



JBoss Enterprise Application Platform 6.3

管理および設定ガイド

Red Hat JBoss Enterprise Application Platform 6 向け

JBoss Enterprise Application Platform 6.3 管理および設定ガイド

Red Hat JBoss Enterprise Application Platform 6 向け

法律上の通知

Copyright © 2014 Red Hat, Inc..

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

概要

本書は、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	15
1.1. RED HAT JBOSS ENTERPRISE APPLICATION PLATFORM 6	15
1.2. JBOSS EAP 6 の機能	15
1.3. JBOSS EAP 6 の操作モード	16
1.4. スタンドアロンサーバー	16
1.5. 管理対象ドメイン	16
1.6. ドメインコントローラー	18
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	18
1.8. ホストコントローラー	19
1.9. サーバークラスタ	20
1.10. JBOSS EAP 6 プロファイル	21
第2章 アプリケーションサーバー管理	22
2.1. JBOSS EAP 6 の起動および停止	22
2.1.1. JBoss EAP 6 の起動	22
2.1.2. JBoss EAP 6 をスタンドアロンサーバーとして起動	22
2.1.3. JBoss EAP 6 を管理対象ドメインとして起動	22
2.1.4. 管理対象ドメインのホストの名前設定	23
2.1.5. 2 台のマシンでの管理対象ドメインの作成	24
2.1.6. 代替設定を用いた JBoss EAP 6 の起動	25
2.1.7. JBoss EAP 6 の停止	27
2.1.8. サーバー実行時に渡すスイッチと引数のリファレンス	29
2.2. サーバーの起動と停止	32
2.2.1. 管理 CLI を使用したサーバーの起動および停止	32
2.2.2. 管理コンソールを使用したサーバーの起動	33
2.2.3. 管理コンソールを使用したサーバーの停止	34
2.3. ファイルシステムパス	34
2.3.1. ファイルシステムパス	34
2.4. 設定ファイル	36
2.4.1. JBoss EAP 6 の設定ファイル	36
2.4.2. 記述子ベースのプロパティ置換	37
2.4.3. 記述子ベースのプロパティ置換の有効化/無効化	39
2.4.4. ファイルの履歴設定	40
2.4.5. 以前の設定でのサーバーの起動	40
2.4.6. 管理 CLI を使用した設定スナップショットの保存	41
2.4.7. 管理 CLI を使用した設定スナップショットのロード	42
2.4.8. 管理 CLI を使用した設定スナップショットの削除	42
2.4.9. 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	43
第3章 管理インターフェース	45
3.1. アプリケーションサーバーの管理	45
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)	45
3.3. 管理コンソールと管理 CLI	46
3.4. 管理コンソール	47
3.4.1. 管理コンソール	47
3.4.2. 管理コンソールへのログイン	47
3.4.3. 管理コンソールの言語の変更	48
3.4.4. EAP コンソールの分析	48
3.4.5. EAP コンソールの Google Analytics の有効化/無効化	49
3.4.6. 管理コンソールを使用したサーバーの設定	51
3.4.7. 管理コンソールでのデプロイメントの追加	52

3.4.8. 管理コンソールでのサーバーの新規作成	52
3.4.9. 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	53
3.4.10. 管理コンソールでのサーバーグループの新規作成	53
3.5. 管理 CLI	54
3.5.1. 管理コマンドラインインターフェース (CLI)	54
3.5.2. 管理 CLI の起動	54
3.5.3. 管理 CLI の終了	54
3.5.4. 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	55
3.5.5. 管理 CLI でのヘルプの取得	55
3.5.6. バッチモードでの管理 CLI の使用	56
3.5.7. CLI のバッチモードコマンド	57
3.5.8. 管理 CLI での操作およびコマンドの使用	58
3.5.9. 管理 CLI 設定オプション	61
3.5.10. 管理 CLI コマンドのリファレンス	63
3.5.11. 管理 CLI 操作のリファレンス	65
3.6. 管理 CLI 操作	67
3.6.1. 管理 CLI によるリソースの属性の表示	67
3.6.2. 管理 CLI でのアクティブユーザーの表示	69
3.6.3. 管理 CLI でのシステムおよびサーバー情報の表示	70
3.6.4. 管理 CLI を使用した操作説明の表示	70
3.6.5. 管理 CLI を使用した操作名の表示	71
3.6.6. 管理 CLI を使用した利用可能なリソースの表示	72
3.6.7. 管理 CLI を使用した利用可能なリソース説明の表示	77
3.6.8. 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	78
3.6.9. 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	78
3.6.10. 管理 CLI での属性の設定	79
3.6.11. 管理 CLI を使用したシステムプロパティーの設定	80
3.7. 管理 CLI コマンド履歴	85
3.7.1. 管理 CLI コマンド履歴の利用	85
3.7.2. 管理 CLI コマンド履歴の表示	85
3.7.3. 管理 CLI コマンド履歴の消去	86
3.7.4. 管理 CLI コマンド履歴の無効化	86
3.7.5. 管理 CLI コマンド履歴の有効化	87
3.8. 管理インターフェース監査ロギング	87
3.8.1. 管理インターフェース監査ロギング	87
3.8.2. 管理 CLI からの管理インターフェース監査ロギングの有効化	87
3.8.3. 管理インターフェースの監査ロギングフォーマッター	88
3.8.4. 管理インターフェースの監査ロギングファイルハンドラー	90
3.8.5. 管理インターフェイスの監査ロギング syslog ハンドラー	90
3.8.6. 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	92
第4章 ユーザー管理	93
4.1. ユーザー作成	93
4.1.1. 管理インターフェースのユーザーの追加	93
4.1.2. ユーザー管理の add-user スクリプトへ引数を渡す	94
4.1.3. add-user コマンド引数	95
4.1.4. ユーザー管理情報の代替プロパティーファイルの指定	96
4.1.5. add-user スクリプトのコマンドラインの例	97
第5章 ネットワークおよびポート設定	99
5.1. インターフェース	99
5.1.1. インターフェース	99
5.1.2. インターフェースの設定	100

5.2. ソケットバインディンググループ	103
5.2.1. ソケットバインディンググループ	103
5.2.2. ソケットバインディングの設定	106
5.2.3. JBoss EAP 6 により使用されるネットワークポート	108
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	111
5.2.5. ポートオフセットの設定	111
5.2.6. リモーティングのメッセージサイズの設定	112
5.3. IPV6	112
5.3.1. IPv6 ネットワーキング向け JVM スタック設定の指定	113
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	113
5.3.3. IPv6 アドレス用 JVM スタック設定の指定	114
第6章 データソース管理	115
6.1. はじめに	115
6.1.1. JDBC	115
6.1.2. JBoss EAP 6 でサポートされるデータベース	115
6.1.3. データソースの型	115
6.1.4. データソースの例	115
6.1.5. -ds.xml ファイルのデプロイメント	116
6.2. JDBC ドライバー	116
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	116
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	117
6.2.3. JDBC ドライバーをダウンロードできる場所	119
6.2.4. ベンダー固有クラスへのアクセス	120
6.3. 非 XA データソース	121
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	121
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	122
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	124
6.4. XA データソース	124
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	124
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	126
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	127
6.4.4. XA リカバリー	128
6.4.4.1. XA リカバリーモジュール	128
6.4.4.2. XA リカバリーモジュールの設定	129
6.5. データソースセキュリティ	130
6.5.1. データソースセキュリティ	130
6.6. データベース接続の検証	131
6.6.1. データベース接続検証設定の指定	131
6.7. データソース設定	133
6.7.1. データソースのパラメーター	133
6.7.2. データソース接続 URL	140
6.7.3. データソースの拡張	140
6.7.4. データソース統計の表示	142
6.7.5. データソースの統計	142
6.8. データソース例	144
6.8.1. PostgreSQL データソースの例	144
6.8.2. PostgreSQL XA データソースの例	145
6.8.3. MySQL データソースの例	146
6.8.4. MySQL XA データソースの例	147
6.8.5. Oracle データソースの例	148
6.8.6. Oracle XA データソースの例	149
6.8.7. Microsoft SQLServer のデータソースの例	150

6.8.8. Microsoft SQLServer XA のデータソースの例	151
6.8.9. IBM DB2 データソースの例	152
6.8.10. IBM DB2 XA のデータソースの例	153
6.8.11. Sybase データソースの例	154
6.8.12. Sybase XA データソースの例	155
第7章 モジュールの設定	157
7.1. はじめに	157
7.1.1. モジュール	157
7.1.2. グローバルモジュール	158
7.1.3. モジュールの依存性	158
7.1.4. サブデプロイメントクラスローダーの分離	159
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	159
7.3. すべてのデプロイメントへのモジュールの追加	160
7.4. カスタムモジュールの作成	161
7.5. 外部 JBOSS モジュールディレクトリーの定義	163
7.6. 参考資料	163
7.6.1. 含まれるモジュール	163
7.6.2. 動的モジュールの名前付け	164
第8章 JSVC	165
8.1. はじめに	165
8.1.1. Jsvc	165
8.1.2. Jsvc を使用した JBoss EAP の起動および停止	165
第9章 グローバル値	170
9.1. バルブ	170
9.2. グローバルバルブ	170
9.3. オーセンティケーターバルブ	170
9.4. グローバルバルブのインストール	170
9.5. グローバルバルブの設定	171
第10章 アプリケーションデプロイメント	173
10.1. アプリケーションデプロイメント	173
10.2. 管理コンソールでのデプロイ	174
10.2.1. 管理コンソールでのアプリケーションデプロイメント管理	174
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	174
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	175
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	176
10.3. 管理 CLI でのデプロイ	177
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	177
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	177
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	178
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	178
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	179
10.4. HTTP API を用いたデプロイ	179
10.4.1. HTTP API を使用したアプリケーションのデプロイ	179
10.5. デプロイメントスキャナーでのデプロイ	182
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	182
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	182
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	183
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	

ブロイ	185
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	186
10.5.6. デプロイメントスキャナー属性のリファレンス	187
10.5.7. デプロイメントスキャナーの設定	188
10.5.8. 管理 CLI でのデプロイメントスキャナーの設定	188
10.6. MAVEN でのデプロイ	191
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	191
10.6.2. Maven によるアプリケーションのデプロイ	191
10.6.3. Maven によるアプリケーションのアンデプロイ	192
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	194
10.8. デプロイメント記述子の上書き	194
第11章 JBOSS EAP 6 のセキュア化	196
11.1. セキュリティーサブシステム	196
11.2. セキュリティーサブシステムの構造	196
11.3. セキュリティーサブシステムの設定	197
11.4. ディープコピーサブジェクトモード	197
11.5. ディープコピーサブジェクトモードの有効化	198
11.6. セキュリティードメイン	199
11.6.1. セキュリティードメイン	199
11.6.2. Picketbox	199
11.6.3. 認証	199
11.6.4. セキュリティードメインでの認証の設定	200
11.6.5. 承認	202
11.6.6. セキュリティードメインにおける承認の設定	202
11.6.7. セキュリティー監査	203
11.6.8. セキュリティー監査の設定	203
11.6.9. 監査ログ	205
11.6.10. セキュリティーマッピング	205
11.6.11. セキュリティードメインでのセキュリティマッピングの設定	205
11.6.12. アプリケーションでのセキュリティドメインの使用	206
11.6.13. JACC (Java Authorization Contract for Containers)	209
11.6.13.1. JACC (Java Authorization Contract for Containers)	209
11.6.13.2. JACC (Java Authorization Contract for Containers) のセキュリティの設定	209
11.6.14. JASPI (Java Authentication SPI for Containers)	210
11.6.14.1. JASPI (Java Authentication SPI for Containers) のセキュリティ	210
11.6.14.2. JASPI (Java Authentication SPI for Containers) のセキュリティの設定	211
11.7. IIOP のセキュア化	211
11.7.1. JBoss IIOP	211
11.7.2. IOR	211
11.7.3. IOR セキュリティーパラメーター	212
11.8. 管理インターフェースセキュリティ	213
11.8.1. デフォルトのユーザーセキュリティ設定	213
11.8.2. 管理インターフェースの詳細設定の概要	214
11.8.3. LDAP	215
11.8.4. 管理インターフェースに対する LDAP を使用した認証	216
11.8.5. HTTP 管理インターフェースの無効化	219
11.8.6. デフォルトセキュリティレルムからのサイレント認証の削除	220
11.8.7. JMX サブシステムへのリモートアクセスの無効化	222
11.8.8. 管理インターフェースのセキュリティレルムの設定	222
11.9. ロールベースアクセス制御を用いた管理インターフェースのセキュア化	223
11.9.1. ロールベースアクセス制御 (RBAC)	223
11.9.2. 管理コンソールおよび CLI でのロールベースアクセス制御	224

11.9.3. サポートされる認証スキーム	224
11.9.4. 標準のロール	225
11.9.5. ロールパーミッション	226
11.9.6. 制約	227
11.9.7. JMX およびロールベースアクセス制御	228
11.9.8. ロールベースアクセス制御の設定	228
11.9.8.1. RBAC 設定タスクの概要	228
11.9.8.2. ロールベースアクセス制御の有効化	229
11.9.8.3. パーミッション組み合わせポリシーの変更	230
11.9.9. ロールの管理	231
11.9.9.1. ロールメンバーシップ	231
11.9.9.2. ユーザーロール割り当ての設定	232
11.9.9.3. 管理 CLI を用いたユーザーロール割り当ての設定	235
11.9.9.4. ロールとユーザーグループ	239
11.9.9.5. グループロール割り当ての設定	239
11.9.9.6. 管理 CLI を用いたグループロール割り当ての設定	242
11.9.9.7. LDAP での承認とグループローディング	246
username-to-dn	246
グループ検索	248
一般的なグループ検索	249
11.9.9.8. スコープ指定ロール	251
11.9.9.9. スコープ指定ロールの作成	252
11.9.10. 制約の設定	254
11.9.10.1. 機密性制約の設定	254
11.9.10.2. アプリケーションリソース制約の設定	255
11.9.10.3. Vault 式制約の設定	256
11.9.11. 制約に関する参考情報	258
11.9.11.1. アプリケーションリソース制約に関する参考情報	258
11.9.11.2. 機密性制約に関する参考情報	259
11.10. ネットワークセキュリティー	268
11.10.1. 管理インターフェースのセキュア化	268
11.10.2. JBoss EAP 6 が使用するネットワークインターフェースの指定	268
11.10.3. JBoss EAP 6 により使用されるネットワークポート	269
11.10.4. JBoss EAP 6 で動作可能なネットワークファイアウォールの設定	272
11.11. JAVA セキュリティーマネージャー	275
11.11.1. Java Security Manager	275
11.11.2. Java Security Manager 内での JBoss EAP 6 の実行	276
11.11.3. Java Security Manager のポリシー	277
11.11.4. Java Security Manager ポリシーの記述	278
11.11.5. Security Manager ポリシーのデバッグ	279
11.12. SSL 暗号化	279
11.12.1. JBoss EAP 6 Web サーバーでの SSL 暗号化の実装	279
11.12.2. SSL 暗号化キーおよび証明書の生成	282
11.12.3. SSL コネクターリファレンス	285
11.13. 機密性の高い文字列のパスワード VAULT	291
11.13.1. パスワード vault システム	291
11.13.2. 機密性の高い文字列を格納する Java キーストアの作成	291
11.13.3. キーストアパスワードのマスキングとパスワード vault の初期化	293
11.13.4. パスワード vault を使用するよう JBoss EAP 6 を設定	294
11.13.5. パスワード Vault のカスタム実装を使用するよう JBoss EAP 6 を設定	295
11.13.6. Java キーストアに暗号化された機密性の高い文字列の保存および読み出し	296
11.13.7. アプリケーションでの機密性の高い文字列の保存および解決	298
11.14. FIPS 140-2 準拠の暗号化	301

11.14.1. FIPS 140-2 の準拠	301
11.14.2. FIPS 140-2 準拠のパスワード	301
11.14.3. Red Hat Enterprise Linux 6 にて SSL の FIPS 140-2 準拠の暗号を有効化	301
11.14.4. Apache HTTP サーバーでの FIPS 140-2 の有効化	304
第12章 セキュリティー管理リファレンス	306
12.1. 含まれる認証モジュール	306
12.2. 含まれる承認モジュール	334
12.3. 含まれるセキュリティーマッピングモジュール	335
12.4. 含まれるセキュリティー監査プロバイダーモジュール	338
第13章 サブシステムの設定	340
13.1. サブシステム設定の概要	340
第14章 ロギングサブシステム	341
14.1. はじめに	341
14.1.1. ロギングの概要	341
14.1.2. 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	341
14.1.3. ブートロギングの設定	341
14.1.4. ガベッジコレクションロギング	342
14.1.5. 暗黙的なロギング API の依存関係	342
14.1.6. デフォルトのログファイルの場所	342
14.1.7. ロギングのフィルター式	343
14.1.8. ログレベル	345
14.1.9. サポート対象のログレベル	345
14.1.10. ログカテゴリー	346
14.1.11. ルートロガーについて	346
14.1.12. ログハンドラー	347
14.1.13. ログハンドラーのタイプ	347
14.1.14. ログフォーマッター	348
14.1.15. ログフォーマッター構文	348
14.2. 管理コンソールでのロギングの設定	349
14.3. CLI でのロギング設定	350
14.3.1. CLI でのルートロガーの設定	350
14.3.2. CLI でのログカテゴリー設定	352
14.3.3. CLI でのコンソールログハンドラーの設定	354
14.3.4. CLI でのファイルログハンドラーの設定	358
14.3.5. CLI での周期ログハンドラーの設定	361
14.3.6. CLI でのサイズログハンドラーの設定	366
14.3.7. CLI での非同期ログハンドラーの設定	371
14.3.8. syslog-handler の設定	375
14.3.9. CLI でのカスタムログフォーマッターの設定	376
14.4. デプロイメントごとのロギング	377
14.4.1. デプロイメントごとのロギング	377
14.4.2. デプロイメントごとのロギングの無効化	377
14.5. ロギングプロファイル	378
14.5.1. ロギングプロファイル	378
14.5.2. CLI を使用した新しいロギングプロファイルの作成	378
14.5.3. CLI を使用したロギングプロファイルの設定	379
14.5.4. アプリケーションでのロギングプロファイルの指定	380
14.5.5. ロギングプロファイル設定の例	380
14.6. ロギング設定プロパティー	382
14.6.1. ルートロガーのプロパティー	382
14.6.2. ログカテゴリーのプロパティー	382

14.6.3. コンソールログハンドラーのプロパティ	383
14.6.4. ファイルログハンドラープロパティ	383
14.6.5. 周期ログハンドラープロパティ	384
14.6.6. サイズログハンドラープロパティ	385
14.6.7. 同期ログハンドラープロパティ	386
14.7. ロギング用 XML 設定例	387
14.7.1. ルートロガーの XML 設定例	387
14.7.2. ログカテゴリーの XML 設定例	387
14.7.3. コンソールログハンドラーの XML 設定例	388
14.7.4. ファイルログハンドラーの XML 設定例	388
14.7.5. 定期ログハンドラーの XML 設定例	388
14.7.6. サイズログハンドラーの XML 設定例	388
14.7.7. 非同期ログハンドラーの XML 設定例	388
第15章 INFINISPAN	390
15.1. INFINISPAN	390
15.2. クラスタリングモード	390
15.3. キャッシュコンテナ	391
15.4. キャッシュストア	393
15.5. INFINISPAN の統計	393
15.6. INFINISPAN 統計収集の有効化	393
15.6.1. 起動設定ファイルでの Infinispan 統計収集の有効化	394
15.6.2. 管理 CLI での Infinispan 統計収集の有効化	394
15.6.3. Infinispan 統計収集の有効化を検証	395
15.7. JGROUPS	395
15.7.1. JGroups	395
第16章 JVM	396
16.1. JVM	396
16.1.1. JVM 設定	396
16.1.2. 管理コンソールでの JVM 状態の表示	397
16.1.3. JVM の設定	398
第17章 WEB サブシステム	401
17.1. WEB サブシステムの設定	401
17.2. デフォルトの WELCOME WEB アプリケーションの置き換え	405
第18章 WEB サービスサブシステム	406
18.1. WEB サービスオプションの設定	406
第19章 HTTP クラスタリングおよび負荷分散	408
19.1. はじめに	408
19.1.1. 高可用性および負荷分散クラスター	408
19.1.2. 高可用性が有益なコンポーネント	408
19.1.3. HTTP コネクターの概要	409
19.1.4. ワーカーノード	411
19.2. コネクター設定	411
19.2.1. JBoss EAP 6 にて HTTP コネクターのスレッドプールを定義	412
19.3. WEB サーバーの設定	415
19.3.1. スタンドアロン Apache HTTP Server	415
19.3.2. JBoss EAP 6 に含まれる Apache HTTP Server のインストール (Zip)	415
19.3.3. Red Hat Enterprise Linux (RHEL) 5、6、および 7 への Apache HTTP Server のインストール (RPM)	417
19.3.4. httpd 上の mod_cluster 設定	419

19.3.5. 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	423
19.3.6. 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	424
19.4. クラスタリング	425
19.4.1. クラスタリングサブシステムに TCP 通信を使用	425
19.4.2. TCP を使用するよう JGroups サブシステムを設定	426
19.4.3. mod_cluster サブシステムのアドバタイズの無効化	427
19.4.4. HornetQ クラスタリングの UDP の TCP への変更	430
19.5. WEB、HTTP コネクター、および HTTP クラスタリング	431
19.5.1. mod_cluster HTTP コネクター	431
19.5.2. mod_cluster サブシステムの設定	432
19.5.3. 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)	444
19.5.4. 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)	447
19.5.5. 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の設定	448
19.5.6. mod_cluster ワーカーノードの設定	449
19.5.7. クラスタ間のトラフィックの移行	455
19.6. APACHE MOD_JK	455
19.6.1. Apache mod_jk HTTP コネクター	455
19.6.2. JBoss EAP 6 が Apache Mod_jk と通信するよう設定	456
19.6.3. Apache HTTP Server への mod_jk モジュールのインストール (ZIP)	457
19.6.4. Apache HTTP Server への Mod_jk モジュールのインストール (RPM)	460
19.6.5. Apache Mod_jk ワーカーの設定リファレンス	463
19.7. APACHE MOD_PROXY	466
19.7.1. Apache mod_proxy HTTP コネクター	466
19.7.2. Apache HTTP Server への mod_proxy HTTP コネクターのインストール	466
19.8. MICROSOFT ISAPI	469
19.8.1. インターネットサーバー API (ISAPI) HTTP コネクター	469
19.8.2. 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	469
19.8.3. Microsoft IIS が ISAPI リダイレクターを使用するよう設定	469
19.8.4. ISAPI リダイレクターがクライアント要求を JBoss EAP 6 に送信するよう設定	471
19.8.5. ISAPI がクライアント要求を複数の JBoss EAP 6 サーバーで分散するよう設定	473
19.9. ORACLE NSAPI	476
19.9.1. Netscape Server API (NSAPI) HTTP コネクター	476
19.9.2. Oracle Solaris での NSAPI コネクターの設定	476
19.9.3. NSAPI を基本的な HTTP コネクターとして設定	478
19.9.4. NSAPI を負荷分散クラスターとして設定	479
第20章 メッセージング	482
20.1. はじめに	482
20.1.1. HornetQ	482
20.1.2. Java Messaging Service (JMS)	482
20.1.3. サポートされているメッセージ形式	482
20.2. トランスポートの設定	483
20.2.1. アクセプターおよびコネクター	483
20.2.2. Netty TCP の設定	483
20.2.3. Netty セキュアソケットレイヤー (SSL) の設定	486
20.2.4. Netty HTTP の設定	488
20.2.5. Netty サーブレットの設定	489
20.3. JAVA NAMING AND DIRECTORY INTERFACE (JNDI)	491
20.4. デッド接続の検出	491
20.4.1. サーバーでデッド接続リソースを閉じる	491
20.4.2. クライアントサイド障害の検出	493

20.5. サイズの大きなメッセージの処理	493
20.5.1. サイズの大きなメッセージの処理	493
20.5.2. HornetQ の大きなメッセージの設定	494
20.5.3. パラメーターの設定	494
20.6. ページング	495
20.6.1. ページング	495
20.6.2. ページファイル	495
20.6.3. ページングフォルダーの設定	495
20.6.4. ページングモード	496
20.7. 迂回	497
20.7.1. 特別な迂回	498
20.7.2. 特別でない迂回	498
20.8. 設定	499
20.8.1. JMS サーバーの設定	499
20.8.2. JMS アドレスの設定	504
20.8.3. HornetQ でのメッセージングの設定	507
20.8.4. HornetQ のロギングの有効化	508
20.8.5. HornetQ Core Bridge の設定	508
20.8.6. JMS ブリッジの設定	509
20.8.7. 遅延再配信の設定	512
20.8.8. デッドレターアドレスの設定	512
20.8.9. メッセージ期限切れアドレス	512
20.8.10. HornetQ 設定属性のリファレンス	513
20.8.11. メッセージの有効期限の設定	520
20.9. メッセージのグループ化	521
20.9.1. メッセージのグループ化	521
20.9.2. クライアント側での HornetQ Core API の使用	521
20.9.3. Java Messaging Service (JMS) クライアントのサーバー設定	521
20.9.4. クラスター化されたグルーピング	522
20.9.5. 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	523
20.10. 複製メッセージの検出	523
20.10.1. 複製メッセージの検出	523
20.10.2. メッセージ送信への複製メッセージ検出の使用	524
20.10.3. 複製 ID キャッシュの設定	525
20.10.4. ブリッジおよびクラスター接続での複製検出の使用	525
20.11. JMS ブリッジ	525
20.11.1. ブリッジ	525
20.11.2. JMS ブリッジの作成	526
20.12. 永続性	528
20.12.1. HornetQ の永続性	528
20.13. HORNETQ クラスタリング	529
20.13.1. サーバーディスカバリー	530
20.13.2. ブロードキャストグループ	530
20.13.2.1. 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	530
20.13.2.2. JGroups ブロードキャストグループ	532
20.13.3. ディスカバリーグループ	532
20.13.3.1. 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	533
20.13.3.2. サーバー上での JGroups ディスカバリーグループの設定	534
20.13.3.3. 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	535
20.13.3.4. コア API のディスカバリー設定	535
20.13.4. サーバー側の負荷分散	536
20.13.4.1. クラスター接続の設定	536
20.14. 高可用性	539

20.14.1. 高可用性とは	539
20.14.2. HornetQ Shared の共有ストア	540
20.14.3. HornetQ ストレージの設定	541
20.14.4. HornetQ のジャーナルタイプ	541
20.14.5. 共有ストアを持つ専用トポロジー向けの HornetQ の設定	542
20.14.6. HornetQ のメッセージレプリケーション	543
20.14.7. レプリケーションに対する HornetQ サーバーの設定	543
20.14.8. 高可用性 (HA) フェールオーバー	545
20.14.9. HornetQ バックアップサーバー上のデプロイメント	545
第21章 TRANSACTION サブシステム	547
21.1. トランザクションサブシステムの設定	547
21.1.1. トランザクション設定の概要	547
21.1.2. トランザクションマネージャーの設定	547
21.1.3. JTA Transaction API を使用するようデータソースを設定	551
21.1.4. XA Datasource の設定	552
21.1.5. トランザクションログメッセージ	552
21.1.6. トランザクションサブシステムのログ設定	553
21.2. トランザクション管理	554
21.2.1. トランザクションの参照と管理	554
21.3. トランザクションに関するリファレンス	559
21.3.1. JBoss Transactions エラーと例外	559
21.3.2. JTA トランザクションの制限	559
21.4. ORB 設定	559
21.4.1. Common Object Request Broker Architecture (CORBA)	559
21.4.2. JTS トランザクション用 ORB の設定	559
21.5. JDBC オブジェクトストアのサポート	561
21.5.1. トランザクションの JDBC ストア	561
第22章 メールサブシステム	563
22.1. 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	563
第23章 ENTERPRISE JAVABEANS	565
23.1. はじめに	565
23.1.1. Enterprise JavaBeans の概要	565
23.1.2. 管理者向け Enterprise JavaBeans の概要	565
23.1.3. エンタープライズ Bean	565
23.1.4. セッション Bean	566
23.1.5. メッセージ駆動型 Bean	566
23.2. BEAN プールの設定	566
23.2.1. Bean プール	566
23.2.2. Bean プールの作成	566
23.2.3. Bean プールの削除	568
23.2.4. Bean プールの編集	569
23.2.5. 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	570
23.3. EJB スレッドプールの設定	571
23.3.1. エンタープライズ Bean スレッドプール	571
23.3.2. スレッドプールの作成	571
23.3.3. スレッドプールの削除	572
23.3.4. スレッドプールの編集	573
23.4. セッション BEAN の設定	574
23.4.1. セッション Bean のアクセスタイムアウト	575
23.4.2. 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	575
23.5. メッセージ駆動型 BEAN の設定	576

23.5.1. 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	576
23.6. EJB3 タイマーサービスの設定	577
23.6.1. EJB3 タイマーサービス	577
23.6.2. EJB3 タイマーサービスの設定	577
23.7. EJB3 非同期呼び出しサービスの設定	578
23.7.1. EJB3 非同期呼び出しサービス	578
23.7.2. EJB3 非同期呼び出しサービスのスレッドプールの設定	578
23.8. EJB3 リモート呼び出しサービスの設定	578
23.8.1. EJB3 リモートサービス	579
23.8.2. EJB3 リモートサービスの設定	579
23.9. EJB 2.X エンティティ BEAN の設定	579
23.9.1. EJB エンティティ Bean	579
23.9.2. コンテナ管理による永続性	579
23.9.3. EJB 2.x のコンテナ管理による永続性の有効化	579
23.9.4. EJB 2.x のコンテナ管理による永続性の設定	580
23.9.5. HiLo キージェネレーター用の CMP サブシステムプロパティ	582
第24章 JAVA CONNECTOR ARCHITECTURE (JCA)	583
24.1. はじめに	583
24.1.1. Java EE Connector API (JCA)	583
24.1.2. Java Connector Architecture (JCA)	583
24.1.3. リソースアダプター	583
24.2. JAVA CONNECTOR ARCHITECTURE (JCA) サブシステムの設定	584
24.3. リソースアダプターのデプロイ	589
24.4. デプロイされたリソースアダプターの設定	590
24.5. リソースアダプター記述子リファレンス	596
24.6. 定義された接続統計の表示	600
24.7. リソースアダプターの統計	601
24.8. WEBSPPHERE MQ リソースアダプターのデプロイ	602
24.9. WEBSPPHERE MQ リソースアダプターのインストール	607
24.10. サードパーティー JMS プロバイダーを使用するよう汎用 JMS リソースアダプターを設定	607
第25章 AMAZON EC2 での JBOSS EAP 6 のデプロイ	612
25.1. はじめに	612
25.1.1. Amazon EC2	612
25.1.2. Amazon Machine Instance (AMI)	612
25.1.3. JBoss Cloud Access	612
25.1.4. JBoss Cloud Access 機能	612
25.1.5. サポートされる Amazon EC2 インスタンスタイプ	613
25.1.6. サポート対象 Red Hat AMI	613
25.2. AMAZON EC2 での JBOSS EAP 6 のデプロイ	614
25.2.1. Amazon EC2 での JBoss EAP 6 のデプロイ (概要)	614
25.2.2. 非クラスター化の JBoss EAP 6	614
25.2.2.1. 非クラスターインスタンス	614
25.2.2.2. 非クラスターインスタンス	614
25.2.2.2.1. 非クラスター化の JBoss EAP 6 インスタンスの起動	614
25.2.2.2.2. 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	616
25.2.2.2.3. 非クラスター化 JBoss EAP 6 インスタンスのテスト	617
25.2.2.3. 非クラスター化管理対象ドメイン	618
25.2.2.3.1. 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	618
25.2.2.3.2. 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	620
25.2.2.3.3. 非クラスター化 JBoss EAP 6 の管理対象ドメインのテスト	621
25.2.2.3.4. 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	623

25.2.3. クラスター化された JBoss EAP 6	624
25.2.3.1. クラスターインスタンスについて	624
25.2.3.2. リレーショナルデータベースサービスデータベースインスタンスの作成	624
25.2.3.3. Virtual Private Cloud	625
25.2.3.4. Virtual Private Cloud (VPC) の作成	626
25.2.3.5. 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	626
25.2.3.6. VPC プライベートサブネットデフォルトルートの設定	628
25.2.3.7. Identity and Access Management (IAM)	629
25.2.3.8. IAM セットアップの設定	629
25.2.3.9. S3 バケット	630
25.2.3.10. S3 バケットセットアップの設定	630
25.2.3.11. クラスターインスタンス	631
25.2.3.11.1. クラスター化された JBoss EAP 6 AMI の起動	631
25.2.3.11.2. クラスター化 JBoss EAP 6 インスタンスのテスト	635
25.2.3.12. クラスター化管理対象ドメイン	635
25.2.3.12.1. 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	636
25.2.3.12.2. 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	638
25.2.3.12.3. クラスター化された JBoss EAP 6 管理対象ドメインのテスト	640
25.3. JBOSS OPERATIONS NETWORK (JON) での監視の確立	641
25.3.1. AMI 監視	641
25.3.2. 接続要件	642
25.3.3. Network Address Translation (NAT)	642
25.3.4. Amazon EC2 および DNS	642
25.3.5. EC2 でのルーティング	643
25.3.6. JON での終了と再起動	643
25.3.7. JBoss Operations Network で登録するインスタンスの設定	644
25.4. ユーザースクリプト設定	644
25.4.1. 永続的な設定パラメーター	644
25.4.2. カスタムスクリプトパラメーター	648
25.5. トラブルシューティング	648
25.5.1. Amazon EC2 のトラブルシューティング	648
25.5.2. 診断情報	648
付録A 補足リファレンス	650
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	650
A.2. RED HAT ENTERPRISE LINUX でのデフォルト JDK の設定	650
付録B 改訂履歴	652

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、オープンな標準に基いて構築され、Java Enterprise Edition 6 の仕様に準拠するミドルウェアプラットフォームです。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技術が JBoss Application Server 7 と統合されます。

JBoss EAP 6 には、必要な場合にだけサービスを有効にできる新しいモジュール構造が含まれます (サービスの起動時間が短縮されます)。

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。

[Report a bug](#)

1.2. JBOSS EAP 6 の機能

表1.1 JBoss EAP 6.3.0 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	<ul style="list-style-type: none">● 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 (セキュリティドメインを含む) 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。XML 設定ファイルの編集は必要なくなりました。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 ディレクトリーには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。共通でサーバー固有の lib ディレクトリーは廃止されました。 domain ディレクトリーと standalone ディレクトリーには、それぞれドメインデプロイメント用のアーティファクトおよび設定ファイルと、スタンドアロンデプロイメント用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	モジュールはオンデマンドで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、セキュリティの利点が提供され、起動および再起動の時間が短縮されます。
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。また、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。
リソースの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。その他の利点として、JBoss EAP 6 では JBoss EAP 5 よりも起動および停止時間が短縮されます。

[Report a bug](#)

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は変わりません。高可用性 (HA) クラスターの機能はどちらのモードからでも利用できることに注意してください。複数のスタンドアロンサーバーを設定して、HA クラスターを構築できます。

[Report a bug](#)

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。

[Report a bug](#)

1.5. 管理対象ドメイン

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。

一元管理された複数の 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 インスタンスは共通の管理ポリシーを共有します。

各ドメインは1つの *ドメインコントローラー*、1つ以上の *ホストコントローラー*、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あり、各サーバーはドメインの管理ポリシーに従って設定されます。ドメインコントローラーはホストコントローラーとしても機能します。

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へ委譲するために設定されます。

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て、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これらのインスタンス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。

サーバーグループは、JBoss EAP がインストールされているサーバーインスタンスのセットで、すべてが1つとして管理および設定されます。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。

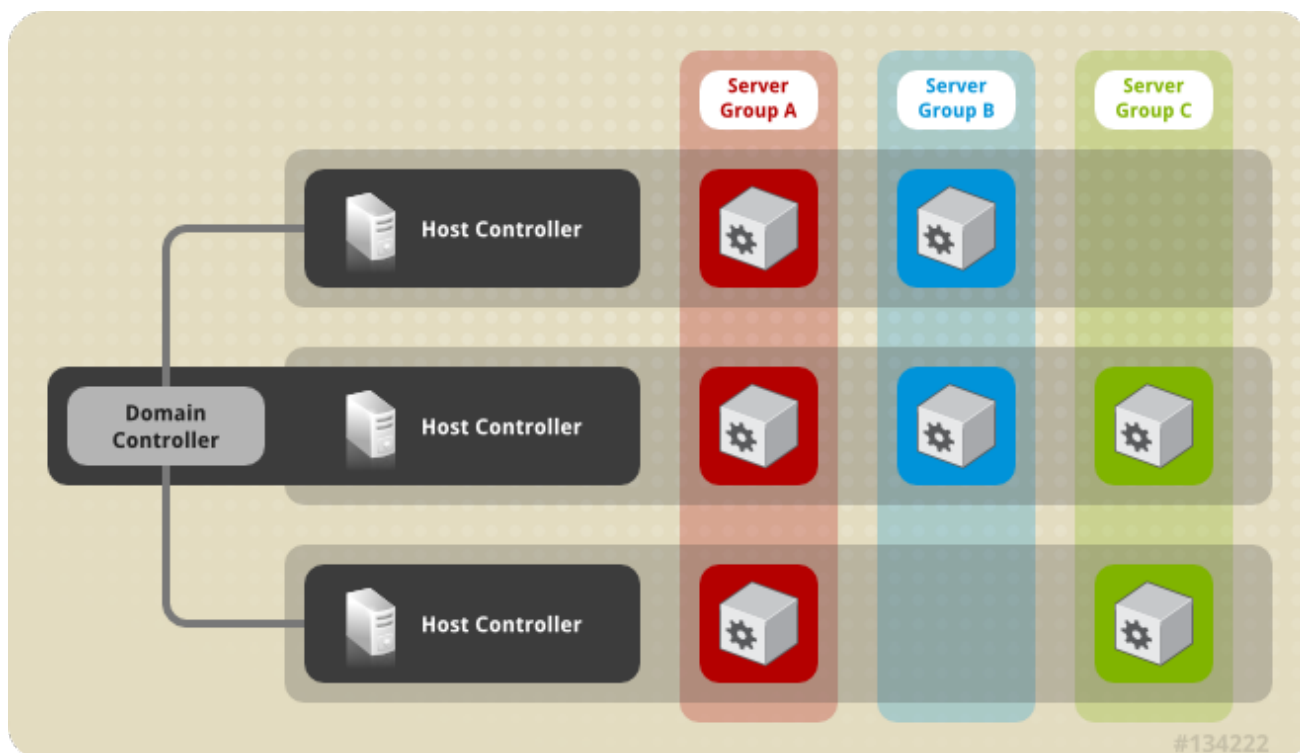


図1.1 管理対象ドメインを表す図

[Report a bug](#)

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に設定されます。

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** ファイルに格納されます。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上にある未展開の JBoss EAP 6 インストールファイル内にあります。

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/** ディレクトリーに存在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しないホストコントローラー上のインストールには必要ありませんが、このようなサーバー上に **domain.xml** ファイルがあっても問題はありません。

domain.xml ファイルには、あるドメインのサーバーインスタンス上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。また、ドメイン設定にはソケットのグループの定義やサーバーグループの定義も含まれます。

[Report a bug](#)

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

管理対象ドメインを設定するとき、ドメインコントローラーのコンタクト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 6.3 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るようになりました。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を用いて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しいマスターへ自動的にフェー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。

```
<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 security-realm>
</domain-controller>
```

```

    </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ドメインコントローラーとして動作するよう昇格できます。



注記

--backup オプションでホストコントローラーを開始すると、コントローラーによってドメイン設定のローカルコピーが維持されます。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。

[Report a bug](#)

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** スクリプトがホスト上で実行されると起動します。

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 **domain/configuration/host.xml** ファイルから設定を読み取ります。このファイルは、ホストのファイルシステム上にある未展開の JBoss EAP 6 インストールファイルに存在します。**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ています。

- 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** に宣言された名前付きインターフェースの定義は **host.xml** にある実際のマシン固有の IP アドレスへマップできます。さらに、**domain.xml** の抽象パス名を **host.xml** のファイルシステムパスへマップできます。

[Report a bug](#)

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。

ドメインは複数のサーバーグループで構成されます。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。たとえば、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。

```
<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対して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** でサーバーごとに上書きできます。しかし、すべてのサーバーグループの必須要素であるため、この要素がないとドメインが起動できません。

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。
- jvm: グループの全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 を開始するために使用される設定を作成します。

[Report a bug](#)

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コンセプトは使用されないようになりました。JBoss EAP 6 では、設定に関するすべての情報を保持する少数の設定ファイルが使用されるようになりました。

モジュールとドライバーは必要に応じてロードされるようになりました。そのため、サーバーを効率的に起動するために以前のバージョンの JBoss EAP 6 で使用されたデフォルトプロファイルの概念は適用されません。

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性が決定、順番付け、および解決され、正しい順番でロードされます。モジュールはデプロイメントで必要が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すると、手作業でモジュールを無効にしたり、ドライバーまたは他のサービスをアンロードしたりできますが、ほとんどの場合でこの作業は必要ありません。モジュールを使用するアプリケーションがない場合、モジュールはロードされません。

[Report a bug](#)

第2章 アプリケーションサーバー管理

2.1. JBoss EAP 6 の起動および停止

2.1.1. JBoss EAP 6 の起動

JBoss EAP 6 を次のいずれかの方法で起動します。

- [「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」](#)

[Report a bug](#)

2.1.2. 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
次のコマンドを実行します: **`EAP_HOME/bin/standalone.sh`**
2. Microsoft Windows Server の場合
次のコマンドを実行します: **`EAP_HOME\bin\standalone.bat`**
3. 他のパラメーターを指定する (任意)
起動スクリプトに渡すことができる他のパラメーターの一覧を出力するには、**`-h`** パラメーターを使います。

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。

[Report a bug](#)

2.1.3. 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、この手順をドメインコントローラーで使用した後に、関連するホストコントローラーおよびドメインに関連する他のホストに対して使用してください。

手順2.2 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
コマンド **`EAP_HOME/bin/domain.sh`** を実行します。
2. Microsoft Windows Server の場合
コマンド **`EAP_HOME\bin\domain.bat`** 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)

起動スクリプトに渡すことができる各種パラメーターを確認するには、**-h** パラメーターを使います。

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。

[Report a bug](#)

2.1.4. 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. **host.xml** 設定ファイルにある **host** 要素の **name** 属性 (設定されている場合)。
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、または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 では **COMPUTERNAME** 環境変数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。

環境変数の設定方法は、ご使用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2.3 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. ファイルにある **host** 要素を見つけます。以下に例を示します。

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

3. この要素が存在する場合は、**name="HOST_NAME"** 属性宣言を削除します。削除後、**host** 要素は以下になるはずです。

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

4. 以下の例のように、**-Djboss.host.name** 引数を渡してサーバーを起動します。

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

手順2.4 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
```

例を以下に示します。

```
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)
```

例を以下に示します。

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

次の結果が表示されるはずです。

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

これは、**host-slave01.xml** ファイルのホスト **name** 属性を以下のように変更します。

```
<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
```

例を以下に示します。

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

[Report a bug](#)

2.1.5. 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。詳細については、「[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 2.5 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. ホストがドメインコントローラーを認証できるようにするために、`add-user.sh` スクリプトを使用して管理ユーザー (例: **slave01**) を追加します。**add-user** 出力の **SECRET_VALUE** に注意してください。
- 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** 設定ファイルでドメインを起動します。
- c. **-bmanagement=\$IP1** を使用して、他のマシンがドメインコントローラーを見えるようにします。

```
[$JBOSS_HOME/bin]$ ./domain.sh --host-config=host-master.xml -bmanagement=$IP1
```

2. マシン 2 での作業

- a. ユーザークレデンシャルを用いて **\$JBOSS_HOME/domain/configuration/host-slave.xml** ファイルを更新します。

```
<?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>
          ...
        </security-realm>
      </security-realms>
    </management>
  </host>
```

- b. ホストを起動します。

```
[$JBOSS_HOME/bin]$ ./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:

```
[$JBOSS_HOME/bin]$ ./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/
```

[Report a bug](#)

2.1.6. 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。しかし、サーバーを起

動してから設定を手動で指定することができます。この手順は管理対象ドメインとスタンドアロンサーバーのどちらを使用するか、またどのオペレーティングシステムを使用するかによって多少の違いがあります。

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、代替の設定ファイルを作成します。管理対象ドメインの場合は、設定ファイルを **EAP_HOME/domain/configuration/** ディレクトリーに置く必要があります。スタンドアロンサーバーの場合は、設定ファイルを **EAP_HOME/standalone/configuration/** ディレクトリーに置く必要があります。



注記

EAP_HOME/docs/examples/configs/ ディレクトリーに複数の設定例が含まれています。これらの例を使用し、クラスタリングや Transaction XTS API などの追加機能を有効にします。

一部の設定例は、使用する前に変更する必要があります。**standalone-picketlink.xml**、**standalone-genericjms.xml**、および **standalone-hornetq-colocated.xml** の設定ファイルを変更せずに使用するとエラーが発生します。

手順2.6 代替設定を用いたインスタンスの起動

1. スタンドアロンサーバー

スタンドアロンサーバーでは、**--server-config** パラメーターに設定ファイルのファイル名をオプションとして指定します。設定ファイルは **EAP_HOME/standalone/configuration/** ディレクトリーに置く必要があります、このディレクトリーからの相対パスで設定ファイル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
```

この例は、**EAP_HOME\standalone\configuration\standalone-alternate.xml** 設定ファイルを使用します。

2. 管理対象ドメイン

管理対象ドメインでは、設定ファイルのファイル名を **--domain-config** パラメーターのオプションとして指定します。設定ファイルは **EAP_HOME/domain/configuration/** ディレクトリーに置く必要があります、このディレクトリーからの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
```

この例は、**EAP_HOME\domain\configuration\domain-alternate.xml** 設定ファイルを使用します。

結果

代替の設定ファイルを使用して JBoss EAP が起動されます。

[Report a bug](#)

2.1.7. 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、対話的に起動されたインスタンスを停止し、サービスにより起動され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の詳細については、「[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の詳細については、「[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2.7 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 が実行されているターミナルで **Ctrl-C** を押します。

- 手順2.8 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 でサービススクリプトを記述したときは、**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** を使用できます。ここで *scriptname* はスクリプトの名前に置き換えます。

- ■ Microsoft Windows Server

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

- 手順2.9 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)

以下のコマンドはいずれも 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** と **jps** 自体の 2 つのプロセスに対する 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。

- 詳細モードで **jps** コマンドを使用すると、見つかった java プロセスに関する詳細情報が提供されます。

以下は、実行している EAP プロセスを PID およびロールで識別する、詳細な **jps** コマンドからの出力の一部になります。

```
$ 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 インスタンスの情報を返すこともできます。

以下は、実行している EAP プロセスを PID およびロールで識別する、詳細な **ps** コマンドからの出力の一部になります。

```
$ 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。

これらのコマンドに **grep** ユーティリティーを使用すると **Process Controller** を識別できます。

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

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

2. **kill PID** を実行して、**TERM** シグナルをプロセスに送信します。*PID* は前述のコマンドの1つを使用して識別されたプロセス ID に置き換えます。

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。

[Report a bug](#)

2.1.8. サーバー実行時に渡すスイッチと引数のリファレンス

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追加の引数とスイッチを受け取ります。これらのパラメーターを使用すると、**standalone.xml**、**domain.xml**、および **host.xml** の設定ファイルで定義されたものとは別の設定でサーバーを起動できます。これには、ソケットバインディングの代替セットや二次設定でのサーバーの起動が含まれることがあります。起動時にヘルプスイッチを渡すと、利用可能なこれらのパラメーターのリストを表示できます。

例2.5

以下の例は、「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」に説明のあるサーバーの起動と似ていますが、**-h** または **--help** スイッチが追加で使用されています。help スイッチの結果は以下の表を参照してください。

スタンドアロンモード:

```
[localhost bin]$ standalone.sh -h
```

ドメインモード:

```
[localhost bin]$ domain.sh -h
```

表2.1 実行時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、管理要求が許可されますが、他のランタイムサービスは起動されず、エンドユーザーの要求は許可されません。
--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。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	システムプロパティ jboss.bind.address を該当する値に設定します。
-b<interface>=<value>	Standalone、Domain	システムプロパティ jboss.bind.address.<interface> を指定の値に設定します。
--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。
-c <config>、-c=<config>	Standalone	使用するサーバー設定ファイルの名前。デフォルト値は 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。デフォルト値は 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、Domain	システムプロパティを設定します。
--domain-config=<config>	Domain	使用するサーバー設定ファイルの名前。デフォルト値は domain.xml です。

引数またはスイッチ	モード	説明
-h, --help	Standalone、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。デフォルト値は 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 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の設定では、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 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の設定では、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 사용되는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。
--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、Domain	該当する URL からシステムプロパティをロードします。
--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を設定します。

引数またはスイッチ	モード	説明
--server-config=<config>	Standalone	使用するサーバー設定ファイルの名前。デフォルト値は standalone.xml です。
-u <value> 、 -u=<value>	Standalone、Domain	システムプロパティ jboss.default.multicast.address を指定の値に設定します。
-v 、 -V 、 --version	Standalone、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。

[Report a bug](#)

2.2. サーバーの起動と停止

2.2.1. 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](#)」

スタンドアロンモードでは、管理 CLI または管理コンソールを使用してサーバーを起動および停止できます。ドメインモードでは、サーバーインスタンスのみを起動できます。これらの管理ツールを使用すると、単一のスタンドアロンサーバーインスタンスを制御したり、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選択的に管理したりできます。ドメインモードで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](#)」の手順を参照してください。管理 CLI を使用する場合は、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インスタンスでプロセスが異なります。

管理 CLI を用いたスタンドアロンサーバーの起動および停止

スタンドアロンサーバーインスタンスは、コマンドラインスクリプトで起動でき、管理 CLI より **shutdown** コマンドを使用してシャットダウンできます。インスタンスが再度必要な場合は、「[JBoss EAP 6 をスタンドアロンサーバーとして起動](#)」の説明どおりに起動プロセスを再実行します。

例2.6 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
```

管理 CLI を用いた管理対象ドメインの起動および停止

管理対象ドメインを実行している場合は、管理コンソールを使用するとドメイン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、ドメイン全体の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2.7 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と同様に、宣言された管理対象ドメインホストをシャットダウンするには **shutdown** コマンドを使用します。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、「master」という名前のサーバーホストを停止します。**tab** キーを使用して文字列を補

完し、利用可能なホスト値などの可視変数を表示します。

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

例2.8 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

この例は、**start** および **stop** 操作を呼び出す前に、サーバー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。**tab** キーを使用して文字列を補完し、利用可能なサーバーグループ名の値などの可視変数を表示します。

```
[domain@localhost:9999 /] /server-group=main-server-group:start-servers
```

```
[domain@localhost:9999 /] /server-group=main-server-group:stop-servers
```

例2.9 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

この例は、**start** および **stop** 操作を呼び出す前に、ホストおよびサーバーの設定を宣言することで、**master** ホスト上の **server-one** という名前のサーバーインスタンスを起動および停止します。**tab** 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。

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

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

[Report a bug](#)

2.2.2. 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2.10 管理対象ドメイン向けのサーバーの起動

1. コンソールの右上にある **Runtime** タブを選択します。Server メニューを展開し、**Overview** を選択します。
2. **Server Instances** のリストから、起動するサーバーを選択します。実行されているサーバーはチェックマークで示されます。

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 **Start Server** テキストをクリックします。クリックすると、ダイアログボックスが開きます。**Confirm** ボタン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。

[Report a bug](#)

2.2.3. 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2.11 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. コンソールの右上にある **Runtime** タブを選択します。Domain メニューを展開し、**Overview** を選択します。
2. **Hosts, groups and server instances** の表に使用可能な **Server Instances** のリストが表示されます。稼働中のサーバーにはチェックマークが付きます。
3. 選択したサーバーにカーソルを合わせ、表示される **Stop Server** テキストをクリックします。確認ダイアログウィンドウが表示されます。
4. **Confirm** をクリックし、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。

[Report a bug](#)

2.3. ファイルシステムパス

2.3.1. ファイルシステムパス

JBoss EAP 6 では、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** の設定には、パスを宣言できるセクションが含まれます。設定の他のセクションは、各インスタンスの絶対パスを宣言せず論理名を使用することにより、これらのパスを参照できます。これにより、特定のホスト設定をユニバーサルな論理名に解決できるため、設定や管理がしやすくなります。

たとえば、ロギングサブシステム設定には、サーバーの **log** ディレクトリーを示す **jboss.server.log.dir** パスへの参照が含まれます。

例2.10 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2.2 標準的なパス

Value	説明
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.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。

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、以下の2つの方法の1つを用いて

jboss.server.base.dir、**jboss.server.log.dir**、または **jboss.server.config.dir** パスをオーバーライドできます。

1. サーバーの起動時に、コマンドラインで引数を渡すことができます。例は次のとおりです。

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

2. サーバー設定ファイルの **JAVA_OPTS** 変数を編集できます。**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はサポートされません。

カスタムパスの追加

カスタムパスを作成することも可能です。たとえば、以下のようにロギングに使用する相対パスを定義できます。

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

my.relative.path を使用するよう、ログハンドラーを変更します。

[Report a bug](#)

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。最も大きな違いの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	<i>EAP_HOME/domain/configuration/domain.xml</i>	管理対象ドメインの主要の設定ファイルです。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が可能です。
host.xml	<i>EAP_HOME/domain/configuration/host.xml</i>	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 および host-slave.xml (詳細については、下記参照) の両方の機能がすべて含まれています。このファイルはスタンドアロンサーバーの場合は存在しません。
host-master.xml	<i>EAP_HOME/domain/configuration/host-master.xml</i>	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では存在しません。
host-slave.xml	<i>EAP_HOME/domain/configuration/host-slave.xml</i>	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では存在しません。

サーバーモード	場所	目的
standalone.xml	<i>EAP_HOME/standalone/configuration/standalone.xml</i>	スタンドアロンサーバーのデフォルトの設定ファイルです。このファイルにはサブシステム、ネットワーク、デプロイメント、ソケットバインディング、その他の設定情報などを含む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ています。スタンドアロンサーバーの起動時にこの設定が自動的に使用されます。
standalone-full.xml	<i>EAP_HOME/standalone/configuration/standalone-full.xml</i>	スタンドアロンサーバーの設定例です。高可用性を要するサブシステムを除く、可能な全サブシステムのサポートが含まれます。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: <i>EAP_HOME/bin/standalone.sh -c standalone-full.xml</i>
standalone-ha.xml	<i>EAP_HOME/standalone/configuration/standalone-ha.xml</i>	このサンプル設定ファイルは、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 <i>mod_cluster</i> および JGroups サブシステムをスタンドアロンサーバーに対して追加します。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、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: <i>EAP_HOME/bin/standalone.sh -c standalone-ha.xml</i>
standalone-full-ha.xml	<i>EAP_HOME/standalone/configuration/standalone-full-ha.xml</i>	これはスタンドアロンサーバーの設定例です。高可用性に必要なサブシステムを含む、可能なすべてのサブシステムのサポートが含まれます。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: <i>EAP_HOME/bin/standalone.sh -c standalone-full-ha.xml</i>

これらはデフォルトの場所です。実行時に異なる設定ファイルを指定できます。

[Report a bug](#)

2.4.2. 記述子ベースのプロパティ置換

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、通常はデプロイメント、テスト、および製品の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おらず、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 Enterprise Application Platform 6 では、*記述子ベースのプロパティ置換* 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置換 は、記述子を基にプロパティを置き換えるため、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置換は、**standalone.xml** または **domain.xml** からグローバルに有効化されます。

```
<subsystem xmlns="urn:jboss:domain:ee:1.1">
  <spec-descriptor-property-replacement>
    true
  </spec-descriptor-property-replacement>
  <jboss-descriptor-property-replacement>
    true
  </jboss-descriptor-property-replacement>
</subsystem>
```

ejb-jar.xml および **persistence.xml** の Java EE 記述子は置き換えできますが、デフォルトでは無効になっています。

JBoss 固有の記述子の置換はデフォルトで有効になっています。記述子は以下で置換できます。

- **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。

```
@ActivationConfigProperty(propertyName = "connectionParameters", propertyValue =
"host=192.168.1.1;port=5445")
```

記述子ベースのプロパティ置換が有効になっている場合、コマンドラインで以下のように **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で指定する場合は以下のようになります。

```
<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>

```

`${jms.connection.parameters:'host=10.10.64.1;port=5445'}` では、デフォルト値を提供しながら、コマンドラインが指定するパラメーターによって接続パラメーターが上書きされます。

[Report a bug](#)

2.4.3. 記述子ベースのプロパティ置換の有効化/無効化

概要

記述子プロパティ置換の限定的な制御が、**jboss-as-ee_1_1.xsd** に導入されました。このタスクには、記述子ベースのプロパティ置換を設定するのに必要な手順が含まれます。

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置換フラグはブール値を持ちます。

- **true** に設定された場合は、プロパティ置換が有効になります。
- **false** に設定された場合は、プロパティ置換が無効になります。

手順2.12 jboss-descriptor-property-replacement

jboss-descriptor-property-replacement は、次の記述子でプロパティ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 では、次のコマンドを実行して **jboss-descriptor-property-replacement** の値を決定します。

```
/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2.13 spec-descriptor-property-replacement

spec-descriptor-property-replacement は、次の記述子でプロパティ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**

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

1. 管理 CLI では、次のコマンドを実行して **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置換が正常に設定されます。

[Report a bug](#)

2.4.4. 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**、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 や管理コンソールといった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しやすいするため、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にタイムスタンプを付けたものを作成します。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向けに保持されます。このアーカイブ機能には、サーバー設定のスナップショットの保存、ロード、および削除が含まれ、リコールおよびロールバックを可能にします。

- [「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」](#)

[Report a bug](#)

2.4.5. 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。同じコンセプトは、**domain.xml** と **host.xml** のそれぞれを使用した管理対象ドメインに適用されます。

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。

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、正常な起動後に最初加えられた変更前のサーバーモデルのインスタンスが再度呼び出されます。

```
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

ドメイン設定の履歴

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
```

[Report a bug](#)

2.4.6. 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ポイントインタイムコピーです。これらのコピーは、管理者が保存およびロードできます。

次の例では **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2.14 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 操作を実行して、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/2011063
0-172258657standalone.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。

[Report a bug](#)

2.4.7. 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ポイントインタイムコピーです。これらのコピーは、管理者が保存およびロードできます。スナップショットのロードのプロセスは、「[以前の設定でのサーバーの起動](#)」で使用された方法に似ており、スナップショットを作成、リスト、および削除するために使用される管理 CLI インターフェースではなくコマンドラインから実行します。

次の例では **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** ファイルにも適用されます。

手順2.15 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識別します。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が呼び出されます。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。

[Report a bug](#)

2.4.8. 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ポイントインタイムコピーです。これらのコピーは、管理者が保存およびロードできます。

次の例では **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** ファイルにも適用されます。

手順2.16 特定のスナップショットの削除

1. ロードするスナップショットを指定します。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が削除されます。

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

2. **: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。以下の例のようにスナップショットの名前を指定します。

```
[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2.17 スナップショットすべてを削除

- **:delete-snapshot(name="all")** コマンドを以下の例のとおり実行し、スナップショットをすべて削除します。

```
[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。

[Report a bug](#)

2.4.9. 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ポイントインタイムコピーです。これらのコピーは、管理者が保存およびロードできます。

次の例では **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** ファイルにも適用されます。

手順2.18 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リストします。

```
[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。

[Report a bug](#)

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、必要なときに実装を設定および管理できる複数の管理ツールを提供します。これらの管理ツールには、新しい管理コンソールや管理コマンドラインインターフェース (CLI) が含まれます。基礎となる管理 API を使用すると、上級ユーザーは必要に応じて独自のツールを開発できます。

[Report a bug](#)

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)

管理クライアント

JBoss EAP 6 は、サーバーを設定および管理する 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 設定ファイルのセット) を提供します。推奨される設定ファイルの編集方法には管理コンソールと管理 CLI が含まれますが、これらの 3 つの方法で設定に加えられた編集は、異なるビューにわたって常に同期され、最終的に XML ファイルに永続化されます。サーバーインスタンスの実行中に XML 設定ファイルで行われた編集は、サーバーモデルによって上書きされることに注意してください。

HTTP API

管理コンソールは、Google Web Toolkit (GWT) で構築された Web インターフェースの例です。管理コンソールは、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、JSON でエンコードされたプロトコルとデタイプな (detyped) 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 Web コンソールによって使用されますが、他のクライアントに対しても統合機能を提供します。

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共存します。HTTP API エンドポイントサービスは、管理操作を実行するためのコンテキストと 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の 2 つの異なるコンテキストを処理します。デフォルトでは、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>
```

Web コンソールは、HTTP 管理 API と同じポートを介して提供されます。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、および公開されたドメイン API を区別することが重要です。

表3.1 管理コンソールへの 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。

ネイティブ API

ネイティブ API ツールの例には管理 CLI があります。この管理ツール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に使用できるため、ユーザー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、デタイプ (detyped) の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、管理レイヤーと統合するためにネイティブ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。管理操作を定義および実行するために、オープンバイナリプロトコルと、非常に少数の Java タイプに基づいた 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。デフォルトでは、ポート 9999 で実行されます。

例3.2 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>
```

[Report a bug](#)

3.3. 管理コンソールと管理 CLI

JBoss EAP 6 では、すべてのサーバーインスタンスと設定が、XML ファイルの編集ではなく管理インターフェースによって管理されます。設定 XML ファイルは引き続き編集できますが、管理インターフェースによる管理は、サーバーインスタンスの永続的な管理に対して追加の検証機能および高度な機能を提供します。サーバーインスタンスの実行中に XML 設定ファイルに加えられた変更はサーバーモデルによって上書きされ、追加された XML コメントもすべて削除されます。サーバーインスタンスの実行中は管理インターフェースのみを使用して設定ファイルを変更する必要があります。

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。

[Report a bug](#)

3.4. 管理コンソール

3.4.1. 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、サーバーの開始および停止、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、サーバー設定の変更の永続化を行います。管理コンソールは管理タスクも実行でき、変更後に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。

[Report a bug](#)

3.4.2. 管理コンソールへのログイン

前提条件

- 「[管理インターフェースのユーザーの追加](#)」の説明にしたがって管理ユーザーを作成する必要があります。

JBoss EAP 6 が稼働している必要があります。

1. 管理コンソールのスタートページへのアクセス

Web ブラウザーを起動し、Web ブラウザーで管理コンソール (<http://localhost:9990/console/App.html>) に移動します。



注記

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

2. 以前作成したアカウントのユーザー名とパスワードを入力し、管理コンソールのログイン画面にログインします。

Authentication Required

A username and password are being requested by <http://localhost:9990>. The site says: "ManagementRealm"

User Name:

Password:

Cancel OK

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>)。

[Report a bug](#)

3.4.3. 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。英語の代わりに次のいずれかの言語を選択できます。

サポート言語

- 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 **Settings** ラベルがあります。これを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。次に、**Save** を選択します。確認ボックスに、アプリケーションをリロードする必要があると表示されます。**Confirm** をクリックします。新しいロケールを使用するために Web ブラウザを更新します。

[Report a bug](#)

3.4.4. EAP コンソールの分析

Google Analytics とは

Google Analytics は、Web サイトの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。訪問数、ページビュー、訪問別ページ数、平均滞在時間など、Web サイトの訪問者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 サイトの存在やそのユーザーの認知度が向上します。

EAP 管理コンソールでの Google Analytics

JBoss EAP 6.3 では、ユーザーは管理コンソールの Google Analytics を有効または無効にできます。Google Analytics の機能は、コンソールがどのように使用され、コンソールのどの部分が重要であるかを Red Hat EAP チームが理解することを目的としています。この情報は、ユーザーの必要性に見合ったコンソールのデザイン、機能、およびコンテンツを導入するために使用されます。

[Report a bug](#)

3.4.5. EAP コンソールの Google Analytics の有効化/無効化

EAP 管理コンソールで Google Analytics を有効にするには、以下の手順に従います。

- 管理コンソールにログインします。
- コンソールの右下にある **Settings** をクリックします。

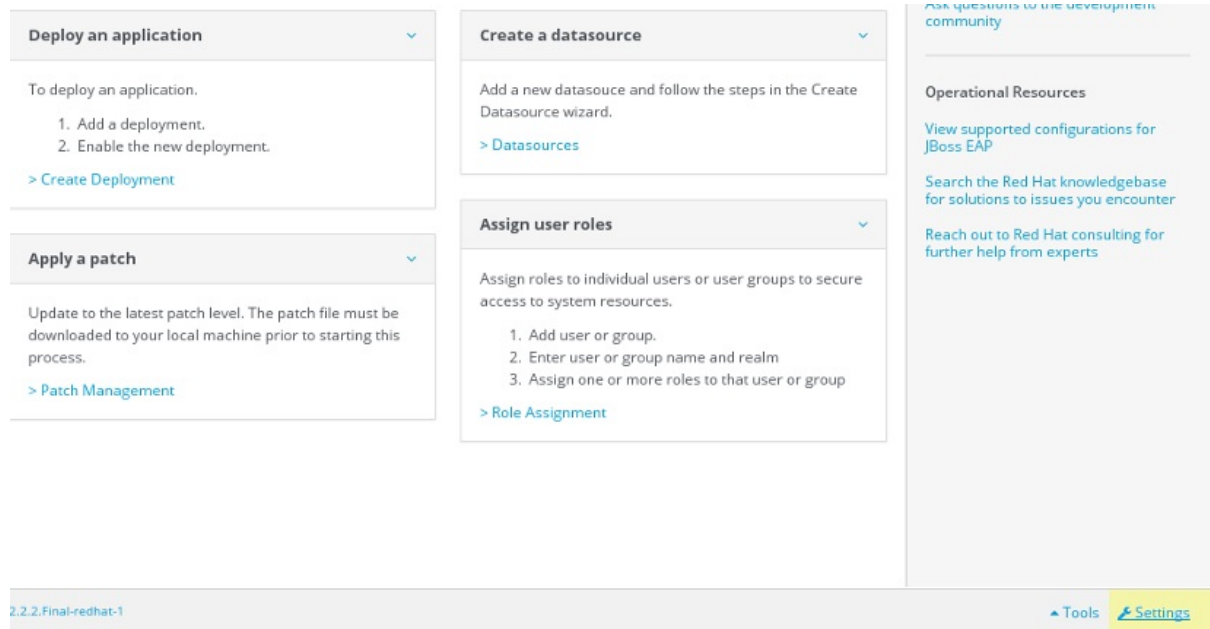
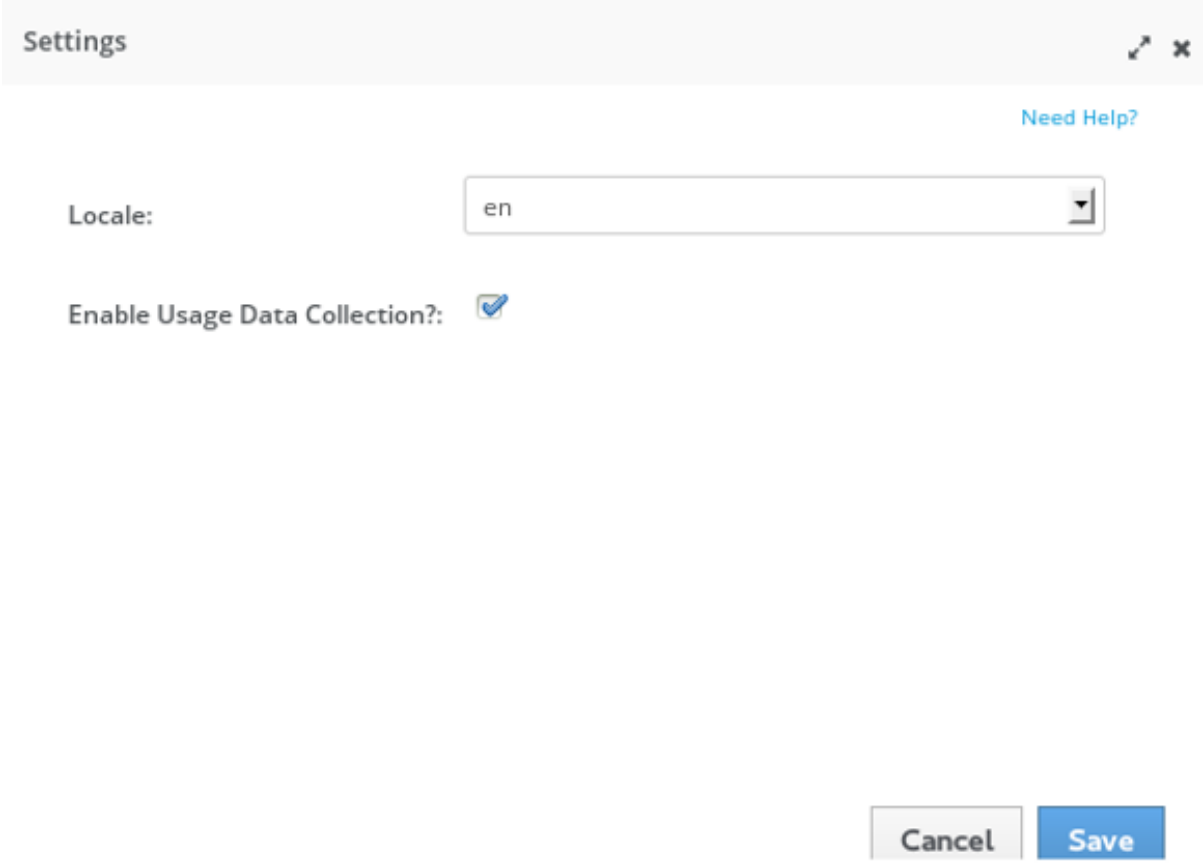


図3.2 管理コンソールのログイン画面

- **Settings** ウィンドウの **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。アプリケーションをリロードし、新しい設定を有効にします。



The screenshot shows a 'Settings' window with a title bar containing the word 'Settings' and window control icons. A blue link 'Need Help?' is in the top right. The 'Locale:' is set to 'en' in a dropdown menu. The 'Enable Usage Data Collection?:' checkbox is checked. At the bottom right are 'Cancel' and 'Save' buttons.

Settings

[Need Help?](#)

Locale: en

Enable Usage Data Collection?: ☒

Cancel Save

図3.3 Settings ウィンドウ (使用率データの収集を有効化)

管理コンソールの Google Analytics を有効にした後に無効にするには、**Settings** ウィンドウの **Enable Usage Data Collection** をクリックして選択を解除します。**Save** ボタンをクリックします。

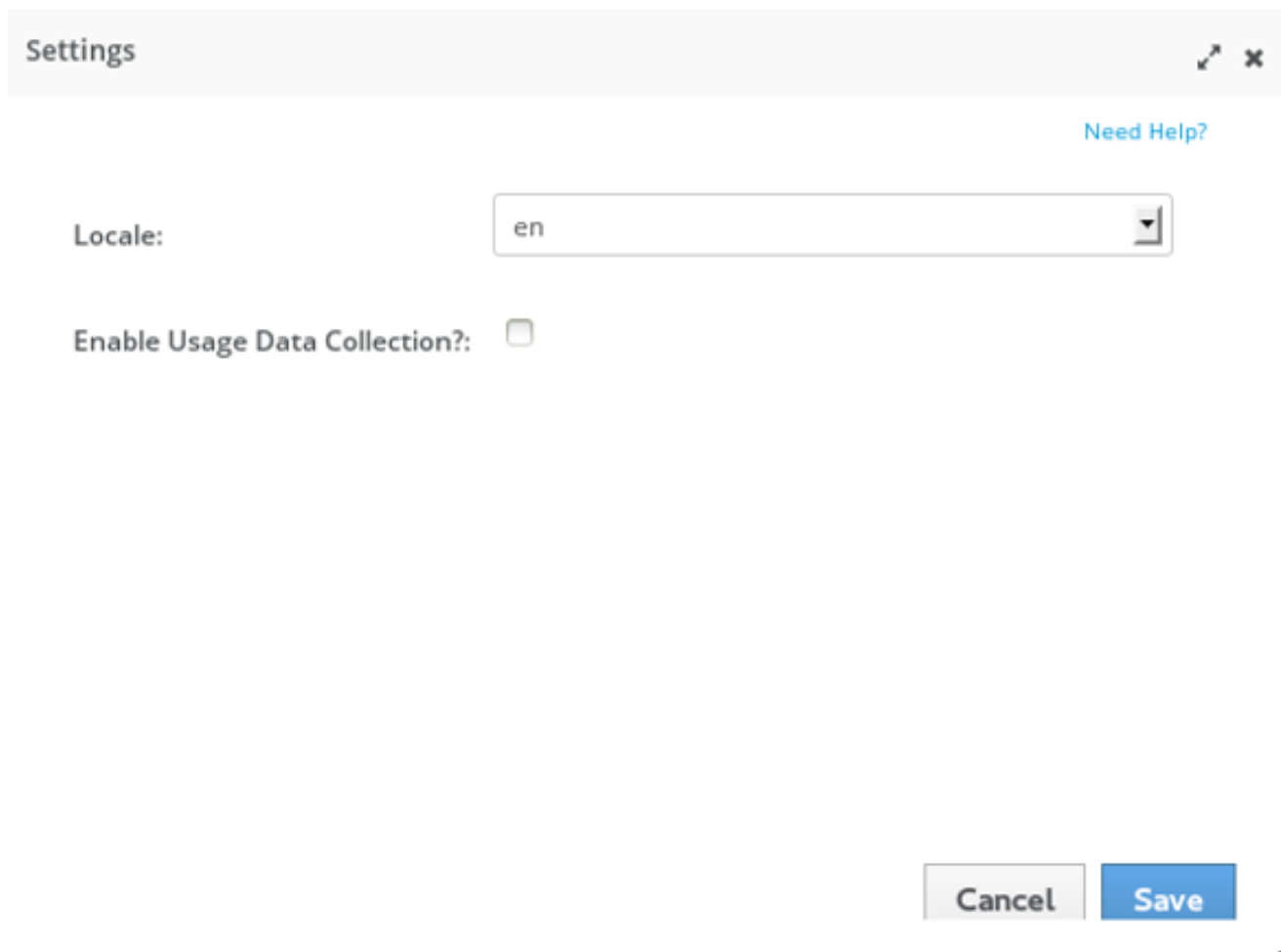
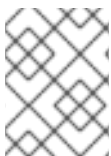


図3.4 Settings ウィンドウ (使用率データの収集を無効化)



注記

EAP 6.3 では、Google Analytics はデフォルトで無効になっています。Google Analytics の使用は任意です。

[Report a bug](#)

3.4.6. 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. コンソールの上部より **Domain** タブを選択します。利用可能なサーバーインスタンスの表が表示されます。
2. **Available Server Configurations** テーブルよりサーバーインスタンスを選択します。
3. 選択したサーバーの詳細の上にある **Edit** ボタンをクリックします。

4. 設定属性を変更します。
5. **Save** をクリックして終了します。

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。

[Report a bug](#)

3.4.7. 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. コンソールの上部より **Runtime** タブを選択します。
 2. スタンドアロンサーバーの場合は、**Server** メニューを展開し、**Manage Deployments** を選択します。管理対象ドメインの場合は、**Domain** メニューを展開し、**Manage Deployments** を選択します。**Manage Deployments** パネルが表示されます。
 3. **Content Repository** タブの **Add** を選択します。**Create Deployment** ダイアログボックスが表示されます。
 4. ダイアログボックスの **Browse** をクリックします。デプロイするファイルを選択し、アップロードします。**Next** をクリックして作業を続行します。
 5. **Create Deployments** ダイアログボックスに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、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。

[Report a bug](#)

3.4.8. 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** ページへの移動
コンソールの上部にある **Domain** タブを選択します。
2. 設定の新規作成
 - a. **Available Server Configuration** の表の上にある **Add** ボタンを選択します。
 - b. **Create Server Configuration** ダイアログで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。

結果

新しいサーバーが作成され、**Server Configurations** リストに表示されます。

[Report a bug](#)

3.4.9. 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 ログinglevel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** タブを選択した後、コンソールの左側にあるドロップダウンリストから該当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どちらの場合でも、コンソールの左側にあるリストの **Core** メニューを展開し、**Logging** エントリーをクリックします。
 - c. コンソール上部の **Log Categories** タグをクリックします。
2. ログ詳細の編集
Log Categories テーブル内のエントリーの詳細を編集します。
 - a. **Log Categories** の表のエントリーを選択し、下にある **Details** セクションの **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。設定したら、**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。

[Report a bug](#)

3.4.10. 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** ビューへの移動
コンソールの上部にある **Domain** タブを選択します。
2. 左側の列にあるメニューの **Server** ラベルを展開します。 **Server Groups** を選択します。
3. サーバーグループの追加
Add ボタンをクリックし新規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. サーバグループ名を入力します。
- b. サーバグループのプロファイルを選択します。
- c. サーバ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、新たに作成したグループを保存します。

結果

新規サーバ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。

[Report a bug](#)

3.5. 管理 CLI

3.5.1. 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、システムの設定、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でき、複数のタスクをグループとして実行できます。

[Report a bug](#)

3.5.2. 管理 CLI の起動

前提条件:

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

手順3.6 Linux または Microsoft Windows Server での CLI の起動

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

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

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

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

[Report a bug](#)

3.5.3. 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
```

[Report a bug](#)

3.5.4. 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3.7 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へ接続するには、**--connect** パラメーターを使用します。

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

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

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

[Report a bug](#)

3.5.5. 管理 CLI でのヘルプの取得

概要

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

前提条件

- 「[管理 CLI の起動](#)」

1. 一般的なヘルプの場合
管理 CLI で、**help** コマンドを入力します。

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

2. 状況依存ヘルプの取得
管理 CLI で、**help -commands** 拡張コマンドを入力します。

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

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

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

結果

CLI ヘルプ情報が表示されます。

[Report a bug](#)

3.5.6. バッチモードでの管理 CLI の使用

概要

バッチ処理により、複数の操作をシーケンスでグループ化し、ユニットとして一緒に実行できます。シーケンス内のいずれかの操作要求が失敗したら、操作のグループ全体がロールバックされます。

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3.8 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 コマンドを使用してバッチモードを開始します。

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

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。

操作要求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

操作要求のシーケンス全体が入力されたら、**run-batch** コマンドを使用してバッチを実行します。

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

バッチの作業に使用できるコマンドの完全リストについては、「[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

頻繁に実行するバッチコマンドは、外部のテキストファイルに保存できます。これらのバッチコマンドをロードするには、ファイルへのフルパスを引数として **batch** コマンドに渡すか、**run-batch** コマンドの引数として直接実行します。

テキストエディターを使用してバッチコマンドファイルを作成できます。1行に1つのコマンドのみがあるようにし、CLI がコマンドにアクセスできる必要があります。

次のコマンドは、バッチモードで **myscript.txt** ファイルをロードします。このファイルのすべてのコマンドは編集または削除できます。新しいコマンドを挿入できます。このバッチセッションで変更された内容は **myscript.txt** ファイルへ保持されません。

```
[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たバッチコマンドを即時実行します。

```
[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。

[Report a bug](#)

3.5.7. CLI のバッチモードコマンド

下表は JBoss EAP 6 の CLI で使用できる有効なバッチコマンドのリストになります。これらのコマンドはバッチの作業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 edited コマンドを提供し、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	移動したい行の番号を最初の引数として指定し、移動先を 2 番目の引数として指定してバッチの行の順序を変更します。例: move-batch-line 3 1
remove-batch-line linenumber	特定の行でバッチ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]	<p>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。バッチの外部にて急に CLI で実行するものがある場合にこのコマンドを使用します。保留したバッチへ戻るには、再度 CLI コマンドラインに batch を入力します。</p> <p>holdback-batch コマンドの使用中にバッチ名を指定すると、バッチがその名前で保存されます。名前が付けられたバッチに戻るには、batch batchname コマンドを使用します。バッチ名を指定せずに 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。名前のない保留バッチは1つのみ存在できます。</p> <p>保留されたバッチをすべて一覧表示するには、batch -l コマンドを使用します。</p>
discard-batch	現在アクティブなバッチを破棄します。

[Report a bug](#)

3.5.8. 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3.9 要求の作成、設定、および実行

1. 操作要求の構築

操作要求では、管理モデルとの低レベルの対話が可能です。この場合、サーバー設定を制御された方法で編集できます。操作要求は、以下の3つの部分から構成されます。

- スラッシュ (/) で始まるアドレス。
- コロン (:) で始まる操作名。
- 丸かっこ「()」で囲まれたパラメーターのオプションセット。

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。アドレスでは以下の構文を使用します。

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

- *node-type* は、リソースノードタイプです。これは、設定 XML の要素名に対して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** 属性に対してマッピングされます。
- スラッシュ (/) を使用してリソースツリーの各レベルを区切ります。

設定 XML ファイルを参照して必要なアドレスを決定しま

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** ファイルと**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。

例3.3 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

リソースノードのタイプによって操作は異なります。操作では以下の構文を使用します。

```
:operation-name
```

- *operation-name* は、要求する操作の名前です。

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任意のリソースアドレスで **read-operation-names** 操作を使用します。

例3.4 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、カンマ (,) で区切られます。

必要なパラメーターを決定するには、リソースノードで操作名をパラメーターとして渡し、**read-operation-description** コマンドを実行します。詳細については、[例3.5「操作パラメーターの決定」](#)を参照してください。

例3.5 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** 操作に必須の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3.6 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。

[Report a bug](#)

3.5.9. 管理 CLI 設定オプション

管理 CLI 設定ファイルである **jboss-cli.xml** は、CLI が起動するたびにロードされます。このファイルは、**\$EAP_HOME/bin** ディレクトリーまたは **jboss.cli.config** システムプロパティーで指定されたディレクトリーに置く必要があります。

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。デフォルト値は **localhost** です。

port

コントローラーへ接続するポート番号。

validate-operation-requests

実行のため要求がコントローラーへ送信される前に操作要求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。タイプはブール値で、デフォルト値は **true** です。

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。

history パラメーター

enabled

history が有効であるかを示します。タイプはブール値で、デフォルト値は **true** です。

file-name

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

file-dir

履歴が保存されるディレクトリーの名前。デフォルト値は **\$USER_HOME** です。

max-size

履歴ファイルの最大サイズ。デフォルト値は 500 です。

resolve-parameter-values

操作要求を送信する前にコマンド引数 (または操作パラメーター) として指定されたシステムプロパティーを解決するか、またはサーバー側で解決が発生するようにするか。タイプはブール値で、デフォルト値は **false** です。

connection-timeout

コントローラーと接続を確立するまでの許容時間。タイプは整数で、デフォルト値は 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、影響す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 を明示的に無効化することを推奨しています。

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 に設定された場合、認識できない証明書を受信したときに CLI がユーザーに伝え、トラストストアへの保存を許可します。タイプはブール値で、デフォルト値は **true** です。

vaultType

code と **module** の両方が指定されていない場合は、デフォルトの実装が使用されます。**code** が指定され、**module** が指定されていない場合は、Picketbox モジュールの指定されたクラスを探します。**module** と **code** が両方指定されている場合は、module によって指定されたモジュールで

code 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

タイプ: 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を指定します。**false** に指定されても、設定が許可したり、>を使用して出力ターゲットがコマンドラインの一部であると指定された場合は、ロガーを使用してメッセージがログに記録されます。デフォルト値は **False** です。

[Report a bug](#)

3.5.10. 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、一般オプションおよび状況依存オプションとのヘルプダイアログを含む管理 CLI ヘルプ機能にアクセスする方法について説明しています。ヘルプコマンドは、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への確立された接続を必要とします。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
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、バッチを再びアクティベートします。保留中のバッチがない場合は、引数なしで呼び出されたこのコマンドによって新しいバッチが開始されます。名前がない保留中のバッチがある場合は、このコマンドによってそのバッチが再びアクティベートされます。名前がある保留中のバッチがある場合は、保留中のバッチの名前を引数にしてこのコマンドを実行することにより、これらのバッチを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、その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 node-type で終わることがあります。その場合は、logging:read-resource などの node-name を指定すれば操作を実行できます。
clear	画面を消去します。

コマンド	説明
command	新しいタイプコマンドを追加し、既存の汎用タイプコマンドを削除およびリストできます。汎用タイプコマンドは、特定のノードタイプに割り当てられたコマンドであり、そのタイプのインスタンス向けの操作を実行可能にします。また、既存インスタンスのタイプにより公開された任意のプロパティを変更することもできます。
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、リポジトリで無効になっている既存のアプリケーションを有効にします。引数なしで実行された場合、既存のデプロイメントすべてがリストされます。
help	ヘルプメッセージを表示します。該当するコマンドの状況依存の結果を提供するには、 --commands 引数とともに使用します。
history	メモリー内の CLI コマンド履歴を表示し、履歴拡張が有効または無効であるかを表示します。必要に応じて履歴拡張をクリア、無効、および有効にするには、引数とともに使用します。
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。
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リストします。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 行ごとに 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で実行された場合にアプリケーションをアンデプロイします。リポジトリからアプリケーションを削除するには引数とともに使用します。アプリケーションを指定せずに実行すると、既存のデプロイメントすべてをリストします。

コマンド	説明
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。

[Report a bug](#)

3.5.11. 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、「[管理 CLI を使用した操作名の表示](#)」で解説している **read-operation-names** を使用すると公開できます。また、操作の説明は、「[管理 CLI を使用した操作説明の表示](#)」で解説している **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。
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され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て、ランタイムの同名の既存コンテンツを置き換え、その置き換えられたコンテンツを使用可能なコンテンツの一覧から削除します。詳細については、リンク先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。

操作名	説明
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および環境変数に対して解決する操作です。
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) 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、コンテンツがランタイムにデプロイされるべきであることを示すものではありません。

操作名	説明
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。

[Report a bug](#)

3.6. 管理 CLI 操作

3.6.1. 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。この操作を使用すると、デフォルトの値や非定義の値を無視し、ユーザーが設定した値のみを表示できます。要求プロパティには次のパラメーターが含まれます。

要求プロパティ

name

選択したリソース下で値を取得する属性の名前。

include-defaults

false を設定すると、デフォルト値を無視し、ユーザーが設定した属性のみを表示するよう操作結果を制限できるブール値パラメーター。

手順3.10 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 より **read-attribute** を使用してリソース属性の値を表示します。操作要求の詳細については、「[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。

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

read-attribute 操作の利点は、特定属性の現在のランタイム値を表示できることです。 **read-resource**

操作でも同様の結果を得ることができますが、**include-runtime** 要求プロパティを追加した場合のみ可能で、そのノードに対して使用できる全リソース一覧の一部のみを表示できます。次の例が示すように、**read-attribute** 操作は粒度の細かい属性クエリを対象としています。

例3.7 **read-attribute** 操作を実行したパブリックインターフェース IP の表示

表示したい属性の名前が分かる場合は、**read-attribute** を使用して現在のランタイムの厳密値を返します。

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

resolved-address 属性はランタイム値であるため、標準的な **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 や他のランタイム値を表示するには、**include-runtime** 要求プロパティ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。

[Report a bug](#)

3.6.2. 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 の操作は、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な操作です。この操作はユーザー名の ID と割り当てられたレルムを表示します。**whoami** の操作は、管理者が複数のレルムで複数のユーザーを管理する場合や、複数のターミナルセッションや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ユーザーを追跡する場合に便利です。

手順3.11 **whoami** を使用した管理 CLI でのアクティブユーザーの表示

- **whoami** の実行
管理 CLI より **whoami** を使用し、アクティブなユーザーのアカウントを表示します。

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

次の例はスタンドアロンサーバーで **whoami** を使用し、アクティブなユーザーは *username* で **ManagementRealm** レルムが割り当てられていることを表しています。

例3.8 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。

[Report a bug](#)

3.6.3. 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3.12 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。

[Report a bug](#)

3.6.4. 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3.13 管理 CLI でのコマンドの実行

- **read-operation-description 操作の実行**
管理 CLI で、**read-operation-description** を使用して、操作に関する情報を表示します。操作ではキーと値のペアの形式でパラメーターを追加して、表示する操作を示す必要があります。操作結果の詳細については、[「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。

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

例3.9 list-snapshots 操作の説明表示

次の例は、**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。

[Report a bug](#)

3.6.5. 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3.14 管理 CLI でのコマンドの実行

- **read-operation-names** の実行
管理 CLI より **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。操作要求の詳細については、「[「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。

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

例3.10 管理 CLI を使用した操作名の表示

次の例は、**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。

[Report a bug](#)

3.6.6. 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。この操作を使用すると、現在のノードまたは子ノードのリソースに関する基本または詳細情報や、操作結果の範囲を拡大または制限するさまざまな要求プロパティを表示できます。要求プロパティには、以下のパラメーターが含まれます。

要求プロパティ

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

再帰的なクエリ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にスレーブの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含めること。

include-runtime

応答に、永続的な設定ではない属性値などのランタイム属性を含めるかどうか。この要求プロパティは、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の要求プロパティ。false に設定された場合は、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。

手順3.15 管理 CLI でのコマンドの実行

1. read-resource 操作の実行

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表示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向けにインストールまたは設定されたシステムリソース、拡張機能、インターフェース、およびサブシステムを表示します。これらの情報は、さらに直接クエリーできます。

例3.11 ルートレベルでの read-resource 操作の使用

```
[standalone@localhost:9999 /]:read-resource
{
  "outcome" => "success",
  "result" => {
    "deployment" => undefined,
    "deployment-overlay" => undefined,
    "management-major-version" => 1,
    "management-micro-version" => 0,
    "management-minor-version" => 4,
```

```
"name" => "longgrass",
"namespaces" => [],
"product-name" => "EAP",
"product-version" => "6.3.0.GA",
"release-codename" => "Janus",
"release-version" => "7.2.0.Final-redhat-3",
"schema-locations" => [],
"system-property" => undefined,
"core-service" => {
  "management" => undefined,
  "service-container" => undefined,
  "server-environment" => undefined,
  "platform-mbean" => 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" => {
  "logging" => undefined,
  "datasources" => undefined,
  "deployment-scanner" => undefined,
  "ee" => undefined,
  "ejb3" => undefined,
  "infinispan" => undefined,
  "jaxrs" => undefined,
  "jca" => undefined,
  "jdr" => undefined,
  "jmx" => undefined,
  "jpa" => undefined,
  "jsf" => undefined,
  "mail" => undefined,
  "naming" => undefined,
  "pojo" => undefined,
  "remoting" => undefined,
  "resource-adapters" => undefined,
  "sar" => undefined,
  "security" => undefined,
  "threads" => undefined,
  "transactions" => undefined,
  "web" => undefined,
  "webservices" => undefined,
  "weld" => undefined
}
}
}

```

2. 子ノードに対する **read-resource** 操作の実行

read-resource 操作は、ルートから子ノードを問い合わせるために実行できます。操作の構造は最初に公開するノードを定義し、ノード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
```

下例では、特定の Web サブシステムノードに対して **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3.12 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、同じ結果を得ることができます。

例3.13 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
  }
}

```

3. 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ことにより、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3.14 **include-runtime** パラメーターを使用して追加で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 "${boss.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
  }
}
```

[Report a bug](#)

3.6.7. 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」](#)

手順3.16 管理 CLI でのコマンドの実行

1. **read-resource-description** の実行

管理 CLI より **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。操作要求の詳細については、[「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。

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

2. オプションパラメーターの使用

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

- リソースの操作の説明を表示するには、**operations** パラメーターを使用します。

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

- b. リソースの継承された操作の説明を表示または非表示にするには、**inherited** パラメーターを使用します。デフォルトの状態は true です。

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

- c. 子リソースの操作の再帰的な説明を表示するには、**recursive** パラメーターを使用します。

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

- d. リソースの説明を表示するには、**locale** パラメーターを使用します。null の場合は、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)
```

結果

利用可能なリソースの説明が表示されます。

[Report a bug](#)

3.6.8. 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 から **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、ランタイムを再起動します。**reload** の完了後、管理 CLI は自動的に再接続します。

操作要求の詳細は、「[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3.15 アプリケーションのリロード

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

[Report a bug](#)

3.6.9. 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

- **shutdown** 操作の実行

- 管理 CLI で、**shutdown** 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。操作要求の詳細については、[「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。

- 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 /]/host=master:shutdown
```

- 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
Connected to IP_ADDRESS:PORT_NUMBER
[192.168.1.10:9999 /] :shutdown
```

IP_ADDRESS をインスタンスの IP アドレスに置き換えます。

結果

アプリケーションサーバーがシャットダウンされます。ランタイムが利用できない場合に、管理 CLI が切断されます。

[Report a bug](#)

3.6.10. 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、選択されたリソース属性を記述または変更するために使用するグローバル操作です。この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。要求プロパティには以下のパラメーターが含まれます。

要求プロパティ

name

選択されたリソース下で値を設定する属性の名前。

value

選択され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 になることがあります。

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、完全リソースパスが指定されたリソースの子ノードまたは管理 CLI のルートノードで実行できます。

例3.16 **write-attribute** 操作によるデプロイメントスキャナーの無効化

以下の例では、**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。操作は、ルートノードから実行されます (タブ補完を使用して正しいリソースパスを入力します)。

```
[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
}
```

また、**read-resource** 操作を使用し、すべてのノードの利用可能なリソース属性をリストして結果を確認することもできます。以下の例では、**scan-enabled** 属性が **false** に設定されていることが分かります。

```
[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。

[Report a bug](#)

3.6.11. 管理 CLI を使用したシステムプロパティの設定

手順3.19 管理 CLI を使用したシステムプロパティの設定

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、ドメインで稼働しているすべてのサーバーへシステムプロパティを追加できます。

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

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

例3.17 システムプロパティ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へシステムプロパティを追加します。

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

例3.18 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へシステムプロパティを追加します。

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

例3.19 システムプロパティ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を追加します。

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

例3.20 システムプロパティ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を読み取ります。

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

例3.21 スタンドアロンサーバーからシステムプロパティの読み取り

```
[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3.22 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3.23 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3.24 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を削除します。

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

例3.25 スタンドアロンサーバーからシステムプロパティを削除

```
[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を削除します。

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

例3.26 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を削除します。

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

例3.27 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を削除します。

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

例3.28 管理対象ドメインのサーバーからシステムプロパティを削除

```
[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"}}}}}}
}
```

[Report a bug](#)

3.7. 管理 CLI コマンド履歴

3.7.1. 管理 CLI コマンド履歴の利用

管理 CLI にはコマンドの履歴機能があり、これはアプリケーションサーバーのインストール時にデフォルトで有効になっています。この履歴はアクティブな CLI セッションの揮発性メモリーでレコードとして保持されるか、ユーザーのホームディレクトリーに **.jboss-cli-history** として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、CLI コマンドを最大 500 件記録できるように設定されます。

history コマンド自体は、現在のセッションの履歴を返しますが、引数を追加すると、セッションメモリーの履歴を無効化、有効化、または消去できます。管理 CLI では、キーボードの矢印キーを使用してコマンドや操作の履歴を前後に移動できます。

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](#)」

[Report a bug](#)

3.7.2. 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** コマンドの実行

管理 CLI で、**history** 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。

[Report a bug](#)

3.7.3. 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** コマンドの実行

管理 CLI で、**history --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た **.jboss-cli-history** ファイルに保持されます。

[Report a bug](#)

3.7.4. 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** コマンドの実行

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、メモリー内およびユーザーのホームディレクトリーに保存された **.jboss-cli-history** ファイルには記録されません。

[Report a bug](#)

3.7.5. 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** コマンドの実行
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、メモリー内および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。

[Report a bug](#)

3.8. 管理インターフェース監査ロギング

3.8.1. 管理インターフェース監査ロギング

監査ロギングが有効な場合、管理 API より実行されたすべての操作は監査ログに記録されます。管理コンソール、管理 CLI インターフェース、またはカスタムのインターフェースから実行されたすべての操作が監査ロギングの対象となります。デフォルトでは、ロギングは無効になっています。

ログエントリーは、保存される前にフォーマッターおよびハンドラーへ渡されます。フォーマッターはログエントリーの形式を指定し、ハンドラーはレコードを指定の宛先に出力します。現在、エントリーを JSON 形式で出力する1つのフォーマッターのみを使用できます。ハンドラーがログレコードをファイルに出力するよう設定したり、Syslog サーバーに転送したりすることができ、その両方を行うこともできます。



注記

監査ロギングは、管理 CLI からのみ設定できます。

[Report a bug](#)

3.8.2. 管理 CLI からの管理インターフェース監査ロギングの有効化

管理インターフェース監査ロギングはデフォルトで無効になっていますが、**file** という名前のファイルハンドラーを使用して **EAP_HOME/standalone/data/audit-log.log** ファイルにログレコードを出力するよう事前設定されています。

管理 CLI から監査ロギングを有効にするには、次のコマンドを使用します。

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

管理インターフェースの監査ロギング設定属性とそれらの値をすべて表示するには、以下のコマンドを入力します。

■

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

管理インターフェースの監査ロギングを有効または無効にする以外に、以下の **logger** 設定属性を使用することもできます。

log-boot

true に設定された場合、サーバーを起動するときの管理操作が監査ログに含まれます。**false** の場合は含まれません。デフォルト値は **false** です。

log-read-only

true に設定された場合、すべての操作が監査ログに記録されます。**false** の場合は、モデルを変更する操作のみが記録されます。デフォルト値は **false** です。

[Report a bug](#)

3.8.3. 管理インターフェースの監査ロギングフォーマッター

フォーマッターはログエントリーの形式を指定します。ログエントリーを JSON 形式で出力する1つのフォーマッターのみを利用できます。各ログエントリーは任意のタイムスタンプで始まり、「JSON フォーマッターフィールド」の表に記載されているフィールドが続きます。

「JSON フォーマッター属性」の表には、ログエントリーの形式変更に使用できるすべての属性が記載されています。

表3.5 JSON フォーマッター属性

属性	説明
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。
date-separator	日付と日付以外のフォーマットされたログメッセージを区別する文字が含まれる文字列。 include-date=false の場合は無視されます。
date-format	java.text.SimpleDateFormat が理解するように、タイムスタンプに使用するデータ形式。 include-date=false の場合は無視されます。
compact	true の場合、JSON を1行で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。
escape-control-characters	true の場合、すべての制御文字 (10 進値が 32 未満の 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。

属性	説明
escape-new-line	true の場合、新しい行を 8 進の ASCII コードでエスケープします (例: #012)。

表3.6 JSON フォーマッターフィールド

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 サブシステムからであることを示す jmx を値として指定できます (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。変更される場合は false になります。
booting	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。稼働しているサーバーと同じマシンの管理 CLI から操作が実行された場合は、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。 <ul style="list-style-type: none"> ● 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 向けの監査ロギングの設定方法については、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の値は 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 ヘ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 の解析で生じたすべての操作がリストされます。起動後は、通常 1 つのエントリーがリストされます。

[Report a bug](#)

3.8.4. 管理インターフェースの監査ロギングファイルハンドラー

ファイルハンドラーは、監査ログの記録がファイルへ出力されるようにするパラメーターを指定します。ファイルハンドラーは、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監査ログ属性」の表には、ログエントリーの出力を変更するために使用できるすべての属性が記載されています。

表3.7 ファイルハンドラー監査ログ属性

属性	説明	読み取り専用
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。	False
path	監査ログファイルのパス。	False
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 が指定された場合、パス属性の値は、この属性によって指定されたパスへの相対値としてみなされます。	False
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。	True
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。	False
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た場合の値は true になります。	True

[Report a bug](#)

3.8.5. 管理インターフェイスの監査ロギング syslog ハンドラー

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よりも、セキュリティオプションが多くなります。複数の syslog ハンドラーを定義できます。

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。

表3.8 syslog ハンドラーの属性

属性	説明	デフォルト値
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。	undefined
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た場合の値は true になります。	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。	USER_LEVEL
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。	0 (ゼロ)
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。	null
host	syslog サーバーのホスト名。	localhost
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。	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。	undefined
port	syslog サーバーがリッスンしているポート。	514
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。	null
syslog-format	syslog 形式: <i>RFC-5424</i> または <i>RFC-3164</i> 。	RFC5424

属性	説明	デフォルト値
truncate	ヘッダーを含むメッセージの長さ(バイト長)が最大長を越える場合に、そのメッセージが省略されるかどうか。 false に設定すると、メッセージが分割され、同じヘッダー値で送信されます。	false

[Report a bug](#)

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



注記

管理対象ドメインへ変更を適用する場合は、**/host=HOST_NAME** 接頭辞を **/core-service** コマンドへ追加します。

この例では、syslog サーバーは 514 番ポートのローカルホストで実行しています。これらのパラメーターをご使用の環境に該当する値に置き換えます。

手順3.24 syslog サーバーへのログの有効化

1. **msyslog** という名前の syslog ハンドラーを作成します。

```
[standalone@localhost:9999 /]batch
[standalone@localhost:9999 /]/core-service=management/access=audit/syslog-
handler=mysyslog:add(formatter=json-formatter)
[standalone@localhost:9999 /]/core-service=management/access=audit/syslog-
handler=mysyslog/protocol=udp:add(host=localhost,port=514)
[standalone@localhost:9999 /]run-batch
```

2. **syslog** ハンドラーへの参照を追加します。

```
[standalone@localhost:9999 /]/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。

[Report a bug](#)

第4章 ユーザー管理

4.1. ユーザー作成

4.1.1. 管理インターフェースのユーザーの追加

概要

最初に利用できるユーザーアカウントがないため、Boss EAP 6 の管理インスタンスは、デフォルトでセキュアになります (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がインストールされていない場合)。これは、単純な設定エラーが原因でセキュリティが侵されることを防ぐための対策です。

JBoss EAP 6 との HTTP 通信は、トラフィックの送信元がローカルホストであっても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、ユーザーが作成されるまでデプロイされないため、エラーが発生します。

以下の手順に従って、最初の管理ユーザーを作成します。このユーザーは、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/ ディレクトリーへ移動します。ご使用のオペレーティングシステムに対応するスクリプトを呼び出します。

Red Hat Enterprise Linux

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

Microsoft Windows Server

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

2. 管理ユーザーの追加を選択します。

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

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

3. ユーザー名とパスワードを入力します。

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

4. グループ情報を入力します。

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

5. 情報を確認します。

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

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

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

7. 追加ユーザーを入力します。

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

8. 非対話的にユーザーを作成します。

コマンドラインで各パラメーターを渡すと非対話的にユーザーを作成できます。ログや履歴ファイルにパスワードが表示されるため、この方法は共有システムでは推奨されません。管理レلم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لم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実行を無効にできます。これは、**username** および **password** パラメーターが指定されている場合のみ適用されます。エラーメッセージは表示されます。

結果

追加したすべてのユーザーは、指定したセキュリティーレلم内でアクティベートされます。**ManagementRealm** レلم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。

関連トピック:

- [「デフォルトのユーザーセキュリティー設定」](#)

[Report a bug](#)

4.1.2. 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できますが、コマンドラインで引数を渡すこともできます。ここでは、コマンドライン引数を `add-user` スクリプトへ渡すときに使用可能なオプションについて説明します。

add-user.sh または **add-user.bat** コマンドで使え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については、[「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、[「add-user スクリプトのコマンドラインの例」](#) を参照してください。

[Report a bug](#)

4.1.3. add-user コマンド引数

下表は、**add-user.sh** または **add-user.bat** コマンドで利用できる引数を表しています。

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、プロパティファイルが含まれるドメイン設定ディレクトリを指定します。省略した場合、デフォルトのディレクトリは EAP_HOME/domain/configuration/ になります。
-sc	SERVER_CONFIGURATION_DIRECTORY	この引数は、プロパティ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を指定します。省略した場合、デフォルトのディレクトリは 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。
-gp --group-properties	GROUP_PROPERTIES_FILE	この引数は、代替のグループプロパティ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。

コマンドライン 引数	引数の値	説明
-p --password	PASSWORD	<p>ユーザーのパスワード。パスワードは次の要件を満たしている必要があります。</p> <ul style="list-style-type: none"> ● 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てはなりません。 ● 1文字以上の数字が含まれてはなりません。 ● 英数字以外の記号が1つ以上含まれてはなりません。
-u --user	USER_NAME	ユーザーの名前。英数字と ./=@\ の記号のみが有効です。
-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。

[Report a bug](#)

4.1.4. ユーザー管理情報の代替プロパティファイルの指定

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。ここでは、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。

手順4.2 代替プロパティファイルの指定

- サーバー設定の代替の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の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。
 - 代替のユーザー設定プロパティファイルを指定するには、**-up** 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ファイルを指定するには、**-gp** 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、既存のプロパティファイルでの操作を目的とするコマンドです。コマンドライン引数に指定された代替のプロパティファイルが存在しない場合は、次のエラーが表示されます。

```
JBAS015234: No appusers.properties files found
```

コマンド引数に関する詳細は、「[add-user コマンド引数](#)」を参照してください。

add-user コマンドの例は、「[add-user スクリプトのコマンドラインの例](#)」を参照してください。

[Report a bug](#)

4.1.5. add-user スクリプトのコマンドラインの例

以下の例は、**add-user.sh** または **add-user.bat** コマンドで引数を渡す方法を表しています。記載があるもの以外は、これらのコマンドはスタンドアロンサーバーの設定を前提としています。

例4.1 デフォルトのプロパティ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

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

上記のコマンドを実行した結果は次のとおりです。

- ユーザー **appuser1** が、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

- **guest** グループのユーザー **appuser1** が、グループ情報が保存されるデフォルトのプロパティ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

例4.2 デフォルトのプロパティファイルを使用した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'
```

上記のコマンドを実行した結果は次のとおりです。

- ユーザー **appuser1** が、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`

- **guest**、**app1group**、および **app2group** グループに属するユーザー **appuser1** が、グループ情報が保存されるデフォルトのプロパティ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`

例4.3 デフォルトのプロパティファイルを使用したデフォルトレルムの管理特権を持つユーザーの作成

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

上記のコマンドを実行した結果は次のとおりです。

- ユーザー **adminuser1** が、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`

- **admin** グループのユーザー **adminuser1** が、グループ情報が保存されるデフォルトのプロパティ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`

例4.4 情報を保存する代替プロパティ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

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

上記のコマンドを実行した結果は次のとおりです。

- ユーザー **appuser1** が、以下のプロパティ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** グループのユーザー **appuser1** が、以下のプロパティ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す。

`/home/someusername/userconfigs/appgroups.properties`

[Report a bug](#)

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

アプリケーションサーバーは、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使用するのではなく論理名を使用して個々のインターフェース宣言を参照できるようになります。また、論理名を使用することにより、管理対象ドメインのサーバーインスタンスに、複数のマシンのさまざまなインターフェース詳細が含まれることがある名前付きインターフェースへのグループ参照で整合性を取ることができます。この方法により、各サーバーインスタンスは、インターフェースグループ全体を簡単に管理できる論理名グループに対応できます。

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。アプリケーションサーバーは、管理およびパブリックインターフェース名のデフォルト設定で出荷されます。この設定では、パブリックインターフェースグループは、Web やメッセージングなどのアプリケーション関連ネットワーク通信で使用することを目的としています。管理インターフェースグループは、HTTP 管理エンドポイントを含む管理レイヤーに必要なすべてのコンポーネントとサービスに使用することを目的としています。インターフェース名自体は、推奨例としてのみ提供され、どのグループのどの名前でも必要に応じて置換または作成できます。

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つ以上の特徴を指定したりできます。次の例は、通常は **standalone.xml** 設定ファイルまたは **host.xml** 設定ファイルで定義される、インターフェース宣言の複数の設定を示しています。これにより、リモートホストグループが独自の固有インターフェース属性を維持できるようになる一方で、ドメインコントローラーの **domain.xml** 設定ファイルのインターフェースグループを引き続き参照できます。

最初の例は、**management** 相対名グループと **public** 相対名グループに対して指定された固有の **inet-address** 値を示します。

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>
```

次の例では、グローバルインターフェースグループが **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>
```


次の例では、相対グループ下のネットワークインターフェースカードを名前 **external** で宣言します。

例5.3 NIC 値で作成された外部グループ

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

次の例では、特定の要件に対して宣言がデフォルトグループとして作成されます。このインスタンスでは、追加要素の特徴によって、有効な一致となるインターフェースの条件が設定されます。これにより、特定のインターフェース宣言グループを作成でき、事前設定された方法でこれらを参照できるため、複数のサーバーインスタンス全体で設定および管理時間を短縮できます。

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>
```

インターフェース宣言はソース設定ファイルで記述および編集できますが、管理 CLI と管理コンソールでは、安全で制御された永続的な環境で設定を変更できます。

[Report a bug](#)

5.1.2. インターフェースの設定

standalone.xml 設定ファイルと **host.xml** 設定ファイルのデフォルトインターフェース設定は、それぞれの相対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。管理コンソールまたは管理 CLI を使用して、下表にある追加の属性と値を設定します。また、必要に応じて、相対インターフェースバインディングを特定の値に置き換えることもできますが、**-b** スイッチは相対値のみを上書きできるため、サーバーの実行時にインターフェース値を渡すことはできません。

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



```

<interface name="unsecure">
  <inet-address value="{boss.bind.address.unsecure:127.0.0.1}"/>
</interface>
</interfaces>

```

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	選択基準を制約するために使用されるアドレス除外タイプの空の要素。
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 に設定された場合は、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取ることができます。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取ることができます。
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。
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-addressType 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	選択基準を制約するために使用されるアドレス除外タイプの空の要素。

インターフェース要素	説明
------------	----

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 (たとえば、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

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 でのインターフェース属性の設定

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ます。次の例では、**inet-address** が 12.0.0.2 と宣言された、新しいインターフェース *interfacename* が作成されます。

```
/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

属性の値が変更されたことを確認するため、**include-runtime=true** パラメーターを使用して **read-resource** 操作を実行し、サーバーモデルで有効な現在の値をすべて表示します。例は次のとおりです。

```
[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. インターフェース画面への移動

i. Configuration タブへ移動します。

画面の上部から **Configuration** タブを選択します。

ii. ドメインモードのみ

画面の左上にある **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の Interfaces の選択

General Configuration メニューを展開します。ナビゲーションメニューから **Interfaces** メニューを選択します。

d. 新しいインターフェースの追加

i. 追加 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. Available Interfaces リストから編集する必要があるインターフェースを選択し、Edit ボタンをクリックします。

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。

[Report a bug](#)

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** 設定ファイルと **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、論理名でこれらのソケットを参照できます。ソケット設定の完全な詳細を含める必要はありません。これにより、マシンによって異なる場合がある相対ソケットを参照できます。

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下に収集されます。ソケットバインディンググループは、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。名前付きグループは、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、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。管理対象ドメインの各サーバーグループに対してソケットバインディングを作成するか、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できます。

名前付き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、単純化された参照を使用できます。また、一般的に、1つのシステムで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と管理にも使用できます。次の例は、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ソケットバインディンググループは、**standard-sockets** 下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 **public** 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">
  <socket-binding name="http" port="8080"/>
  <socket-binding name="https" port="8443"/>
  <socket-binding name="jacob" port="3528"/>
  <socket-binding name="jacob-ssl" port="3529"/>
  <socket-binding name="jmx-connector-registry" port="1090" interface="management"/>
  <socket-binding name="jmx-connector-server" port="1091" interface="management"/>
  <socket-binding name="jndi" port="1099"/>
  <socket-binding name="messaging" port="5445"/>
  <socket-binding name="messaging-throughput" port="5455"/>
  <socket-binding name="osgi-http" port="8090" interface="management"/>
  <socket-binding name="remoting" port="4447"/>
  <socket-binding name="txn-recovery-environment" port="4712"/>
  <socket-binding name="txn-status-manager" port="4713"/>
</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ソケットバインディンググループには **standard-sockets**、**ha-sockets**、**full-sockets**、および **full-ha-sockets** の4つのグループが含まれます。これらのグループは **public** と呼ばれる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="osgi-http" interface="management" port="8090"/>
    <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-group>
```

```

<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="osgi-http" interface="management" port="8090"/>
<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="jacob" interface="unsecure" port="3528"/>
  <socket-binding name="jacob-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="osgi-http" interface="management" port="8090"/>
  <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="jacob" interface="unsecure" port="3528"/>
  <socket-binding name="jacob-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="osgi-http" interface="management" port="8090"/>
<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** ソースファイルと **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には、管理コンソールまたは管理 CLI を使用することが推奨されます。管理コンソールを使用すると、**General Configuration** セクション下に専用の Socket Binding Group 画面があるグラフィカルユーザーインターフェースが使用できるなどの利点があります。管理 CLI は、アプリケーションサーバー設定の上位および下位レベル全体でバッチ処理やスクリプトの使用を可能にする、コマンドラインを基にした 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、変更を永続化したり、サーバー設定に保存したりできます。

[Report a bug](#)

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の1つである **standard-sockets** が含まれ、その他のグループを作成できません。その他のグループを作成するには、別のスタンドアロンサーバー設定ファイルを作成します。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することが可能で、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。下表は各ソケットバインディングに使用できる属性を表しています。

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基本ポート。すべてのポートの値をインクリメントまたはデクリメントすることにより、サーバーがこの基本値を上書きするよう設定できることに注意してください。	必須

属性	説明	ロール
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。定義されない場合は、囲んでいるソケットバインディンググループから default-interface 属性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 の値が常に使用されなければならない、インクリメントまたはデクリメントを適用して上書きされてはならないことを宣言します。	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

add 操作を使用して、必要な場合に新しいアドレス設定を作成します。このコマンドは、管理 CLI セッションのルートから実行できます。この場合、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** 属性が 1234 として宣言されます。以下の例は、**standard-sockets** ソケットバインディンググループ上で編集を行うスタンドアロンサーバーおよび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。タブ補完を使用して、入力する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。以下の例では、**port** 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

include-runtime=true パラメーターを用いて **read-resource** 操作を実行し、値の変更を確認します。サーバーモデルでアクティブな現在の値をすべて表示します。

```
[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 タブへ移動**

画面の上部にある **Configuration** タブを選択します。

c. **ナビゲーションメニューより Socket Binding を選択**

General Configuration メニューを展開し、**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** リストで希望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. **Add** 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. リスト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. 必要な値を入力します (**Name**、**Interface**、**Port** など)。

iii. **Save** をクリックして終了します。

[Report a bug](#)

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。



注記

数値ポートオフセットは、同じ物理サーバーで複数のサーバーを実行する場合にポートの競合を緩和するために設定できます。サーバーが数値の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に対する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、サーバーは **100** のポートオフセットを使用する場合、その 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**

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	standar d- socket
ajp	8009		Apache JServ プロトコル。HTTP クラスターリングおよび負荷分散に使用します。	○	○	○	○
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。	○	○	○	○
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。	○	○	○	○
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。	○	○	×	×
jacorb-ssl	3529		SSL 暗号化 CORBA サービス。	○	○	×	×
jgroups - diagnostics		7500	マルチキャスト。HA クラスターでのピアの検出に使用されます。管理インターフェースを使用しても設定できません。	○	×	○	×
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。	○	×	○	×

名前	ポート	マルチ キャスト ポート	説明	full-ha- sockets	full- sockets	ha- socket	standar d- socket
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。	○	×	○	×
jgroups -tcp-fd	57600		TCP を介した HA 障害検出に使用されます。	○	×	○	×
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。	○	×	○	×
jgroups -udp-fd	54200		UDP を介した HA 障害検出に使用されます。	○	×	○	×
messaging	5445		JMS サービス。	○	○	×	×
messaging- group			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。	○	○	×	×
messaging- throughput	5455		JMS Remoting により使用されます。	○	○	×	×
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。	○	×	○	×
osgi- http	8090		OSGi サブシステムを使用する内部コンポーネントにより使用されます。管理インターフェースを使用して設定できません。	○	○	○	○

名前	ポート	マルチ キャスト ポート	説明	full-ha- sockets	full- sockets	ha- socket	standar d- socket
remoting	4447		リモート EJB の呼び出しに使用されます。	○	○	○	○
txn-recovery-environment	4712		JTA トランザクションリカバリーマネージャ。	○	○	○	○
txn-status-manager	4713		JTA / JTS トランザクションマネージャ。	○	○	○	○

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 によって使用されるポート

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。

[Report a bug](#)

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、該当するサーバーのソケットバインディンググループにより提供されるポート値に追加される数値オフセットです。これにより、単一のサーバーが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します (グループ内の他のサーバーと競合しないようオフセットを使用)。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、その HTTP ポートは 8180 になります。

[Report a bug](#)

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属性に書き込みます。次の例は、`server-two` の **socket-binding-port-offset** 値を 250 に更新します。このサー

バーはデフォルト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となります。

```
[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

include-runtime=true パラメーターを用いて **read-resource** 操作を実行し、値の変更を確認します。サーバーモデルでアクティブな現在の値をすべて表示します。

```
[domain@localhost:9999 /] /host=master/server-config=server-two/:read-resource(include-runtime=true)
```

o. 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. Domain タブの選択

画面の上部にある **Domain** 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** リストのサーバーを選択し、属性リストの上にある **Edit** をクリックします。

ii. **Port Offset** フィールドで必要な値を入力します。

iii. **Save** をクリックして終了します。

[Report a bug](#)

5.2.6. リモートイングのメッセージサイズの設定

リモートイングサブシステムは、リモートイングプロトコルに対して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。最大受信メッセージサイズ (**MAX_INBOUND_MESSAGE_SIZE**) および最大送信メッセージサイズ (**MAX_OUTBOUND_MESSAGE_SIZE**) を設定すると、メッセージが適切な制限内のサイズで送受信されます。

リモートイ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。

送信側が最大許容限界 (**MAX_OUTBOUND_MESSAGE_SIZE**) を超えるメッセージを送信した場合、サーバーによって例外が発生し、データの送信がキャンセルされます。しかし、接続は開いたままになり、送信側は必要であれば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限界 (**MAX_INBOUND_MESSAGE_SIZE**) を越えた場合、接続が開いたまま、メッセージは非同期に閉じられます。

[Report a bug](#)

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:
EAP_HOME/bin/standalone.conf を開きます。
 - 管理対象ドメインの場合:
EAP_HOME/bin/domain.conf を開きます。
2. IPv4 Stack Java プロパティを `false` に変更します。

```
-Djava.net.preferIPv4Stack=false
```

例を以下に示します。

```
# 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
```

[Report a bug](#)

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. 画面の上部にある **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. 詳細リストの **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。

[Report a bug](#)

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:
EAP_HOME/bin/standalone.conf を開きます。
 - 管理対象ドメインの場合:
EAP_HOME/bin/domain.conf を開きます。
2. 以下の Java プロパティを Java VM オプションに追加します。

```
-Djava.net.preferIPv6Addresses=true
```

例を以下に示します。

```
# 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
```

[Report a bug](#)

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 の仕様は、次のサイトで定義されています: <http://jcp.org/en/jsr/detail?id=221>

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。

[Report a bug](#)

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 <https://access.redhat.com/site/articles/111663> で参照できます。

[Report a bug](#)

6.1.3. データソースの型

一般的なリソースタイプには、**datasources** と **XA datasources** の 2 つがあります。

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。

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。

[Report a bug](#)

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、軽量な関係データベース管理システムであり、開発者がアプリケーションを迅速に構築できるようにします。また、このプラットフォームのデータソースの例になります。

**警告**

本番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あり、本番稼働用として堅牢またはスケーラブルではありません。

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。

[Report a bug](#)

6.1.5. -ds.xml ファイルのデプロイメント

JBoss EAP 6 では、データソースはサーバー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** ファイルを JBoss EAP 6 にデプロイするには、<http://www.ironjacamar.org/documentation.html> の「Schemas」下にある Data sources の 1.1 にしがいます。

**警告**

この機能は開発目的でのみ使用するようになっています。JBoss の管理ツールではサポートされないため、本番環境での使用は推奨されません。

**重要**

***-ds.xml** ファイルをデプロイする場合、すでにデプロイまたは定義されている <driver> エントリーへの参照を使用しなければなりません。

[Report a bug](#)

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

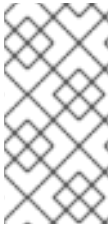
概要

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。そのため、管理対象ドメインを使用する場合は、サーバーグループ内の複数のサーバー全体でドライバーをデプロイできます。

前提条件

このタスクを実行する前に、以下の前提条件を満たしている必要があります。

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



注記

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

手順6.1 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。使用しない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。

[Report a bug](#)

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 ドライバーをダウンロードする場所の一覧は [「JDBC ドライバーをダウンロードできる場所」](#) を参照してください。
- アーカイブを展開します。

手順6.2 コアモジュールとしての JDBC ドライバーのインストール

1. **EAP_HOME/modules/** ディレクトリー下にファイルパス構造を作成します。たとえば、MySQL JDBC ドライバーの場合には、**EAP_HOME/modules/com/mysql/main/** のようなディレクトリー構造を作成します。

2. JDBC ドライバー JAR を **main/** サブディレクトリーにコピーします。
3. **main/** サブディレクトリーで、「**モジュール**」の例に似た **module.xml** ファイルを作成します。**module** XSD は **EAP_HOME/docs/schema/module-1_2.xsd** ファイルに定義されます。
4. サーバーを起動します。
5. 管理 CLI を起動します。
6. 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たクラスの数によって異なります。たとえば、MySQL 5.1.20 JDBC JAR 内の **/META-INF/services/java.sql.Driver** ファイルは以下の 2 つのクラスをリストします。

- **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。これを行わないと、以下のようなエラーが発生します。

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6.11 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 コマンドの例

```
/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6.2 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 コマンドの例

```
/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6.3 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 コマンドの例

```
/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6.4 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 コマンドの例

```
/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)
```

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。

[Report a bug](#)

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 で使用す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する場所を示しています。これらのリンク先は、Red Hat が管理および監視していない他社の Web サイトであります。データベースの最新のドライバーについては、データベースのベンダーのドキュメントや 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
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/

[Report a bug](#)

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の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 の一部でないベンダー固有の機能を使用する必要がある場合に必要です。



警告

これは、高度な手順であり、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6.3 アプリケーションへの依存関係の追加

- ◦ **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。

依存関係: *MODULE_NAME*

例6.5 依存関係の例

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** フォルダーで **jboss-deployment-structure.xml** という名前のファイルを作成します。

例6.6 サンプル jboss-deployment-structure.xml ファイル

```
<jboss-deployment-structure>
  <deployment>
    <dependencies>
      <module name="com.mysql" />
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

例6.7 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();
```

[Report a bug](#)

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。



注記

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

手順6.4 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
 - CLI ツールを起動し、サーバーに接続します。
 - 以下の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。

```
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
```

- o. 管理コンソール

- a. 管理コンソールへログインします。
- b. 管理コンソールの **Datasources** パネルに移動します。
 - i. コンソールの上部から **Configuration** タブを選択します。
 - ii. ドメインモードの場合は、左上のドロップダウンボックスからプロファイルを選択します。
 - iii. コンソールの左側にある **Subsystems** メニューを展開し、**Connector** メニューを展開します。
 - iv. コンソールの左側にあるメニューより **Datasources** を選択します。
- c. 新しいデータソースを作成します。
 - i. **Datasources** パネルの上部にある **Add** を選択します。
 - ii. **Create Datasource** ウィザードで新しいデータソースの属性を入力し、**Next** ボタンを押します。
 - iii. **Create Datasource** ウィザードで JDBC ドライバーの詳細を入力し、**Next** をクリックします。
 - iv. **Create Datasource** ウィザードで接続設定を入力します。
 - v. **Test Connection** ボタンをクリックしてデータソースへの接続をテストし、設定が正しいことを確認します。
 - vi. **Done** をクリックして終了します。

結果

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

[Report a bug](#)

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」](#)。



注記

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 と統合する場合、必ず **jta** パラメーターを **true** に設定してください。

手順6.5 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. コンソールの上部から **Configuration** タブを選択します。
 - ii. ドメインモードの場合は、左上のドロップダウンボックスからプロファイルを選択します。
 - iii. コンソールの左側にある **Subsystems** メニューを展開し、**Connector** メニューを展開します。
 - iv. 展開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下に表示されます。
 - ii. **Edit** をクリックし、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ます。変更は **standalone.xml** または **domain.xml** ファイルのどちらかと、管理インターフェースで確認できます。

- 新しいデータソースを作成する場合は [「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場合は [「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。

[Report a bug](#)

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6.6 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. コンソールの上部から **Configuration** タブを選択します。
 - ii. ドメインモードの場合は、左上のドロップダウンボックスからプロファイルを選択します。
 - iii. コンソールの左側にある **Subsystems** メニューを展開し、**Connector** メニューを展開します。
 - iv. **Datasources** を選択します。
- c. 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場合は [「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。

[Report a bug](#)

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

必読トピック:

- [「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。



注記

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

手順6.7 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。

```
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. コンソールの上部から **Configuration** タブを選択します。
 - ii. ドメインモードの場合は、左上のドロップダウンボックスからプロファイルを選択します。
 - iii. コンソールの左側にある **Subsystems** メニューを展開し、**Connector** メニューを展開します。
 - iv. **Datasources** を選択します。

- c. **XA Datasource** タブを選択します。
- d. 新しい XA データソースを作成します。
 - i. **追加** をクリックします。
 - ii. **Create XA Datasource** ウィザードに新しい XA データソースの属性を入力し、**Next** をクリックします。
 - iii. **Create XA Datasource** ウィザードに JDBC ドライバーの詳細を入力し、**Next** をクリックします。
 - iv. XA プロパティを入力し、**Next** をクリックします。
 - v. **Create XA Datasource** ウィザードで接続設定を入力します。
 - vi. **Test Connection** ボタンをクリックして XA データソースへの接続をテストし、設定が正しいことを確認します。
 - vii. **Done** をクリックして終了します。

結果

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

関連トピック:

- [「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」](#)

[Report a bug](#)

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6.8 管理 CLI または管理コンソールのいずれかを使用した XA データソースの編集

- ○ 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
```

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

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

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

iv. **Datasources** を選択します。

c. **XA Datasource** タブを選択します。

d. データソースを編集します。

i. **Available XA Datasources** リストより関連する XA データソースを選択します。下にある **Attributes** パネルに XA データソースの属性が表示されます。

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 データソースの属性を編集し、作業が終了したら **Save** ボタンを選択します。

結果

XA データソースが設定されます。変更は **standalone.xml** または **domain.xml** ファイルのどちらかと、管理インターフェースで確認できます。

- 新しいデータソースを作成する場合は「[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場合は「[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。

[Report a bug](#)

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6.9 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. コンソールの上部から **Configuration** タブを選択します。
 - ii. ドメインモードの場合は、左上のドロップダウンボックスからプロファイルを選択します。
 - iii. コンソールの左側にある **Subsystems** メニューを展開し、**Connector** メニューを展開します。
 - iv. **Datasources** を選択します。
- c. **XA Datasource** タブを選択します。
- d. 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 データソースを永久的に削除します。

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場合は [「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。

[Report a bug](#)

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 リソースは、設定が関連付けられたリカバリーモジュールを必要とします。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、データベースに登録できます。

[Report a bug](#)

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が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。

これらの各設定属性は、データソースの作成時または作成後に設定できます。設定属性は、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。XA データソースの設定の一般情報については、「[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に対するコンマ区切りの <i>key=value</i> ペアのリストに設定します。

ベンダー固有の設定情報

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。

```
WARN [com.arjuna.ats.jta.logging.logger18N] [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** で設定された Oracle ユーザーがリカバリーに必要なテーブルにアクセスする必要があります。以下の SQL ステートメントは、Oracle バグ 5945463 用の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ステートメントです。

```
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 よりも前の Oracle 11 バージョンを使用する場合は、最後の **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

準備済みトランザクション (XA など) を有効にする手順については、PostgreSQL ドキュメンテーションを参照してください。PostgreSQL の JDBC ドライバーのバージョン 8.4-701 では、**org.postgresql.xa.PGXACConnection** にバグがあり、特定の状況でリカバリーが中断されます。このバグは新しいバージョンでは修正されています。

MySQL

<http://bugs.mysql.com/bug.php?id=12161> に基づき、XA トランザクションリカバリーは一部の MySQL 5 バージョンでは動作しませんでした。この問題は MySQL 6.1 で修正されました。詳細については、バグの URL または MySQL ドキュメンテーションを参照してください。

IBM DB2

IBM DB2 では、クラッシュまたは障害の発生後にアプリケーションサーバーが再起動されたときの指定の再同期ステージの間でのみ **XAResource.recover** メソッドが呼び出されることを想定します。これは、DB2 実装の設計に関する決まりであり、本書の範囲外となります。

Sybase

Sybase は、XA トランザクションがデータベース上で有効にな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。**enable xact coordination** は、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
```

[Report a bug](#)

6.5. データソースセキュリティ

6.5.1. データソースセキュリティ

データソースセキュリティは、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、セキュリティリスクが高くなります。

データソースセキュリティのソリューションには、セキュリティドメインまたはパスワード vault のいずれかを使用することが推奨されます。以下にそれぞれの例を示します。詳細については、以下のドキュメントを参照してください。

- セキュリティドメイン: [「セキュリティドメイン」](#)
- パスワード vault: [「パスワード vault システム」](#)

例6.8 セキュリティドメインの例

```
<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>
```

例6.9 パスワード vault の例

```
<security>
  <user-name>admin</user-name>

  <password>${VAULT::ds_ExampleDS::password::N2NhZDYzOTMtNWE0OS00ZGQ0LWE4MmEt
    MWNIMDMyNDdmNmI2TEIORV9CUkVBS3ZhdWx0}</password>
</security>
```

[Report a bug](#)

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

データベースメンテナンス、ネットワーク問題、または他の停止イベントにより、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。以下の手順に従い、JBoss EAP 6 でデータベース接続検証を有効にするデータソース設定を指定してください。

手順6.10 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> オプションが **true** に設定されている場合は、データ接続が、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> オプションが **true** に設定されている場合は、**<background-validation-millis>** 値とともに使用され、バックグラウンド検証実行の頻度を決定します。**<background-validation-millis>** パラメーターのデフォルト値は 0 ミリ秒であり、デフォルトで無効になります。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 **<background-validation-millis>** 値を決定するには注意が必要です。値が小さいと、プールの検証頻度が多くなり、無効な接続がプールからすぐに削除されます。ただし、値が小さいと、より多くのデータベースリソースが使用されます。値が大きい場合は、接続検証チェックの頻度が少なくなり、より少ないデータベースリソースが使用されますが、デッド接続は長時間検出されません。



注記

<validate-on-match> オプションが **true** に設定されている場合は、**<background-validation>** オプションを **false** に設定する必要があります。それとは逆に、**<background-validation>** オプションが **true** に設定されている場合は、**<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 RDBMS に対して最適化されるため、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.MySQLReplicationValidConnectionChecker
 - 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> 用 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. <exception-sorter> クラス名の設定

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ブ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

[Report a bug](#)

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。
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 spy 機能を有効にします。この機能は、データソースへの JDBC トラフィックをすべてログに記録します。ロギングカテゴリーの jboss.jdbc.spy もロギングサブシステムのログレベルである DEBUG に設定する必要があることに注意してください。
use-ccm	キャッシュ接続マネージャーを有効にします。
new-connection-sql	接続プールに接続が追加された時に実行する SQL ステートメント。
transaction-isolation	次のいずれかになります。 <ul style="list-style-type: none"> ● 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	セキュリティ設定である子要素が含まれます。表 6.8「 セキュリティパラメーター 」を参照してください。
validation	検証設定である子要素が含まれます。表 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	Driver.connect(url,props) メソッドに渡される任意の接続プロパティー。各 connection-property は、文字列名と値のペアを指定します。プロパティー名は名前、値は要素の内容に基づいています。
pool	プーリング設定である子要素が含まれます。表 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。 <i>name=value</i> で指定されます。 setName という形式で setter メソッドが存在する場合、プロパティーは setName(value) という形式の setter メソッドを呼び出すことで設定されます。
xa-datasource-class	実装クラス javax.sql.XADataSource の完全修飾名。
driver	JDBC ドライバーが含まれるクラスローダーモジュールへの一意参照。 <i>driverName#majorVersion.minorVersion</i> の形式にのみ対応しています。
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を指定します。要素が空の場合は true を示します。デフォルト値は、 false です。
use-strict-min	pool-size が厳密かどうかを指定します。デフォルト値は false に設定されています。
flush-strategy	<p>エラーの場合にプールをフラッシュするかどうかを指定します。有効な値は次の通りです。</p> <ul style="list-style-type: none"> ● FailingConnectionOnly ● IdleConnections ● EntirePool <p>デフォルト値は FailingConnectionOnly です。</p>
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 あるいは false のどちらを返すかを指定します。
interleaving	XA 接続ファクトリーのインターリービングを有効にするかどうかを指定します。
no-tx-separate-pools	<p>コンテキスト毎に sub-pool を作成するかどうかを指定します。これには Oracle のデータソースが必要ですが、このデータソースは JTA トランザクションの内部、外部に関わらず、XA 接続の利用ができなくなります。</p> <p>このオプションを使用すると 2 つの実際のプールが作成されるため、プールサイズの合計が max-pool-size の 2 倍になります。</p>
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を指定します。

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	認証処理を行う JAAS security-manager 名が含まれます。この名前は、JAAS ログイン設定の application-policy/name 属性に関連します。
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	SQLException.isValidConnection(Connection) メソッドを提供し接続を検証するインターフェース org.jboss.jca.adapters.jdbc.ValidConnectionChecker の実装。例外が発生すると接続が破棄されます。存在する場合、 check-valid-connection-sql パラメーターが上書きされます。
check-valid-connection-sql	プール接続の妥当性を確認する SQL ステートメント。これは、管理接続をプールから取得し利用する場合に呼び出される場合があります。
validate-on-match	<p>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。</p> <p>通常、validate-on-match に true を指定したときには background-validation を true に指定することはありません。クライアントが使用する前に接続を検証する必要がある場合に Validate-on-match が必要になります。このパラメーターはデフォルトでは false になっています。</p>

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。 validate-on-match を使用しない場合、バックグラウンドの検証はパフォーマンスを最適化します。 validate-on-match が true のときに background-validation を使用すると、チェックが冗長になることがあります。バックグラウンド検証では、不良の接続がクライアントに提供される可能性があります (検証スキャンと接続がクライアントに提供されるまでの間に接続が悪くなります)。そのため、クライアントアプリケーションはこの接続不良の可能性に対応する必要があります。
background-validation-millis	バックグラウンド検証を実行する期間 (ミリ秒単位)。
use-fast-fail	true の場合、接続が無効であれば最初に接続を割り当てしようとした時点で失敗します。デフォルト値は false です。
stale-connection-checker	ブール値の isStaleConnection(SQLException e) メソッドを提供する org.jboss.jca.adapters.jdbc.StaleConnectionChecker のインスタンス。このメソッドが true を返すと、 SQLException のサブクラスである org.jboss.jca.adapters.jdbc.StaleConnectionException に例外がラップされます。
exception-sorter	ブール値である isExceptionFatal(SQLException e) メソッドを提供する org.jboss.jca.adapters.jdbc.ExceptionSorter のインスタンス。このメソッドは、例外が connectionErrorOccurred メッセージとして javax.resource.spi.ConnectionEventListener のすべてのインスタンスへブロードキャストされるかどうかを検証します。

表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	lock() の代わりに tryLock() を使用します。これは、ロックが使用できない場合に即座に失敗するのではなく、設定された秒数間ロックの取得を試みます。デフォルト値は 60 秒です。たとえば、タイムアウトを 5 分に設定するには、 <use-try-lock>300</use-try-lock> を設定します。
blocking-timeout-millis	接続待機中にブロックする最大時間 (ミリ秒)。この時間を超過すると、例外が発生します。これは、接続許可の待機中のみブロックし、新規接続の作成に長時間要している場合は例外が発生しません。デフォルト値は 30000 (30 秒) です。

パラメーター	説明
idle-timeout-minutes	アイドル接続が切断されるまでの最大時間 (分単位)。実際の最大時間は idleRemover のスキャン時間によって異なります。idleRemover のスキャン時間はプールの最小 idle-timeout-minutes の半分になります。
set-tx-query-timeout	トランザクションがタイムアウトするまでの残り時間を基にクエリーのタイムアウトを設定するかどうかを指定します。トランザクションが存在しない場合は設定済みのクエリーのタイムアウトが使用されます。デフォルト値は false です。
query-timeout	クエリーのタイムアウト (秒)。デフォルト値はタイムアウトなしです。
allocation-retry	例外を発生させる前に接続の割り当てを再試行する回数。デフォルト値は 0 で、初回の割り当て失敗で例外が発生します。
allocation-retry-wait-millis	接続の割り当てを再試行するまで待機する期間 (ミリ秒単位)。デフォルト値は 5000 (5 秒) です。
xa-resource-timeout	ゼロでない場合、この値は XAResource.setTimeout メソッドへ渡されます。

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
track-statements	<p>接続がプールへ返され、ステートメントが準備済みステートメントキャッシュへ返された時に、閉じられていないステートメントをチェックするかどうかを指定します。false の場合、ステートメントは追跡されません。</p> <p>有効な値</p> <ul style="list-style-type: none"> ● true: ステートメントと結果セットが追跡され、ステートメントが閉じられていない場合は警告が出力されます。 ● false: ステートメントと結果セットのいずれも追跡されません。 ● nowarn: ステートメントは追跡されますが、警告は出力されません。これがデフォルト設定となっています。
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。

パラメーター	説明
share-prepared-statements	閉じずに同じステートメントを 2 回要求した場合に、同じ基盤の準備済みステートメントを使用するかどうかを指定します。デフォルト値は false です。

表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。

[Report a bug](#)

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	<code>jdbc:postgresql://SERVER_NAME:PORT/DATABASE_NAME</code>
MySQL	<code>jdbc:mysql://SERVER_NAME:PORT/DATABASE_NAME</code>
Oracle	<code>jdbc:oracle:thin:@ORACLE_HOST:PORT:ORACLE_SID</code>
IBM DB2	<code>jdbc:db2://SERVER_NAME:PORT/DATABASE_NAME</code>
Microsoft SQLServer	<code>jdbc:microsoft:sqlserver://SERVER_NAME:PORT;DatabaseName=DATABASE_NAME</code>

[Report a bug](#)

6.7.3. データソースの拡張

データソースのデプロイメントは、JDBC リソースアダプターにある複数の拡張を使用して接続の検証を改善でき、例外が接続を再確立するべきかどうかを確認できます。これらの拡張は次のとおりです。

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.StaleConnection	<stale-connection-checker>	org.jboss.jca.adapters.jdbc.StaleConnectionException の古いSQLExceptionsをラップします。
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.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

[Report a bug](#)

6.7.4. データソース統計の表示

以下のコマンドを適切に編集すると、**jdbc** および **pool** の定義済みデータソースより統計を表示できます。

手順6.11

- ```
/subsystem=datasources/data-source=ExampleDS/statistics=jdbc:read-resource(include-runtime=true)
```
- ```
/subsystem=datasources/data-source=ExampleDS/statistics=pool:read-resource(include-runtime=true)
```



注記

統計はすべてランタイムのみの情報で、デフォルト値は **false** であるため、必ず ***include-runtime=true*** 引数を指定するようにしてください。

[Report a bug](#)

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。

名前	説明
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。
WaitCount	接続を待機する必要がある要求の数。

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。

名前	説明
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、**Core** と **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)
```

[Report a bug](#)

6.8. データソース例

6.8.1. PostgreSQL データソースの例

例6.10

PostgreSQL のデータソースの設定例は以下のとおりです。データソースの有効化、ユーザーの追加、および検証オプションの設定が行われます。

```
<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>
      <background-validation>true</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker">
</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。

```

<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>

```

[Report a bug](#)

6.8.2. PostgreSQL XA データソースの例

例6.11

PostgreSQL XA のデータソースの設定例は以下のとおりです。データソースの有効化、ユーザーの追加、および検証オプションの設定が行われます。

```

<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>
      <background-validation>true</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker">
      </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。

```
<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>
```

[Report a bug](#)

6.8.3. MySQL データソースの例

例6.12

MySQL のデータソース設定例は以下のとおりです。データソースの有効化、ユーザーの追加、および検証オプションの設定が行われます。

```
<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>
      <background-validation>true</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。

```
<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>

```

[Report a bug](#)

6.8.4. MySQL XA データソースの例

例6.13

MySQL XA のデータソース設定例は以下のとおりです。データソースの有効化、ユーザーの追加、および検証オプションの設定が行われます。

```

<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">mysql</xa-datasource-property>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <background-validation>true</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。

```

<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>

```

[Report a bug](#)

6.8.5. Oracle データソースの例



注記

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

例6.14

Oracle のデータソース設定例は以下のとおりです。データソースの有効化、ユーザーの追加、および検証オプションの設定が行われます。

```
<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>
      <background-validation>true</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">
      <xa-datasource-class>oracle.jdbc.xa.client.OracleXADataSource</xa-datasource-class>
    </driver>
  </drivers>
</datasources>
```

上記の Oracle データソースの **module.xml** ファイルの例は次のとおりです。

```
<module xmlns="urn:jboss:module:1.1" name="com.oracle">
  <resources>
    <resource-root path="ojdbc6.jar"/>
  </resources>
  <dependencies>
    <module name="javax.api"/>
  </dependencies>
</module>
```



```

<module name="javax.transaction.api"/>
</dependencies>
</module>

```

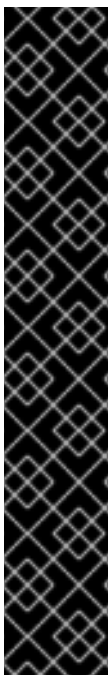
[Report a bug](#)

6.8.6. Oracle XA データソースの例



注記

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



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)

例6.15

Oracle XA のデータソース設定例は以下のとおりです。データソースの有効化、ユーザーの追加、および検証オプションの設定が行われます。

```

<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>

```

```

    <background-validation>true</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。

```

<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>

```

[Report a bug](#)

6.8.7. Microsoft SQLServer のデータソースの例

例6.16

Microsoft SQLServer のデータソース設定例は以下のとおりです。データソースの有効化、ユーザーの追加、および検証オプションの設定が行われます。

```

<datasources>
  <datasource jndi-name="java:/MSSQLDS" pool-name="MSSQLDS">
    <connection-
url>jdbc:microsoft:sqlserver://localhost:1433;DatabaseName=MyDatabase</connection-url>
    <driver>sqlserver</driver>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <validation>
      <background-validation>true</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker"></valid-
connection-checker>

```

```

    </validation>
  </datasource>
</drivers>
  <driver name="sqlserver" module="com.microsoft">
    <xa-datasource-class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-
datasource-class>
  </driver>
</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。

```

<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>

```

[Report a bug](#)

6.8.8. Microsoft SQLServer XA のデータソースの例

例6.17

Microsoft SQLServer XA のデータソース設定例は以下のとおりです。データソースの有効化、ユーザーの追加、および検証オプションの設定が行われます。

```

<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>
      <background-validation>true</background-validation>
      <valid-connection-checker class-
name="org.jboss.jca.adapters.jdbc.extensions.mssql.MSSQLValidConnectionChecker"></valid-
connection-checker>
    </validation>
  </xa-datasource>
</drivers>
  <driver name="sqlserver" module="com.microsoft">
    <xa-datasource-class>com.microsoft.sqlserver.jdbc.SQLServerXADataSource</xa-

```

```
datasource-class>
</driver>
</drivers>
</datasources>
```

上記の 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。

```
<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>
```

[Report a bug](#)

6.8.9. IBM DB2 データソースの例

例6.18

IBM DB2 のデータソース設定例は以下のとおりです。データソースの有効化、ユーザーの追加、および検証オプションの設定が行われます。

```
<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>
      <background-validation>true</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">
    <xa-datasource-class>com.ibm.db2.jdbc.DB2XADataSource</xa-datasource-class>
```

```

    </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。

```

<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>

```

[Report a bug](#)

6.8.10. IBM DB2 XA のデータソースの例

例6.19

IBM DB2 XA のデータソース設定例は以下のとおりです。データソースの有効化、ユーザーの追加、および検証オプションの設定が行われます。

```

<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>
    <security>
      <user-name>admin</user-name>
      <password>admin</password>
    </security>
    <xa-pool>
      <is-same-rm-override>false</is-same-rm-override>
    </xa-pool>
    <validation>
      <background-validation>true</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>
      </recover-plugin>
    </recovery>
  </xa-datasource>
</datasources>

```

```

        <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。

```

<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>

```

[Report a bug](#)

6.8.11. Sybase データソースの例

例6.20

Sybase データソースの設定例は以下のとおりです。データソースの有効化、ユーザーの追加、および検証オプションの設定が行われます。

```

<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>
      <background-validation>true</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。

```

<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>

```

[Report a bug](#)

6.8.12. Sybase XA データソースの例

例6.21

Sybase XA データソースの設定例は以下のとおりです。データソースの有効化、ユーザーの追加、および検証オプションの設定が行われます。

```

<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>
    <validation>
      <background-validation>true</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。

```
<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>
```

[Report a bug](#)

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、静的および動的モジュールと呼ばれる 2 つのタイプのモジュールを識別します。この 2 つのタイプのモジュールは、パッケージ化された方法のみが異なります。すべてのモジュールは同じ機能を提供します。

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/** ディレクトリーに事前定義されます。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**) および必要な 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って提供されるすべての API (Java EE API や、JBoss Logging などのその他の API を含む) は、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>
```

main/ サブディレクトリー名以外のモジュール名 **com.mysql** はその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、**JBOSS_HOME/modules** ディレクトリー内の **system** ディレクトリーにあります。そのため、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 またはそれ以降のバージョンの上部レイヤーにある Red Hat 提供のレイヤー製品も、モジュールを **system** ディレクトリー内に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 管理者は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。

モジュール毎のディレクトリーになるよう、カスタムモジュールが **JBOSS_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。こうすると、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。これにより、ユーザー提供のモジュールが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。**system** 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から派生されます。デプロイメントはモジュールとしてロードされるため、依存関係を設定でき、他のデプロイメントは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。

[Report a bug](#)

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 がすべての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。このモジュールをアプリケーションサーバーのグローバルモジュールリストへ追加すると、どのモジュールもグローバルモジュールになります。モジュールへの変更は必要ありません。

[Report a bug](#)

7.1.3. モジュールの依存性

モジュール依存関係とは、あるモジュールが機能するには別のモジュールのクラスが必要になるという宣言です。モジュールはいくつでも他のモジュールの依存関係を宣言することができます。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き、モジュールクラスローダーが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。

明示的な依存関係は開発者が設定に宣言します。静的モジュールでは、依存関係を `modules.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、任意の依存関係として指定できます。任意の依存関係をロードできなくても、モジュールによるロードは失敗しません。しかし、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。モジュールがロードされるときに依存関係が使用できなければなりません。

暗黙的な依存関係は、デプロイメントで特定の状態やメタデータが見つかったときに、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。

デプロイメントを設定して特定の暗黙的な依存関係を除外することも可能です。この設定は `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、暗示的な依存関係のエクスポートを指定できます。エクスポートされた依存関係は、エクスポートするモジュールに依存するモジュールへ提供されます。

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。

[Report a bug](#)

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べきでない場合 (厳格なサブデプロイメントの分離が必要な場合) は、この挙動を有効にできます。

[Report a bug](#)

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の分離を無効化する方法を説明します。このタスクは、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。この作業を行う前に、サーバーを停止する必要があります。最終リリースの管理ツールでは、このような設定がサポートされるようになるため、これは一時的な措置になります。

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によって異なります。ま

た、デフォルト以外の場所とファイル名が使用されることがあります。デフォルトの設定ファイルは、**domain/configuration/domain.xml** (管理対象ドメインの場合) と **standalone/configuration/standalone.xml** (スタンドアロンサーバー) です。

3. EE サブシステム設定の特定

設定ファイルの EE サブシステム設定要素を特定します。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 サブシステム要素には **urn:jboss:domain:ee:1.2** のネームスペースがあります。

```
<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

以下のように、**ear-subdeployments-isolated** 要素を EE サブシステム要素の子として追加し、**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。

[Report a bug](#)

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され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一覧は、「[含まれるモジュール](#)」を参照してください。モジュールが他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. 管理コンソールにログインします (「[管理コンソールへのログイン](#)」)。
2. **EE Subsystem** パネルへ移動します。
 - a. コンソールの上部から **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より **Container** → **EE** の順で選択します。
3. **Subsystem Defaults** セクションの **Add** をクリックします。 **Create Module** 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** をクリックしてキャンセルします。
 - **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、何も変更されません。

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。

[Report a bug](#)

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. **module/** ディレクトリー構造を作成し、ファイルを追加します。
 - a. **EAP_HOME/module** ディレクトリー下にディレクトリー構造を作成し、ファイルや JAR が含まれるようにします。例は次のとおりです。

```
$ cd EAP_HOME/modules/
$ mkdir -p myorg-conf/main/properties
```

- b. 作成した **EAP_HOME/modules/myorg-conf/main/properties/** ディレクトリーにプロパティファイルを移動します。

- c. 次の XML が含まれる **module.xml** ファイルを **EAP_HOME/modules/myorg-conf/main/** ディレクトリーに作成します。

```
<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** サブシステムを編集します。JBoss CLI を使用するか、手作業でファイルを編集します。

- 次の手順に従って JBoss CLI を使用し、サーバー設定ファイルを編集します。

- 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** サブシステムに **myorg-conf** <global-modules> 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"}])
```

次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、**EAP_HOME/standalone/configuration/standalone.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編集された **ee** サブシステム要素の例になります。

```
<subsystem xmlns="urn:jboss:domain:ee:1.0" >
  <global-modules>
    <module name="myorg-conf" slot="main" />
  </global-modules>
</subsystem>
```



```
</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は、以下のようなコードを使用してプロパティファイルをロードできるようになります。

```
Thread.currentThread().getContextClassLoader().getResource("my.properties");
```

[Report a bug](#)

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/** ディレクトリーのモジュールを探します。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するには、**JBOSS_MODULEPATH** 環境変数を定義するか、起動設定ファイルに変数を設定します。本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 では、以下のようにディレクトリーのリストをコロンで区別します。

```
export
JBOSS_MODULEPATH=EAP_HOME/modules:/home/username/external/modules/directory/
```

Windows では、以下のようにディレクトリーのリストをセミコロンで区別します。

```
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たくない場合は、JBoss EAP 起動設定ファイルで **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は、**EAP_HOME/bin/domain.conf** ファイルになります。

以下は、**JBOSS_MODULEPATH** 変数を **standalone.conf** ファイルに設定するコマンドの例になります。

```
JBOSS_MODULEPATH="EAP_HOME/modules:/home/username/external/modules/directory/"
```

[Report a bug](#)

7.6. 参考資料

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や、それらのモジュールがサポートされるかどうかを表すリストについては、カスタマーポータル[の https://access.redhat.com/articles/1122333](https://access.redhat.com/articles/1122333) を参照してください。

[Report a bug](#)

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。

1. 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** のモジュール名は **deployment.inventory.war** となり、**store.jar** のモジュール名は **deployment.store.jar** となります。

2. 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** 内にある **reports.war** のサブデプロイメントのモジュール名は **deployment.accounts.ear.reports.war** になります。

[Report a bug](#)

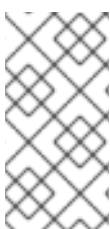
第8章 JSVC

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上で実行できるようにす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 で同様の機能を使用する場合は、Red Hat カスタマーポータルからダウンロード可能な **Native Utilities for Windows Server** 用ファイルに含まれる **prunsvr.exe** を参照してください。

[Report a bug](#)

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方法は、スタンドアロンとドメインのどちらのモードが実行されているかによって異なります。JBoss EAP がドメインモードで実行されている場合は、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 を起動する場合、使用するコマンドに関係なく、特権ユーザーで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、ご使用のオペレーティングシステム用の「Native Utilities」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。サーバーを起動および停止するために使用されるアカウントには、JBoss EAP がインストールされたディレクトリーの読み書き権限が必要です。
- 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 に Jsvc をインストールした方法によって異なることに注意してください。コマンドの変数は、以下の表を参照して使用するファイルを判断してください。

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	<code>\${eap-installation-location}/jboss-eap-\${version}</code>
JSVC-BIN	<code>EAP_HOME/modules/system/layers/base/native/sbin/jsvc</code>
JSVC-JAR	<code>EAP_HOME/modules/system/layers/base/native/sbin/commons-daemon.jar</code>
CONF-DIR	<code>EAP_HOME/standalone/configuration</code>
LOG-DIR	<code>EAP_HOME/standalone/log</code>

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	<code>/usr/share/jbossas</code>
JSVC-BIN	<code>/usr/bin/jsvc-eap6/jsvc</code>
JSVC-JAR	<code>EAP_HOME/modules/system/layers/base/native/sbin/commons-daemon.jar</code>
CONF-DIR	<code>/etc/jbossas/standalone</code>
LOG-DIR	<code>/var/log/jbossas/standalone</code>

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。ドメインモードでは、`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	<code>\${eap-installation-location}/jboss-eap-\${version}</code>
JSVC-BIN	<code>EAP_HOME/modules/system/layers/base/native/sbin/jsvc</code>
JSVC-JAR	<code>EAP_HOME/modules/system/layers/base/native/sbin/commons-daemon.jar</code>
CONF-DIR	<code>EAP_HOME/domain/configuration</code>
LOG-DIR	<code>EAP_HOME/domain/log</code>

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 が正常に終了されなかった場合、Jsvc は JBoss EAP 6 を自動的に再起動します。JBoss EAP 6 が正常に終了されると、Jsvc も停止します。

[Report a bug](#)

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要求に挿入される Java クラスです。バルブはサブレットフィルターの前にパイプラインへ挿入されます。バルブは要求を渡す前に変更を加えたり、認証や要求のキャンセルなどの他の処理を実行したりできます。

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、設定とパッケージ化の方法のみが異なります。

- 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については、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します。アプリケーションレベルでバルブを設定する方法の手順については、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。

[Report a bug](#)

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へ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、グローバルバルブとなります。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「[グローバルバルブの設定](#)」を参照してください。

[Report a bug](#)

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、要求の証明情報を認証するバルブです。オーセンティケーターバルブは `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。

[Report a bug](#)

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。このタスクでは、モジュールのインストール方法について説明します。

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。

- モジュールに対して **module.xml** ファイルを作成する必要があります。

module.xml ファイルの例は「[モジュール](#)」を参照してください。

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 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。

[Report a bug](#)

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。これには JBoss CLI ツールを使用します。

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の参照に使用される名前)
- **MODULENAME** (設定され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)

```
/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=testvalve:add-param(param-name="NAME", param-value="VALUE")
```

```
/subsystem=web/valve=testvalve: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。

[Report a bug](#)

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 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を提供します。管理者向けには、管理コンソールと管理 CLI によって、本番環境でアプリケーションデプロイメントを管理できる理想的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が提供されます。開発者用には、アプリケーションデプロイメントのテストオプションとして、設定可能なファイルシステム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が提供されます。

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」

Development

- **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」](#)

[Report a bug](#)

10.2. 管理コンソールでのデプロイ

10.2.1. 管理コンソールでのアプリケーションデプロイメント管理

管理コンソールよりアプリケーションをデプロイすると、使いやすさが利点のグラフィカルインターフェースを使用できます。ご使用のサーバーもしくはサーバーグループにデプロイされているアプリケーションを一目で確認できるほか、必要に応じてアプリケーションを無効にしたり、コンテンツリポジトリから削除したりできます。

[Report a bug](#)

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

1. コンソールの上部より **Runtime** タブを選択します。
2. ◦ 管理対象ドメインの場合は、**Domain** メニューを展開します。
 - スタンドアロンサーバーの場合は、**Server** メニューを展開します。
3. **Manage Deployments** を選択します。
4. 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とそれらの状態が表示されます。
 - 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た後に **En/Disable** をクリックします。
 - 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有効にすることを確定します。
 - **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、それらの状態を表す **Available Deployment Content** の表が含まれます。
 - 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。

- b. アプリケーションを追加したい各サーバーグループのボックスにチェックマークを付け、**Save** をクリックして終了します。
- c. **Server Groups** タブをクリックし **Server Groups** の表を表示します。アプリケーションが選択したサーバーグループにデプロイされます。

結果

該当のサーバーあるいはサーバーグループにアプリケーションがデプロイされます。

[Report a bug](#)

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

1. a. コンソールの上部より **Runtime** タブを選択します。
 - b. ■ 管理対象ドメインの場合は、**Domain** メニューを展開します。
 - スタンドアロンサーバーの場合は、**Server** メニューを展開します。
 - c. **Manage Deployments** を選択します。
2. 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とそれらの状態が表示されます。
 - 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、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定します。
 - **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**
Manage Deployments Content 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。
 - b. アプリケーションをアンデプロイするサーバーの名前を **Server Group** の表から選択します。**View** をクリックしてアプリケーションを確認します。

- 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定します。
- 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。**Group Deployments** の表の各サーバーグループに対してアプリケーションの状態が確定されます。

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。

[Report a bug](#)

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

1.
 - a. コンソールの上部より **Runtime** タブを選択します。
 - b.
 - 管理対象ドメインの場合は、**Domain** メニューを展開します。
 - スタンドアロンサーバーの場合は、**Server** メニューを展開します。
 - c. **Manage Deployments** を選択します。
2. 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とそれらの状態が表示されます。
 - 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、選択され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することを確定します。
 - **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**
Manage Deployments Content 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。
- b. アプリケーションをアンデプロイするサーバーの名前を **Server Group** の表から選択します。 **View** をクリックしてアプリケーションを確認します。
- c. アプリケーションを選択し、 **Remove** をクリックして、選択され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することを確定します。
- 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。 **Group Deployments** の表の各サーバーグループに対してアプリケーションの状態が確定されます。

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。

[Report a bug](#)

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。スタンドアロンインスタンスの場合は単一サーバーのデプロイメント状態を管理でき、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。

[Report a bug](#)

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

- **deploy コマンドの実行**
管理 CLI で、アプリケーション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。

```
[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。

[Report a bug](#)

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** コマンドはアンデプロイし、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とともに、管理 CLI **undeploy** 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** ともに、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

指定されたアプリケーションはアンデプロイされます。**undeploy** コマンドは、成功した場合に管理 CLI に出力を表示しません。

[Report a bug](#)

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

- **deploy コマンドの実行**

管理対象ドメインにアプリケーションをデプロイするには、管理 CLI を使用して、アプリケーションをデプロイします。

管理 CLI で、アプリケーション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。すべてのサーバーグループをデプロイするために **--all-server-groups** パラメーターを追加します。

```
[domain@localhost:9999 /] deploy /path/to/test-application.war --all-server-groups
```

- または、**--server-groups** パラメーターを追加して、デプロイメントに固有な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。

[Report a bug](#)

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すると、アプリケーションがデプロイされたすべての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。

[Report a bug](#)

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。

手順10.8 DeployDmrToJson.java を使用したアプリケーションのデプロイ

1. **DeployDmrToJson.java** を使用して JSON への要求を生成し、アプリケーションをデプロイします。

例10.1 DeployDmrToJson.java クラス

```
import org.jboss.dmr.ModelNode;
import java.net.URL;

public class DeployDmrToJson
{
    public static void main(String[] args) throws Exception
    {
        if(args.length < 1)
            throw new IllegalArgumentException("The first argument must be a URL");

        URL url = new URL(args[0]);
        String[] pathElements = url.getFile().split("/");
        String name = pathElements[pathElements.length-1];

        ModelNode deploy = getDeploy(url.toExternalForm(), name);
        ModelNode undeploy = getUndeploy(name);

        System.out.println("Deploy\n-----\n");
        System.out.println("Formatted:\n" + deploy.toJSONString(false));
        System.out.println("Unformatted:\n" + deploy.toJSONString(true));
        System.out.println("\nUndeploy\n-----\n");
        System.out.println("Formatted:\n" + undeploy.toJSONString(false));
        System.out.println("Unformatted:\n" + undeploy.toJSONString(true));
    }

    public static ModelNode getUndeploy(String name)
    {
        ModelNode undeployRequest = new ModelNode();
        undeployRequest.get("operation").set("undeploy");
        undeployRequest.get("address", "deployment").set(name);

        ModelNode removeRequest = new ModelNode();
        removeRequest.get("operation").set("remove");
        removeRequest.get("address", "deployment").set(name);

        ModelNode composite = new ModelNode();
        composite.get("operation").set("composite");
        composite.get("address").setEmptyList();
        final ModelNode steps = composite.get("steps");
        steps.add(undeployRequest);
        steps.add(removeRequest);
        return composite;
    }

    public static ModelNode getDeploy(String url, String name)
    {
        ModelNode deployRequest = new ModelNode();
        deployRequest.get("operation").set("deploy");
        deployRequest.get("address", "deployment").set(name);
    }
}
```



```

ModelNode addRequest = new ModelNode();
addRequest.get("operation").set("add");
addRequest.get("address", "deployment").set(name);
addRequest.get("content").get(0).get("url").set(url);

ModelNode composite = new ModelNode();
composite.get("operation").set("composite");
composite.get("address").setEmptyList();
final ModelNode steps = composite.get("steps");
steps.add(addRequest);
steps.add(deployRequest);
return composite;
}
}

```

2. 以下のように、コマンドを使用してクラスを実行します。

例10.2 コマンドの実行

```

java -cp .:$JBASS_HOME/modules/org/jboss/dmr/main/jboss-dmr-1.1.1.Final-redhat-1.jar
DeployDmrToJson \
file:///Users/username/support/helloWorld.war/dist/helloWorld.war

```

3. クラスが実行されると、以下のコマンド形式が表示されます。必要に応じて、**deploy** または **undeploy** コマンドを使用します。

例10.3 デプロイおよびアンデプロイコマンド

```

Deploy
-----

Formatted:
{
  "operation" : "composite",
  "address" : [],
  "steps" : [
    {
      "operation" : "add",
      "address" : {"deployment" : "helloWorld.war"},
      "content" : [{"url" :
"file:/Users/username/support/helloWorld.war/dist/helloWorld.war"}]
    },
    {
      "operation" : "deploy",
      "address" : {"deployment" : "helloWorld.war"}
    }
  ]
}
Unformatted:
{"operation" : "composite", "address" : [], "steps" : [{"operation" : "add", "address" :
{"deployment" : "helloWorld.war"}, "content" : [{"url" :

```

```
"file:/Users/username/support/helloWorld.war/dist/helloWorld.war"}]],{"operation" :
"deploy", "address" : {"deployment" : "helloWorld.war"}}}]}
```

Undeploy

Formatted:

```
{
  "operation" : "composite",
  "address" : [],
  "steps" : [
    {
      "operation" : "undeploy",
      "address" : {"deployment" : "helloWorld.war"}
    },
    {
      "operation" : "remove",
      "address" : {"deployment" : "helloWorld.war"}
    }
  ]
}
```

Unformatted:

```
{"operation" : "composite", "address" : [], "steps" : [{"operation" : "undeploy", "address" :
{"deployment" : "helloWorld.war"}}, {"operation" : "remove", "address" : {"deployment" :
"helloWorld.war"}}]}
```

4. 以下のコマンドを使用して、アプリケーションをデプロイまたはアンデプロイします。 **json request** を上記の要求に置き換えます。

例10.4 コマンドの実行

```
curl -f --digest -u "<user>:<pass>" -H Content-Type:\ application/json -d '<json request>'
"http://localhost:9990/management"
```

[Report a bug](#)

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ことにより、迅速な開発サイクルに適した方法でアプリケーションのテストと構築を行うことができます。デプロイメントスキャナーは、デプロイメントの頻度やさまざまなアプリケーションタイプの動作などのニーズに合わせて設定することができます。

[Report a bug](#)

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

このタスクでは、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スタンドアロンサーバーインスタンスに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」](#)の説明にあるように、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方法は2つあります。自動または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。[「管理 CLI でのデプロイメントスキャナーの設定」](#)を読んでから、いずれかのデプロイメント方法を開始してください。

◦ 自動デプロイメント

デプロイメントスキャナーは、[「管理 CLI でのデプロイメントスキャナーの設定」](#)で定義されたようにフォルダー状態の変更を検出し、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では、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 の **`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10.5 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。マーカーファイルがデプロイメントフォルダーで作成され、デプロイメントが正常に終了したことを示します。アプリケーションは、管理コンソールで、**Enabled** と示されます。

例10.6 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
```

[Report a bug](#)

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](#)」の説明にあるように、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ともに使用しないでください。管理コンソールによって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、デプロイメントディレクトリーに含まれるマーカーファイルまたは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。意図しない再デプロイメントや、他のエラーの可能性を最低限にするには、本番稼働環境での管理に管理 CLI および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、**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維持されます。

○ アプリケーションの削除によるアンデプロイ

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。

[Report a bug](#)

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」](#)」の説明にあるように、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で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サイクル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、個々の設定に応じて選択できます。
 - **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と変更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。次の Linux の例では、**touch** コマンドが使用されます。

例10.7 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** ファイルマーカーは以前のファイルを置き換えます。

- **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が再デプロイをトリガーします。「[「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」](#)」に記載された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」](#)」の説明のように、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。アンデプロイメント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。詳細については「[「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。

[Report a bug](#)

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にあるアプリケーションの状態をマークします。マーカーファイルは、アプリケーションと同じ名前を持ち、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。以下の表に、各マーカーファイルのタイプおよび応答の定義を示します。

例10.8 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。展開された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編集がライブでプッシュされることを防ぎます。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。
.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た場合、コンテンツは自動デプロイメントで再び可視状態になります。
.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。

ファイル名接尾辞	生成	説明
.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待っているサーバーに送信されます。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。

[Report a bug](#)

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、これらの属性は **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については、「[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。

表10.2 デプロイメントスキャナーの属性

名前	説明	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
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 属性が指定された場合は、 path 値が、そのディレクトリーまたはパスに対する相対追加分として機能します。	文字列	deployment s

名前	説明	タイプ	デフォルト値
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	リポジトリのスキャン間の時間 (ミリ秒単位)。値が1未満の場合は、スキャナーが起動時にのみ動作します。	Int	5000

[Report a bug](#)

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。これらには、スキャン間隔、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。

[Report a bug](#)

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、バッチスクリプトの使用やリアルタイムによる属性の公開および変更には、管理 CLI を使用できます。**read-attribute** および **write-attribute** グローバル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挙動を変更できます。デプロイメントスキャナー属性の詳細については「[デプロイメントスキャナー属性のリファレンス](#)」の説明を参照してください。

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** で確認できます。

```
<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 を使用して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。これには、ルートノードで **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移動して行います。また、このレベルで **ls** コマンドを

使用して属性を表示することもできます。

◦ **read-resource** 操作を使用したデプロイメントスキャナー属性の表示

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。

```
[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表示

ls コマンドに任意の **-l** 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** コマンドとその引数の詳細を確認するには、**ls --help** と入力して CLI ヘルプエントリを表示します。管理 CLI のヘルプメニューの詳細については、「[管理 CLI でのヘルプの取得](#)」を参照してください。

```
[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、書き込む属性名と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。この例を使用するには、**cd** コマンドとタブ補完を使用してデフォルトスキャナーノードを表示し、移動します。

```
[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、展開された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、例の **newpathname** 値を監視するデプロイメントスキャナーの新しいパス名に置き換えます。変更を反映するにはサーバーでリロードする必要があります。

```
[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 ファイルのパスセクションで定義されたファイルシステムパスに対する相対参照を変更します。変更を反映するには、サーバーでリロードする必要があります。

```
[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

write-attribute 操作を使用して、**scan-enabled** 値を **false** に設定することにより、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。

[Report a bug](#)

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。

[Report a bug](#)

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 を用いてアプリケーションをデプロイする方法について説明します。ここで使用される例は、JBoss EAP 6 のクイックスタートにある **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。このプラグインを使用すると、アプリケーションサーバーでアプリケーションを簡単にデプロイおよびアンデプロイできます。

手順10.13 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10.9 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、アプリケーション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10.10 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10.11 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/32/4b4ef9a4bbe7206d3674a89
807203a2092fc70/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。

[Report a bug](#)

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 を用いてアプリケーションをアンデプロイする方法について説明します。ここで使用される例は、JBoss EAP 6 のクイックスタートにある **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。このプラグインを使用すると、アプリケーションサーバーでアプリケーションを簡単にデプロイおよびアンデプロイできます。

手順10.14 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10.12 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10.13 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10.14 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-
6.3/standalone/data/content/44/e1f3c55c84b777b0fc201d69451223c09c9da5/content
```

```
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。

[Report a bug](#)

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、サーバー起動時にアプリケーションがデプロイされる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徹底することが可能で、再起動後の順序を永続化することも可能です。

手順10.15 EAP 6.0.X 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をサポートします。バッチモードでは、コマンドや操作をグループ化でき、グループ化したコマンドや操作をアトミックユニットとして一緒に実行できます。1つ以上のコマンドや操作が失敗すると、正常に実行された他のコマンドや操作はロールバックされます。

手順10.16 EAP 6.1.X 以降におけるデプロイメント順序の制御

Inter Deployment Dependencies (デプロイメント間依存関係) と呼ばれる EAP 6.1.X 以降の新機能を使用すると、トップレベルのデプロイメント間の依存関係を宣言できます。

1. **jboss-all.xml** ファイルがない場合は、**app.ear/META-INF** フォルダに作成します。**app.ear** はアプリケーションアーカイブで、このアーカイブの前にデプロイされる他のアプリケーションアーカイブに依存します。
2. 次のように、このファイルに **jboss-deployment-dependencies** エントリーを作成します。下記では、**framework.ear** は依存されるアプリケーションアーカイブで、**app.ear** アプリケーションアーカイブの前にデプロイする必要があります。

```
<jboss xmlns="urn:jboss:1.0">
  <jboss-deployment-dependencies xmlns="urn:jboss:deployment-dependencies:1.0">
    <dependency name="framework.ear" />
  </jboss-deployment-dependencies>
</jboss>
```

[Report a bug](#)

10.8. デプロイメント記述子の上書き

EAP 6.1.x には、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起動時に上書きできる新機能が含まれています。**デプロイメントオーバーレイ**は、アーカイブで上書きされなければならないファイルのルールセットを表します。また、上書きされるファイルの代わりに使用

する必要がある新しいファイルへのリンクも提供します。上書きされたファイルがデプロイメントアーカイブにない場合、デプロイメントへ戻されます。

手順10.17 管理 CLI を使用したデプロイメント記述子の上書き

次の手順は、**app.war** というデプロイされたアプリケーションが存在し、このアプリケーションの **WEB-INF/web.xml** ファイルを **/home/user/web.xml** にある別の **web.xml** で上書きすることを前提とします。

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。これを行う方法は2つあります。

- 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へリンクします。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がサーバー上に存在する必要があることに注意してください。名前は指定しますが、まだこの段階では実際のデプロイメントへはリンクしません。

3. アプリケーションの再デプロイ

```
/deployment=app.war:redploy
```

[Report a bug](#)

第11章 JBOSS EAP 6 のセキュア化

11.1. セキュリティーサブシステム

security サブシステムは、アプリケーションのセキュリティーインフラストラクチャーを提供します。サブシステムは、現在のリクエストに関連するセキュリティーコンテキストを使用して認証マネージャー、承認マネージャー、監査マネージャー、およびマッピングマネージャーの機能を関係するコンテナに提供します。

security サブシステムはデフォルトで事前設定されているため、セキュリティー要素を変更する必要はほとんどありません。変更が必要となる可能性がある唯一のセキュリティー要素は、*deep-copy-subject-mode* を使用するかどうかです。ほとんどの場合で、管理者はセキュリティードメインの設定に集中します。

ディープコピーモード

ディープコピーサブジェクトモードの詳細は、「[ディープコピーサブジェクトモード](#)」を参照してください。

セキュリティードメイン

セキュリティードメインとは、認証、承認、監査、およびマッピングを制御するために単一または複数のアプリケーションが使用する *Java Authentication and Authorization Service (JAAS)* 宣言型セキュリティー設定のセットです。デフォルトでは、**jboss-ejb-policy**、**jboss-web-policy**、および **other** の3つのセキュリティードメインが含まれています。アプリケーションの要件に対応するため、必要な数のセキュリティードメインを作成できます。セキュリティードメインの詳細は、「[アプリケーションでのセキュリティードメインの使用](#)」を参照してください。

[Report a bug](#)

11.2. セキュリティーサブシステムの構造

セキュリティーサブシステムは、管理対象ドメインまたはスタンドアロン設定ファイルに設定されます。設定要素のほとんどは、Web ベースの管理コンソールまたはコンソールベースの管理 CLI を使用して設定することが可能です。以下の XML はセキュリティーサブシステムの例を表しています。

例11.1 セキュリティーサブシステムの設定例

```
<subsystem xmlns="urn:jboss:domain:security:1.2">
  <security-management>
    ...
  </security-management>
  <security-domains>
    <security-domain name="other" cache-type="default">
      <authentication>
        <login-module code="Remoting" flag="optional">
          <module-option name="password-stacking" value="useFirstPass"/>
        </login-module>
        <login-module code="RealmUsersRoles" flag="required">
          <module-option name="usersProperties"
value="${jboss.domain.config.dir}/application-users.properties"/>
          <module-option name="rolesProperties"
value="${jboss.domain.config.dir}/application-roles.properties"/>
          <module-option name="realm" value="ApplicationRealm"/>
          <module-option name="password-stacking" value="useFirstPass"/>
        </login-module>
      </authentication>
    </security-domain>
  </security-domains>
</subsystem>
```



```

        </login-module>
    </authentication>
</security-domain>
<security-domain name="jboss-web-policy" cache-type="default">
    <authorization>
        <policy-module code="Delegating" flag="required"/>
    </authorization>
</security-domain>
<security-domain name="jboss-ejb-policy" cache-type="default">
    <authorization>
        <policy-module code="Delegating" flag="required"/>
    </authorization>
</security-domain>
</security-domains>
<vault>
    ...
</vault>
</subsystem>

```

<security-management>、**<subject-factory>**、および **<security-properties>** 要素はデフォルト設定では存在しません。**<subject-factory>** および **<security-properties>** 要素は JBoss EAP 6.1 およびそれ以降のバージョンで廃止になりました。

[Report a bug](#)

11.3. セキュリティーサブシステムの設定

セキュリティーサブシステムの設定には、管理 CLI または Web ベースの管理コンソールを使用できます。

セキュリティーサブシステム内の各トップレベル要素には、セキュリティー設定の異なる情報が含まれています。セキュリティーサブシステムの設定例は「[セキュリティーサブシステムの構造](#)」を参照してください。

<security-management>

このセクションは、security サブシステムの高レベルの動作をオーバーライドします。スレッドの安全性を強化するため、セキュリティートークンヘコピーまたはリンクするかどうかを指定するオプション設定 **deep-copy-subject-mode** が含まれます。

<security-domains>

複数のセキュリティードメインを保持するコンテナ要素です。セキュリティードメインには認証、承認、マッピング、監査モジュール、JASPI 認証、および JSSE 設定の情報が含まれることがあります。アプリケーションはセキュリティードメインを指定してセキュリティー情報を管理します。

<security-properties>

java.security.Security クラスに設定されるプロパティーの名前と値が含まれます。

[Report a bug](#)

11.4. ディープコピーサブジェクトモード

ディープコピーサブジェクトモードが無効であるときに(デフォルト)、セキュリティーデータ構造をコピーすると、データ構造全体がコピーされずに元のデータ構造への参照が作成されます。この挙動は効率的ですが、同一 ID の複数のスレッドによってフラッシュまたはログアウトの操作が行われ、サブジェクトが消去されると、データが破損しやすくなります。

ディープコピーサブジェクトモードでは、クローン可能と指定されていると、データ構造およびデータ構造に関連するデータの完全コピーが作成されます。このモードはよりスレッドセーフですが、効率は悪くなります。

ディープコピーサブジェクトモードは、セキュリティーサブシステムの一部として設定されます。

[Report a bug](#)

11.5. ディープコピーサブジェクトモードの有効化

Web ベースの管理コンソールまたは管理 CLI より、ディープコピーセキュリティーモードを有効にできます。

手順11.1 管理コンソールからディープコピーセキュリティーモードを有効化

1. 管理コンソールへのログイン

「[管理コンソールへのログイン](#)」を参照してください。

2. 管理対象ドメイン: 適切なプロファイルを選択します。

管理対象ドメインではプロファイルごとにセキュリティーサブシステムが設定されているため、各プロファイルに対して個別にディープコピーセキュリティーモードを有効または無効にできます。

プロファイルを選択するには、画面の上部にあるの右上にある **Configuration** をクリックし、左上にある **Profile** ドロップダウンボックスよりプロファイルを選択します。

3. **Security Subsystem** 設定メニューを開きます。

Security メニューを展開し、**Security Subsystem** を選択します。

4. **Deep Copy Subject** モードを有効にします。

Edit をクリックします。**Deep Copy Subjects** の横にあるボックスにチェックを入れ、ディープコピーサブジェクトモードを有効にします。

管理 CLI を使用したディープコピーサブジェクトモードの有効化

管理 CLI を使用してこのオプションを有効にしたい場合、以下のコマンドの1つを使用します。

例11.2 管理対象ドメイン

```
/profile=full/subsystem=security/:write-attribute(name=deep-copy-subject-mode,value=TRUE)
```

例11.3 スタンドアロンサーバー

```
/subsystem=security/:write-attribute(name=deep-copy-subject-mode,value=TRUE)
```

[Report a bug](#)

11.6. セキュリティードメイン

11.6.1. セキュリティードメイン

セキュリティードメインは JBoss EAP 6 のセキュリティーサブシステムの一部になります。すべてのセキュリティー設定は、管理対象ドメインの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によって集中管理されるようになりました。

セキュリティードメインは認証、承認、セキュリティーマッピング、監査の設定によって構成されます。セキュリティードメインは *Java Authentication and Authorization Service (JAAS)* の宣言的セキュリティーを実装します。

認証とはユーザーアイデンティティーを検証することを言います。セキュリティー用語では、このユーザーを *プリンシパル*と呼びます。認証と承認は異なりますが、含まれている認証モジュールの多くは承認の処理も行います。

承認は、認証されたユーザーが、システムまたは操作の特定のリソースへアクセスできるパーミッションまたは特権を持っているかどうかをサーバーが判断するセキュリティーポリシーです。セキュリティー用語では、*ロール*とも呼ばれます。

セキュリティーマッピングとは、情報をアプリケーションに渡す前にプリンシパル、ロール、または属性から情報を追加、編集、削除する機能のことです。

監査マネージャーは、*プロバイダーモジュール*を設定して、セキュリティーイベントの報告方法を制御できるようにします。

セキュリティードメインを使用する場合、アプリケーション自体から特定のセキュリティー設定をすべて削除することが可能です。これにより、一元的にセキュリティーパラメーターを変更できるようにします。このような設定構造が有効な一般的な例には、アプリケーションをテスト環境と実稼動環境間で移動するプロセスがあります。

[Report a bug](#)

11.6.2. Picketbox

Picketbox は、認証、許可、監査、およびマッピング機能を、JBoss EAP 6 で実行されている Java アプリケーションに提供する基礎的なセキュリティーフレームワークです。単一構成の単一フレームワークで、次の機能を提供します。

- 「[認証](#)」
- 「[承認](#)」 およびアクセス制御
- 「[セキュリティー監査](#)」
- 「[セキュリティーマッピング](#)」 のプリンシパル、ロール、および属性

[Report a bug](#)

11.6.3. 認証

認証とは、サブジェクトを識別し、身分が本物であるかを検証することを言います。最も一般的な認証メカニズムはユーザー名とパスワードの組み合わせです。その他の一般的な認証メカニズムは共有キー、スマートカード、または指紋などを使用します。Java Enterprise Edition の宣言的セキュリティーでは、成功した認証の結果のことをプリンシパルと呼びます。

JBoss EAP 6 は認証モジュールのプラグ可能なシステムを使用して、組織ですでに使用されている認証システムへ柔軟に対応し、統合を実現します。各セキュリティドメインには1つ以上の設定された認証モジュールが含まれます。各モジュールには、挙動をカスタマイズするための追加の設定パラメーターが含まれています。Web ベースの管理コンソール内に認証サブシステムを設定するのが最も簡単な方法です。

認証と承認が関連している場合も多くありますが、認証と承認は同じではありません。含まれている多くの認証モジュールは承認も処理します。

[Report a bug](#)

11.6.4. セキュリティドメインでの認証の設定

セキュリティドメインの認証を設定するには、管理コンソールにログインして以下の手順を実行します。

手順11.2 セキュリティドメインの認証設定

1. セキュリティドメインの詳細ビューを開きます。
 - a. 管理コンソールの上部にある **Configuration** ラベルをクリックします。
 - b. プロファイルビューの左上にある **Profile** 選択ボックスから編集するプロファイルを選択します。
 - c. **Security** メニューを展開し、**Security Domains** を選択します。
 - d. 編集したいセキュリティドメインの **View** リンクをクリックします。
2. 認証サブシステム設定に移動します。

ビューの上部にある **Authentication** ラベルを選択します (未選択である場合)。

設定領域が **Login Modules** と **Details** の2つの領域に分割されます。ログインモジュールは、設定の基本単位です。セキュリティドメインには複数のログインモジュールを含めることができ、各ログインモジュールには複数の属性とオプションを含めることができます。

3. 認証モジュールを追加します。

Add をクリックして、JAAS 認証モジュールを追加します。モジュールの詳細を入力します。

Code はモジュールのクラス名です。**Flag** 設定は、モジュールが同じセキュリティドメイン内で他の認証モジュールにどのように関係するかを制御します。

フラグの説明

Java Enterprise Edition 6 の仕様には、以下のようなセキュリティモジュールの説明があります。次のリスト

は、<http://docs.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html> に記載されています。詳細については、そのドキュメントを参照してください。

フラグ	説明
required	LoginModule は成功する必要があります。成功または失敗すると、LoginModule リストを下方に進み認証が続行されます。

フラグ	説明
requisite	LoginModule は成功する必要があります。成功すると、LoginModule リストの下方向に進み認証が続行されます。失敗すると、即座に制御がアプリケーションに戻されます (LoginModule リストの下方向に進まず、認証が続行されません)。
sufficient	LoginModule は成功する必要はありません。成功した場合は、即座に制御がアプリケーションに戻されます (LoginModule リストの下方向に進まず、認証が続行されません)。失敗すると、LoginModule リストの下方向に進み認証が続行されます。
optional	LoginModule は成功する必要はありません。成功または失敗した場合、LoginModule リストの下方向に進み認証が続行されます。

4. 認証設定を編集します。

モジュールを追加したら、画面の **Details** セクションで **Edit** をクリックすることにより、**Code** または **Flags** を変更できます。**Attributes** タブが選択されていることを確認します。

5. モジュールオプションを追加、編集、または削除します (任意)。

モジュールにオプションを追加する必要がある場合は、**Login Modules** リストのエントリーをクリックし、ページの **Details** セクションの **Module Options** タブを選択します。**Add** ボタンをクリックし、オプションのキーと値を指定します。**Remove** ボタンを使用してオプションを削除します。

結果

認証モジュールが、セキュリティドメインに追加され、セキュリティドメインを使用するアプリケーションですぐに利用できます。

jboss.security.security_domain モジュールのオプション

デフォルトでは、セキュリティドメインに定義された各ログインモジュールには **jboss.security.security_domain** モジュールオプションが自動的に追加されます。既知のオプションのみが定義されるようチェックを行うログインモジュールでは、このオプションによって問題が発生します。このようなログインモジュールの1つが IBM Kerberos ログインモジュールの **com.ibm.security.auth.module.Krb5LoginModule** です。

このモジュールオプションを追加する挙動を無効にするには、JBoss EAP 6 の起動時にシステムプロパティを **true** に設定します。以下を起動パラメーターに追加します。

```
-Djboss.security.disable.secdomain.option=true
```

Web ベースの管理コンソールを使用してこのプロパティを設定することも可能です。スタンドアロンサーバーでは、システムプロパティを設定の **Profile** セクションに設定できます。管理対象ドメインでは、各サーバーグループのシステムプロパティを設定できます。

[Report a bug](#)

11.6.5. 承認

承認とはアイデンティティーを基にリソースへのアクセスを許可または拒否するメカニズムのことです。プリンシパルに追加できる宣言的セキュリティロールのセットとして実装されます。

JBoss EAP 6 はモジュラーシステムを使用して承認を設定します。各セキュリティドメインに1つ以上の承認ポリシーが含まれるようにすることができます。各ポリシーには動作を定義する基本モジュールがあり、特定のフラグや属性より設定されます。Web ベースの管理コンソールを使用すると承認サブシステムを最も簡単に設定できます。

承認は認証とは異なり、通常は認証後に承認が行われます。認証モジュールの多くは承認も処理します。

[Report a bug](#)

11.6.6. セキュリティドメインにおける承認の設定

セキュリティドメインの承認を設定するには、管理コンソールにログインして以下の手順を実行します。

手順11.3 セキュリティドメインの承認設定

1. セキュリティドメインの詳細ビューを開きます。
 - a. 管理コンソールの上部にある **Configuration** ラベルをクリックします。
 - b. 管理対象ドメインでは、左上にある **Profile** ドロップダウンボックスで編集するプロファイルを選択します。
 - c. **Security** メニューを展開し、**Security Domains** を選択します。
 - d. 編集したいセキュリティドメインの **View** リンクをクリックします。

2. 承認サブシステム設定に移動します。

画面の上部にある **Authorization** ラベルを選択します。

設定領域が **Policies** と **Details** の2つの領域に分割されます。ログインモジュールは、設定の基本単位です。セキュリティドメインには複数の承認ポリシーを含めることができ、各ログインモジュールには複数の属性とオプションを含めることができます。

3. ポリシーを追加します。

Add をクリックして、JAAS 承認ポリシーモジュールを追加します。モジュールの詳細を入力します。

Code はモジュールのクラス名です。**Flag** は、モジュールが同じセキュリティドメイン内で他の承認ポリシーモジュールにどのように関係するかを制御します。

フラグの説明

Java Enterprise Edition 6 の仕様には、以下のようなセキュリティモジュールの説明があります。次のリスト

は、<http://docs.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html> に記載されています。詳細については、そのドキュメントを参照してください。

フラグ	説明
required	LoginModule は成功する必要があります。成功または失敗すると、LoginModule リストを下方方向に進み承認が続行されます。
requisite	LoginModule は成功する必要があります。成功すると、LoginModule リストの下方方向に進み承認が続行されます。失敗すると、即座に制御がアプリケーションに戻されます (LoginModule リストの下方方向に進まず、承認が続行されません)。
sufficient	LoginModule は成功する必要はありません。成功した場合は、即座に制御がアプリケーションに戻されます (LoginModule リストの下方方向に進まず、承認が続行されません)。失敗すると、LoginModule リストの下方方向に進み承認が続行されます。
optional	LoginModule は成功する必要はありません。成功または失敗した場合、LoginModule リストを下方方向に進み承認が続行されます。

4. 承認設定を編集します。

モジュールを追加したら、画面の **Details** セクションで **Edit** をクリックすることにより、**Code** または **Flags** を変更できます。**Attributes** タブが選択されていることを確認します。

5. モジュールオプションを追加、編集、または削除します (任意)。

モジュールにオプションを追加する必要がある場合は、**Policies** リストのエントリーをクリックし、ページの **Details** セクションの **Module Options** タブを選択します。**Add** をクリックし、オプションのキーと値を指定します。**Remove** ボタンを使用してオプションを削除します。

結果

承認ポリシーモジュールが、セキュリティドメインに追加され、セキュリティドメインを使用するアプリケーションですぐに利用できます。

[Report a bug](#)

11.6.7. セキュリティー監査

セキュリティー監査とは、セキュリティーサブシステム内で発生したイベントに応答するため、ログへの書き込みなどのイベントをトリガーすることです。監査のメカニズムは、認証、承認、およびセキュリティーマッピングの詳細と共に、セキュリティードメインの一部として設定されます。

監査にはプロバイダーモジュールが使用されます。含まれているプロバイダーモジュールを使用するか、独自のモジュールを実装することができます。

[Report a bug](#)

11.6.8. セキュリティー監査の設定

セキュリティードメインのセキュリティー監査を設定するには、管理コンソールにログインして以下の手順を実行します。

手順11.4 セキュリティードメインのセキュリティー監査の設定

1. セキュリティードメインの詳細ビューを開きます。

- a. 画面の上部にある **Configuration** をクリックします。
- b. 管理対象ドメインでは、左上にある **Profile** 選択ボックスから編集するプロファイルを選択します。
- c. **Security** メニューを展開し、**Security Domains** を選択します。
- d. 編集したいセキュリティードメインの **View** リンクをクリックします。

2. 監査サブシステム設定に移動します。

画面の上部にある **Audit** タブを選択します。

設定領域が **Provider Modules** と **Details** の2つの領域に分割されます。プロバイダーモジュールは、設定の基本単位です。セキュリティードメインには複数のプロバイダーモジュールを含めることができ、各ログインモジュールには属性とオプションを含めることができます。

3. プロバイダーモジュールを追加します。

Add をクリックします。**Code** セクションに、プロバイダーモジュールのクラス名を入力します。

4. モジュールの挙動を確認します。

監査モジュールの目的は、セキュリティーサブシステムのイベントを監視する方法を提供することです。ログファイルの記述、メールによる通知、またはその他の監査メカニズムによって監視を行うことが可能です。

たとえば、JBoss EAP 6 にはデフォルトで **LogAuditProvider** モジュールが含まれています。この監査モジュールを前述の手順に従って有効にすると、**EAP_HOME** ディレクトリー内の **log** サブフォルダーにある **audit.log** ファイルにセキュリティー通知を書き込みます。

前述の手順が **LogAuditProvider** のコンテキスト内で動作するかを確認するには、通知が発生するようなアクションを実行した後、監査ログファイルをチェックします。

含まれるセキュリティー監査プロバイダーモジュールの完全一覧は [「含まれるセキュリティー監査プロバイダーモジュール」](#) を参照してください。

5. モジュールオプションを追加、編集、または削除します (任意)。

モジュールにオプションを追加する場合は、**Policies** リストのエントリーをクリックし、ページの **Details** セクションの **Module Options** タブを選択します。**Add** をクリックし、オプションのキーと値を指定します。

すでに存在するオプションを編集するには、**Remove** をクリックして削除し、**Add** をクリックして正しいオプションで再度追加します。

結果

セキュリティー監査モジュールは、セキュリティードメインに追加され、セキュリティードメインを使用するアプリケーションですぐに利用できます。

[Report a bug](#)

11.6.9. 監査ログ

LogAuditProvider モジュールが有効な場合、アプリケーションおよびログインモジュールでの認証および承認を記録する監査ログが JBoss EAP 6 によって維持されます。デフォルトでは、このファイルの名前は **audit.log** になり、*EAP_HOME* ディレクトリーの **log** サブディレクトリーに保存されます。挙動は、ロギングサブシステムのログハンドラーの設定によって決定されます。**syslog** ログハンドラーを使用すると、ファイルの代わりまたはファイルとともに **LogAuditProvider** モジュールの出力を **syslog** サーバーに送信できます。

デフォルトでは、**LogAuditProvider** モジュールは累積の **audit.log** ファイルへのみ出力します。**AUDIT** と呼ばれる定期的にローテーションを行うファイルハンドラーを実装するには、以下の管理 CLI コマンドを入力します。

```
/subsystem=logging/periodic-rotating-file-handler=AUDIT/:add(suffix=.yyyy-MM-dd,formatter=%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n,level=TRACE,file={"relative-to" => "jboss.server.log.dir","path" => "audit.log"})
```

関連トピック:

- 「ログハンドラー」

[Report a bug](#)

11.6.10. セキュリティーマッピング

セキュリティーマッピングを使用すると、認証または承認が実行された後、情報がアプリケーションに渡される前に認証情報と承認情報を組み合わせることができます。

プリンシパル (認証)、ロール (承認)、またはクレデンシャル (プリンシパルやロールでない属性) をマッピングすることが可能です。

ロールマッピングは、認証後にサブジェクトへロールを追加、置換、または削除するために使用されます。

プリンシパルマッピングは、認証後にプリンシパルを変更するために使用されます。

属性マッピングは、外部システムからの属性値をアプリケーションで使用するために変換したり、逆にそのような外部システムへ属性を変換したりするために使用されます。

[Report a bug](#)

11.6.11. セキュリティードメインでのセキュリティーマッピングの設定

セキュリティードメインのセキュリティーマッピングを設定するには、管理コンソールにログインして以下の手順を実行します。

手順11.5 セキュリティードメインでのセキュリティーマッピングの設定

1. セキュリティードメインの詳細ビューを開きます。
 - a. 管理コンソールの上部にある **Configuration** ラベルをクリックします。
 - b. 管理対象ドメインでは、左上にある **Profile** 選択ボックスからプロファイルを選択します。
 - c. **Security** メニューを展開し、**Security Domains** を選択します。

d. 編集したいセキュリティードメインの **View** リンクをクリックします。

2. マッピングサブシステム設定に移動します。

画面の上部にある **Mapping** ラベルを選択します。

設定領域が **Modules** と **Details** の2つの領域に分割されます。マッピングモジュールは、設定の基本単位です。セキュリティードメインには複数のマッピングモジュールを含めることができ、各ログインモジュールには複数の属性とオプションを含めることができます。

3. セキュリティーマッピングモジュールを追加します。

Add をクリックします。

モジュールの詳細を記入します。**Code** がモジュールのクラス名です。**Type** フィールドは、このモジュールが実行するマッピングのタイプを示します。許可される値は、principal、role、attribute、または credential です。

4. セキュリティーマッピングモジュールを編集します。

モジュールを追加したら、**Code** または **Type** を変更できます。

a. **Attributes** タブを選択します。

b. 画面の **Details** セクションにある **Edit** をクリックします。

5. モジュールオプションを追加、編集、または削除します (任意)。

モジュールにオプションを追加する場合は、**Policies** リストのエントリーをクリックし、ページの **Details** セクションの **Module Options** タブを選択します。**Add** をクリックし、オプションのキーと値を指定します。

すでに存在するオプションを編集するには、**Remove** をクリックして削除し、新たな値で再度追加します。

Remove ボタンを使用して、オプションを削除します。

結果

セキュリティーマッピングモジュールは、セキュリティードメインに追加され、セキュリティードメインを使用するアプリケーションですぐに利用できます。

[Report a bug](#)

11.6.12. アプリケーションでのセキュリティードメインの使用

概要

アプリケーションでセキュリティードメインを使用するには、最初にサーバーの設定でセキュリティードメインを定義し、アプリケーションのデプロイメント記述子でアプリケーションに対して有効にする必要があります。その後、使用する EJB に必要なアノテーションを追加する必要があります。ここでは、アプリケーションでセキュリティードメインを使用するために必要な手順について取り上げます。



警告

アプリケーションが、認証キャッシュを使用するセキュリティードメインの一部である場合、セキュリティードメインの他のアプリケーションもそのアプリケーションのユーザー認証を使用できます。

手順11.6 セキュリティードメインを使用するようアプリケーションを設定

1. セキュリティードメインの定義

サーバーの設定ファイルでセキュリティードメインを定義した後、アプリケーションの記述子ファイルでアプリケーションに対して有効にする必要があります。

a. サーバーの設定ファイルへセキュリティードメインを設定

セキュリティードメインは、サーバーの設定ファイルの **security** サブシステムに設定されます。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、**domain/configuration/domain.xml** ファイルになります。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は **standalone/configuration/standalone.xml** ファイルになります。

other、**jboss-web-policy**、および **jboss-ejb-policy** セキュリティードメインはデフォルトとして JBoss EAP 6 に提供されます。次の XML の例は、サーバーの設定ファイルの **security** サブシステムよりコピーされました。

セキュリティードメインの **cache-type** 属性は、高速な認証チェックを行うためにキャッシュを指定します。簡単なマップをキャッシュとして使用する **default** か、Infinispan キャッシュを使用する **infinispan** を値として使用できます。

```
<subsystem xmlns="urn:jboss:domain:security:1.2">
  <security-domains>
    <security-domain name="other" cache-type="default">
      <authentication>
        <login-module code="Remoting" flag="optional">
          <module-option name="password-stacking" value="useFirstPass"/>
        </login-module>
        <login-module code="RealmDirect" flag="required">
          <module-option name="password-stacking" value="useFirstPass"/>
        </login-module>
      </authentication>
    </security-domain>
    <security-domain name="jboss-web-policy" cache-type="default">
      <authorization>
        <policy-module code="Delegating" flag="required"/>
      </authorization>
    </security-domain>
    <security-domain name="jboss-ejb-policy" cache-type="default">
      <authorization>
        <policy-module code="Delegating" flag="required"/>
      </authorization>
    </security-domain>
  </security-domains>
</subsystem>
```

管理コンソールまたは CLI を使用して、追加のセキュリティドメインを必要に応じて設定できます。

b. アプリケーションの記述子ファイルでのセキュリティドメインの有効化

セキュリティドメインはアプリケーションの **WEB-INF/web.xml** ファイルにある **<jboss-web>** 要素の **<security-domain>** 子要素に指定されます。次の例は **my-domain** という名前のセキュリティドメインを設定します。

```
<jboss-web>
  <security-domain>my-domain</security-domain>
</jboss-web>
```

これが **WEB-INF/jboss-web.xml** 記述子に指定できる多くの設定の1つになります。

2. EJB への必要なアノテーションの追加

@SecurityDomain および **@RolesAllowed** アノテーションを使用してセキュリティーを EJB に設定します。次の EJB コードの例は、**guest** ロールのユーザーによる **other** セキュリティードメインへのアクセスを制限します。

```
package example.ejb3;

import java.security.Principal;

import javax.annotation.Resource;
import javax.annotation.security.RolesAllowed;
import javax.ejb.SessionContext;
import javax.ejb.Stateless;

import org.jboss.ejb3.annotation.SecurityDomain;

/**
 * Simple secured EJB using EJB security annotations
 * Allow access to "other" security domain by users in a "guest" role.
 */
@Stateless
@RolesAllowed({ "guest" })
@SecurityDomain("other")
public class SecuredEJB {

    // Inject the Session Context
    @Resource
    private SessionContext ctx;

    /**
     * Secured EJB method using security annotations
     */
    public String getSecurityInfo() {
        // Session context injected using the resource annotation
        Principal principal = ctx.getCallerPrincipal();
        return principal.toString();
    }
}
```

その他のコード例は、Red Hat カスタマーポータルより入手できる JBoss EAP 6 Quickstarts パンドルの **ejb-security** クイックスタートを参照してください。

[Report a bug](#)

11.6.13. JACC (Java Authorization Contract for Containers)

11.6.13.1. JACC (Java Authorization Contract for Containers)

JACC (Java Authorization Contract for Containers) はコンテナと承認サービスプロバイダー間のインターフェースを定義する規格で、これによりコンテナによって使用されるプロバイダーの実装が可能になります。JACC は JSR-115 に定義されており、<http://jcp.org/en/jsr/detail?id=115> の Java Community Process Web サイトで確認できます。Java EE バージョン 1.3 より、コアの Java Enterprise Edition (Java EE) 仕様の一部となっています。

JBoss EAP 6 はセキュリティーサブシステムのセキュリティー機能内に JACC のサポートを実装します。

[Report a bug](#)

11.6.13.2. JACC (Java Authorization Contract for Containers) のセキュリティーの設定

JACC (Java Authorization Contract for Containers) を設定するには、適切なモジュールでセキュリティードメインを設定し、適切なパラメーターが含まれるよう **jboss-web.xml** を編集する必要があります。

セキュリティードメインへの JACC サポートの追加

セキュリティードメインに JACC サポートを追加するには、**required** フラグセットで **JACC** 承認ポリシーをセキュリティードメインの承認スタックへ追加します。以下は JACC サポートを持つセキュリティードメインの例になりますが、セキュリティードメインは直接 XML には設定されず、管理コンソールまたは管理 CLI で設定されます。

```
<security-domain name="jacc" cache-type="default">
  <authentication>
    <login-module code="UsersRoles" flag="required">
    </login-module>
  </authentication>
  <authorization>
    <policy-module code="JACC" flag="required"/>
  </authorization>
</security-domain>
```

JACC を使用するよう Web アプリケーションを設定

jboss-web.xml はデプロイメントの **WEB-INF/** ディレクトリーに存在し、Web コンテナに対する追加の JBoss 固有の設定を格納し、上書きします。JACC が有効になっているセキュリティードメインを使用するには、**<security-domain>** 要素が含まれるようにし、さらに **<use-jboss-authorization>** 要素を **true** に設定する必要があります。以下は、上記の JACC セキュリティードメインを使用するよう適切に設定されているアプリケーションになります。

```
<jboss-web>
  <security-domain>jacc</security-domain>
  <use-jboss-authorization>true</use-jboss-authorization>
</jboss-web>
```

JACC を使用するよう EJB アプリケーションを設定

セキュリティードメインと JACC を使用するよう EJB を設定する方法は Web アプリケーションの場合

とは異なります。EJB の場合、**ejb-jar.xml** 記述子にてメソッドまたはメソッドのグループ上でメソッドパーミッションを宣言できます。**<ejb-jar>** 要素内では、**<method-permission>** 要素のすべての子に JACC ロールに関する情報が含まれます。詳細は設定例を参照してください。**EJBMethodPermission** クラスは Java Enterprise Edition 6 API の一部で、<http://docs.oracle.com/javaee/6/api/javax/security/jacc/EJBMethodPermission.html> で説明されています。

例11.4 EJB の JACC メソッドパーミッション例

```
<ejb-jar>
  <assembly-descriptor>
    <method-permission>
      <description>The employee and temp-employee roles may access any method of the
EmployeeService bean </description>
      <role-name>employee</role-name>
      <role-name>temp-employee</role-name>
      <method>
        <ejb-name>EmployeeService</ejb-name>
        <method-name>*</method-name>
      </method>
    </method-permission>
  </assembly-descriptor>
</ejb-jar>
```

Web アプリケーションと同様にセキュリティドメインを使用して EJB の認証および承認メカニズムを指定することも可能です。セキュリティドメインは **<security>** 子要素の **jboss-ejb3.xml** 記述子に宣言されます。セキュリティドメインの他に、EJB が実行されるプリンシパルを変更する *run-as* プリンシパルを指定することもできます。

例11.5 EJB におけるセキュリティドメイン宣言の例

```
<ejb-jar>
  <assembly-descriptor>
    <security>
      <ejb-name>*</ejb-name>
      <security-domain>myDomain</security-domain>
      <run-as-principal>myPrincipal</run-as-principal>
    </security>
  </assembly-descriptor>
</ejb-jar>
```

[Report a bug](#)

11.6.14. JASPI (Java Authentication SPI for Containers)

11.6.14.1. JASPI (Java Authentication SPI for Containers) のセキュリティー

Java Authentication SPI for Containers (JASPI または JASPIC) は Java アプリケーションのプラグ可能なインターフェースです。Java Community Process の JSR-196 に定義されています。この仕様の詳細は <http://www.jcp.org/en/jsr/detail?id=196> を参照してください。

[Report a bug](#)

11.6.14.2. JASPI (Java Authentication SPI for Containers) のセキュリティーの設定

JASPI プロバイダーに対して認証するには、**<authentication-jaspi>** 要素をセキュリティードメインに追加します。設定は標準的な認証モジュールと似ていますが、ログインモジュール要素は **<login-module-stack>** 要素で囲まれています。設定の構成は次のとおりです。

例11.6 authentication-jaspi 要素の構成

```
<authentication-jaspi>
  <login-module-stack name="...">
    <login-module code="..." flag="...">
      <module-option name="..." value="..."/>
    </login-module>
  </login-module-stack>
  <auth-module code="..." login-module-stack-ref="...">
    <module-option name="..." value="..."/>
  </auth-module>
</authentication-jaspi>
```

ログインモジュール自体は標準的な認証モジュールと全く同じように設定されます。

Web ベースの管理コンソールは JASPI 認証モジュールの設定を公開しないため、JBoss EAP 6 を完全に停止してから、設定を **EAP_HOME/domain/configuration/domain.xml** または **EAP_HOME/standalone/configuration/standalone.xml** へ直接追加する必要があります。

[Report a bug](#)

11.7. IIOP のセキュア化

11.7.1. JBoss IIOP

IIOP (Internet Inter-ORB Protocol) は CORBA サーバー上でリモート機能呼び出すために CORBA クライアントによって使用される通信プロトコルです。JBoss IIOP は、EJB 仕様によって定義されしており、JBoss EAP サーバー上にデプロイされたエンタープライズ Bean への CORBA/IIOP アクセスをサポートします。これにより、Java で書かれた RMI および IIOP クライアントや、Java、C++、およびその他の言語で書かれた CORBA クライアントがエンタープライズ Bean メソッドを利用できます。IIOP エンジン は CORBA に完全対応する他のエンジンに置き換えることができます。デフォルトのエンジンは、CORBA 標準の無料の実装である JacORB です。

[Report a bug](#)

11.7.2. IOR

IOR (Interoperable Object Reference) は、オブジェクトを識別するために CORBA システムで使用されます。IOR は以下で構成されます。

- オブジェクトタイプ
- サーバーのホスト名
- サーバーのポート番号

- オブジェクトを識別するオブジェクトキー

ホスト名とポート番号は、サーバーと通信するために使用されます。オブジェクトキーは、オブジェクトを識別するためサーバーによって使用されます。

[Report a bug](#)

11.7.3. IOR セキュリティーパラメーター

前提条件

- 以下の管理 CLI コマンドで IOR 設定を有効にします。

```
/subsystem=jacorb/ior-settings=default:add
```

- JacORB サブシステムが有効になっていることを確認します。たとえば、**full** プロファイルではすでに有効になっていますが、デフォルトのプロファイル (**web**) では有効になっていません。JacORB サブシステムを有効にする方法の詳細は、「[JTS トランザクション用 ORB の設定](#)」を参照してください。

iorSASContextType は、IOR セキュア属性サービスを設定するために使用される属性を指定します。これらのパラメーターを設定するには、管理 CLI を使用する必要があります。

表11.1 iorSASContextType

パラメーター	説明	有効な値
caller-propagation	呼び出し側が SAS コンテキストで伝播されるべきかどうかを示します	none、supported

```
/subsystem=jacorb/ior-settings=default/setting=sas-context:add
/subsystem=jacorb/ior-settings=default/setting=sas-context:write-attribute(name=caller-propagation,
value=NONE/SUPPORTED)
```

iorASContextType は、IOR 認証サービスを設定するために使用される属性を指定します。以下のパラメーターは任意です。

表11.2 iorASContextType

パラメーター	説明	タイプ	有効な値
auth-method	認証メソッド	文字列	none、username_password
realm	認証サービスのレルム名	文字列	デフォルト値: Default
required	認証が必要であるかどうかを示します。	ブール値	true、false


```
/subsystem=jacorb/ior-settings=default/setting=as-context:add
/subsystem=jacorb/ior-settings=default/setting=as-context:write-attribute(name=ATTRIBUTE,
value=VALUE)
```

iorTransportconfigType は、IOR トランスポートを設定するために使用される属性を指定します。

表11.3 iorTransportconfigType

パラメーター	説明	有効な値
integrity	トランスポートに整合性の保護が必要であるかどうかを示します。	none 、 supported 、または required
confidentiality	トランスポートに機密性の保護が必要であるかどうかを示します。	none 、 supported 、または required
trust-in-target	トランスポートが trust-in-target を必要とするかどうかを示します。	none 、 supported
trust-in-client	トランスポートが trust-in-client を必要とするかどうかを示します。	none 、 supported 、または required
detect-replay	トランスポートにリプレイの検出を必要とするかどうかを示します。	none 、 supported 、または required
detect-misordering	トランスポートに誤順序の検出を必要とするかどうかを示します。	none 、 supported 、または required

```
/subsystem=jacorb/ior-settings=default/setting=transport-config:add
/subsystem=jacorb/ior-settings=default/setting=transport-config:write-attribute(name=ATTRIBUTE,
value=VALUE)
```

[Report a bug](#)

11.8. 管理インターフェースセキュリティ

11.8.1. デフォルトのユーザーセキュリティ設定

はじめに

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

- ローカルインターフェースは、ローカルクライアントとローカルクライアントが接続するサーバーとの間の SASL コントラクトによって保護されます。このセキュリティメカニズムは、ローカルファイルシステムにアクセスするクライアントの機能に基づきます。ローカルシステムへアクセスできるとクライアントによるユーザーの追加が可能で、他のセキュリティメカ

ニズムを無効にするよう設定を変更できるからです。これにより、ファイルシステムへ物理的にアクセスできると、他のセキュリティーメカニズムが不要になるという原則が厳守されます。このメカニズムは4つの手順で実現されます。



注記

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

1. ローカル SASL メカニズムを用いて認証する要求が含まれるメッセージをクライアントがサーバーに送信します。
 2. サーバーは1度限りのトークンを生成して、固有のファイルに書き込み、そのファイルのフルパスが含まれるメッセージをクライアントに送信します。
 3. クライアントはファイルよりトークンを読み取り、サーバーへ送信し、ファイルシステムへローカルアクセスできるかを検証します。
 4. サーバーはトークンを検証し、ファイルを削除します。
- ローカル HTTP クライアントを含むリモートクライアントはレルムベースのセキュリティーを使用します。管理インターフェースを使用して JBoss EAP 6 インスタンスをリモートで設定するパーミッションを持つデフォルトのレルムは **ManagementRealm** です。このレルム (またはユーザーが作成したレルム) にユーザーを追加できるスクリプトが提供されます。ユーザーごとに、ユーザー名とハッシュ化されたパスワードがファイルに保存されます。

管理対象ドメイン

EAP_HOME/domain/configuration/mgmt-users.properties

スタンドアロンサーバー

EAP_HOME/standalone/configuration/mgmt-users.properties

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

[Report a bug](#)

11.8.2. 管理インターフェースの詳細設定の概要

EAP_HOME/domain/configuration/host.xml または **EAP_HOME/standalone/configuration/standalone.xml** の管理インターフェース設定は、ホストコントロールロープロセスのバインド先となるネットワークインターフェース、利用可能な管理インターフェースのタイプ、および各インターフェースのユーザーを認証するために使用される認証システムのタイプを制御します。本トピックでは、ご使用の環境に合わせて管理インターフェースを設定する方法について説明します。

管理サブシステムは、以下の4つの設定可能な子要素が含まれる **<management>** 要素で構成されます。セキュリティーレルムと送信接続はそれぞれ最初に定義されてから、管理インターフェースに属性として適用されます。

- **<security-realms>**
- **<outbound-connections>**

- <management-interfaces>
- <audit-log>



注記

監査ロギングの詳細は、『管理および設定ガイド』の「管理インターフェース監査ロギング」の項を参照してください。

セキュリティーレルム

セキュリティーレルムは、管理 API、管理 CLI、または Web ベースの管理コンソールを介して JBoss EAP 6 の管理を許可されているユーザーの認証と承認を行います。

デフォルトのインストールに含まれる 2 つの異なるファイルベースのセキュリティーレルムは **ManagementRealm** と **ApplicationRealm** です。これらのセキュリティーレルムはそれぞれ - **users.properties** ファイルを使用してユーザーおよびハッシュ化されたパスワードを保存し、- **roles.properties** を使用してユーザーとロール間のマッピングを保存します。サポートは LDAP 対応のセキュリティーレルムにも含まれています。



注記

独自のアプリケーションにセキュリティーレルムを使用することも可能です。このトピックで説明するセキュリティーレルムは管理インターフェース固有のものです。

送信接続

一部のセキュリティーレルムは、LDAP サーバーなどの外部インターフェースに接続します。送信接続は、この接続の確立方法を定義します。事前に定義された接続タイプ **ldap-connection** は、LDAP サーバーに接続してクレデンシャルを検証するための必須およびオプションの属性をすべて設定します。

LDAP 認証の設定方法の詳細は、「[管理インターフェースに対する LDAP を使用した認証](#)」を参照してください。

管理インターフェース

管理インターフェースには、JBoss EAP の接続および設定方法に関するプロパティーが含まれています。この情報には、名前付きのネットワークインターフェース、ポート、セキュリティーレルム、およびインターフェースに関するその他の設定可能な情報が含まれます。デフォルトのインストールには 2 つのインターフェースが含まれています。

- **http-interface** は Web ベースの管理コンソールの設定です。
- **native-interface** はコマンドライン管理 CLI および REST ライクな管理 API の設定です。

ホスト管理サブシステムの設定可能要素はそれぞれ関連しています。セキュリティーレルムは送信接続を参照し、管理インターフェースはセキュリティーレルムを参照します。

関連情報は「[管理インターフェースのセキュア化](#)」を参照してください。

[Report a bug](#)

11.8.3. LDAP

LDAP (Lightweight Directory Access Protocol) はネットワーク全体でディレクトリー情報を格納および分散するプロトコルです。このディレクトリー情報には、ユーザー、ハードウェアデバイス、アクセスロール、制限に関する情報などが含まれます。

LDAP の一般的な実装には OpenLDAP、Microsoft Active Directory、IBM Tivoli Directory Server、Oracle Internet Directory などがあります。

JBoss EAP 6 には、Web アプリケーションや EJB アプリケーションの認証承認権限として LDAP サーバーを使用できるようにする複数の認証および承認モジュールがあります。

[Report a bug](#)

11.8.4. 管理インターフェースに対する LDAP を使用した認証

管理コンソール、管理 CLI、管理 API の認証ソースとして LDAP ディレクトリーサーバーを使用するには、以下の手順を実行する必要があります。

1. LDAP サーバーへアウトバウンド接続を作成します。
2. LDAP 対応のセキュリティーレルムを作成します。
3. 管理インターフェースの新規セキュリティードメインを参照します。

LDAP サーバーへの送信接続の作成

LDAP 送信接続には、以下の属性を使用することができます。

表11.4 LDAP 送信接続の属性

属性	必須	説明
url	必要	ディレクトリーサーバーの URL アドレス。
search-dn	不要	検索の実行を許可されているユーザーの完全識別名 (DN)
search-credentials	不要	検索の実行を許可されているユーザーのパスワード。
initial-context-factory	不要	接続を確立する際に使用する初期コンテキストファクトリー。デフォルトでは com.sun.jndi.ldap.LdapCtxFactory に設定されています。
security-realm	不要	接続を確立するときに、使用する設定済みの SSLContext を取得するために参照するセキュリティーレルム。

例11.7 LDAP 送信接続の追加

この例では、以下のプロパティーセットを使用して送信接続を追加します。

- DN の検索: **cn=search,dc=acme,dc=com**

- 証明情報の検索: **myPass**
- URL: **ldap://127.0.0.1:389**

最初のコマンドによってセキュリティーレルムが追加されます。

```
/host=master/core-service=management/security-realm=ldap_security_realm:add
```

2 つ目のコマンドによって、LDAP 接続が追加されます。

```
/host=master/core-service=management/ldap-connection=ldap_connection/:add(search-credential=myPass,url=ldap://127.0.0.1:389,search-dn="cn=search,dc=acme,dc=com")
```

LDAP 対応セキュリティーレルムの作成

管理インターフェースは、デフォルトで設定されているプロパティファイルをベースとするセキュリティーレルムの代わりに LDAP サーバーに対して認証を行うことができます。LDAP のオーセンティケーターは、最初にリモートディレクトリーサーバーへの接続を確立します。次に、LDAP レコードの完全修飾識別名 (DN) を探すため、ユーザーが認証システムに渡したユーザー名を使用して検索を実行します。クレデンシャルとしてユーザーの DN とユーザー提供のパスワードを使用して、新規接続が確立されます。この LDAP サーバーに対するこの検証に成功すると、DN が有効であることが検証されます。

LDAP セキュリティーレルムは次の設定属性を使用します。

connection

LDAP ディレクトリーへ接続するために使用される **outbound-connections** に定義されている接続の名前。

advanced-filter

指定のユーザー ID を基にユーザーを検索するのに使用される完全に定義されたフィルター。フィルターには **{0}** という形式の変数が含まれている必要があります。これは、後でユーザー指定のユーザー名に置き換えられます。

base-dn

ユーザー検索を開始するためのコンテキストの識別名

recursive

LDAP ディレクトリーツリー全体にわたって再帰的に検索を行うか、指定のコンテキストのみを検索するかを指定します。デフォルトでは **false** に設定されています。

user-dn

識別名を保持するユーザーの属性。これは、後で認証のテストに使用されます。デフォルトでは **dn** に設定されています。

username-attribute

ユーザーを検索するための属性の名前。このフィルターは、ユーザーによって入力されたユーザー名が指定の属性と一致する簡単な検索を行います。

allow-empty-passwords

この属性は、空白のパスワードが許可されるかどうかを決定します。この属性のデフォルト値は **false** です。

username-filter または **advanced-filter** を指定する必要があります。

advanced-filter 属性には、標準の LDAP 構文のフィルタークエリーが含まれています。例を以下に示します。

```
(&(sAMAccountName={0})(memberOf=cn=admin,cn=users,dc=acme,dc=com))
```

例11.8 LDAP 対応のセキュリティーレームを示す XML

この例には、以下のパラメーターを使用します。

- connection - **ldap_connection**
- base-dn - **cn=users,dc=acme,dc=com.**
- username-filter - **attribute="sambaAccountName"**

```
<security-realm name="ldap_security_realm">
  <authentication>
    <ldap connection="ldap_connection" base-dn="cn=users,dc=acme,dc=com">
      <username-filter attribute="sambaAccountName" />
    </ldap>
  </authentication>
</security-realm>
```



警告

空の LDAP パスワードを許可しないようにすることが重要になります。ご使用の環境で具体的に空の LDAP パスワードを許可したい場合を除き、深刻なセキュリティー上の問題となります。

EAP 6.1 には CVE-2012-5629 のパッチが含まれています。このパッチは、LDAP ログインモジュールの allowEmptyPasswords オプションが設定されていない場合にこのオプションを False に設定します。6.1 以前のバージョンでは、このオプションを手作業で設定する必要があります。

例11.9 LDAP セキュリティーレームの追加

以下のコマンドは、LDAP 認証をセキュリティーレームに追加し、ドメインの master という名前のホストに対して属性を設定します。

```
/host=master/core-service=management/security-
realm=ldap_security_realm/authentication=ldap:add(base-dn="DC=mycompany,DC=org",
recursive=true, username-attribute="MyAccountName", connection="ldap_connection")
```

管理インターフェースへの新規セキュリティーレلمの適用

セキュリティーレلمの作成が完了したら、管理インターフェースの設定でそのドメインを参照する必要があります。管理インターフェースは、HTTP ダイジェスト認証用のセキュリティーレلمを使用します。

例11.10 HTTP インターフェースへのセキュリティーレلمの適用

この設定が有効になり、ホストコントローラーを再起動した後、Web ベースの管理コンソールは LDAP を使用してユーザーの認証を行います。

```
/host=master/core-service=management/management-interface=http-interface/:write-attribute(name=security-realm,value=ldap_security_realm)
```

例11.11 セキュリティーレلمのネイティブインターフェースへの適用

以下のコマンドを使用して同じ設定をネイティブインターフェースに適用します。

```
/host=master/core-service=management/management-interface=native-interface/:write-attribute(name=security-realm,value=ldap_security_realm)
```

[Report a bug](#)

11.8.5. HTTP 管理インターフェースの無効化

管理対象ドメインでは、ドメインメンバーのサーバーではなく、ドメインコントローラー上の HTTP インターフェースへのアクセスのみが必要です。また、実稼働サーバー上では、Web ベースの管理コンソールを完全に無効化することが可能です。

注記

JBoss Operations Network などの他のクライアントも HTTP インターフェースを使用し稼働します。これらのサービスを使用したり、管理コンソール自体を無効にしたい場合は、インターフェースを完全に無効化する代わりに HTTP インターフェースの **console-enabled** 属性を **false** に設定できます。

```
/host=master/core-service=management/management-interface=http-interface/:write-attribute(name=console-enabled,value=false)
```

HTTP インターフェースへのアクセスを無効にすると、Web ベースの管理コンソールへのアクセスも無効になるため、HTTP インターフェースを完全に削除して無効化できます。

次の JBoss CLI コマンドを使用すると、再度追加する場合に備えて HTTP インターフェースの現在の内容を読み込むことができます。

例11.12 HTTP インターフェースの設定の読み込み

```
/host=master/core-service=management/management-interface=http-interface/:read-resource(recursive=true,proxies=false,include-runtime=false,include-defaults=true)
```



```
{
  "outcome" => "success",
  "result" => {
    "console-enabled" => true,
    "interface" => "management",
    "port" => expression "${jboss.management.http.port:9990}",
    "secure-port" => undefined,
    "security-realm" => "ManagementRealm"
  }
}
```

HTTP インターフェースを削除するには、次のコマンドを実行します。

例11.13 HTTP インターフェースの削除

```
/host=master/core-service=management/management-interface=http-interface/:remove
```

アクセスを再度有効化するには、以下のコマンドを実行し、デフォルト値を使用して HTTP インターフェースを再作成します。

例11.14 HTTP インターフェースの再作成

```
/host=master/core-service=management/management-interface=http-interface:add(console-enabled=true,interface=management,port="${jboss.management.http.port:9990}",security-realm=ManagementRealm)
```

[Report a bug](#)

11.8.6. デフォルトセキュリティーレームからのサイレント認証の削除

概要

JBoss EAP 6 には、ローカル管理 CLI ユーザーに対するサイレント認証方法が含まれます。これにより、ローカルユーザーは、ユーザー名またはパスワード認証なしで管理 CLI にアクセスできるようになります。この機能は、利便性のために有効であり、ローカルユーザーが認証なしで管理 CLI スクリプトを実行する場合に役に立ちます。この機能は、ローカル設定へのアクセスにより、ユーザーが独自のユーザー詳細を追加できる (または、セキュリティーチェックを無効にする) ため、役に立つ機能です。

ローカルユーザーのサイレント認証は便利ですが、さらに強力なセキュリティー制御が必要な場合に無効にできます。これは、設定ファイルの **security-realm** セクション内で **local** 要素を削除することにより、実現できます。これは、スタンドアロンサーバーインスタンス用の **standalone.xml** と管理対象ドメイン用の **host.xml** の両方に適用されます。特定のサーバー設定への影響について理解している場合のみ、**local** 要素の削除を考慮するようにしてください。

推奨されるサイレント認証の削除方法は、管理 CLI を使用して、次の例に示された **local** 要素を直接削除することです。

例11.15 security-realm の local 要素の例

```
<security-realms>
```



```

<security-realm name="ManagementRealm">
  <authentication>
    <local default-user="$local"/>
    <properties path="mgmt-users.properties" relative-to="jboss.server.config.dir"/>
  </authentication>
</security-realm>
<security-realm name="ApplicationRealm">
  <authentication>
    <local default-user="$local" allowed-users="*/>
    <properties path="application-users.properties" relative-to="jboss.server.config.dir"/>
  </authentication>
  <authorization>
    <properties path="application-roles.properties" relative-to="jboss.server.config.dir"/>
  </authorization>
</security-realm>
</security-realms>

```

前提条件

- [「JBoss EAP 6 の起動」](#)
- [「管理 CLI の起動」](#)

手順11.7 デフォルトセキュリティーレームからのサイレント認証の削除

- 管理 CLI でのサイレント認証の削除
必要に応じて、管理レームとアプリケーションレームから **local** 要素を削除します。

a. 管理レームから **local** 要素を削除します。

■ スタンドアロンの場合

```
/core-service=management/security-
realm=ManagementRealm/authentication=local:remove
```

■ 管理対象ドメインの場合

```
/host=HOST_NAME/core-service=management/security-
realm=ManagementRealm/authentication=local:remove
```

b. アプリケーションレームから **local** 要素を削除します。

■ スタンドアロンの場合

```
/core-service=management/security-
realm=ApplicationRealm/authentication=local:remove
```

■ 管理対象ドメインの場合

```
/host=HOST_NAME/core-service=management/security-
realm=ApplicationRealm/authentication=local:remove
```

結果

サイレント認証モードが、**ManagementRealm** と **ApplicationRealm** から削除されます。

[Report a bug](#)

11.8.7. JMX サブシステムへのリモートアクセスの無効化

JMX サブシステムにリモートアクセスすると、JDK およびアプリケーション管理操作をリモートでトリガーすることができます。インストールをセキュア化するには、リモート接続コネクタまたは JMX サブシステムを削除してこの機能を無効化します。例の管理 CLI コマンドは管理対象ドメインを対象としています。スタンドアロンサーバーの場合はコマンドから **/profile=default** 接頭辞を削除してください。



注記

管理対象ドメインでは、リモート処理コネクタはデフォルトで JMX サブシステムから削除されています。以下のコマンドは、開発中にリモート処理コネクタを追加した場合に備えて、参考のために記載します。

例11.16 JMX サブシステムからのリモートコネクタの削除

```
/profile=default/subsystem=jmx/remoting-connector=jmx/:remove
```

例11.17 JMX サブシステムの削除

管理対象ドメインの場合は、各プロファイルに対してこのコマンドを実行します。

```
/profile=default/subsystem=jmx/:remove
```

[Report a bug](#)

11.8.8. 管理インターフェースのセキュリティーレームの設定

管理インターフェースはセキュリティーレームを使用して JBoss EAP 6 の認証および設定メカニズムへのアクセスを制御します。セキュリティーレームは Unix グループと似ている、ユーザーの認証に使用できるユーザーとパスワードのデータベースです。

デフォルトの管理レーム

デフォルトでは、管理インターフェースは **ManagementRealm** セキュリティーレームを使用するように設定されています。ManagementRealm はユーザーとパスワードの組み合わせを **mgmt-users.properties** ファイルに格納します。

例11.18 デフォルトの ManagementRealm

```
/host=master/core-service=management/security-realm=ManagementRealm/:read-resource(recursive=true,proxies=false,include-runtime=false,include-defaults=true)
{
  "outcome" => "success",
  "result" => {
    "authorization" => undefined,
    "server-identity" => undefined,
```

```

    "authentication" => {"properties" => {
        "path" => "mgmt-users.properties",
        "plain-text" => false,
        "relative-to" => "jboss.domain.config.dir"
    }}
  }
}

```

セキュリティーレームを新たに作成します。

以下のコマンドは **TestRealm** というセキュリティーレームを作成し、関連するプロパティファイルのディレクトリーを設定します。

例11.19 セキュリティーレームを新たに作成します。

```

/host=master/core-service=management/security-realm=TestRealm/:add
/host=master/core-service=management/security-
realm=TestRealm/authentication=properties/:add(path=TestUsers.properties, relative-
to=jboss.domain.config.dir)

```

既存のセキュリティードメインを用いたセキュリティーレーム認証の設定

セキュリティードメインを使用して管理インターフェースに対して認証を行うには、以下の手順に従います。

最初にセキュリティーレームを作成します。次に、そのセキュリティーレームを管理インターフェースの **security-realm** 属性の値として指定します。

例11.20 HTTP 管理インターフェースに使用するセキュリティーレームの指定

```

/host=master/core-service=management/management-interface=http-interface/:write-
attribute(name=security-realm,value=TestRealm)

```

[Report a bug](#)

11.9. ロールベースアクセス制御を用いた管理インターフェースのセキュア化

11.9.1. ロールベースアクセス制御 (RBAC)

ロールベースアクセスコントロール (RBAC) は、管理ユーザーのパーミッションセットを指定するメカニズムです。これにより、無制限のアクセスを必要とせずに複数のユーザーが JBoss EAP 6.3 の管理業務を共有できます。JBoss EAP 6.3 は管理ユーザーの「職務の分離」を実現することで、不必要な特権を与えずに組織における個人またはグループの業務分散を容易にします。これにより、設定、デプロイメント、および管理の柔軟性を維持しながらサーバーやデータのセキュリティーを可能な限り強化できます。

JBoss EAP 6.3 のロールベースアクセス制御は、ロールパーミッションと制約の組み合わせにより動作します。

異なる固定パーミッションを持つ7つのロールが事前定義されています。事前定義されたロールは Monitor、Operator、Maintainer、Deployer、Auditor、Administrator、および SuperUser です。各管理ユーザーには1つ以上のロールが割り当てられます。各ロールは、サーバーを管理するときにユーザーが実行できる操作を指定します。

[Report a bug](#)

11.9.2. 管理コンソールおよび CLI でのロールベースアクセス制御

ロールベースアクセス制御 (RBAC) が有効になっている場合、ユーザーに割り当てられたロールによって、ユーザーがアクセスできるリソースやリソースの属性を用いて実行できる操作が決定されます。

管理コンソール

管理コンソールでは、ユーザーに割り当てられたロールのパーミッションによっては、制御およびビューの一部が無効化 (灰色で表示) されたり、全く表示されないことがあります。

リソース属性に対する読み取りパーミッションがない場合、その属性は管理コンソールでは空白で表示されます。たとえば、ほとんどのロールは、データソースのユーザー名およびパスワードフィールドを読み取りできません。

リソース属性に対する書き込みパーミッションがない場合、その属性はリソースの編集フォームで無効化 (灰色で表示) されます。リソースに対する書き込みパーミッションがない場合、リソースの編集ボタンは表示されません。

ユーザーがリソースまたは属性へアクセスするパーミッションを持たない場合 (ロールに対して「アドレス不可能」な場合)、コンソールではそのユーザーに対するリソースまたは属性が表示されません。アクセス制御システム自体がこの例で、デフォルトでは少数のロールのみに対して表示されません。

管理 CLI または API

RBAC が有効である場合、API での 管理 CLI や管理 API の動作が若干異なります。

読み取りできないリソースや属性は結果からフィルターされます。フィルターされた項目がロールによってアドレス可能である場合、結果の **response-headers** セクションにこれらの名前が **filtered-attributes** としてリストされます。リソースまたは属性がロールによってアドレス可能でない場合は、リストされません。

アドレス不可能なリソースにアクセスしようとすると、**resource not found** (リソースが見つかりません) というエラーが発生します。

適切な読み書きパーミッションのないユーザーが、アドレス可能なリソースを読み書きしようとすると、**Permission Denied** (パーミッションが拒否されました) というエラーが返されます。

[Report a bug](#)

11.9.3. サポートされる認証スキーム

ロールベースアクセス制御は、JBoss EAP 6.3 に含まれる標準の認証プロバイダーと動作します。標準の認証プロバイダーは **username/password**、**client certificate**、および **local user** です。

Username/Password

ユーザーはユーザー名とパスワードの組み合わせを使用して認証されます。この組み合わせは、**mgmt-users.properties** ファイルまたは LDAP サーバーに対して検証されます。

Client Certificate

トラストストアを使用します。

Local User

同じマシン上で実行されているサーバーの場合、**jboss-cli.sh** は自動的に Local User として認証します。デフォルトでは、Local User は **SuperUser** グループのメンバーになります。

使用されるプロバイダーに関係なく、JBoss EAP はロールをユーザーに割り当てます。しかし、**mgmt-users.properties** ファイルまたは LDAP サーバーで認証する場合、これらのシステムはユーザーグループ情報を提供できます。ユーザーグループ情報は、ロールをユーザーに割り当てるために JBoss EAP が使用することも可能です。

[Report a bug](#)

11.9.4. 標準のロール

JBoss EAP 6 は、Monitor、Operator、Maintainer、Deployer、Auditor、Administrator、および SuperUser の 7 つの事前定義されたユーザーロールを提供します。各ロールは特定のユースケース向けに設定されており、異なるパーミッションセットを持ちます。Monitor、Operator、Maintainer、Administrator、および SuperUser ロールは、それぞれ前のレベルのロールから継承したパーミッションと追加のパーミッションを持ちます。Auditor および Deployer ロールはそれぞれ Monitor および Maintainer ロールに似ていますが、特別なパーミッションおよび制限が追加されています。

Monitor

Monitor ロールのユーザーは、最も少ないパーミッションを持ち、現在の設定およびサーバーの状態のみを読み取りできます。これは、サーバーのパフォーマンスを追跡し、報告する必要があるユーザー向けのロールです。

Monitor はサーバー設定を変更したり、機密データおよび操作にアクセスしたりできません。

Operator

Operator ロールは、サーバーのランタイム状態を変更する機能を追加して、Monitor ロールを拡張します。これにより、Operator はサーバーをリロードおよびシャットダウンでき、JMS 宛先を一時停止および再開できます。Operator ロールは、アプリケーションサーバーの物理または仮想ホストを管理するユーザーに向いており、必要時にサーバーが正しくシャットダウンおよび再起動されるようにします。

Operator はサーバー設定を変更したり、機密データおよび操作にアクセスしたりできません。

Maintainer

Maintainer ロールは、機密データおよび操作以外のすべての設定と、ランタイム状態を表示および変更できます。Maintainer ロールは、機密データおよび操作へアクセスできない汎用のロールです。Maintainer ロールは、パスワードやその他の機密情報へのアクセス権限を付与せずに、ユーザーがサーバーをほぼ完全に管理できるようにします。

Maintainer は機密データまたは操作へアクセスできません。

Administrator

Administrator ロールは、監査ロギングシステムを除くサーバー上のすべてのリソースおよび操作へ無制限にアクセスできます。Administrator ロールは、機密のデータおよび操作へアクセスできます。このロールはアクセス制御システムも設定できます。Administrator ロールは、機密データを処理する場合や、ユーザーとロールを設定する場合のみ必要です。

Administrator は監査ロギングシステムへアクセスできず、Administrator 自身を Auditor または SuperUser ロールへ変更できません。

SuperUser

SuperUser ロールには制限がなく、監査ロギングシステムを含むサーバーのすべてのリソースおよび操作へ完全にアクセスできます。このロールは、以前のバージョンの JBoss EAP 6 (6.0 および 6.1) での管理者ユーザーと同等です。RBAC が無効である場合、すべての管理ユーザーは SuperUser ロールと同等のパーミッションを持ちます。

Deployer

Deployer ロールは Monitor と同じパーミッションを持ちますが、デプロイメントの設定や状態を変更でき、アプリケーションリソースとして有効になっている他のリソースタイプも変更できます。

Auditor

Auditor ロールは Monitor ロールと同じパーミッションを持ちますが、機密データを表示でき (変更はできません)、監査ロギングシステムへ完全にアクセスできます。Auditor ロールは SuperUser ロール以外で唯一監査ロギングシステムへアクセスできるロールです。

Auditor は機密データやリソースを変更できません。読み取りのみが許可されます。

[Report a bug](#)

11.9.5. ロールパーミッション

各ロールの権限は、各ロールのパーミッションによって定義されます。すべてのロールにすべてのパーミッションがあるわけではありません。SuperUser はすべてのパーミッションを持ちますが、Monitor のパーミッションは最も少なくなります。

各パーミッションは、リソースの単一のカテゴリに対して読み書きのアクセスを付与します。

カテゴリは、ランタイム状態、サーバー設定、機密データ、監査ログ、およびアクセス制御システムです。

表11.5「[ロールパーミッション](#)」は各ロールのパーミッションを示しています。

表11.5 ロールパーミッション

	Monitor	Operator	Maintainer	Deployer	Auditor	Administrator	SuperUser
設定と状態の読み取り	○	○	○	○	○	○	○

機密データの読み取り [2]					○	○	○
機密データの変更 [2]						○	○
監査ログの読み取り/ 変更					○		○
ランタイム状態の変更		○	○	○[1]		○	○
永続化設定の変更			○	○[1]		○	○
アクセス制御の読み 取り/変更						○	○

[1] パーミッションはアプリケーションリソースに制限されます。

[2] 機密性制約 (Sensitivity Constraint) を使用して「機密データ」として考慮されるリソースを設定します。

[Report a bug](#)

11.9.6. 制約

制約とは、指定のリソースリストに対するアクセス制御設定の名前付きセットです。RBAC システムは制約とロールパーミッションの組み合わせを使用して、特定ユーザーが管理操作を実行できるかどうかを決定します。

制約は、アプリケーション、機密性、および Vault 式の 3 つに分類されます。

アプリケーション制約

アプリケーション制約は、Deployer ロールのユーザーがアクセスできるリソースおよび属性のセットを定義します。デフォルトでは、有効になっているアプリケーション制約は、デプロイメントやデプロイメントオーバーレイが含まれるコアのみです。また、アプリケーション制約はデータソース、ロギング、メール、メッセージング、ネーミング、リソースアダプター、およびセキュリティに対しても含まれますが、デフォルトでは有効になっていません。これらの制約により、Deployer ユーザーはアプリケーションをデプロイできるだけでなく、これらのアプリケーションが必要とするリソースを設定および維持できます。

アプリケーション制約の設定は、管理 API の `/core-service=management/access=authorization/constraint=application-classification` にあります。

機密性制約

機密性制約は、「機密」とされるリソースのセットを定義します。通常、機密リソースとは、パスワードなどの秘密のリソースや、ネットワーキング、JVM 設定、システムプロパティなどのサーバー操作に深刻な影響を与えるリソースのことを言います。アクセス制御システム自体も機密であ

るとみなされます。

機密リソースへの書き込みを許可されるロールは、Administrator と SuperUser のみです。Auditor ロールは、機密リソースの読み取りのみ可能です。他のロールは機密リソースへアクセスできません。

機密性制約の設定は、管理 API の **/core-service=management/access=authorization/constraint=sensitivity-classification** にあります。

vault 式制約

vault 式制約は、vault 式の読み取りまたは書き込みが機密操作としてみなされるかどうかを定義します。デフォルトでは、vault 式の読み書き両方が機密操作とみなされます。

vault 式制約の設定は、管理 API の **/core-service=management/access=authorization/constraint=vault-expression** にあります。

制約は管理コンソールでは設定できません。

[Report a bug](#)

11.9.7. JMX およびロールベースアクセス制御

ロールベースのアクセス制御は、3 つの方法で JMX に適用されます。

1. JBoss EAP 6 の管理 API は JXM 管理 Bean として公開されます。これらの管理 Bean は「コア MBean」と呼ばれ、コア MBean へのアクセスは基盤の管理 API と同じように制御およびフィルターされます。
2. JMX サブシステムは、書き込みパーミッションが「機密」で設定されます。そのため、Administrator および SuperUser ロールのユーザーのみがそのサブシステムに変更を追加できます。Auditor ロールのユーザーはこのサブシステムの設定を読み取りできます。
3. デフォルトでは、デプロイされたアプリケーションおよびサービスによって登録された管理 Bean (コアでない MBean) へすべての管理ユーザーがアクセスできますが、Maintainer、Operator、Administrator、および SuperUser ロールのユーザーのみが書き込み可能です。

[Report a bug](#)

11.9.8. ロールベースアクセス制御の設定

11.9.8.1. RBAC 設定タスクの概要

RBAC が有効になっている場合、Administration および SuperUser ロールのユーザーのみがアクセス制御システムを表示し、変更を追加できます。

管理コンソールは、以下の一般的な RBAC タスクに対してインターフェースを提供します。

- 各ユーザーへ割り当てられた (または除外された) ロールの表示および設定
- 各グループへ割り当てられた (または除外された) ロールの表示および設定。
- ロールごとのグループおよびユーザーメンバーシップの表示。
- ロールごとのデフォルトメンバーシップの設定。

- スコープ指定されたロールの作成。

管理 CLI は完全なアクセス制御システムへのアクセスを提供します。そのため、管理コンソールで行えることはすべて管理 CLI で行えますが、管理コンソールでは実行できない複数のタスクを管理 CLI で実行できます。

管理 CLI で実行できる追加のタスクは次のとおりです。

- RBAC の有効化および無効化。
- パーミッション組み合わせポリシーの変更
- アプリケーションリソースおよびリソース機密性の制約の設定

[Report a bug](#)

11.9.8.2. ロールベースアクセス制御の有効化

デフォルトでは、ロールベースアクセス制御 (RBAC) システムは無効になっています。有効にするには、プロバイダー属性を **simple** から **rbac** に変更します。この変更を行うには管理 CLI を使用しますが、サーバーがオフラインの場合はサーバー設定 XML ファイルを編集して変更できます。RBAC が稼働中のサーバー上で無効または有効になっている場合は、サーバー設定をリロードして変更を反映する必要があります。

一旦 RBAC を有効にすると、無効にできるのは Administrator または SuperUser ロールのユーザーのみです。デフォルトでは、サーバーと同じマシン上で実行されている場合、管理 CLI が **SuperUser** ロールとして実行されます。

手順11.8 RBAC の有効化

- 管理 CLI で RBAC を有効にするには、アクセス承認リソースの **write-attribute** 操作を使用して、プロバイダー属性を **rbac** に設定します。

```
/core-service=management/access=authorization:write-attribute(name=provider, value=rbac)
```

```
[standalone@localhost:9999 /] /core-service=management/access=authorization:write-attribute(name=provider, value=rbac)
{
  "outcome" => "success",
  "response-headers" => {
    "operation-requires-reload" => true,
    "process-state" => "reload-required"
  }
}
[standalone@localhost:9999 /] ./:reload
{
  "outcome" => "success",
  "result" => undefined
}
```

手順11.9 RBAC の無効化

- 管理 CLI で RBAC を無効にするには、アクセス承認リソースの **write-attribute** 操作を使用して、プロバイダー属性を **simple** に設定します。

■

```
/core-service=management/access=authorization:write-attribute(name=provider,
value=simple)
```

```
[standalone@localhost:9999 /] /core-service=management/access=authorization:write-
attribute(name=provider, value=simple)
{
  "outcome" => "success",
  "response-headers" => {
    "operation-requires-reload" => true,
    "process-state" => "reload-required"
  }
}
[standalone@localhost:9999 /] ./reload
{
  "outcome" => "success",
  "result" => undefined
}
```

サーバーがオフラインの場合は、XML 設定を編集して RBAC を有効または無効にできます。これを行うには、管理要素の **access-control** 要素にある **provider** 属性を編集します。有効にする場合は値を **rbac** に設定し、無効にする場合は **simple** に設定します。

```
<management>

  <access-control provider="rbac">
    <role-mapping>
      <role name="SuperUser">
        <include>
          <user name="$local"/>
        </include>
      </role>
    </role-mapping>
  </access-control>

</management>
```

[Report a bug](#)

11.9.8.3. パーミッション組み合わせポリシーの変更

パーミッション組み合わせポリシーは、ユーザーに複数のロールが割り当てられている場合にどのようにパーミッションを判断するかを決定します。このポリシーは、**permissive** または **rejecting** に設定できます。デフォルトは **permissive** です。

permissive に設定されると、アクションを許可するロールがユーザーに割り当てられている場合に、そのアクションが許可されます。

rejecting に設定された場合、複数のロールが割り当てられたユーザーはすべてのアクションが禁止されます。そのため、ポリシーが **rejecting** に設定された場合は、各ユーザーに1つのロールのみが割り当てられるようにします。ポリシーが **rejecting** に設定されると、複数のロールが割り当てられたユーザーは管理コンソールや管理 CLI を使用できません。

パーミッション組み合わせポリシーを設定するには、**permission-combination-policy** 属性を **permissive** または **rejecting** に設定します。これは管理 CLI を使用して設定できますが、サーバーがオフラインの場合はサーバー設定 XML ファイルを編集して設定できます。

手順11.10 パーミッション組み合わせポリシーの設定

- アクセス承認リソースの **write-attribute** 操作を使用して **permission-combination-policy** 属性を必要なポリシー名に設定します。

```
/core-service=management/access=authorization:write-attribute(name=permission-combination-policy, value=POLICYNAME)
```

有効なポリシー名は rejecting と permissive です。

```
[standalone@localhost:9999 /] /core-service=management/access=authorization:write-attribute(name=permission-combination-policy, value=rejecting)
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

サーバーがオフラインの場合は、XML 設定を編集してパーミッション組み合わせポリシーの値を変更できます。これを行うには、アクセス制御要素の **permission-combination-policy** 属性を編集します。

```
<access-control provider="rbac" permission-combination-policy="rejecting">
  <role-mapping>
    <role name="SuperUser">
      <include>
        <user name="$local"/>
      </include>
    </role>
  </role-mapping>
</access-control>
```

[Report a bug](#)

11.9.9. ロールの管理

11.9.9.1. ロールメンバシップ

ロールベースアクセス制御 (RBAC) が有効になっている場合、管理ユーザーに割り当てられたロールがそのユーザーに許可されるアクションを決定します。JBoss EAP 6.3 は、ユーザーおよびグループメンバシップを基に、include (含まれる) および exclude (除外される) を使用して、ユーザーが属するロールを決定します。

以下の場合に、ロールがユーザーに割り当てられたとみなされます。

1. ユーザーが以下のいずれかである場合。
 - ロールに含まれるユーザーとしてリストに記載されている。
 - ロールに含まれるとリストに記載されたグループのメンバーである。
2. ユーザーが以下のいずれかに該当しない場合。
 - ロールから除外されるユーザーとしてリストに記載されている。
 - ロールから除外されるとリストに記載されたグループのメンバーである。

exclude は include よりも優先度が高くなります。

ユーザーおよびグループに対するロールの include および exclude 設定は、管理コンソールと管理 CLI の両方を使用して設定できます。

SuperUser または Administrator ロールのユーザーのみがこの設定を行えます。

[Report a bug](#)

11.9.9.2. ユーザーロール割り当ての設定

include (含まれる) および exclude (除外) されるユーザーロールは、管理コンソールおよび 管理 CLI で設定できます。ここでは、管理コンソールを使用した設定のみを説明します。

この設定が行えるのは **SuperUser** または **Administrator** ロールのユーザーのみです。

以下の手順で、管理コンソールのユーザーロール設定を確認します。

1. 管理コンソールへログインします。
2. **Administration** タブをクリックします。
3. **Access Control** メニューを展開し、**Role Assignment** を選択します。
4. **USERS** タブを選択します。

手順11.11 ユーザーの新しいロール割り当ての作成

1. 管理コンソールへログインします。
2. **Role Assignment** セクションの **Users** タブに移動します。
3. ユーザーリストの右上にある **Add** ボタンをクリックします。 **Add User** ダイアログが表示されます。

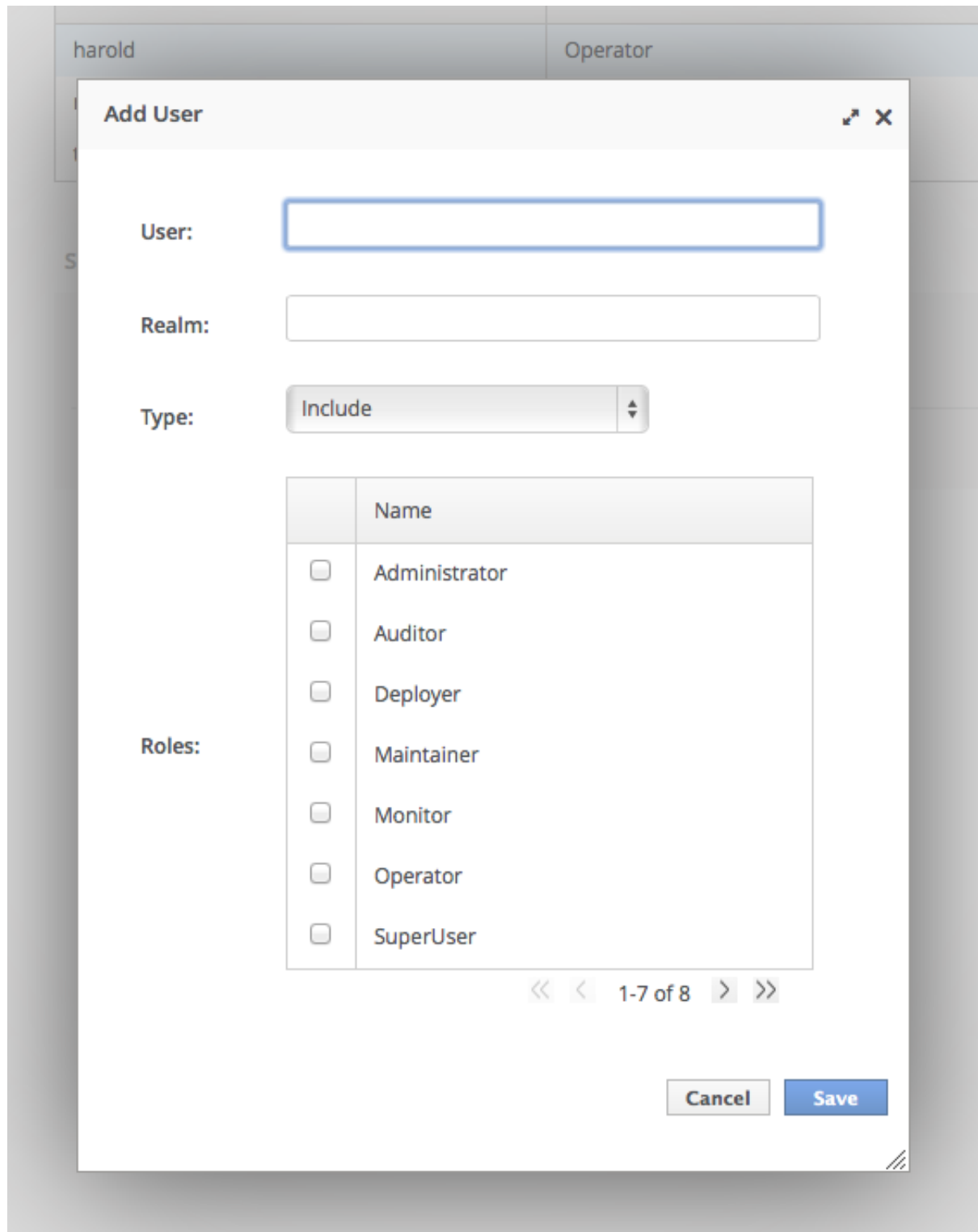


図11.1 Add User ダイアログ

4. ユーザー名を指定し、任意でレルムを指定します。
5. Type メニューを include または exclude に設定します。
6. include または exclude するロールのチェックボックスをクリックします。Control キー (OSX では Command キー) を押しながらクリックすると、複数のチェックボックスを選択できます。
7. **Save** をクリックして終了します。

正常に保存されると、**Add User** タイプロックが閉じられます。ユーザーのリストが更新され、変更内容が反映されます。正常に保存されなかった場合は、**Failed to save role assignment** というメッセージが表示されます。

手順11.12 ユーザーロール割り当ての更新

- 1. 管理コンソールへログインします。
- 2. **Role Assignment** セクションの **Users** タブに移動します。
- 3. リストよりユーザーを選択します。
- 4. **Edit** をクリックします。選択パネルが編集モードになります。

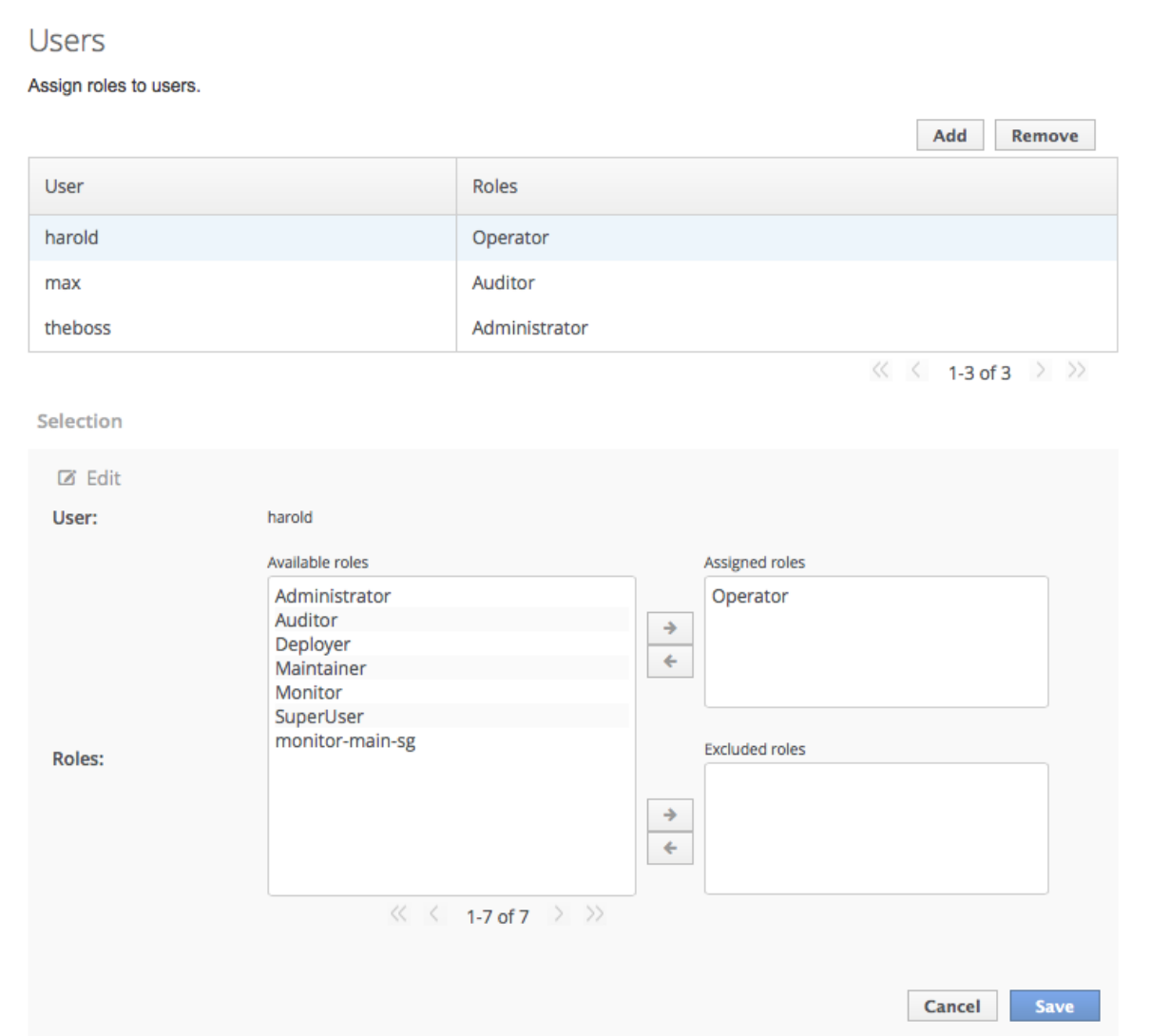


図11.2 Selection Edit ビュー

ここで、ユーザーに割り当てられたロールや除外されたロールを追加および削除できます。

- 1. 割り当てられたロールを追加するには、左側の使用可能なロールのリストからロールを選択し、割り当てられたロールリストの横にある、右矢印のボタンをクリックします。ロールが、使用可能なロールのリストから割り当てられたロールのリストに移動します。

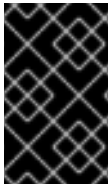
2. 割り当てられたロールを削除するには、割り当てられたロールのリストからロールを選択し、割り当てられたロールリストの横にある左矢印のボタンをクリックします。ロールが、割り当てられたロールのリストから使用可能なロールのリストへ移動します。
3. 除外されたロールを追加するには、左側の使用可能なロールのリストからロールを選択し、除外されたロールリストの横にある、右矢印のボタンをクリックします。ロールが、使用可能なロールのリストから除外されたロールのリストに移動します。
4. 除外されたロールを削除するには、除外されたロールのリストからロールを選択し、除外されたロールリストの横にある左矢印のボタンをクリックします。ロールが、除外されたロールのリストから使用可能なロールのリストへ移動します。
5. **Save** をクリックして終了します。

正常に保存されると、編集ビューが閉じられます。ユーザーのリストが更新され、変更内容が反映されます。正常に保存されなかった場合は、**Failed to save role assignment** というメッセージが表示されます。

手順11.13 ユーザーのロール割り当ての削除

1. 管理コンソールへログインします。
2. Role Assignment セクションの **Users** タブへ移動します。
3. リストよりユーザーを選択します。
4. **Remove** をクリックします。 **Remove Role Assignment** 確認プロンプトが表示されます。
5. **Confirm** をクリックします。

正しく削除されると、ユーザーロール割り当てのリストにユーザーが表示されなくなります。



重要

ロール割り当てのリストからユーザーを削除しても、ユーザーはシステムから削除されず、ユーザーにロールが割り当てられなくなる保証はありません。ロールはグループメンバーシップから割り当てられる可能性があります。

[Report a bug](#)

11.9.9.3. 管理 CLI を用いたユーザーロール割り当ての設定

include (含まれる) および exclude (除外) されるユーザーロールは、管理コンソールおよび 管理 CLI で設定できます。ここでは、管理 CLI を使用した設定のみを説明します。

ユーザーおよびグループをロールへマッピングする設定は、管理 API の **role-mapping** 要素とする **/core-service=management/access=authorization** にあります。

SuperUser または Administrator ロールのユーザーのみがこの設定を行えます。

コマンドへのアクセスを容易にするため、管理 CLI では **/core-service=management/access=authorization** の場所を変更します。

```
[standalone@localhost:9999] cd /core-service=management/access=authorization
```

手順11.14 ロール割り当て設定の表示

1. `:read-children-names` 操作を使用して、設定されたロールの完全リストを取得します。

```
/core-service=management/access=authorization:read-children-names(child-type=role-mapping)
```

```
[standalone@localhost:9999 access=authorization] :read-children-names(child-type=role-mapping)
{
  "outcome" => "success",
  "result" => [
    "Administrator",
    "Deployer",
    "Maintainer",
    "Monitor",
    "Operator",
    "SuperUser"
  ]
}
```

2. 指定されたロールマッピングの **read-resource** 操作を使用して、特定ロールの完全詳細を取得します。

```
/core-service=management/access=authorization/role-mapping=ROLENAME:read-resource(recursive=true)
```

```
[standalone@localhost:9999 access=authorization] ./role-mapping=Administrator:read-resource(recursive=true)
{
  "outcome" => "success",
  "result" => {
    "include-all" => false,
    "exclude" => undefined,
    "include" => {
      "user-theboss" => {
        "name" => "theboss",
        "realm" => undefined,
        "type" => "USER"
      },
      "user-harold" => {
        "name" => "harold",
        "realm" => undefined,
        "type" => "USER"
      },
      "group-SysOps" => {
        "name" => "SysOps",
        "realm" => undefined,
        "type" => "GROUP"
      }
    }
  }
}
[standalone@localhost:9999 access=authorization]
```


手順11.15 新規ロールの追加

この手順は、ロールのロールマッピングエントリーを追加する方法を示しています。ロールを設定する前にこの手順を実行する必要があります。

- **add** 操作を使用して、新しいロール設定を追加します。

```
/core-service=management/access=authorization/role-mapping=ROLENAME:add
```

ROLENAME は新しいマッピングに対するロール名です。

```
[standalone@localhost:9999 access=authorization] ./role-mapping=Auditor:add
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

手順11.16 ロールに include されるユーザーの追加

この手順では、ユーザーをロールの include されたリストに追加する方法を説明します。

ロールの設定が行われていない場合は、最初にロールマッピングエントリーの設定を行う必要があります。

- **add** 操作を使用して、ユーザーエントリーをロールの include リストに追加します。

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/include=ALIAS:add(name=USERNAME, type=USER)
```

ROLENAME は設定されたロールの名前です。

ALIAS はこのマッピングの一意名です。Red Hat は、**user-*USERNAME*** などのエイリアスに命名規則を使用することを推奨します。

USERNAME は、include リストに追加されたユーザーの名前です。

```
[standalone@localhost:9999 access=authorization] ./role-mapping=Auditor/include=user-
max:add(name=max, type=USER)
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

手順11.17 ロールに exclude されるユーザーの追加

この手順では、ユーザーをロールの exclude されたリストに追加する方法を説明します。

ロールの設定が行われていない場合は、最初にロールマッピングエントリーの設定を行う必要があります。

- **add** 操作を使用して、ユーザーエントリーをロールの exclude リストに追加します。

```
/core-service=management/access=authorization/role-
mapping=ROLENAME/exclude=ALIAS:add(name=USERNAME, type=USER)
```

ROLENAME は設定されたロールの名前です。

USERNAME は、exclude リストに追加されたユーザーの名前です。

ALIAS はこのマッピングの一意名です。Red Hat は、**user-USERNAME** などのエイリアスに命名規則を使用することを推奨します。

```
[standalone@localhost:9999 access=authorization] ./role-mapping=Auditor/exclude=user-max:add(name=max, type=USER)
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

手順11.18 ユーザーロールの include 設定の削除

この手順では、ロールマッピングからユーザー include エントリーを削除する方法を説明します。

- **remove** 操作を使用してエントリーを削除します。

```
/core-service=management/access=authorization/role-mapping=ROLENAME/include=ALIAS:remove
```

ROLENAME は設定されたロールの名前です。

ALIAS はこのマッピングの一意名です。Red Hat は、**user-USERNAME** などのエイリアスに命名規則を使用することを推奨します。

```
[standalone@localhost:9999 access=authorization] ./role-mapping=Auditor/include=user-max:remove
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

include リストからユーザーを削除しても、ユーザーはシステムから削除されず、ロールがユーザーに割り当てられなくなる保証はありません。ロールはグループメンバーシップを基に割り当てられる可能性があります。

手順11.19 ユーザーロールの exclude 設定の削除

この手順では、ロールマッピングからユーザー exclude エントリーを削除する方法を説明します。

- **remove** 操作を使用してエントリーを削除します。

```
/core-service=management/access=authorization/role-mapping=ROLENAME/exclude=ALIAS:remove
```

ROLENAME は設定されたロールの名前です。

ALIAS はこのマッピングの一意名です。Red Hat は、**user-USERNAME** などのエイリアスに命名規則を使用することを推奨します。

```
[standalone@localhost:9999 access=authorization] ./role-mapping=Auditor/exclude=user-max:remove
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

exclude リストからユーザーを削除しても、ユーザーはシステムから削除されず、ロールがユーザーに割り当てられる保証はありません。ロールはグループメンバーシップを基に除外される可能性があります。

[Report a bug](#)

11.9.9.4. ロールとユーザーグループ

mgmt-users.properties ファイルまたは LDAP サーバーを使用して認証されるユーザーはユーザーグループのメンバーである可能性があります。ユーザーグループは、1名以上のユーザーに割り当てできる任意のラベルです。

RBAC システムは、ユーザーがメンバーであるユーザーグループに応じて自動的にロールをユーザーに割り当てるよう設定できます。また、グループメンバーシップを基にユーザーをロールから exclude (除外) することも可能です。

mgmt-users.properties ファイルを使用する場合、グループ情報は **mgmt-groups.properties** ファイルに保存されます。LDAP を使用する場合、グループ情報は LDAP サーバーに保存され、LDAP サーバーの管理者によって維持されます。

[Report a bug](#)

11.9.9.5. グループロール割り当ての設定

ユーザーグループのユーザーのメンバーシップを基に、ロールをユーザーに割り当てできます。

include (含まれる) または exclude (除外) されるグループロールは、管理コンソールおよび 管理 CLI で設定できます。ここでは、管理コンソールを使用した設定のみを説明します。

この設定が行えるのは **SuperUser** または **Administrator** ロールのユーザーのみです。

以下の手順で、管理コンソールのグループロール設定を確認します。

1. 管理コンソールへログインします。
2. **Administration** タブをクリックします。
3. **Access Control** メニューを展開し、**Role Assignment** を選択します。
4. **GROUPS** タブを選択します。

手順11.20 グループの新しいロール割り当ての作成

1. 管理コンソールへログインします。
2. **Role Assignment** セクションの **GROUPS** タブに移動します。
3. ユーザーリストの右上にある **Add** ボタンをクリックします。**Add Group** ダイアログが表示されます。

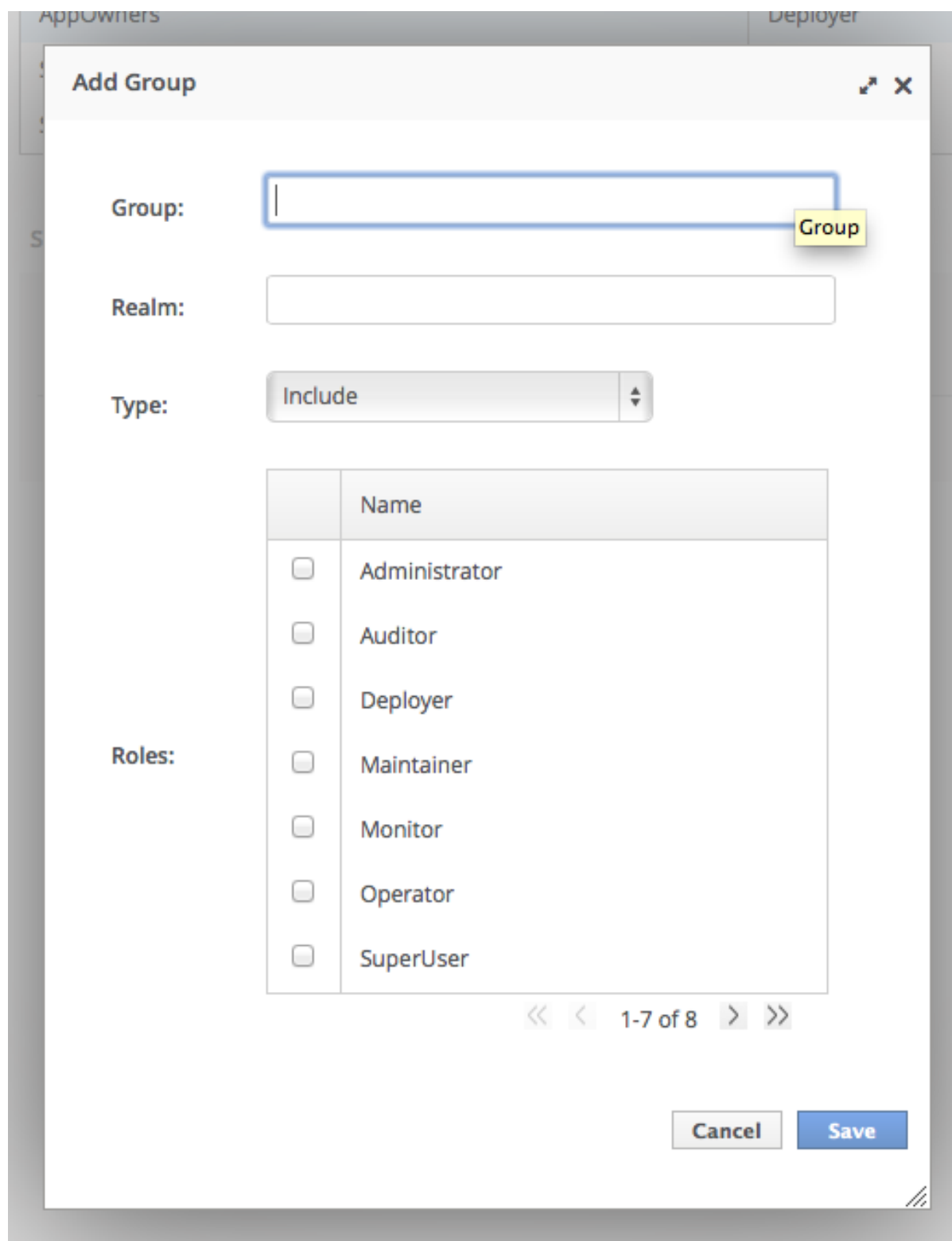


図11.3 Add Group ダイアログ

4. グループ名を指定し、任意でレルムを指定します。
5. Type メニューを include または exclude に設定します。
6. include または exclude するロールのチェックボックスをクリックします。Control キー (OSX では Command キー) を押しながらクリックすると、複数のチェックボックスを選択できます。
7. **Save** をクリックして終了します。

正常に保存されると、**Add Group** ダイアログが閉じられます。グループのリストが更新され、変更内容が反映されます。正常に保存されなかった場合は、**Failed to save role assignment** というメッセージが表示されます。

手順11.21 グループロール割り当ての更新

1. 管理コンソールへログインします。
2. Role Assignment セクションの **GROUPS** タブへ移動します。
3. リストよりグループを選択します。
4. Edit をクリックします。Selection ビューが編集モードになります。

Groups

Assign roles to groups.

Groups		Add	Remove
Group	Roles		
AppOwners	Deployer		
Supervisors	Monitor		
SysOps	Operator		

1-3 of 3

Selection

☒ Edit

Group: AppOwners

Available roles

- Administrator
- Auditor
- Maintainer
- Monitor
- Operator
- SuperUser
- monitor-main-sg

Assigned roles

- Deployer

Excluded roles

Roles:

1-7 of 7

Cancel Save

図11.4 Selection ビュー編集モード

ここで、グループに割り当てられたロールや除外されたロールを追加および削除できます。

- 割り当てられたロールを追加するには、左側の使用可能なロールのリストからロールを選択し、割り当てられたロールリストの横にある、右矢印のボタンをクリックします。ロールが、使用可能なロールのリストから割り当てられたロールのリストに移動します。

割り当てられたロールを削除するには、割り当てられたロールの横にある、左矢印のボタンをクリックします。

- 割り当てられたロールを削除するには、割り当てられたロールのリストからロールを選択し、割り当てられたロールリストの横にある左矢印のボタンをクリックします。ロールが、割り当てられたロールのリストから使用可能なロールのリストへ移動します。
- 除外されたロールを追加するには、左側の使用可能なロールのリストからロールを選択し、除外されたロールリストの横にある、右矢印のボタンをクリックします。ロールが、使用可能なロールのリストから除外されたロールのリストに移動します。
- 除外されたロールを削除するには、除外されたロールのリストからロールを選択し、除外されたロールリストの横にある左矢印のボタンをクリックします。ロールが、除外されたロールのリストから使用可能なロールのリストへ移動します。

5. **Save** をクリックして終了します。

正常に保存されると、編集ビューが閉じられます。グループのリストが更新され、変更内容が反映されます。正常に保存されなかった場合は、**Failed to save role assignment** というメッセージが表示されます。

手順11.22 グループのロール割り当ての削除

1. 管理コンソールへログインします。
2. **Role Assignment** セクションの **GROUPS** タブへ移動します。
3. リストよりグループを選択します。
4. **Remove** をクリックします。 **Remove Role Assignment** 確認プロンプトが表示されます。
5. **Confirm** をクリックします。

正しく削除されると、グループロール割り当てのリストにロールが表示されなくなります。

ロール割り当てのリストからグループを削除しても、ユーザーグループはシステムから削除されず、そのグループのメンバーにロールが割り当てられなくなる保証はありません。各グループメンバーに直接ロールが割り当てられる可能性があります。

[Report a bug](#)

11.9.9.6. 管理 CLI を用いたグループロール割り当ての設定

include (含まれる) または exclude (除外) されるグループロールは、管理コンソールおよび 管理 CLI で設定できます。ここでは、管理 CLI を使用した設定のみを説明します。

ユーザーおよびグループをロールへマッピングする設定は、管理 API の **role-mapping** 要素とする **/core-service=management/access=authorization** にあります。

この設定を行えるのは、SuperUser または Administrator ロールのユーザーのみです。

コマンドへのアクセスを容易にするため、管理 CLI では **/core-service=management/access=authorization** の場所を変更します。

```
[standalone@localhost:9999] cd /core-service=management/access=authorization
```

手順11.23 グループロール割り当て設定の表示

1. **read-children-names** 操作を使用して、設定されたロールの完全リストを取得します。

```
/core-service=management/access=authorization:read-children-names(child-type=role-
mapping)
```

```
[standalone@localhost:9999 access=authorization] :read-children-names(child-type=role-
mapping)
{
  "outcome" => "success",
  "result" => [
    "Administrator",
    "Deployer",
    "Maintainer",
    "Monitor",
    "Operator",
    "SuperUser"
  ]
}
```

2. 指定されたロールマッピングの **read-resource** 操作を使用して、特定ロールの完全詳細を取得します。

```
/core-service=management/access=authorization/role-mapping=ROLENAME:read-
resource(recursive=true)
```

```
[standalone@localhost:9999 access=authorization] ./role-mapping=Administrator:read-
resource(recursive=true)
{
  "outcome" => "success",
  "result" => {
    "include-all" => false,
    "exclude" => undefined,
    "include" => {
      "user-theboss" => {
        "name" => "theboss",
        "realm" => undefined,
        "type" => "USER"
      },
      "user-harold" => {
        "name" => "harold",
        "realm" => undefined,
        "type" => "USER"
      },
      "group-SysOps" => {
        "name" => "SysOps",
        "realm" => undefined,
        "type" => "GROUP"
      }
    }
  }
}
[standalone@localhost:9999 access=authorization]
```

手順11.24 新規ロールの追加

この手順は、ロールのロールマッピングエントリーを追加する方法を示しています。ロールを設定する前にこの手順を実行する必要があります。

- **add** 操作を使用して、新しいロール設定を追加します。

```
/core-service=management/access=authorization/role-mapping=ROLENAME:add
```

```
[standalone@localhost:9999 access=authorization] ./role-mapping=Auditor:add  
{"outcome" => "success"}  
[standalone@localhost:9999 access=authorization]
```

手順11.25 グループに include されるユーザーの追加

この手順では、グループをロールの include されたリストに追加する方法を説明します。

ロールの設定が行われていない場合は、最初にロールマッピングエントリーの設定を行う必要があります。

- **add** 操作を使用して、グループエントリーをロールの include リストに追加します。

```
/core-service=management/access=authorization/role-  
mapping=ROLENAME/include=ALIAS:add(name=GROUPNAME, type=GROUP)
```

ROLENAME は設定されたロールの名前です。

GROUPNAME は、include リストに追加されたグループの名前です。

ALIAS はこのマッピングの一意名です。Red Hat は、**group-*GROUPNAME*** などのエイリアスに命名規則を使用することを推奨します。

```
[standalone@localhost:9999 access=authorization] ./role-mapping=Auditor/include=group-  
investigators:add(name=investigators, type=GROUP)  
{"outcome" => "success"}  
[standalone@localhost:9999 access=authorization]
```

手順11.26 ロールに exclude されるグループの追加

この手順では、グループをロールの exclude されたリストに追加する方法を説明します。

ロールの設定が行われていない場合は、最初にロールマッピングエントリーを作成する必要があります。

- **add** 操作を使用して、グループエントリーをロールの exclude リストに追加します。

```
/core-service=management/access=authorization/role-  
mapping=ROLENAME/exclude=ALIAS:add(name=GROUPNAME, type=GROUP)
```

ROLENAME は設定されたロールの名前です。

GROUPNAME は、include リストに追加されたグループの名前です。

ALIAS はこのマッピングの一意名です。Red Hat は、**group-*GROUPNAME*** などのエイリアスに命名規則を使用することを推奨します。


```
[standalone@localhost:9999 access=authorization] ./role-mapping=Auditor/exclude=group-supervisors:add(name=supervisors, type=GROUP)
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

手順11.27 グループロールの include 設定の削除

この手順では、ロールマッピングからグループ include エントリーを削除する方法を説明します。

- **remove** 操作を使用してエントリーを削除します。

```
/core-service=management/access=authorization/role-mapping=ROLENAME/include=ALIAS:remove
```

ROLENAME は設定されたロールの名前です。

ALIAS はこのマッピングの一意名です。Red Hat は、**group-*GROUPNAME*** などのエイリアスに命名規則を使用することを推奨します。

```
[standalone@localhost:9999 access=authorization] ./role-mapping=Auditor/include=group-investigators:remove
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

include リストからグループを削除しても、グループはシステムから削除されず、このグループのユーザーにロールが割り当てられなくなる保証はありません。ロールは、グループのユーザーへ個別に割り当てられる可能性があります。

手順11.28 ユーザーグループの exclude エントリーの削除

この手順では、ロールマッピングからグループ exclude エントリーを削除する方法を説明します。

- **remove** 操作を使用してエントリーを削除します。

```
/core-service=management/access=authorization/role-mapping=ROLENAME/exclude=ALIAS:remove
```

ROLENAME は設定されたロールの名前です。

ALIAS はこのマッピングの一意名です。Red Hat は、**group-*GROUPNAME*** などのエイリアスに命名規則を使用することを推奨します。

```
[standalone@localhost:9999 access=authorization] ./role-mapping=Auditor/exclude=group-supervisors:remove
{"outcome" => "success"}
[standalone@localhost:9999 access=authorization]
```

exclude リストからグループを削除しても、グループはシステムから削除されず、ロールがグループのメンバーに割り当てられる保証はありません。ロールはグループメンバーシップを基に除外される可能性があります。

[Report a bug](#)

11.9.9.7. LDAP での承認とグループローディング

LDAP ディレクトリーには、属性によって相互参照されるユーザーアカウントとグループのエントリーが含まれます。LDAP サーバーの設定によっては、**memberOf** 属性を用いてユーザーエンティティがユーザーが属するグループをマップしたり、**uniqueMember** 属性を用いてグループエンティティが属するユーザーをマップしたりすることがあります。また、両方のマッピングが LDAP サーバーによって維持されることもあります。

通常、ユーザーは簡単なユーザー名を使用してサーバーに対して認証を行います。グループメンバーシップ情報を検索する場合、使用中のディレクトリーサーバーに応じて、検索がこの単純名を使用して実行されたり、ディレクトリーのユーザーエントリーの識別名を使用して実行されたりします。

必ず最初に、サーバーへ接続するユーザーを認証する手順が実行されます。ユーザーの認証に成功した後、サーバーはサーバーグループをロードします。認証手順と承認手順はそれぞれ LDAP サーバーへの接続が必要になります。レルムはグループをロードする手順の認証接続を再使用して、このプロセスを最適化します。以下の設定手順のとおり、承認セクション内でルールを定義し、ユーザーの単純名を識別名に変換できます。認証中に行われる「ユーザー名から識別名へのマッピング」検索の結果はキャッシュされ、**force** が **false** に設定されている場合は承認クエリー中に再使用されます。**force** が **true** の場合は、承認中 (グループのロード中) に検索が再実行されます。通常、これは認証と承認が異なるサーバーによって実行される場合に行われます。

```
<authorization>
  <ldap connection="...">
    <!-- OPTIONAL -->
    <username-to-dn force="true">
      <!-- Only one of the following. -->
      <username-is-dn />
      <username-filter base-dn="..." recursive="..." user-dn-attribute="..." attribute="..." />
      <advanced-filter base-dn="..." recursive="..." user-dn-attribute="..." filter="..." />
    </username-to-dn>

    <group-search group-name="..." iterative="..." group-dn-attribute="..." group-name-attribute="...">
      <!-- One of the following -->
      <group-to-principal base-dn="..." recursive="..." search-by="...">
        <membership-filter principal-attribute="..." />
      </group-to-principal>
      <principal-to-group group-attribute="..." />
    </group-search>
  </ldap>
</authorization>
```

重要

これらの例では、実証目的で一部の属性をデフォルト値に指定します。デフォルト値を指定する属性は、サーバーによって永続化されると設定から削除されます。例外は **force** 属性で、デフォルト値の **false** に設定されていても必要となります。

username-to-dn

username-to-dn 要素は、ユーザー名をエントリーの識別名へマップする方法を指定します。この要素は、以下の両方の条件に見合う場合のみ必要となります。

- 認証および承認手順は異なる LDAP サーバーに対するものである。
- グループ検索が識別名を使用する。

1:1 username-to-dn

リモートユーザーによって入力されたユーザー名はユーザーの識別名であると指定します。

```
<username-to-dn force="false">
  <username-is-dn />
</username-to-dn>
```

これは 1:1 マッピングを定義し、追加の設定はありません。

username-filter

次のオプションは、前述の認証手順の簡単なオプションと似ています。指定のユーザー名に対して一致する指定の属性が検索されます。

```
<username-to-dn force="true">
  <username-filter base-dn="dc=people,dc=harold,dc=example,dc=com" recursive="false"
    attribute="sn" user-dn-attribute="dn" />
</username-to-dn>
```

設定可能な属性は次のとおりです。

- **base-dn**: 検索を開始するコンテキストの識別名。
- **recursive**: サブコンテキストが検索対象となるかどうか。デフォルトは **false** です。
- **attribute**: 指定のユーザー名に対して一致されるユーザーエントリーの属性。デフォルトは **uid** です。
- **user-dn-attribute**: ユーザーの識別名を取得するために読み取る属性。デフォルトは **dn** です。

advanced-filter

詳細フィルターを指定するオプションです。認証セクションでは、カスタムフィルターを使用してユーザーの識別名を見つけられます。

```
<username-to-dn force="true">
  <advanced-filter base-dn="dc=people,dc=harold,dc=example,dc=com" recursive="false"
    filter="sAMAccountName={0}" user-dn-attribute="dn" />
</username-to-dn>
```

username-filter の例と同じ属性は、意味とデフォルト値も同じです。新たな属性は以下の1つのみです。

- **filter**: ユーザー名が **{0}** プレースホルダーで置き換えられる、ユーザーのエントリーの検索に使用されるカスタムフィルター。



重要

フィルターの定義後も XML が有効である必要があるため、**&** などの特殊文字が使用される場合は、適切な形式が使用されるようにしてください。たとえば、**&** 文字には **&** を使用します。

グループ検索

グループメンバーシップ情報の検索に2つのスタイルを使用できます。1つ目のスタイルは、ユーザーがメンバーであるグループを参照する属性が含まれるユーザーのエントリーで、2つ目のスタイルは、ユーザーエントリーを参照する属性が含まれるグループです。

使用するスタイルを選択できる場合、Red Hat はグループを参照するユーザーのエントリーに対する設定を使用することを推奨します。これは、検索を実行せずに既知の識別名の属性を読み取り、グループ情報をロードできるからです。別の方法は、ユーザーを参照するグループを特定するために大がかりな検索が必要となります。

設定を記述する前に、LDIF の例を見てみましょう。

例11.21 LDIF の例: プリンシパルからグループ

この例では、ユーザー **TestUserOne** は **GroupOne** のメンバーで、**GroupOne** は **GroupFive** のメンバーです。グループメンバーシップは、**memberOf** 属性を使用して表されます。**memberOf** 属性は、ユーザー (またはグループ) がメンバーであるグループの識別名に設定されます。

ここには示されていませんが、複数の **memberOf** 属性を設定することも可能です (ユーザーが直接メンバーであるグループごとに1つ)。

```
dn: uid=TestUserOne,ou=users,dc=principal-to-group,dc=example,dc=org
objectClass: extensibleObject
objectClass: top
objectClass: groupMember
objectClass: inetOrgPerson
objectClass: uidObject
objectClass: person
objectClass: organizationalPerson
cn: Test User One
sn: Test User One
uid: TestUserOne
distinguishedName: uid=TestUserOne,ou=users,dc=principal-to-group,dc=example,dc=org
memberOf: uid=GroupOne,ou=groups,dc=principal-to-group,dc=example,dc=org
memberOf: uid=Slashy/Group,ou=groups,dc=principal-to-group,dc=example,dc=org
userPassword::
e1NTSEF9WFpURzhLVjc4WVZBQUJNbEI3Ym96UVAva0RTNIFNWUpLOTdTMUE9PQ==

dn: uid=GroupOne,ou=groups,dc=principal-to-group,dc=example,dc=org
objectClass: extensibleObject
objectClass: top
objectClass: groupMember
objectClass: group
objectClass: uidObject
uid: GroupOne
distinguishedName: uid=GroupOne,ou=groups,dc=principal-to-group,dc=example,dc=org
memberOf: uid=GroupFive,ou=subgroups,ou=groups,dc=principal-to-group,dc=example,dc=org

dn: uid=GroupFive,ou=subgroups,ou=groups,dc=principal-to-group,dc=example,dc=org
objectClass: extensibleObject
objectClass: top
objectClass: groupMember
objectClass: group
objectClass: uidObject
```

```
uid: GroupFive
distinguishedName: uid=GroupFive,ou=subgroups,ou=groups,dc=principal-to-
group,dc=example,dc=org
```

例11.22 LDIF の例: グループからプリンシパル

この例でも、ユーザー **TestUserOne** は **GroupOne** のメンバーで、**GroupOne** は **GroupFive** のメンバーですが、相互参照にはグループからユーザーへ属性 **uniqueMember** が使用されます。

グループメンバーシップの相互参照に使用される属性は繰り返しが可能で、**GroupFive** を確認すると、別のユーザー **TestUserFive** への参照があります (ここでは示されていません)。

```
dn: uid=TestUserOne,ou=users,dc=group-to-principal,dc=example,dc=org
objectClass: top
objectClass: inetOrgPerson
objectClass: uidObject
objectClass: person
objectClass: organizationalPerson
cn: Test User One
sn: Test User One
uid: TestUserOne
userPassword::
e1NTSEF9SJr0OTRDR1ItaHc1VVZQOEJvbXhUYjI1dkFVd1lQTmRLSEdzaWc9PQ==

dn: uid=GroupOne,ou=groups,dc=group-to-principal,dc=example,dc=org
objectClass: top
objectClass: groupOfUniqueNames
objectClass: uidObject
cn: Group One
uid: GroupOne
uniqueMember: uid=TestUserOne,ou=users,dc=group-to-principal,dc=example,dc=org

dn: uid=GroupFive,ou=subgroups,ou=groups,dc=group-to-principal,dc=example,dc=org
objectClass: top
objectClass: groupOfUniqueNames
objectClass: uidObject
cn: Group Five
uid: GroupFive
uniqueMember: uid=TestUserFive,ou=users,dc=group-to-principal,dc=example,dc=org
uniqueMember: uid=GroupOne,ou=groups,dc=group-to-principal,dc=example,dc=org
```

一般的なグループ検索

前述の 2 つの方法の例を確認する前に、両方の方法に共通する属性を定義する必要があります。

```
<group-search group-name="..." iterative="..." group-dn-attribute="..." group-name-attribute="..." >
...
</group-search>
```

- **group-name**: この属性は、ユーザーがメンバーであるグループのリストとして返されるグループ名に使用される形式を指定するために使用されます。これは、単純なグループ名またはグループの識別名になります。識別名が必要な場合は、この属性を **DISTINGUISHED_NAME** に

設定できます。デフォルトは **SIMPLE** です。

- **iterative**: ユーザーがメンバーであるグループを特定した後に、グループがメンバーであるグループを特定するため、グループを基に繰り返し検索するかどうかを指定するために使用される属性です。繰り返し検索が有効であると、他のグループのメンバーでないグループが見つかるか、サイクルが検出されるまで検索を続行します。デフォルトは **false** です。

巡回のグループメンバーシップは問題ではありません。検索済みグループの再検索を防ぐため、各検索の記録が保存されます。



重要

繰り返し検索が正しく実行されるようにするには、グループエントリーがユーザーエントリーと同じに見える必要があります。ユーザーがメンバーであるグループを特定するために使用された方法で、グループがメンバーであるグループを特定します。これは、グループ対グループのメンバーシップで相互参照に使用される属性の名前が変更されたり、参照の方向が変更されたりする場合は不可能です。

- **group-dn-attribute**: 属性が識別名であるグループのエントリーです。デフォルトは **dn** です。
- **group-name-attribute**: 属性が単純名であるグループのエントリーです。デフォルトは **uid** です。

例11.23 プリンシパルからグループへの設定例

前述の LDIF の例を基にした、ユーザーのグループを繰り返しロードする設定の例は次のとおりです。相互参照に使用される属性はユーザーの **memberOf** 属性です。

```
<authorization>
  <ldap connection="LocalLdap">
    <username-to-dn>
      <username-filter base-dn="ou=users,dc=principal-to-group,dc=example,dc=org"
recursive="false" attribute="uid" user-dn-attribute="dn" />
    </username-to-dn>
    <group-search group-name="SIMPLE" iterative="true" group-dn-attribute="dn" group-name-
attribute="uid">
      <principal-to-group group-attribute="memberOf" />
    </group-search>
  </ldap>
</authorization>
```

この設定で最も重要なことは、**principal-to-group** 要素が単一の属性で追加されていることです。

- **group-attribute**: ユーザーがメンバーであるグループの識別名と一致する、ユーザーエントリー上の属性名。デフォルトは **memberOf** です。

例11.24 グループからプリンシパルへの設定例

この例は、前述のグループからプリンシパルへの LDIF の例に対する繰り返し検索を示しています。

```
<authorization>
  <ldap connection="LocalLdap">
    <username-to-dn>
```

```

    <username-filter base-dn="ou=users,dc=group-to-principal,dc=example,dc=org"
recursive="false" attribute="uid" user-dn-attribute="dn" />
  </username-to-dn>
  <group-search group-name="SIMPLE" iterative="true" group-dn-attribute="dn" group-name-
attribute="uid">
    <group-to-principal base-dn="ou=groups,dc=group-to-principal,dc=example,dc=org"
recursive="true" search-by="DISTINGUISHED_NAME">
      <membership-filter principal-attribute="uniqueMember" />
    </group-to-principal>
  </group-search>
</ldap>
</authorization>

```

ここでは、**group-to-principal** 要素が追加されています。この要素は、ユーザーエントリーを参照するグループの検索がどのように実行されるかを定義するために使用されます。以下の属性が設定されます。

- **base-dn**: 検索を開始するために使用するコンテキストの識別名。
- **recursive**: サブコンテキストも検索されるかどうか。デフォルトは **false** です。
- **search-by**: 検索で使用するロール名の形式です。有効な値は **SIMPLE** および **DISTINGUISHED_NAME** です。デフォルトは **DISTINGUISHED_NAME** です。

group-to-principal 要素内に、相互参照を定義する **membership-filter** 要素があります。

- **principal-attribute**: ユーザーエントリーを参照する、グループエントリー上の属性名。デフォルトは **member** です。

[Report a bug](#)

11.9.9.8. スコープ指定ロール

スコープ指定ロールは、指定された1つ以上のサーバーグループまたはホストに対してのみ、1つの標準ロールのパーミッションを付与するユーザー定義のロールです。スコープ指定ロールにより、必要なサーバーグループやホストのみに限定されるパーミッションを管理ユーザーに付与できます。

Administrator または SuperUser ロールが割り当てられたユーザーがスコープ指定ロールを作成できます。

スコープ指定ロールは5つの特性によって定義されます。

1. 一意名。
2. ベースになる標準ロール。
3. サーバーグループまたはホストへ適用されるかどうか。
4. スコープ指定ロールが制限されるサーバーグループまたはホストのリスト。
5. すべてのユーザーが自動的に含まれるかどうか。デフォルトは **false** です。

スコープ指定ロールの作成後、標準ロールと同様にユーザーやグループへ割り当てできます。

スコープ指定ロールを作成しても、新しいパーミッションは定義できません。スコープ指定ロールは、既存ロールのパーミッションを制限されたスコープで適用するために使用されます。たとえば、単一のサーバーグループに制限される Deployer ロールを基にスコープ指定ロールを作成できます。

ロールは、ホストとサーバーグループの2つのスコープのみに限定されます。

ホストスコープ指定ロール

ホストスコープ指定されたロールは、そのロールのパーミッションを1つ以上のホストに制限します。これにより、関連する `/host=*` リソースツリーにアクセスできますが、他のホスト固有のリソースは表示されません。

サーバーグループスコープ指定ロール

サーバーグループスコープ指定のロールは、そのロールのパーミッションを1つ以上のサーバーグループに制限します。さらに、ロールパーミッションは、指定されたサーバーグループに関連するプロファイル、ソケットバインディンググループ、サーバー設定、およびサーバーリソースにも適用されます。サーバーグループに論理的に関連しないリソース内のサブリソースは、ユーザーに対して表示されません。

ホストおよびサーバーグループスコープ指定ロールは、その他の管理対象ドメイン設定に対して Monitor ロールのパーミッションを持ちます。

[Report a bug](#)

11.9.9.9. スコープ指定ロールの作成

スコープ指定ロールは、指定された1つ以上のサーバーグループまたはホストに対してのみ、1つの標準ロールのパーミッションを付与するユーザー定義のロールです。ここでは、スコープ指定ロールの作成方法を説明します。

この設定が行えるのは **SuperUser** または **Administrator** ロールのユーザーのみです。

管理コンソールのスコープ指定ロール設定は、以下の手順で確認できます。

1. 管理コンソールへログインします。
2. **Administration** タブをクリックします。
3. **Access Control** メニューを展開し、**Role Assignment** を選択します。
4. **ROLES** タブを選択し、そのタブ内にある **Scoped Roles** タブを選択します。

管理コンソールの **Scoped Roles** セクションは、現在設定されているスコープ指定ロールのリストが含まれるテーブルと、テーブルで現在選択されているロールの詳細を表示する **Selection** パネルの2つの主なエリアで構成されます。

スコープ指定ロールの設定タスクを実行する手順は次のとおりです。

手順11.29 新しいスコープ指定ロールの追加

1. 管理コンソールへログインします。
2. **Roles** タブの **Scoped Roles** エリアに移動します。
3. **Add** をクリックします。 **Add Scoped Role** ダイアログが表示されます。

4. 次の詳細を指定します。

- **Name:** 新しいスコープ指定ロールの一意名。
 - **Base Role:** このロールのパーミッションがベースとなるロール。
 - **Type:** このロールがホストまたはサーバーグループに制限されるかどうか。
 - **Scope:** ロールが制限されるホストまたはサーバーグループのリスト。複数のエントリーを選択できます。
 - **Include All:** このロールにすべてのユーザーが自動的に含まれるかどうか。デフォルトは no です。
5. **Save** ボタンをクリックすると、ダイアログが閉じられ、新規作成されたロールがテーブルに表示されます。

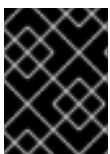
手順11.30 スコープ指定ロールの編集

1. 管理コンソールへログインします。
2. **Roles** タブの **Scoped Roles** エリアに移動します。
3. テーブル上で編集するスコープ指定ロールをクリックします。ロールの詳細がテーブルの下の **Selection** パネルに表示されます。
4. **Selection** パネルの **Edit** をクリックします。 **Selection** パネルが編集モードになります。
5. 変更する必要がある詳細を変更し、 **Save** ボタンをクリックします。 **Selection** パネルが以前の状態に戻ります。 **Selection** パネルとテーブルの両方に、新たに更新された詳細が表示されます。

手順11.31 スコープ指定ロールメンバーの表示

1. 管理コンソールへログインします。
2. **Roles** タブの **Scoped Roles** エリアに移動します。
3. テーブル上で、 **Members** を表示したいスコープ指定ロールをクリックし、 **Members** をクリックします。 **Members of role** ダイアログが表示されます。このダイアログには、ロールに含まれるまたは除外されるユーザーとグループが表示されます。
4. 情報を確認した後、 **Done** ボタンをクリックします。

手順11.32 スコープ指定ロールの削除



重要

ユーザーまたはグループが **Scoped Role** に割り当てられている場合、 **Scoped Role** は削除できません。ロールの割り当てを解除してから削除してください。

1. 管理コンソールへログインします。
2. **Roles** タブの **Scoped Roles** エリアに移動します。

3. テーブル上で、削除するスコープ指定ロールを選択します。
4. **Remove** ボタンをクリックします。 **Remove Scoped Role** ダイアログが表示されます。
5. **Confirm** ボタンをクリックします。ダイアログが閉じられ、ロールが削除されます。

[Report a bug](#)

11.9.10. 制約の設定

11.9.10.1. 機密性制約の設定

各機密性制約は、「機密」とみなされるリソースのセットを定義します。通常、機密リソースとは、パスワードなどの秘密にしなければならないリソースや、ネットワーキング、JVM 設定、システムプロパティーなどのサーバーに深刻な影響を与えるリソースのことです。アクセス制御システム自体も機密とみなされます。リソースの機密性は、特定リソースの読み取り、書き込み、またはアドレス指定の可能なロールを制限します。

機密性制約の設定は、管理 API の **/core-service=management/access=authorization/constraint=sensitivity-classification** にあります。

管理モデル内で、各機密性制約は **classification** として識別されます。classification (分類) は **types** にグループ化されます。39 個の分類が含まれ、それらは 13 個のタイプにグループ化されます。

機密性の制約を設定するには、**write-attribute** 操作を使用して **configured-requires-read**、**configured-requires-write**、または **configured-requires-addressable** 属性を設定します。操作のタイプを機密にするには、**true** を値として設定します。機密にしない場合は **false** を設定します。デフォルトではこれらの属性は設定されておらず、**default-requires-read**、**default-requires-write**、および **default-requires-addressable** の値が使用されます。configured 属性の値が設定されると、デフォルトの代わりにその値が使用されます。デフォルト値は変更できません。

例11.25 読み取りシステムプロパティーを機密操作にする

```
[domain@localhost:9999 /] cd /core-
service=management/access=authorization/constraint=sensitivity-
classification/type=core/classification=system-property
[domain@localhost:9999 classification=system-property] :write-attribute(name=configured-
requires-read, value=true)
{
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => {"main-server-group" => {"host" => {"master" => {
    "server-one" => {"response" => {"outcome" => "success"}},
    "server-two" => {"response" => {"outcome" => "success"}}
  }}}
}
[domain@localhost:9999 classification=system-property] :read-resource
{
  "outcome" => "success",
  "result" => {
    "configured-requires-addressable" => undefined,
    "configured-requires-read" => true,
    "configured-requires-write" => undefined,
    "default-requires-addressable" => false,
    "default-requires-read" => false,
```

```

"default-requires-write" => true,
"applies-to" => {
  "/host=master/system-property=" => undefined,
  "/host=master/core-service=platform-mbean/type=runtime" => undefined,
  "/server-group=*/system-property=" => undefined,
  "/host=master/server-config=*/system-property=" => undefined,
  "/host=master" => undefined,
  "/system-property=" => undefined,
  "/" => undefined
}
}
}
[domain@localhost:9999 classification=system-property]

```

どのロールがどの操作を実行できるかは、表11.6「機密性制約の設定結果」に説明されている属性の設定によって異なります。

表11.6 機密性制約の設定結果

Value	requires-read	requires-write	requires-addressable
true	読み取りは機密です。 Auditor、Administrator、 および SuperUser のみ読み 取りできます。	書き込みは機密です。 Administrator および SuperUser のみ書き込みで きます。	アドレス指定は機密です。 Auditor、Administrator、 および SuperUser のみアド レス可能です。
false	読み取りは機密ではありません。 すべての管理ユーザーが読み 取りできます。	書き込みは機密ではありません。 Maintainer、Administrator、 および SuperUser のみ書 き込みできます。 Deployers はアプリケーションリソース のアプリケーションを書き込 みできます。	アドレス指定は機密ではありません。 すべての管理ユーザーがアド レス指定できます。

[Report a bug](#)

11.9.10.2. アプリケーションリソース制約の設定

各アプリケーションリソース制約は、アプリケーションおよびサービスのデプロイメントに関連するリソース、属性、および操作のセットを定義します。アプリケーションリソース制約が有効になっていると、Deployer ロールの管理ユーザーは適用されるリソースへアクセスできます。

アプリケーション制約の設定は、管理モデルの `/core-service=management/access=authorization/constraint=application-classification/` にあります。

管理モデル内で、各アプリケーションリソース制約は **classification** として識別されます。classification (分類) は **types** にグループ化されます。14 個の分類が含まれ、それらは 8 つのタイプにグループ化されます。各分類には、分類設定が適用されるリソースパスパターンのリストである **applies-to** 要素が含まれます。

デフォルトでは、有効になっているアプリケーションリソース分類は **core** のみです。core にはデプロイメント、デプロイメントオーバーレイ、およびデプロイメント操作が含まれます。

アプリケーションリソースを有効にするには、**write-attribute** 操作を使用して、分類の **configured-application attribute** を **true** に設定します。アプリケーションリソースを無効にするには、この属性を **false** に設定します。デフォルトでは、この属性は設定されていないため、**default-application attribute** の値が使用されます。デフォルトの値は変更できません。

例11.26 logger-profile アプリケーションリソースの分類を有効にする

```
[domain@localhost:9999 /] cd /core-
service=management/access=authorization/constraint=application-
classification/type=logging/classification=logging-profile
[domain@localhost:9999 classification=logging-profile] :write-attribute(name=configured-
application, value=true)
{
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => {"main-server-group" => {"host" => {"master" => {
    "server-one" => {"response" => {"outcome" => "success"}},
    "server-two" => {"response" => {"outcome" => "success"}}
  }}}
}
[domain@localhost:9999 classification=logging-profile] :read-resource
{
  "outcome" => "success",
  "result" => {
    "configured-application" => true,
    "default-application" => false,
    "applies-to" => {"profile=*/subsystem=logging/logging-profile=" => undefined}
  }
}
[domain@localhost:9999 classification=logging-profile]
```

重要

アプリケーションリソース制約は、設定に一致するすべてのリソースに適用されます。たとえば、**Deployer** ユーザーに、あるデータソースリソースへのアクセスを許可し、他のデータソースリソースへのアクセスを拒否することはできません。このような分離レベルが必要な場合は、リソースを異なるサーバーグループに設定し、各グループに対して異なるスコープ指定の **Deployer** ロールを作成することが推奨されます。

[Report a bug](#)

11.9.10.3. Vault 式制約の設定

デフォルトでは、vault 式の読み書きは機密操作になっています。vault 式の制約を設定すると、読み取りと書き込みのどちらかまたは両方を非機密の操作にすることができます。この制約を変更すると、vault 式を読み書きできるロールの数を増やすことができます。

vault 式の制約は、管理モデルの **/core-service=management/access=authorization/constraint=vault-expression** にあります。

vault 式の制約を設定するには、**write-attribute** 操作を使用して **configured-requires-write** および **configured-requires-read** 属性を **true** または **false** に設定します。デフォルトでは、これらの属性は設定されていないため、**default-requires-read** および **default-requires-write** の値が使用されます。デフォルトの値は変更できません。

例11.27 vault 式への書き込みを非機密操作にする

```
[domain@localhost:9999 /] cd /core-service=management/access=authorization/constraint=vault-expression
[domain@localhost:9999 constraint=vault-expression] :write-attribute(name=configured-requires-write, value=false)
{
  "outcome" => "success",
  "result" => undefined,
  "server-groups" => {"main-server-group" => {"host" => {"master" => {
    "server-one" => {"response" => {"outcome" => "success"}},
    "server-two" => {"response" => {"outcome" => "success"}}
  }}}
}
[domain@localhost:9999 constraint=vault-expression] :read-resource
{
  "outcome" => "success",
  "result" => {
    "configured-requires-read" => undefined,
    "configured-requires-write" => false,
    "default-requires-read" => true,
    "default-requires-write" => true
  }
}
[domain@localhost:9999 constraint=vault-expression]
```

どのロールが vault 式へ読み書きできるかは、[表11.7「vault 式制約の設定結果」](#) に説明されている設定によって異なります。

表11.7 vault 式制約の設定結果

Value	requires-read	requires-write
true	読み取り操作は機密です。 Auditor 、 Administrator 、および SuperUser のみ読み取りできます。	書き込み操作は機密です。 Administrator および SuperUser のみ書き込みできます。
false	読み取り操作は機密ではありません。 すべての管理ユーザーが読み取りできます。	書き込み操作は機密ではありません。 Maintainer 、 Administrator 、および SuperUser が書き込みできます。 Deployers はアプリケーションリソースに Vault 式がある場合に書き込みできます。

[Report a bug](#)

11.9.11. 制約に関する参考情報

11.9.11.1. アプリケーションリソース制約に関する参考情報

タイプ: core

分類: deployment-overlay

- デフォルト: true
- PATH: /deployment-overlay=*
- PATH: /deployment=*
- PATH: /

操作:

upload-deployment-stream、full-replace-deployment、upload-deployment-url、upload-deployment-bytes

タイプ: datasources

分類: datasource

- デフォルト: false
- PATH: /deployment=*/subdeployment=*/subsystem=datasources/data-source=*
- PATH: /subsystem=datasources/data-source=*
- PATH: /subsystem=datasources/data-source=ExampleDS
- PATH: /deployment=*/subsystem=datasources/data-source=*

分類: jdbc-driver

- デフォルト: false
- PATH: /subsystem=datasources/jdbc-driver=*

分類: xa-data-source

- デフォルト: false
- PATH: /subsystem=datasources/xa-data-source=*
- PATH: /deployment=*/subsystem=datasources/xa-data-source=*
- PATH: /deployment=*/subdeployment=*/subsystem=datasources/xa-data-source=*

タイプ: logging

分類: logger

- デフォルト: false
- PATH: /subsystem=logging/logger=*
- PATH: /subsystem=logging/logging-profile=*/logger=*

分類: logging-profile

- デフォルト: false
- PATH: /subsystem=logging/logging-profile=*

タイプ: mail

分類: mail-session

- デフォルト: false
- PATH: /subsystem=mail/mail-session=*

タイプ: naming

分類: binding

- デフォルト: false
- PATH: /subsystem=naming/binding=*

タイプ: resource-adapters

分類: resource-adapters

- デフォルト: false
- PATH: /subsystem=resource-adapters/resource-adapter=*

タイプ: security

分類: security-domain

- デフォルト: false
- PATH: /subsystem=security/security-domain=*

[Report a bug](#)

11.9.11.2. 機密性制約に関する参考情報

タイプ: core

分類: access-control

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /core-service=management/access=authorization
- PATH: /subsystem=jmx ATTRIBUTE: non-core-mbean-sensitivity

分類: credential

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=mail/mail-session=*/server=pop3 ATTRIBUTE: username , password
- PATH: /subsystem=mail/mail-session=*/server=imap ATTRIBUTE: username , password
- PATH: /subsystem=datasources/xa-data-source=* ATTRIBUTE: user-name, recovery-username, password, recovery-password
- PATH: /subsystem=mail/mail-session=*/custom=* ATTRIBUTE: username, password
- PATH: /subsystem=datasources/data-source=*" ATTRIBUTE: user-name, password
- PATH: /subsystem=remoting/remote-outbound-connection=*" ATTRIBUTE: username
- PATH: /subsystem=mail/mail-session=*/server=smtp ATTRIBUTE: username, password
- PATH: /subsystem=web/connector=*/configuration=ssl ATTRIBUTE: key-alias, password
- PATH: /subsystem=resource-adapters/resource-adapter=*/connection-definitions=*" ATTRIBUTE: recovery-username, recovery-password

分類: domain-controller

- requires-addressable: false
- requires-read: false
- requires-write: true

分類: domain-names

- requires-addressable: false
- requires-read: false
- requires-write: true

分類: extensions

- requires-addressable: false

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /extension=*

分類: jvm

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=platform-mbean/type=runtime ATTRIBUTE: input-arguments, boot-class-path, class-path, boot-class-path-supported, library-path

分類: management-interfaces

- requires-addressable: false
- requires-read: false
- requires-write: true
- /core-service=management/management-interface=native-interface
- /core-service=management/management-interface=http-interface

分類: module-loading

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=module-loading

分類: patching

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=patching/addon=*
- PATH: /core-service=patching/layer=*
- PATH: /core-service=patching

分類: read-whole-config

- requires-addressable: false

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: / OPERATION: read-config-as-xml

分類: security-domain

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /subsystem=security/security-domain=*

分類: security-domain-ref

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /subsystem=datasources/xa-data-source=* ATTRIBUTE: security-domain
- PATH: /subsystem=datasources/data-source=* ATTRIBUTE: security-domain
- PATH: /subsystem=ejb3 ATTRIBUTE: default-security-domain
- PATH: /subsystem=resource-adapters/resource-adapter=*/connection-definitions=*
ATTRIBUTE: security-domain, recovery-security-domain, security-application, security-domain-and-application

分類: security-realm

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /core-service=management/security-realm=*

分類: security-realm-ref

- requires-addressable: true
- requires-read: true
- requires-write: true
- PATH: /subsystem=remoting/connector=* ATTRIBUTE: security-realm

• PATH: /subsystem=remoting/connector=* ATTRIBUTE: security-realm

- PATH: /core-service=management/management-interface=native-interface ATTRIBUTE: security-realm
- PATH: /core-service=management/management-interface=http-interface ATTRIBUTE: security-realm
- PATH: /subsystem=remoting/remote-outbound-connection=* ATTRIBUTE: security-realm

分類: security-vault

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=vault

分類: service-container

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=service-container

分類: snapshots

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: / ATTRIBUTE: take-snapshot, list-snapshots, delete-snapshot

分類: socket-binding-ref

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: /subsystem=mail/mail-session=*/server=pop3 ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=mail/mail-session=*/server=imap ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=remoting/connector=* ATTRIBUTE: socket-binding
- PATH: /subsystem=web/connector=* ATTRIBUTE: socket-binding
- PATH: /subsystem=remoting/local-outbound-connection=* ATTRIBUTE: outbound-socket-binding-ref

- PATH: /subsystem=remoting/local-outbound-connection=* ATTRIBUTE: outbound-socket-binding-ref
- PATH: /socket-binding-group=*/local-destination-outbound-socket-binding=* ATTRIBUTE: socket-binding-ref
- PATH: /subsystem=remoting/remote-outbound-connection=* ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=mail/mail-session=*/server=smtp ATTRIBUTE: outbound-socket-binding-ref
- PATH: /subsystem=transactions ATTRIBUTE: process-id-socket-binding, status-socket-binding, socket-binding

分類: socket-config

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /interface=* OPERATION: resolve-internet-address
- PATH: /core-service=management/management-interface=native-interface ATTRIBUTE: port, interface, socket-binding
- PATH: /socket-binding-group=*
- PATH: /core-service=management/management-interface=http-interface ATTRIBUTE: port, secure-port, interface, secure-socket-binding, socket-binding
- PATH: / OPERATION: resolve-internet-address
- PATH: /subsystem=transactions ATTRIBUTE: process-id-socket-max-ports

分類: system-property

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /core-service=platform-mbean/type=runtime ATTRIBUTE: system-properties
- PATH: /system-property=*
- PATH: / OPERATION: resolve-expression

タイプ: datasources

分類: data-source-security

- requires-addressable: false

- requires-read: true
- requires-write: true
- PATH: /subsystem=datasources/xa-data-source=* ATTRIBUTE: user-name, security-domain, password
- PATH: /subsystem=datasources/data-source=* ATTRIBUTE: user-name, security-domain, password

タイプ: jdr

分類: jdr

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /subsystem=jdr OPERATION: generate-jdr-report

タイプ: jmx

分類: jmx

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /subsystem=jmx

タイプ: mail

分類: mail-server-security

- requires-addressable: false
- requires-read: false
- requires-write: true
- PATH: /subsystem=mail/mail-session=*/server=pop3 ATTRIBUTE: username, tls, ssl, password
- PATH: /subsystem=mail/mail-session=*/server=imap ATTRIBUTE: username, tls, ssl, password
- PATH: /subsystem=mail/mail-session=*/custom=* ATTRIBUTE: username, tls, ssl, password
- PATH: /subsystem=mail/mail-session=*/server=smtp ATTRIBUTE: username, tls, ssl, password

タイプ: naming

分類: jndi-view

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=naming OPERATION: jndi-view

分類: naming-binding

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: /subsystem=naming/binding=*

タイプ: remoting

分類: remoting-security

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=remoting/connector=* ATTRIBUTE: authentication-provider, security-realm
- PATH: /subsystem=remoting/remote-outbound-connection=* ATTRIBUTE: username, security-realm
- PATH: /subsystem=remoting/connector=*/security=sasl

タイプ: resource-adapters

分類: resource-adapter-security

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=resource-adapters/resource-adapter=*/connection-definitions=*
ATTRIBUTE: security-domain, recovery-username, recovery-security-domain, security-application, security-domain-and-application, recovery-password

タイプ: security

分類: misc-security

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=security ATTRIBUTE: deep-copy-subject-mode

タイプ: web

分類: web-access-log

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: /subsystem=web/virtual-server=*/configuration=access-log

分類: web-connector

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: /subsystem=web/connector=*

分類: web-ssl

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=web/connector=*/configuration=ssl

分類: web-sso

- requires-addressable: false
- requires-read: true
- requires-write: true
- PATH: /subsystem=web/virtual-server=*/configuration=sso

分類: web-valve

- requires-addressable: false
- requires-read: false
- requires-write: false
- PATH: /subsystem=web/valve=*

[Report a bug](#)

11.10. ネットワークセキュリティー

11.10.1. 管理インターフェースのセキュア化

一般的な開発シナリオでは、セキュア化されていない JBoss EAP 6 を管理インターフェースで実行し、迅速に設定を変更できるようにします。

実稼働に向けた開発段階では、少なくとも以下の方法で管理インターフェースをセキュアにします。

- [「JBoss EAP 6 が使用するネットワークインターフェースの指定」](#)
- [「JBoss EAP 6 で動作可能なネットワークファイアウォールの設定」](#)

また、デフォルトのサイレントローカル認証モードを使用すると、ローカルクライアント (サーバーマシン側の) はユーザー名やパスワードを必要とせずに管理 CLI へ接続できます。これは、ローカルユーザーや管理 CLI スクリプトには便利な機能です。この機能を無効にする場合は [「デフォルトセキュリティーレームからのサイレント認証の削除」](#) を参照してください。

[Report a bug](#)

11.10.2. JBoss EAP 6 が使用するネットワークインターフェースの指定

概要

サービスを必要とするクライアントにのみアクセスできるようにサービスを分離すると、ネットワークのセキュリティーが強化されます。JBoss EAP 6 には、デフォルト設定の 2 つのインターフェースが含まれ、どちらもデフォルトで IP アドレス **127.0.0.1** または **localhost** にバインドされます。インターフェースの 1 つは **management** と呼ばれ、管理コンソール、CLI、および API によって使用されます。他のインターフェースは **public** と呼ばれ、アプリケーションをデプロイするために使用されます。これらのインターフェースは、特別なものではありませんが、作業を始める土台として提供されます。

management インターフェースはデフォルトでポート **9990** および **9999** を使用し、**public** インターフェースはポート **8080** または **8443** (HTTPS を使用する場合) を使用します。

管理インターフェース、パブリックインターフェース、またはその両方の IP アドレスを変更できます。

**警告**

リモートホストからアクセス可能な他のネットワークインターフェースに管理インターフェースを公開する場合は、セキュリティの問題に注意してください。ほとんどの場合、管理インターフェースにリモートアクセスを提供することはお勧めしません。

1. JBoss EAP 6 を停止します。

オペレーティングシステムに適切な方法で割り込みを送信して JBoss EAP 6 を停止します。JBoss EAP 6 をフォアグラウンドアプリケーションとして実行している場合、通常は **Ctrl+C** を押してこれを行います。

2. バインドアドレスを指定して JBoss EAP 6 を再起動します。

-b コマンドラインスイッチを使用して、特定のインターフェースで JBoss EAP 6 を起動します。

例11.28 パブリックインターフェースの指定

```
EAP_HOME/bin/domain.sh -b 10.1.1.1
```

例11.29 管理インターフェースの指定

```
EAP_HOME/bin/domain.sh -bmanagement=10.1.1.1
```

例11.30 各インターフェースへの異なるアドレスの指定

```
EAP_HOME/bin/domain.sh -bmanagement=127.0.0.1 -b 10.1.1.1
```

例11.31 すべてのネットワークインターフェースへのパブリックインターフェースのバインド

```
EAP_HOME/bin/domain.sh -b 0.0.0.0
```

XML 設定ファイルを直接編集してデフォルトのバインドアドレスを変更できますが、これを行うと **-b** コマンドラインスイッチを使用してランタイム時に IP アドレスを指定できなくなるため、お勧めしません。この作業を行う場合は、XML ファイルを編集する前に JBoss EAP 6 を完全に停止する必要があります。

[Report a bug](#)

11.10.3. 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。



注記

数値ポートオフセットは、同じ物理サーバーで複数のサーバーを実行する場合にポートの競合を緩和するために設定できます。サーバーが数値のポートオフセットを使用する場合は、サーバグループのソケットバインディンググループに対する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、サーバーは **100** のポートオフセットを使用する場合、その 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**

表11.8 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。	○	○	○	○
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。	○	○	○	○
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。	○	○	○	○
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。	○	○	×	×
jacorb-ssl	3529		SSL 暗号化 CORBA サービス。	○	○	×	×

名前	ポート	マルチ キャスト ポート	説明	full-ha- sockets	full- sockets	ha- socket	standar d- socket
jgroups - diagno stics		7500	マルチキャスト。HA クラスターでのピア の検出に使用されま す。管理インター フェースを使用し ても設定できません。	○	×	○	×
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。	○	×	○	×
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。	○	×	○	×
jgroups -tcp-fd	57600		TCP を介した HA 障 害検出に使用されま す。	○	×	○	×
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。	○	×	○	×
jgroups -udp-fd	54200		UDP を介した HA 障 害検出に使用されま す。	○	×	○	×
messa ging	5445		JMS サービス。	○	○	×	×
messa ging- group			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。	○	○	×	×
messa ging- throug hput	5455		JMS Remoting によ り使用されます。	○	○	×	×

名前	ポート	マルチ キャスト ポート	説明	full-ha- sockets	full- sockets	ha- socket	standar d- socket
mod_cl uster		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。	○	×	○	×
osgi- http	8090		OSGi サブシステムを使用する内部コンポーネントにより使用されます。管理インターフェースを使用して設定できません。	○	○	○	○
remotin g	4447		リモート EJB の呼び出しに使用されます。	○	○	○	○
txn- recover y- environ ment	4712		JTA トランザクションリカバリーマネージャ。	○	○	○	○
txn- status- manag er	4713		JTA / JTS トランザクションマネージャ。	○	○	○	○

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 によって使用されるポート

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。

[Report a bug](#)

11.10.4. JBoss EAP 6 で動作可能なネットワークファイアウォールの設定

概要

ほとんどの本番環境では、ネットワークセキュリティ全体の方針の一部としてファイアウォールを使用します。複数のサーバーインスタンスがお互いに通信したり、Web サーバーやデータベースなどの外部サービスと通信したりする必要がある場合は、ファイアウォールでこれを考慮する必要があります。

適切に管理されたファイアウォールでは、操作に必要なポートのみが開かれ、特定の IP アドレス、サブネット、およびネットワークプロトコルに対するポートへのアクセスが制限されます。

本書では、ファイアウォールの完全な説明は範囲外になります。

前提条件

- 開く必要があるポートを判断します。
- ファイアウォールソフトウェアについて理解する必要があります。この手順では、Red Hat Enterprise Linux 6 の **system-config-firewall** コマンドを使用します。Microsoft Windows Server には、ファイアウォールが組み込まれ、各プラットフォーム用の複数のサードパーティー製ファイアウォールソリューションが利用可能です。Microsoft Windows Server では、PowerShell を使用してファイアウォールを設定できます。

前提条件

この手順では、以下を前提として環境のファイアウォールを設定します。

- オペレーティングシステムは Red Hat Enterprise Linux 6 です。
- JBoss EAP 6 はホスト **10.1.1.2** で実行されます。オプションで、サーバーには独自のファイアウォールがあります。
- ネットワークファイアウォールサーバーは、ホスト **10.1.1.1** のインターフェース **eth0** で実行され、外部インターフェース **eth1** を持ちます。
- ポート **5445** (JMS で使用されるポート) のトラフィックを JBoss EAP 6 に転送します。ネットワークファイアウォールで他のトラフィックは許可されません。

手順11.33 ネットワークファイアウォールと JBoss EAP 6 が連携するための管理

1. 管理コンソールへのログイン

管理コンソールにログインします。デフォルトでは、<http://localhost:9990/console/> で実行されます。

2. ソケットバインディンググループが使用するソケットバインディングを決定します。

- a. 管理コンソールの上部にある **Configuration** ラベルをクリックします。
- b. **General Configuration** メニューを展開します。 **Socket Binding** を選択します。
- c. **Socket Binding Declarations** 画面が表示されます。最初に、 **standard-sockets** グループが表示されます。他のグループを選択する場合は、右側のコンボボックスから選択します。



注記

スタンドアロンサーバーを使用する場合は、1つのソケットバインディンググループのみが存在します。

ソケット名とポートのリストが表示されます (1 ページあたり 8 つの値)。テーブルの矢印ナビゲーションを使用してページを移動できます。

3. 開く必要があるポートを判断します。

特定ポートの機能および環境の要件によっては、一部のポートをファイアウォール上で開く必要がある場合があります。

4. JBoss EAP 6 にトラフィックを転送するようファイアウォールを設定します。

以下の手順を実行して、必要なポートでトラフィックを許可するようネットワークファイアウォールを設定します。

- a. root ユーザーとしてファイアウォールマシンにログインし、コマンドプロンプトにアクセスします。
- b. **system-config-firewall** コマンドを実行してファイアウォール設定ユーティリティを起動します。ファイアウォールシステムにログインした方法に応じて、GUI またはコマンドラインユーティリティが起動します。このタスクでは、SSH 経由でコマンドラインインターフェースを使用してログインしていることを前提とします。
- c. キーボードで **TAB** キーを使用して **Customize** ボタンに移動し、**ENTER** キーを押します。**Trusted Services** 画面が表示されます。
- d. どの値も変更せずに、**TAB** キーを使用して **Forward** ボタンに移動し、**ENTER** を押して次の画面に進みます。**Other Ports** 画面が表示されます。
- e. **TAB** キーを使用して **<Add>** ボタンに移動し、**ENTER** を押します。**Port and Protocol** 画面が表示されます。
- f. **Port / Port Range** フィールドに **5445** と入力し、**TAB** キーを使用して **Protocol** フィールドに移動し、**tcp** と入力します。**TAB** キーを使用して **OK** ボタンに移動し、**ENTER** を押します。
- g. **TAB** キーを使用して、**Forward** ボタンに移動し、**Port Forwarding** 画面にアクセスします。
- h. **TAB** キーを使用して **<Add>** ボタンに移動し、**ENTER** キーを押します。
 - i. 以下の値を入力してポート **5445** のポート転送を設定します。
 - 送信元インターフェース: **eth1**
 - プロトコル: **tcp**
 - ポート/ポート範囲: **5445**
 - 宛先 IP アドレス: **10.1.1.2**
 - ポート/ポート範囲: **5445**
- TAB** キーを使用して **OK** ボタンに移動し、**ENTER** を押します。
- j. **TAB** キーを使用して **Close** ボタンに移動し、**ENTER** を押します。
- k. **TAB** キーを使用して **OK** ボタンに移動し、**ENTER** を押します。変更内容を適用するには、警告を読み、**Yes** をクリックします。

5. JBoss EAP 6 ホストでファイアウォールを設定します。

一部の組織では、JBoss EAP 6 サーバー自体でファイアウォールを設定し、運用に必要なすべてのポートを閉じます。「JBoss EAP 6 により使用されるネットワークポート」を参照して開くポートを決定し、残りのポートを閉じます。Red Hat Enterprise Linux 6 のデフォルトの設

定では、Secure Shell (SSH) に使用される **22** と、マルチキャスト DNS に使用される **5353** 以外のポートが閉じられます。ポートを設定する場合は、誤ってロックアウトされないよう物理的にアクセスしてください。

結果

ファイアウォール設定で指定したとおり、ファイアウォールによって内部 JBoss EAP 6 サーバーへトラフィックが転送されるようになります。サーバーでファイアウォールを有効にした場合は、アプリケーションを実行するために必要なポート以外はすべて閉じられます。

手順11.34 PowerShell を使用した Microsoft Windows 上でのファイアウォールの設定

1. デバッグの目的でファイアウォールを停止し、現在のネットワークの挙動がファイアウォールの設定と関係しているかどうかを判断します。

```
Start-Process "$psHome\powershell.exe" -Verb Runas -ArgumentList '-command "NetSh Advfirewall set allprofiles state off"'
```

2. ポート 23364 上で UDP 接続を許可します。以下に例を示します。

```
Start-Process "$psHome\powershell.exe" -Verb Runas -ArgumentList '-command "NetSh Advfirewall firewall add rule name="UDP Port 23364" dir=in action=allow protocol=UDP localport=23364"'
Start-Process "$psHome\powershell.exe" -Verb Runas -ArgumentList '-command "NetSh Advfirewall firewall add rule name="UDP Port 23364" dir=out action=allow protocol=UDP localport=23364"'
```

手順11.35 mod_cluster アドバタイズを許可するよう Red Hat Enterprise Linux 7 でファイアウォールを設定

- Red Hat Enterprise Linux 7 で mod_cluster のアドバタイズを有効にするには、以下のようにファイアウォールで UDP ポートを有効にする必要があります。

```
firewall-cmd --permanent --zone=public --add-port=23364/udp
```



注記

UDP マルチキャストをアドバタイズする mod_cluster バランサーのデフォルトアドレスおよびポートは 224.0.1.105:23364 です。

[Report a bug](#)

11.11. JAVA セキュリティーマネージャー

11.11.1. Java Security Manager

Java Security Manager

Java Security Manager は、Java 仮想マシン (JVM) サンドボックスの外部境界を管理するクラスで、JVM 内で実行するコードが JVM 外のリソースと対話する方法を制御します。Java Security Manager が有効な場合、Java API は安全でない可能性のある操作を行う前に Security Manager が承認するか確認します。

Java Security Manager は、セキュリティーポリシーを使用して該当するアクションを許可または拒否するかどうかを決定します。

[Report a bug](#)

11.11.2. Java Security Manager 内での JBoss EAP 6 の実行

Java Security Manager ポリシーを指定するには、ブートストラッププロセス中にドメインまたはサーバーインスタンスに渡す Java オプションを編集する必要があります。このため、**domain.sh** スクリプトまたは **standalone.sh** スクリプトにパラメーターをオプションとして渡すことはできません。次の手順を実行すると、インスタンスが Java Security Manager ポリシー内で実行されるよう設定できます。

前提条件

- この手順を実行する前に、Java Development Kit (JDK) に含まれる **policytool** コマンド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。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。この代わりに、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書き、**EAP_HOME/bin/server.policy** として手作業で保存することもできます。
- 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。

複数のシステムにドメインメンバーが分散されている場合は、ドメインの各物理ホストまたはインスタンスに対して次の手順を実行してください。

手順11.36 JBoss EAP 6 向けのセキュリティーマネージャーの設定

1. 設定ファイルを開きます。

編集のために設定ファイルを開きます。このファイルの場所は、管理対象ドメインとスタンドアロンサーバーのどちらを使用しているかによって異なります。このファイルは、サーバーまたはドメインを起動するために使用される実行可能ファイルではありません。

◦ 管理対象ドメイン

- Linux の場合: **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

- Linux の場合: **EAP_HOME/bin/standalone.conf**
- Windows の場合: **EAP_HOME\bin\standalone.conf.bat**

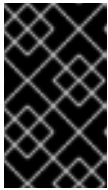
2. ファイルに Java オプションを追加します。

確実に Java オプションが使用されるようにするため、以下で始まるコードブロックに Java オプションを追加します。

```
if [ "x$JAVA_OPTS" = "x" ]; then
```

-Djava.security.policy の値を編集して、セキュリティーポリシーの場所を指定できます。改行を入れずに1行で指定する必要があります。**-Djava.security.policy** プロパティーを設定するときに **==** を使用して指定すると、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

のみを使用します。`=`を使用して指定すると、セキュリティーマネージャーは指定されたポリシーと **JAVA_HOME/lib/security/java.security** の **policy.url** セクションに指定されたポリシーを一緒に使用します。



重要

JBoss Enterprise Application Platform 6.2.2 およびそれ以降のリリースでは、システムプロパティー **jboss.modules.policy-permissions** を **true** に設定する必要があります。

例11.32 domain.conf

```
JAVA_OPTS="$JAVA_OPTS -Djava.security.manager -
Djava.security.policy== $PWD/server.policy -Djboss.home.dir=/path/to/EAP_HOME -
Djboss.modules.policy-permissions=true"
```

例11.33 domain.conf.bat

```
set "JAVA_OPTS=%JAVA_OPTS% -Djava.security.manager -
Djava.security.policy==\path\to\server.policy -Djboss.home.dir=\path\to\EAP_HOME -
Djboss.modules.policy-permissions=true"
```

例11.34 standalone.conf

```
JAVA_OPTS="$JAVA_OPTS -Djava.security.manager -
Djava.security.policy== $PWD/server.policy -Djboss.home.dir=$JBOSS_HOME -
Djboss.modules.policy-permissions=true"
```

例11.35 standalone.conf.bat

```
set "JAVA_OPTS=%JAVA_OPTS% -Djava.security.manager -
Djava.security.policy==\path\to\server.policy -Djboss.home.dir=%JBOSS_HOME% -
Djboss.modules.policy-permissions=true"
```

3. **ドメインサーバーを起動します。**
ドメインまたはサーバーを通常どおり起動します。

[Report a bug](#)

11.11.3. Java Security Manager のポリシー

Security Policy

コードの異なるクラスに対する定義済みのパーミッション。Java Security Manager はアプリケーションがリクエストしたアクションとセキュリティポリシーを比較します。ポリシーによってアクションが許可される場合、Java Security Manager はアクションの実行を許可します。ポリシーによってアクションが許可されない場合は、Java

Security Managerはこのアクションを拒否します。セキュリティポリシーは、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を基にパーミッションを定義できます。

使用する Java Security Manager やセキュリティポリシーは、Java 仮想マシンのオプション `java.security.manager` や `java.security.policy` を使用して設定されます。

基本情報

セキュリティポリシーのエントリは、**policytool** に関係のある以下の設定要素から構成されます。

CodeBase

コードの元の URL の場所 (ホストとドメインの情報以外)。オプションのパラメーターです。

SignedBy

コードを署名するためにプライベートキーが使用された署名者を参照するキーストアで使用されたエイリアス。これは、単一値またはカンマ区切りの値リストになります。オプションのパラメーターです。省略された場合は、署名の有無に関わらず Java Security Manager に影響はありません。

Principal

principal_type と **principal_name** のペアのリスト。これは、実行スレッドのプリンシパルセット内に存在する必要があります。Principals エントリは任意です。このエントリを省略すると、実行スレッドのプリンシパルによる Java Security Manager への影響はありません。

Permissions

permission は、コードに与えられるアクセス権です。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。

[Report a bug](#)

11.11.4. Java Security Manager ポリシーの記述

はじめに

ほとんどの JDK および JRE ディストリビューションには、Java Security Manager セキュリティポリシーを作成および編集するための **policytool** という名前のアプリケーションが含まれます。**policytool** の詳細については、<http://docs.oracle.com/javase/6/docs/technotes/tools/> を参照してください。

手順11.37 新しい Java Security Manager ポリシーの設定

1. **policytool** を起動します。
policytool ツールを次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
スタートメニューまたは Java インストールの `bin\` から、**policytool.exe** を実行します。
場所は異なることがあります。
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、**Done** をクリックします。

3. **既存のポリシーを編集します。**

既存のポリシーのリスト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。**

既存のポリシーのリストからポリシーを選択し、**Remove Policy Entry** ボタンを選択します。

[Report a bug](#)

11.11.5. Security Manager ポリシーのデバッグ

デバッグ情報を有効にすると、セキュリティーポリシーに関する問題のトラブルシューティングに便利です。**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。コマンド **java -Djava.security.debug=help** は、すべてのデバッグオプションのヘルプ出力を表示します。デバッグレベルを **all** に設定すると、原因不明の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ときに役に立ちますが、一般的な使用には多すぎる量の情報が表示されます。一般的に適切なデフォルト値は **access:failure** です。

手順11.38 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。

- 管理対象ドメインで JBoss EAP 6 インスタンスが実行されている場合、以下の行は Linux では **bin/domain.conf** ファイル、Windows では **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、以下の行は Linux では **bin/standalone.conf** ファイル、Windows では **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。

[Report a bug](#)

11.12. SSL 暗号化

11.12.1. JBoss EAP 6 Web サーバーでの SSL 暗号化の実装

はじめに

多くの Web アプリケーションでは、クライアントとサーバー間で SSL 暗号化接続 (**HTTPS** 接続とも呼ばれます) が必要です。以下の手順を使用すると、サーバーまたはサーバーグループで **HTTPS** を有効にできます。



警告

Red Hat は、影響す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 を明示的に無効化することを推奨しています。

前提条件

- SSL 暗号化キーセットと SSL 暗号化証明書。これらは、証明書署名認証局から購入したり、コマンドラインユーティリティーを使用して生成したりできます。Red Hat Enterprise Linux のユーティリティーを使用して暗号化キーを生成するには、[「SSL 暗号化キーおよび証明書の生成」](#)を参照してください。
- 特定の環境と設定に関する以下の詳細。
 - 証明書ファイルが保存されている完全なディレクトリー名。
 - 暗号化キーの暗号化パスワード。
- 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に接続する稼働中の管理 CLI。
- 適切な暗号化スイートを選択します。

暗号化スイート

暗号化スイートを構成するために使用される暗号プリミティブは複数あります。推奨される暗号プリミティブは最初の表に記載されています。2 つ目の表には、既存のソフトウェアとの互換性を維持するために使用できる **可能性がある** 暗号プリミティブが記載されていますが、推奨されるプリミティブよりも安全性が低くなります。



警告

Red Hat は、厳選された強力な暗号を **cipher-suite** に使用することを推奨します。弱い暗号を使用するとセキュリティリスクが増大します。互換性の問題がある可能性があるため、特定の暗号化スイートを決定する前に JDK ベンダーのドキュメントを確認してください。

表11.9 推奨される暗号プリミティブ

2048 ビットキーと OAEP を使用する RSA
CBC モードの AES-128

SHA-256
HMAC-SHA-256
HMAC-SHA-1

表11.10 その他の暗号プリミティブ

1024 以上のキーサイズとレガシーパディングを使用する RSA
AES-192
AES-256
3DES (トリプル DES で、2 つまたは 3 つの 56 ビットキー)
RC4 (非推奨)
SHA-1
HMAC-MD5

コネクターの SSL プロパティに設定できるパラメーターの完全リストは、「[SSL コネクターリファレンス](#)」を参照してください。



注記

この手順では、管理対象ドメインを使用する JBoss EAP 6 設定に適切なコマンドを使用します。スタンドアロンサーバーを使用する場合は、管理 CLI コマンドの先頭から **/profile=default** を削除して管理 CLI コマンドを変更します。



警告

Red Hat は、影響す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 を明示的に無効化することを推奨しています。

手順11.39 JBoss Web Server が HTTPS を使用するよう設定

1. 新しい HTTPS コネクターを追加します。

https スキームと **https** ソケットバインディング (デフォルトは **8443**) を使用し、セキュアとして設定される **HTTPS** という名前のセキュアなコネクターを作成します。

```
/profile=default/subsystem=web/connector=HTTPS/:add(socket-binding=https,scheme=https,protocol=HTTP/1.1,secure=true)
```

2. SSL 暗号化証明書およびキーを設定します。

例の値を独自の値に置き換え、SSL 証明書を設定します。この例では、キーストアはサーバー設定ディレクトリー (管理対象ドメインでは **EAP_HOME/domain/configuration/**) へコピーされることを前提としています。

```
/profile=default/subsystem=web/connector=HTTPS/ssl=configuration:add(name=https,certificate-key-file="{jboss.server.config.dir}/keystore.jks",password=SECRET, key-alias=KEY_ALIAS, cipher-suite=CIPHERS)
```

3. プロトコルを TLSv1 に設定します。

```
/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=protocol,value=TLSv1)
```

4. アプリケーションをデプロイします。

設定したプロファイルを使用するサーバーグループにアプリケーションをデプロイします。スタンドアロンサーバーを使用する場合、アプリケーションをサーバーにデプロイします。アプリケーションに対する HTTPS 要求は新しい SSL 暗号化接続を使用します。

[Report a bug](#)

11.12.2. SSL 暗号化キーおよび証明書の生成

SSL で暗号化された HTTP 接続 (HTTPS) や、他のタイプの SSL で暗号化された通信を使用するには、署名された暗号化証明書が必要です。証明書を認証局 (CA) から購入したり、自己署名証明書を使用したりできます。自己署名証明書を信頼できるとみなすサードパーティーは少数ですが、内部テストを目的とした使用には適しています。

この手順を実行すると、Red Hat Enterprise Linux で利用可能なユーティリティを使用して自己署名証明書を作成できます。

前提条件

- Java Development Kit 実装で提供される **keytool** ユーティリティが必要です。このコマンドは、Red Hat Enterprise Linux 上の OpenJDK により **/usr/bin/keytool** にインストールされます。
- **keytool** コマンドの構文およびパラメーターについて理解してください。この手順では、非常に一般的な手順を実行します。本書では、SSL 証明書または **keytool** コマンドに特有の説明は範囲外となります。

手順11.40 SSL 暗号化キーおよび証明書の生成

1. パブリックキーおよびプライベートキーとともにキーストアを生成します。

以下のコマンドを実行し、**jboss** というエイリアスを持つ **server.keystore** という名前のキーストアをカレントディレクトリーに生成します。

```
keytool -genkeypair -alias jboss -keyalg RSA -keystore server.keystore -storepass mykeystorepass --dname "CN=jsmith,OU=Engineering,O=mycompany.com,L=Raleigh,S=NC,C=US"
```

この keytool コマンドで使用されるパラメーターの説明は次のとおりです。

パラメーター	説明
-genkeypair	keytool コマンドは公開鍵と秘密鍵が含まれるキーペアを生成します。
-alias	キーストアのエイリアス。この値は任意ですが、エイリアス jboss はデフォルトで JBoss Web サーバーによって使用されます。
-keyalg	キーペア生成のアルゴリズム。この例では RSA になります。
-keystore	キーストアファイルの名前と場所。デフォルトの場所はカレントディレクトリーです。選択する名前は任意です。この例では、ファイルの名前は server.keystore になります。
-storepass	このパスワードは、キーの読み取りを可能にするためキーストアに対して認証を行うために使用されます。パスワードは 6 文字以上である必要があり、キーストアがアクセスされた時に提供しなければなりません。この例では、 mykeystorepass が使用されています。このパラメーターを省略すると、コマンドの実行時に入力するよう要求されます。
-keypass	<p>実際の鍵のパスワードです。</p> <div>  <div> <p>注記</p> <p>実装の制限により、ストアと同じパスワードを使用する必要があります。</p> </div> </div>

パラメーター	説明
--dname	<p>キーの識別名を記述する引用符で囲まれた文字列 (例: "CN=jsmith,OU=Engineering,O=mycompany.com,L=Raleigh,C=US")。以下のコンポーネントが連結された文字列になります。</p> <ul style="list-style-type: none"> ○ CN - 共通名またはホスト名。ホスト名が jsmith.mycompany.com の場合、CN は jsmith になります。 ○ OU - 組織単位 (例: Engineering)。 ○ O - 組織名 (例: mycompany.com)。 ○ L - 地域 (例: Raleigh または London)。 ○ S - 州 (例: NC)。このパラメーターの使用は任意です。 ○ C - 2 文字の国コード (例: US または UK)。

上記のコマンドを実行すると、次の情報が要求されます。

- コマンドラインで **-storepass** パラメーターを使用しなかった場合、キーストアのパスワードを入力するよう要求されます。次に要求されたら新しいパスワードを再入力します。
- コマンドラインで **-keypass** パラメーターを使用しなかった場合、キーのパスワードを入力するよう要求されます。**Enter** を押し、キーストアのパスワードと同じ値を設定します。

コマンドが終了すると、ファイル **server.keystore** にエイリアス **jboss** を持つ単一のキーが含まれるようになります。

2. キーを検証します。

以下のコマンドを使用して、キーが正常に動作することを確認します。

```
keytool -list -keystore server.keystore
```

キーストアのパスワードを入力するよう求められます。キーストアの内容 (この場合は **jboss** という名前の単一キー) が表示されます。**jboss** キーの種類が **PrivateKeyEntry** であることに注意してください。これは、キーストアにこのキーのパブリックおよびプライベートエントリが含まれることを示します。

3. 証明書署名要求を生成します。

次のコマンドを実行し、手順1で作成したキーストアより公開鍵を使用して証明書署名要求を生成します。

```
keytool -certreq -keyalg RSA -alias jboss -keystore server.keystore -file certreq.csr
```

キーストアに対する認証を行うために、パスワードを入力するよう求められます。**keytool** コマンドにより、現在の作業ディレクトリーに **certreq.csr** という名前の証明書署名要求が新規作成されます。

4. 新しく生成された証明書署名要求をテストします。

以下のコマンドを使用して証明書の内容をテストします。

```
openssl req -in certreq.csr -noout -text
```

証明書の詳細が表示されます。

5. オプション: 証明書署名要求を認証局 (CA) に送信します。

認証局 (CA) は、証明書を認証できます。この結果、証明書は、サードパーティークライアントが信用できると見なされます。CA により、署名済み証明書が提供されます。また、オプションで1つまたは複数の中間証明書が提供されます。

6. オプション: キーストアからの自己署名証明書のエクスポート

テストまたは内部使用のためにのみ証明書が必要な場合は、自己署名証明書を使用できます。次のように、手順1で作成したキーストアからエクスポートします。

```
keytool -export -alias jboss -keystore server.keystore -file server.crt
```

キーストアに対して認証するためパスワードの入力が求められます。**server.crt** という名前の自己署名証明書が現在の作業ディレクトリーに作成されます。

7. 署名済み証明書を中間証明書とともにインポートします。

CA で指示された順序で各証明書をインポートします。各証明書をインポートするには、**intermediate.ca** または **server.crt** を実際のファイル名に置き換えます。証明書が別のファイルとして提供されない場合は、各証明書に対して個別のファイルを作成し、その内容をファイルに貼り付けます。



注記

署名済み証明書および証明書キーは機密情報です。サーバー間での転送方法に注意してください。

```
keytool -import -keystore server.keystore -alias intermediateCA -file intermediate.ca
```

```
keytool -importcert -alias jboss -keystore server.keystore -file server.crt
```

8. 証明書が正常にインポートされたことをテストします。

以下のコマンドを実行し、要求された場合にキーストアパスワードを入力します。キーストアの内容が表示され、証明書がリストの一部になります。

```
keytool -list -keystore server.keystore
```

結果

署名済み証明書はキーストアに含まれ、HTTPS Web サーバー通信を含む SSL 接続を暗号化するために使用できます。

[Report a bug](#)

11.12.3. SSL コネクタリファレンス

JBoss Web コネクタには、次の SSL 設定属性を含めることができます。提供された CLI コマンドは、プロファイル **default** を使用した管理対象ドメイン向けに設計されています。プロファイル名を、管理対象ドメインに対して設定する名前に変更するか、コマンドの **/profile=default** 部分を省略します

(スタンドアロンサーバーの場合)。

表11.11 SSL コネクター属性

属性	説明	CLI コマンド
name	SSL コネクターの表示名。	属性 name は読み取り専用です。
verify-client	<p>HTTP/HTTPS コネクターまたはネイティブ APR コネクターが使用されるかによって、verify-client の可能な値は異なります。</p> <p>HTTP/HTTPS コネクター</p> <p>可能な値は true、false、または want です。接続を受け入れる前にクライアントから有効な証明書チェーンが必要な場合は、true に設定します。SSL スタックでクライアント証明書を要求し、クライアント証明書が提示されない場合でもエラーが発生しないようにするには、want に設定します。CLIENT-CERT 認証を使用するセキュリティ制約によって保護されたリソースをクライアントが要求する場合を除き、証明書チェーンを必要としないときは false (デフォルト値) に設定します。</p> <p>ネイティブ APR コネクター</p> <p>可能な値は optional、require、optionalNoCA、および none (または none と同様の他の文字列) です。これらの値は、証明が任意、必須、認証局なしの任意、または不要であるかどうかを決定します。デフォルトは none で、クライアントは証明書を提出する機会がありません。</p>	<p>最初のコマンド例は HTTPS コネクターを使用します。</p> <pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=verify-client,value=want)</pre> <p>2 つ目のコマンド例は APR コネクターを使用します。</p> <pre>/profile=default/subsystem=web/connector=APR/ssl=configuration/:write-attribute(name=verify-client,value=require)</pre>
verify-depth	クライアントが有効な証明を持たないと判断するまでにチェックされる中間証明書発行者の最大数。デフォルト値は 10 です。	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=verify-depth,value=10)</pre>

属性	説明	CLI コマンド
certificate-key-file	署名済みサーバー証明書が格納されるキーストアの完全ファイルパスおよびファイル名。JSSE 暗号化の場合、この証明書ファイルが唯一のファイルになり、OpenSSL は複数のファイルを使用します。デフォルト値は JBoss EAP 6 を実行しているユーザーのホームディレクトリー内にある .keystore ファイルになります。 keystoreType がファイルを使用しない場合は、パラメーターを空の文字列に設定します。	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=certificate-key-file,value=../domain/configuration/server.keystore)</pre>
certificate-file	OpenSSL 暗号化を使用する場合は、このパラメーターの値を、サーバー証明書を含むファイルに対するパスに設定します。	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=certificate-file,value=server.crt)</pre>
password	トラストストアおよびキーストアのパスワード。以下の例では、 <i>PASSWORD</i> を実際のパスワードに置き換えます。	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=password,value=PASSWORD)</pre>

属性	説明	CLI コマンド
protocol	<p>使用する SSL プロトコルのバージョン。サポートされる値は基礎の SSL 実装 (JSSE または OpenSSL) によって異なります。 Java SSE のドキュメント を参照してください。</p> <p>また、プロトコルの組み合わせをコンマで区切って指定することもできます (例: TLSv1, TLSv1.1,TLSv1.2)。</p> <div>警告<p>Red Hat は、影響す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 を明示的に無効化することを推奨しています。</p></div>	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=protocol,value=ALL)</pre> <pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=protocol,value="TLSv1,TLSv1.1,TLSv1.2")</pre>

属性	説明	CLI コマンド
cipher-suite	<p>許可される暗号のリスト。JSSE の構文ではカンマ区切りのリストを使用する必要があります。OpenSSL の構文ではコロン区切りのリストを使用する必要があります。必ず1つの構文のみを使用してください。</p> <p>デフォルトは HIGH:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5です。</p> <p>例では可能な暗号が2つのみですが、実際にはこれ以上の暗号を使用することになります。</p> <div data-bbox="596 725 702 1200" data-label="Image"> </div> <p>重要</p> <p>弱い暗号を使用するとセキュリティリスクが増大します。NIST が推奨する暗号化スイートは http://www.nist.gov/manuscript-publication-search.cfm?pub_id=915295 を参照してください。</p> <p>使用可能な OpenSSL の暗号のリストは https://www.openssl.org/docs/apps/ciphers.html#CIPHER_STRINGS を参照してください。 @SECLEVEL、SUITEB128、SUITEB128ONLY、および SUITEB192 はサポートされないことに注意してください。</p>	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=cipher-suite,value="TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA")</pre>
key-alias	<p>キーストア内のサーバー証明書に使用されるエイリアス。以下の例では、KEY_ALIAS を実際の証明書のエイリアスに置き換えます。</p>	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=key-alias,value=KEY_ALIAS)</pre>
truststore-type	<p>トラストストアのタイプ。PKCS12 や Java の標準 JKS など、さまざまなタイプのトラストストアが使用可能です。</p>	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=truststore-type,value=jks)</pre>

属性	説明	CLI コマンド
keystore-type	キーストアのタイプ。 PKCS12 や Java の標準 JKS など、さまざまなタイプのキーストアが使用可能です。	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=keystore-type,value=jks)</pre>
ca-certificate-file	CA 証明書が含まれるファイル。JSSE の場合、これは truststoreFile であり、キーストアと同じパスワードを使用します。クライアント証明書を検証するには、 ca-certificate-file ファイルが使用されます。	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=certificate-file,value=ca.crt)</pre>
ca-certificate-password	ca-certificate-file の証明書パスワード。以下の例では、 MASKED_PASSWORD を実際のマスクされたパスワードに置き換えます。	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=ca-certificate-password,value=MASKED_PASSWORD)</pre>
ca-revocation-url	呼び出しリストが含まれるファイルまたは URL。JSSE の場合は、 crlFile を参照し、SSL の場合は、 SSLCARevocationFile を参照します。	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=ca-revocation-url,value=ca.crl)</pre>
session-cache-size	SSLSession キャッシュのサイズ。この属性は、JSSE コネクターへのみ適用されます。デフォルト値は 0 で、無制限のキャッシュサイズが指定されます。	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=session-cache-size,value=100)</pre>
session-timeout	キャッシュされた SSLSession が期限切れになるまでの秒数。この属性は JSSE コネクターへのみ適用されます。デフォルト値は 86400 秒 (24 時間) です。	<pre>/profile=default/subsystem=web/connector=HTTPS/ssl=configuration/:write-attribute(name=session-timeout,value=43200)</pre>

[Report a bug](#)

11.13. 機密性の高い文字列のパスワード VAULT

11.13.1. パスワード vault システム

JBoss EAP 6 にはパスワード vault が含まれています。このパスワード vault は、機密性の高い文字列を暗号化して暗号化されたキーストアに保存し、アプリケーションや検証システムに対して復号化します。

XML デプロイメント記述子などのプレーンテキストの設定ファイルには、パスワードなどの機密性の高い情報を指定する必要があります。

JBoss EAP のパスワード vault を使用して、プレーンテキストファイルの機密性の高い文字列をセキュアに保存します。

[Report a bug](#)

11.13.2. 機密性の高い文字列を格納する Java キーストアの作成

前提条件

- Java Runtime Environment (JRE) によって提供される **keytool** ユーティリティが必要です。このファイルのパスを見つけます。Red Hat Enterprise Linux では **/usr/bin/keytool** になります。



警告

JCEKS キーストアの実装は Java のベンダーによって異なるため、使用する JDK と同じベンダーの **keytool** ユーティリティを使用してキーストアを生成する必要があります。

あるベンダーの JDK の **keytool** によって生成されたキーストアを別のベンダーの JDK で実行されている JBoss EAP インスタンスで使用すると、以下の例外が発生します。

```
java.io.IOException: com.sun.crypto.provider.SealedObjectForKeyProtector
```

手順11.41 Java キーストアの設定

- キーストアと他の暗号化された情報を格納するディレクトリーを作成します。
キーストアと他の重要な情報を保存するディレクトリーを作成します。この残りの手順では、ディレクトリーが **EAP_HOME/vault/** であることを前提とします。このディレクトリーには機密情報が含まれるため、限られたユーザーのみがアクセスできるようにする必要があります。JBoss EAP が実行されるユーザーアカウントでは、最低でも読み書きアクセスが必要です。
- keytool** ユーティリティで使用するパラメーターを決定します。
以下のパラメーターの値を決定します。

alias

`alias` はキーストアに保存された `vault` またはその他のデータの一意な識別子です。大文字と小文字を区別します。

storetype

`storetype` はキーストアのタイプを指定します。値は **jceks** が推奨されます。

keyalg

暗号化に使用するアルゴリズム。利用可能な他の選択肢については、JRE およびオペレーティングシステムのドキュメンテーションを参照してください。

keysize

暗号化キーのサイズは、ブルートフォース攻撃で復号化を行う難しさに影響します。適切な値については、**keytool** ユーティリティーで提供されるドキュメントを参照してください。

storepass

storepass の値は、キーを読み取りできるようにするためキーストアに対する認証で使われるパスワードです。パスワードは 6 文字以上である必要があり、キーストアがアクセスされたときにパスワードが提供される必要があります。このパラメーターを省略すると、コマンドの実行時に入力するよう要求されます。

keypass

keypass の値は特定のキーへのアクセスに使用されるパスワードで、**storepass** パラメーターの値と一致する必要があります。

validity

validity の値は、キーの有効期間 (日数) になります。

keystore

keystore の値は、キーストアの値が保存されるファイルパスおよびファイル名になります。キーストアファイルはデータが最初に追加されたときに作成されます。

ファイルパスで正しい区切り文字を使用するようにしてください。Red Hat Enterprise Linux や類似のオペレーティングシステムの場合は `/` (スラッシュ) を使用し、Microsoft Windows Server の場合は `\` (バックスラッシュ) を使用する必要があります。

keytool ユーティリティーには他にも多くのオプションがあります。詳細については、JRE または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。

3. **keytool** コマンドを実行します。

オペレーティングシステムのコマンドラインインターフェースを開き、収集した情報を使用して **keytool** ユーティリティーを実行します。

例11.36 Java キーストアの作成

```
$ keytool -genseckey -alias vault -storetype jceks -keyalg AES -keysize 128 -storepass vault22 -keypass vault22 -validity 730 -keystore EAP_HOME/vault/vault.keystore
```

結果

ファイル **EAP_HOME/vault/vault.keystore** にキーストアが作成されます。このキーストアにはエイリアスが **vault** であるキーが1つ保存され、パスワードなどの JBoss EAP の暗号化された文字列を保存するために使用されます。

[Report a bug](#)

11.13.3. キーストアパスワードのマスキングとパスワード **vault** の初期化

前提条件

- 「機密性が高い文字列を格納する Java キーストアの作成」
1. **vault.sh** コマンドを実行します。
EAP_HOME/bin/vault.sh を実行します。0 を入力して新しい対話セッションを開始します。
 2. 暗号化されたファイルが保存されるディレクトリーを入力します。
 限られたユーザーのみがこのディレクトリーにアクセスできるようにする必要があります。JBoss EAP が実行されるユーザーアカウントでは、最低でも読み書きアクセスが必要です。「機密性が高い文字列を格納する Java キーストアの作成」に従ってキーストアを作成した場合、キーストアは **EAP_HOME/vault/** というディレクトリーにあります。



注記

必ずディレクトリー名の最後にスラッシュが含まれるようにしてください。ご使用のオペレーティングシステムに応じて / または \ を使用します。

3. キーストアへのパスを入力します。
 キーストアファイルへのフルパスを入力します。この例では **EAP_HOME/vault/vault.keystore** を使用します。
4. キーストアパスワードを暗号化します。
 次の手順に従って、設定ファイルやアプリケーションで安全に使用できるようキーストアのパスワードを暗号化します。
 - a. キーストアパスワードを入力します。
 入力を促されたらキーストアのパスワードを入力します。
 - b. salt 値を入力します。
 8 文字の salt 値を入力します。salt 値は繰り返す回数 (下記) と共にハッシュ値の作成に使用されます。
 - c. 繰り返す回数を入力します。
 繰り返す回数の値を入力します。
 - d. マスクされたパスワード情報を書き留めておきます。
 マスクされたパスワード、salt、および繰り返す回数は標準出力へ書き出されます。これらの情報を安全な場所書き留めておきます。攻撃者がこれらの情報を使用してパスワードを復号化する可能性があるからです。
 - e. vault のエイリアスを入力します。
 入力を促されたら、vault のエイリアスを入力します。「機密性が高い文字列を格納する Java キーストアの作成」に従って vault を作成した場合、エイリアスは **vault** になります。
5. 対話コンソールを終了します。

2を入力して対話コンソールを終了します。

結果

設定ファイルとデプロイメントで使用するため、キーストアパスワードがマスキングされます。また、vault が完全設定され、すぐ使用できる状態になります。

[Report a bug](#)

11.13.4. パスワード vault を使用するよう JBoss EAP 6 を設定

概要

設定ファイルにあるパスワードや機密性の高いその他の属性をマスキングする前に、これらを保存し復号化するパスワード vault を JBoss EAP 6 が認識するようにする必要があります。次の手順に従ってこの機能を有効にします。

前提条件

- 「[機密性の高い文字列を格納する Java キーストアの作成](#)」
- 「[キーストアパスワードのマスキングとパスワード vault の初期化](#)」

手順11.42 パスワード vault の設定

1. コマンドの適切な値を決定します。

キーストアの作成に使用されるコマンドによって決定される以下のパラメーターの値を決定します。キーストア作成の詳細は「[機密性の高い文字列を格納する Java キーストアの作成](#)」および「[キーストアパスワードのマスキングとパスワード vault の初期化](#)」を参照してください。

パラメーター	説明
KEYSTORE_URL	ファイルシステムのパスまたはキーストアファイル。通常 vault.keystore のようになります。
KEYSTORE_PASSWORD	キーストアのアクセスに使用されるパスワード。この値はマスクされる必要があります。
KEYSTORE_ALIAS	キーストアエイリアスの名前。
SALT	キーストアの値を暗号化および復号化するために使用される salt。
ITERATION_COUNT	暗号化アルゴリズムが実行される回数。
ENC_FILE_DIR	キーストアコマンドが実行されるディレクトリーへのパス。通常、パスワード vault が含まれるディレクトリーになります。
host (管理対象ドメインのみ)	設定するホストの名前。

2. 管理 CLI を使用してパスワード vault を有効にします。

次のコマンドの1つを実行します。実行するコマンドは、管理対象ドメインまたはスタンドアロンサーバー設定のどちらを使用するかによって異なります。コマンドの値は、手順の最初で使用した値に置き換えます。



注記

Microsoft Windows Server を使用する場合、CLI コマンドではディレクトリーパスにある \ 文字に \ を追加してエスケープします。たとえば、**C:\data\vault\vault.keystore** のようになります。これは、単一の \ 文字は文字のエスケープに使用されるためです。

管理対象ドメイン

```
/host=YOUR_HOST/core-service=vault:add(vault-options=[("KEYSTORE_URL" =>
"PATH_TO_KEYSTORE"), ("KEYSTORE_PASSWORD" => "MASKED_PASSWORD"),
("KEYSTORE_ALIAS" => "ALIAS"), ("SALT" => "SALT"), ("ITERATION_COUNT" =>
"ITERATION_COUNT"), ("ENC_FILE_DIR" => "ENC_FILE_DIR")])
```

スタンドアロンサーバー

```
/core-service=vault:add(vault-options=[("KEYSTORE_URL" =>
"PATH_TO_KEYSTORE"), ("KEYSTORE_PASSWORD" => "MASKED_PASSWORD"),
("KEYSTORE_ALIAS" => "ALIAS"), ("SALT" => "SALT"), ("ITERATION_COUNT" =>
"ITERATION_COUNT"), ("ENC_FILE_DIR" => "ENC_FILE_DIR")])
```

仮の値を用いたコマンドの例は次のとおりです。

```
/core-service=vault:add(vault-options=[("KEYSTORE_URL" =>
"/home/user/vault/vault.keystore"), ("KEYSTORE_PASSWORD" => "MASK-3y28rCZlcKR"),
("KEYSTORE_ALIAS" => "vault"), ("SALT" => "12438567"), ("ITERATION_COUNT" => "50"),
("ENC_FILE_DIR" => "/home/user/vault")])
```

結果

パスワード vault を使用してマスキングされた文字列を復号化するよう JBoss EAP 6 が設定されます。vault に文字列を追加し、設定で使用する場合は「[Java キーストアに暗号化された機密性の高い文字列の保存および読み出し](#)」を参照してください。

[Report a bug](#)

11.13.5. パスワード Vault のカスタム実装を使用するよう JBoss EAP 6 を設定

概要

独自の **SecurityVault** 実装を使用して、設定ファイルのパスワードや機密性の高いその他の属性をマスクできます。

手順11.43 パスワード vault のカスタム実装の使用

1. インターフェース **SecurityVault** を実装するクラスを作成します。
2. 前の手順のクラスが含まれるモジュールを作成し、インターフェースが **SecurityVault** である **org.picketbox** の依存関係を指定します。

- 以下の属性を用いて vault 要素を追加して、JBoss EAP サーバー設定でカスタムのパスワード vault を有効にします。

code

SecurityVault を実装するクラスの完全修飾名。

module

カスタムクラスが含まれるモジュールの名前。

オプションで **vault-options** パラメーターを使用して、パスワード Vault のカスタムクラスを初期化できます。以下に例を示します。

```
/core-service=vault:add(code="custom.vault.implementation.CustomSecurityVault",  
module="custom.vault.module", vault-options=[("KEYSTORE_URL" =>  
"PATH_TO_KEYSTORE"), ("KEYSTORE_PASSWORD" => "MASKED_PASSWORD"),  
("KEYSTORE_ALIAS" => "ALIAS"), ("SALT" => "SALT"), ("ITERATION_COUNT" =>  
"ITERATION_COUNT"), ("ENC_FILE_DIR" => "ENC_FILE_DIR")])
```

結果

パスワード vault のカスタム実装を使用して、マスクされた文字列が復号化されるよう JBoss EAP 6 が設定されます。

[Report a bug](#)

11.13.6. Java キーストアに暗号化された機密性の高い文字列の保存および読み出し

概要

パスワードや、機密性の高いその他の文字列がプレーンテキストの設定ファイルに含まれるのはセキュアではありません。JBoss EAP 6 には、このような機密性の高い文字列をマスキングして暗号化されたキーストアに保存する機能や、設定ファイルでマスクされた値を使用する機能が含まれています。

前提条件

- 「機密性の高い文字列を格納する Java キーストアの作成」
- 「キーストアパスワードのマスキングとパスワード vault の初期化」
- 「パスワード vault を使用するよう JBoss EAP 6 を設定」
- EAP_HOME/bin/vault.sh** アプリケーションはコマンドラインインターフェースからアクセスできなければなりません。

手順11.44 Java キーストアの設定

- vault.sh** コマンドを実行します。
EAP_HOME/bin/vault.sh を実行します。0 を入力して新しい対話セッションを開始します。
- 暗号化されたファイルが保存されるディレクトリーを入力します。
「機密性の高い文字列を格納する Java キーストアの作成」に従ってキーストアを作成した場合、キーストアは **EAP_HOME/vault/** というディレクトリーにあります。ほとんどの場合、暗号化した情報をキーストアと同じ場所に保存することは適切です。このディレクトリーには機

密性の高い情報が含まれるため、アクセスできるユーザーを制限する必要があります。最低でも、JBoss EAP を実行するユーザーアカウントには読み書き権限が必要です。



注記

必ずディレクトリー名の最後にスラッシュが含まれるようにしてください。ご使用のオペレーティングシステムに応じて / または \ を使用します。

3. キーストアへのパスを入力します。
キーストアファイルへのフルパスを入力します。この例では **EAP_HOME/vault/vault.keystore** を使用します。
4. キーストアパスワード、vault 名、ソルト、繰り返す回数を入力します。
入力を促されたら、キーストアパスワード、vault 名、ソルト、繰り返す回数を入力します。ハッシュが実行されます。
5. パスワードを保存するオプションを選択します。
オプション **0** を選択して、パスワードや機密性の高い他の文字列を保存します。
6. 値を入力します。
入力を促されたら、値を 2 回入力します。値が一致しない場合は再度入力するよう要求されます。
7. vault ブロックを入力します。
同じリソースに関連する属性のコンテナである vault ブロックを入力します。属性名の例としては **ds_ExampleDS** などが挙げられます。データソースまたは他のサービス定義で、暗号化された文字列への参照の一部を形成します。
8. 属性名を入力します。
保存する属性の名前を入力します。 **password** が属性名の例の 1 つになります。

結果

以下のようなメッセージによって、属性が保存されたことが示されます。

```
Secured attribute value has been stored in vault.
```

9. 暗号化された文字列に関する情報を書き留めます。
メッセージは vault ブロック、属性名、共有キー、および設定で文字列を使用する場合のアドバイスを表示する標準出力を出力します。安全な場所にこの情報を書き留めておくようにしてください。出力例は次のとおりです。

```
*****
Vault Block:ds_ExampleDS
Attribute Name:password
Configuration should be done as follows:
VAULT::ds_ExampleDS::password::1
*****
```

10. 設定で暗号化された文字列を使用します。
プレーンテキストの文字列の代わりに、前の設定手順の文字列を使用します。以下は、上記の暗号化されたパスワードを使用するデータソースになります。

...

```

<subsystem xmlns="urn:jboss:domain:datasources:1.0">
  <datasources>
    <datasource jndi-name="java:jboss/datasources/ExampleDS" enabled="true" use-java-
context="true" pool-name="H2DS">
      <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1</connection-url>
      <driver>h2</driver>
      <pool></pool>
      <security>
        <user-name>sa</user-name>
        <password>${VAULT::ds_ExampleDS::password::1}</password>
      </security>
    </datasource>
  </drivers>
  <driver name="h2" module="com.h2database.h2">
    <xa-datasource-class>org.h2.jdbcx.JdbcDataSource</xa-datasource-class>
  </driver>
</drivers>
</datasources>
</subsystem>
...

```

式が許可されるドメインまたはスタンドアロン設定ファイルであれば、どこでも暗号化された文字列を使用することができます。



注記

特定のサブシステム内で式が許可されるかを確認するには、そのサブシステムに対して次の CLI コマンドを実行します。

```
/host=master/core-service=management/security-realm=TestRealm:read-
resource-description(recursive=true)
```

このコマンドの出力で、**expressions-allowed** パラメーターの値を探します。値が true であればこのサブシステムの設定内で式を使用できます。

文字列をキーストアに格納した後、次の構文を使用してクリアテキストの文字列を暗号化された文字列に置き換えます。

```
${VAULT::VAULT_BLOCK::ATTRIBUTE_NAME::ENCRYPTED_VALUE}
```

実環境の値の例は次のとおりです。vault ブロックは **ds_ExampleDS**、属性は **password** です。

```
<password>${VAULT::ds_ExampleDS::password::1}</password>
```

[Report a bug](#)

11.13.7. アプリケーションでの機密性の高い文字列の保存および解決

概要

JBoss EAP 6 の設定要素は、セキュリティー vault メカニズムを通じて Java キーストアに保存される値に対して暗号化された文字列を解決する機能をサポートしています。この機能に対するサポートを独自のアプリケーションに追加することができます。

最初に、vault にパスワードを追加します。次に、クリアテキストのパスワードを vault に保存されているパスワードに置き換えます。この方法を使用してアプリケーションの機密性の高い文字列を分かりにくくすることができます。

前提条件

この手順を実行する前に、vault ファイルを格納するディレクトリーが存在することを確認してください。JBoss EAP 6 を実行するユーザーが vault ファイルを読み書きできるパーミッションを持っていれば、vault ファイルの場所はどこでも構いません。この例では、**vault/** ディレクトリーを **/home/USER/vault/** ディレクトリーに置きます。vault 自体は **vault/** ディレクトリーの中にある **vault.keystore** と呼ばれるファイルになります。

例11.37 vault へのパスワードの文字列の追加

EAP_HOME/bin/vault.sh コマンドを用いて文字列を vault へ追加します。次の画面出力にコマンドと応答がすべて含まれています。ユーザー入力の値は強調文字で表されています。出力の一部は書式上、削除されています。Microsoft Windows ではコマンド名は **vault.bat** になります。Microsoft Windows のファイルパスでは、ディレクトリーの分離記号として **/** ではなく **** が使用されることに注意してください。

```
[user@host bin]$ ./vault.sh
*****

**** JBoss Vault *****
*****

Please enter a Digit:: 0: Start Interactive Session 1: Remove Interactive Session 2: Exit
0
Starting an interactive session
Enter directory to store encrypted files:/home/user/vault/
Enter Keystore URL:/home/user/vault/vault.keystore
Enter Keystore password: ...
Enter Keystore password again: ...
Values match
Enter 8 character salt:12345678
Enter iteration count as a number (Eg: 44):25

Enter Keystore Alias:vault
Vault is initialized and ready for use
Handshake with Vault complete
Please enter a Digit:: 0: Store a password 1: Check whether password exists 2: Exit
0
Task: Store a password
Please enter attribute value: sa
Please enter attribute value again: sa
Values match
Enter Vault Block:DS
Enter Attribute Name:thePass
Secured attribute value has been stored in vault.

Please make note of the following:
*****

Vault Block:DS
Attribute Name:thePass
Configuration should be done as follows:
VAULT::DS::thePass::1
*****
```

Please enter a Digit:: 0: Store a password 1: Check whether password exists 2: Exit
2

Java コードに追加される文字列は、出力の最後の値である **VAULT** で始まる行です。

次のサーブレットは、クリアテキストのパスワードの代わりに vault された文字列を使用します。違いを確認できるようにするため、クリアテキストのパスワードはコメントアウトされています。

例11.38 vault されたパスワードを使用するサーブレット

```
package vaulterror.web;

import java.io.IOException;
import java.io.Writer;

import javax.annotation.Resource;
import javax.annotation.sql.DataSourceDefinition;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;

/*@DataSourceDefinition(
    name = "java:jboss/datasources/LoginDS",
    user = "sa",
    password = "sa",
    className = "org.h2.jdbcx.JdbcDataSource",
    url = "jdbc:h2:tcp://localhost/mem:test"
)*/
@DataSourceDefinition(
    name = "java:jboss/datasources/LoginDS",
    user = "sa",
    password = "VAULT::DS::thePass::1",
    className = "org.h2.jdbcx.JdbcDataSource",
    url = "jdbc:h2:tcp://localhost/mem:test"
)
@WebServlet(name = "MyTestServlet", urlPatterns = { "/my/" }, loadOnStartup = 1)
public class MyTestServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    @Resource(lookup = "java:jboss/datasources/LoginDS")
    private DataSource ds;

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
        ServletException, IOException {
        Writer writer = resp.getWriter();
```



```
        writer.write((ds != null) + "");  
    }  
}
```

これでサーブレットが vault された文字列を解決できるようになります。

[Report a bug](#)

11.14. FIPS 140-2 準拠の暗号化

11.14.1. FIPS 140-2 の準拠

連邦情報処理標準 (FIPS: Federal Information Processing Standard) 140-2 は、アメリカ政府による、暗号化ソフトウェアモジュール認定のためのコンピューターセキュリティ標準です。多くの場合、FIPS 140-2 へ準拠することが政府機関や民間企業が使用するソフトウェアシステムの要件となります。

JBoss EAP 6 は外部モジュールによる暗号化を使用します。また、FIPS 140-2 に準拠する暗号化モジュールを使用するよう設定できます。

[Report a bug](#)

11.14.2. FIPS 140-2 準拠のパスワード

FIPS 準拠のパスワードは以下の条件を満たす必要があります。

1. 7 文字以上であること。
2. 以下の文字クラスのうち、3 クラス以上の文字が含まれること。
 - ASCII 数字
 - 小文字の ASCII
 - 大文字の ASCII
 - 英数字以外の ASCII
 - ASCII 以外の文字

パスワードの最初の文字が大文字の ASCII である場合、2. にある大文字の ASCII として見なされません。

パスワードの最後の文字が ASCII 数字である場合、制限 2 の ASCII 数字とは見なされません。

[Report a bug](#)

11.14.3. Red Hat Enterprise Linux 6 にて SSL の FIPS 140-2 準拠の暗号を有効化

ここでは、JBoss EAP 6 の Web コンテナ (JBoss Web) を SSL に対する FIPS 140-2 準拠の暗号化に設定する方法を説明します。ここでは Red Hat Enterprise Linux 6 での手順のみを取り上げます。

このタスクでは、FIPS モードの Mozilla NSS ライブラリーを使用します。

前提条件

- Red Hat Enterprise Linux 6 が FIPS 140-2 に準拠するよう設定されている必要があります。<https://access.redhat.com/knowledge/solutions/137833> を参照してください。

手順11.45 SSL に対して FIPS 140-2 準拠の暗号化を有効にする

1. データベースの作成

jboss ユーザーが所有するディレクトリーに NSS データベースを作成します。

```
$ mkdir -p /usr/share/jboss-as/nssdb
$ chown jboss /usr/share/jboss-as/nssdb
$ modutil -create -dbdir /usr/share/jboss-as/nssdb
```

2. NSS 設定ファイルの作成

次の内容が含まれる **nss_pkcs11_fips.cfg** という名前の新しいテキストファイルを **/usr/share/jboss-as** ディレクトリーに作成します。

```
name = nss-fips
nssLibraryDirectory=/usr/lib64
nssSecmodDirectory=/usr/share/jboss-as/nssdb
nssModule = fips
```

NSS 設定ファイルには以下が指定されている必要があります。

- 名前
- NSS ライブラリーが存在するディレクトリー
- 手順1に従って作成された NSS データベースが存在するディレクトリー

Red Hat Enterprise Linux 6 の 64 ビットバージョンを実行していない場合は、**/usr/lib64** の代わりに **/usr/lib** を **nssLibraryDirectory** に設定します。

3. SunPKCS11 プロバイダーの有効化

JRE (**\$JAVA_HOME/jre/lib/security/java.security**) の **java.security** 設定ファイルを編集し、次の行を追加します。

```
security.provider.1=sun.security.pkcs11.SunPKCS11 /usr/share/jboss-as/nss_pkcs11_fips.cfg
```

この行に指定されている設定ファイルは手順2で作成されたファイルであることに注意してください。

このプロバイダーを優先するため、このファイルにある他の **security.provider.X** 行の値 (X) に 1 を足す必要があります。

4. NSS ライブラリーに対して FIPS モードを有効にする

次のように **modutil** コマンドを実行し、FIPS モードを有効にします。

```
modutil -fips true -dbdir /usr/share/jboss-as/nssdb
```

ここで指定するディレクトリーは手順1で作成したものであることに注意してください。

この時点で、セキュリティーライブラリーエラーが発生し、NSS 共有オブジェクトの一部に対してライブラリー署名の再生成が必要になることがあります。

5. FIPS トークンのパスワードの変更

次のコマンドを使用して FIPS トークンのパスワードを設定します。トークンの名前は **NSS FIPS 140-2 Certificate DB** でなければならないことに注意してください。

```
modutil -changepw "NSS FIPS 140-2 Certificate DB" -dbdir /usr/share/jboss-as/nssdb
```

FIPS トークンに使用されるパスワードは FIPS 準拠のパスワードでなければなりません。

6. NSS ツールを使用した証明書の作成

次のコマンドを入力し、NSS ツールを使用して証明書を作成します。

```
certutil -S -k rsa -n jbossweb -t "u,u,u" -x -s "CN=localhost, OU=MYOU, O=MYORG,  
L=MYCITY, ST=MYSTATE, C=MY" -d /usr/share/jboss-as/nssdb
```

7. PKCS11 キーストアを使用するよう HTTPS コネクタを設定する

JBoss CLI ツールで次のコマンドを使用し、HTTPS コネクタを追加します。

```
/subsystem=web/connector=https/:add(socket-  
binding=https,scheme=https,protocol=HTTP/1.1,secure=true)
```

次に、以下のコマンドを使用して SSL 設定を追加します。PASSWORD を手順 5 の FIPS 準拠のパスワードに置き換えます。

```
/subsystem=web/connector=https/ssl=configuration:add(name=https,password=PASSWORD,ke  
ystore-type=PKCS11,  
cipher-  
suite="SSL_RSA_WITH_3DES_EDE_CBC_SHA,SSL_DHE_RSA_WITH_3DES_EDE_CBC_S  
HA,  
TLS_RSA_WITH_AES_128_CBC_SHA,TLS_DHE_DSS_WITH_AES_128_CBC_SHA,  
TLS_DHE_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA,  
TLS_DHE_DSS_WITH_AES_256_CBC_SHA,TLS_DHE_RSA_WITH_AES_256_CBC_SHA,  
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA,TLS_ECDH_ECDSA_WITH_AES_128_  
CBC_SHA,  
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_ECDSA_WITH_3DES_EDE  
_CBC_SHA,  
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES_256_  
CBC_SHA,  
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA,TLS_ECDH_RSA_WITH_AES_128_CBC_S  
HA,  
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_  
SHA,  
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_CBC_  
SHA,  
TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA,TLS_ECDH_anon_WITH_AES_128_CBC_  
SHA,  
TLS_ECDH_anon_WITH_AES_256_CBC_SHA")
```

8. 検証

次のコマンドを実行し、JVM が PKCS11 キーストアから公開鍵を読み取れることを検証しま
す。

```
keytool -list -storetype pkcs11
```

例11.39 FIPS 140-2 に準拠した HTTPS コネクターの XML 設定

```

<connector name="https" protocol="HTTP/1.1" scheme="https" socket-binding="https"
secure="true">
  <ssl name="https" password="*****"
    cipher-
suite="SSL_RSA_WITH_3DES_EDE_CBC_SHA,SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA,

    TLS_RSA_WITH_AES_128_CBC_SHA,TLS_DHE_DSS_WITH_AES_128_CBC_SHA,
    TLS_DHE_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA,

    TLS_DHE_DSS_WITH_AES_256_CBC_SHA,TLS_DHE_RSA_WITH_AES_256_CBC_SHA,

    TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA,TLS_ECDH_ECDSA_WITH_AES_128_CBC_
SHA,

    TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC
_SHA,

    TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_ECDSA_WITH_AES_256_CBC
_SHA,

    TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA,TLS_ECDH_RSA_WITH_AES_128_CBC_SHA,

    TLS_ECDH_RSA_WITH_AES_256_CBC_SHA,TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
,

    TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA,

    TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA,TLS_ECDH_anon_WITH_AES_128_CBC_SHA,

    TLS_ECDH_anon_WITH_AES_256_CBC_SHA"
    keystore-type="PKCS11"/>
</connector>

```

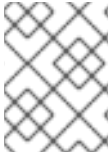
読みやすくするため、**cipher-suite** 属性には改行が挿入されていることに注意してください。

[Report a bug](#)

11.14.4. Apache HTTP サーバーでの FIPS 140-2 の有効化

Apache HTTP サーバーで 140-2 暗号化を有効にするには、**SSLFIPS on** ディレクティブを Apache HTTP サーバー設定ファイル (**httpd.conf** または **ssl.conf**) に挿入します。このディレクティブは **VirtualHost** 設定セクションの外部で使用する必要があります。

SSLFIPS on ディレクティブは SSL ライブラリーの FIPS_mode フラグを有効にします。このモードはすべての SSL ライブラリー操作に適用されます。変更を有効にするには、このディレクティブを追加した後に Apache HTTP サーバーを再起動する必要があります。



注記

FIPS を有効にするには、**FIPS_mode** フラグをサポートする FIPS 対応の OpenSSL がシステムにインストールされている必要があります。

[Report a bug](#)

第12章 セキュリティー管理リファレンス

12.1. 含まれる認証モジュール

以下の認証モジュールが JBoss EAP 6 に含まれます。これらのモジュールの一部は許可と認証を処理します。通常、**Role** という単語が **Code** 名に含まれます。

これらのモジュールを設定する場合は、モジュールを参照するために **Code** 値またはフルネーム (パッケージ修飾) 使用します。

認証モジュール

- [表12.1 「**RealmDirect**」](#)
- [表12.2 「**RealmDirect** モジュールオプション」](#)
- [表12.3 「**Client**」](#)
- [表12.4 「**Client** モジュールオプション」](#)
- [表12.5 「**Remoting**」](#)
- [表12.6 「**Remoting** モジュールオプション」](#)
- [表12.7 「**Certificate**」](#)
- [表12.8 「**Certificate** モジュールオプション」](#)
- [表12.9 「**CertificateRoles**」](#)
- [表12.10 「**CertificateRoles** モジュールオプション」](#)
- [表12.11 「**Database**」](#)
- [表12.12 「**Database** モジュールオプション」](#)
- [表12.13 「**DatabaseCertificate**」](#)
- [表12.14 「**DatabaseCertificate** モジュールオプション」](#)
- [表12.15 「**Identity**」](#)
- [表12.16 「**Identity** モジュールオプション」](#)
- [表12.17 「**Ldap**」](#)
- [表12.18 「**Ldap** モジュールオプション」](#)
- [表12.19 「**LdapExtended**」](#)
- [表12.20 「**LdapExtended** モジュールオプション」](#)
- [表12.21 「**RoleMapping**」](#)
- [表12.22 「**RoleMapping** モジュールオプション」](#)

- [表12.23 「RunAs」](#)
- [表12.24 「RunAs オプション」](#)
- [表12.25 「Simple」](#)
- [表12.26 「ConfiguredIdentity」](#)
- [表12.27 「ConfiguredIdentity モジュールオプション」](#)
- [表12.28 「SecureIdentity」](#)
- [表12.29 「SecureIdentity モジュールオプション」](#)
- [表12.30 「PropertiesUsers」](#)
- [表12.31 「SimpleUsers」](#)
- [表12.32 「LdapUsers」](#)
- [表12.33 「Kerberos」](#)
- [表12.34 「Kerberos モジュールオプション」](#)
- [表12.35 「SPNEGO」](#)
- [表12.36 「SPNEGO モジュールオプション」](#)
- [表12.37 「AdvancedLdap」](#)
- [表12.38 「AdvancedLdap モジュールオプション」](#)
- [表12.39 「AdvancedADLdap」](#)
- [表12.40 「UsersRoles」](#)
- [表12.41 「UsersRoles モジュールオプション」](#)
- [カスタム認証モジュール](#)

表12.1 RealmDirect

コード	RealmDirect
クラス	org.jboss.as.security.RealmDirectLoginModule
説明	セキュリティーレルムと直接インターフェースで接続するログインモジュール実装。このログインモジュールは、バックングストアとのやりとりがすべてレルムへ委譲されるようにするため、定義の重複や同期化が必要なくなります。リモーティング呼び出しや管理インターフェースに対して使用されま す。

表12.2 RealmDirect モジュールオプション

オプション	タイプ	デフォルト	説明
realm	文字列	ApplicationRealm	指定するレルムの名前。

表12.3 Client

コード	Client
クラス	org.jboss.security.ClientLoginModule
説明	このログインモジュールは、JBoss EAP 6 がクライアントとして動作するときに呼び出し元 ID とクレデンシャルを確立するよう設定されています。サーバー認証に使用されるセキュリティドメインの一部として使用しないでください。

表12.4 Client モジュールオプション

オプション	タイプ	デフォルト	説明
multi-threaded	true または false	false	各スレッドが独自のプリンシパルとクレデンシャルストレージを持つ場合は、 true に設定します。VM 内のすべてのスレッドが同じ ID とクレデンシャルを共有するよう指定する場合は false に設定します。
password-stacking	useFirstPass または false	false	このログインモジュールが ID として使用する LoginContext に格納された情報を探そう指定する場合は、 LoginContext に設定します。このオプションは、他のログインモジュールをスタックする場合に使用できます。
restore-login-identity	true または false	false	login() メソッドの先頭に示された ID とクレデンシャルを logout() メソッドの呼び出し後に復元する必要がある場合は true に設定します。

表12.5 Remoting

コード	Remoting
クラス	org.jboss.as.security.remoting.RemotingLoginModule

説明	このログインモジュールは、現在認証中の要求が Remoting 接続上で受信された要求であるかどうかを確認するために使用されます。Remoting 接続上で受信された要求である場合、Remoting 認証処理中に作成された ID が使用され、現在の要求に関連付けされます。要求が Remoting 接続上で受信されなかった場合は、このモジュールは何もせず、JAAS ベースのログインは次のモジュールへ続行されます。
----	---

表12.6 Remoting モジュールオプション

オプション	タイプ	デフォルト	説明
password-stacking	useFirstPass または false	false	このログインモジュールが ID を見つけるため LoginContext に格納された情報を最初に探すことを示す useFirstPass の値。このオプションは、他のログインモジュールをこのモジュールとスタックする場合に使用できます。
principalClass	完全修飾クラス名	なし	プリンシパル名に String 引数を取るコンストラクターを含む Principal 実装クラス。
unauthenticatedIdentity	プリンシパル名。	なし	認証情報を含まない要求に割り当てられるプリンシパル名を定義します。これにより、保護されていないサブレットは特定のロールを必要としない EJB でメソッドを呼び出すことができますようになります。このようなプリンシパルには関連付けられたロールがなく、セキュアでない JEB または unchecked permission 制約に関連付けられた EJB メソッドのみにアクセスできます。

表12.7 Certificate

コード	Certificate
クラス	org.jboss.security.auth.spi.BaseCertLoginModule

説明	このログインモジュールは、 X509 Certificates に基づいてユーザーを認証するよう設計されています。この使用例は、Web アプリケーションの CLIENT-CERT 認証です。
----	---

表12.8 Certificate モジュールオプション

オプション	タイプ	デフォルト	説明
securityDomain	文字列	other	信頼済み証明書を保持するトラストストア用 JSSE 設定を持つセキュリティードメインの名前。
verifier	class	なし	ログイン証明書の検証に使用する org.jboss.security.auth.certs.X509CertificateVerifier のクラス名。

表12.9 CertificateRoles

コード	CertificateRoles
クラス	org.jboss.security.auth.spi.CertRolesLoginModule
説明	このログインモジュールは、Certificate ログインモジュールを拡張して、プロパティファイルからロールマッピング機能を追加します。同じすべてのオプションを Certificate ログインモジュールとして取得し、次のオプションを追加します。

表12.10 CertificateRoles モジュールオプション

オプション	タイプ	デフォルト	説明
-------	-----	-------	----

オプション	タイプ	デフォルト	説明
rolesProperties	文字列	roles.properties	<p>各ユーザーに割り当てるロールを含むリソースまたはファイルの名前。ロールプロパティーファイルの形式は username=role1,role2 である必要があります。username は証明書の DN で、= (等記号) やスペースをすべてエスケープします。正しい形式を用いた例は次のとおりです。</p> <pre>CN\=unit-tests-client,\ OU\=Red\ Hat\ Inc.,\ O\=Red\ Hat\ Inc.,\ ST\=North\ Carolina,\ C\=US</pre>
defaultRolesProperties	文字列	defaultRoles.properties	rolesProperties ファイルが見つからない場合に使用するリソースまたはファイルの名前。
roleGroupSeparator	単一の文字。	. (ピリオド1つ)	rolesProperties ファイルでどの文字をロールグループセパレーターとして使用するか。

表12.11 Database

コード	Database
クラス	org.jboss.security.auth.spi.DatabaseServerLoginModule
説明	<p>認証とロールマッピングをサポートする JDBC ベースのログインモジュール。これは、次の定義を使用して、2つの論理テーブルに基づきます。</p> <ul style="list-style-type: none"> ● Principals: PrincipalID (text), Password (text) ● Roles: PrincipalID (text), Role (text), RoleGroup (text)

表12.12 Database モジュールオプション

オプション	タイプ	デフォルト	説明
digestCallback	完全修飾クラス名	なし	入力パスワードをハッシュ化するソルト値などのプレまたはポストダイジェストコンテンツを含む DigestCallback 実装のクラス名。 hashAlgorithm が指定された場合にのみ使用されます。
dsJndiName	JNDI リソース	java:/DefaultDS	認証情報を保持する JNDI リソースの名前。このオプションは必須です。
hashAlgorithm	文字列	プレーンパスワードを使用	パスワードをハッシュ化するのに使用されるメッセージダイジェストアルゴリズム。サポートされるアルゴリズムは Java セキュリティプロバイダーに依存しますが、 MD5 、 SHA-1 、および SHA-256 がサポートされます。
hashCharset	文字列	プラットフォームのデフォルトのエンコーディング	パスワードの文字列をバイトアレイに変換するときに使用する文字セット/エンコーディングの名前。これにはサポートされるすべての Java 文字セット名が含まれます。
hashEncoding	文字列	Base64	使用する文字列エンコーディング形式。
ignorePasswordCase	boolean	false	パスワードの比較で大文字と小文字の区別を無視するかどうかを示すフラグ。
inputValidator	完全修飾クラス名	なし	クライアントによって提供されるユーザー名およびパスワードを検証するために使用される InputValidator 実装のインスタンス。
principalsQuery	準備済み SQL ステートメント	select Password from Principals where PrincipalID=?	プリンシパルに関する情報を取得するための準備済み SQL クエリー。
rolesQuery	準備済み SQL ステートメント	なし	ロールの情報を取得するための準備済み SQL クエリー。 select Role, RoleGroup from Roles where PrincipalID=? と同等である必要があります。ここで、Role はロール名で、RoleGroup 列の値は常に R が大文字である Roles または CallerPrincipal である必要があります。
storeDigestCallback	完全修飾クラス名	なし	ストア/予想されるパスワードをハッシュ化するソルト値などのプレまたはポストダイジェストコンテンツを含む DigestCallback 実装のクラス名。 hashStorePassword または hashUserPassword が true で、 hashAlgorithm が指定された場合のみ使用されます。

オプション	タイプ	デフォルト	説明
suspendResume	boolean	true	データベース操作中に既存の JTA トランザクションを一時停止するかどうか。
throwValidatorError	boolean	false	検証エラーがクライアントへ公開されるべきかどうかを示すフラグ。
transactionManagerJndiName	JNDI リソース	java:/TransactionManager	ログインモジュールによって使用されるトランザクションマネージャーの JNDI 名。

表12.13 DatabaseCertificate

コード	DatabaseCertificate
クラス	org.jboss.security.auth.spi.DatabaseCertLoginModule
説明	このログインモジュールは、Certificate ログインモジュールを拡張して、データベーステーブルからロールマッピング機能を追加します。同じオプションと次の追加オプションが存在します。

表12.14 DatabaseCertificate モジュールオプション

オプション	タイプ	デフォルト	説明
dsJndiName	JNDI リソース	java:/DefaultDS	認証情報を保持する JNDI リソースの名前。このオプションは必須です。
rolesQuery	準備済み SQL ステートメント	select Role,RoleGroup from Roles where PrincipalID=?	ロールをマップするために実行される SQL 準備済みステートメント。これは、 select Role, RoleGroup from Roles where PrincipalID=? と同等である必要があります。Role はロール名で、RoleGroup 列の値は常に R が大文字である Roles または CallerPrincipal である必要があります。
suspendResume	true または false	true	データベース操作中に既存の JTA トランザクションを一時停止するかどうか。

表12.15 Identity

コード	Identity
クラス	org.jboss.security.auth.spi.IdentityLoginModule
説明	モジュールオプションで指定されたプリンシパルをモジュールに対して認証されたサブジェクトと関連付けます。使用される Principal クラスのタイプは org.jboss.security.SimplePrincipal です。プリンシパルオプションが指定されない場合は、名前が guest のプリンシパルが使用されます。

表12.16 Identity モジュールオプション

オプション	タイプ	デフォルト	説明
principal	文字列	guest	プリンシパルに使用する名前。
roles	カンマ区切りの文字列リスト	なし	サブジェクトに割り当てられるロールのカンマ区切りリスト。

表12.17 Ldap

コード	Ldap
クラス	org.jboss.security.auth.spi.LdapLoginModule
説明	ユーザー名とパスワードが、JNDI LDAP プロバイダーを使用してアクセスできる LDAP サーバーに格納された場合に、LDAP サーバーに対して認証します。多くのオプションは、LDAP プロバイダーまたは環境によって決定されるため、必須ではありません。

表12.18 Ldap モジュールオプション

オプション	タイプ	デフォルト	説明
java.naming.factory.initial	クラス名	com.sun.jndi.Ldap.LdapCtxFactory	InitialContextFactory 実装クラス名。
java.naming.provider.url	ldap:// URL	java.naming.security.protocol の値が SSL の場合は ldap://localhost:636 で、その他の場合は ldap://localhost:389 。	LDAP サーバーの URL。

オプション	タイプ	デフォルト	説明
java.naming.security.authentication	none 、 simple 、または SASL メカニズムの名前。	simple	LDAP サーバーにバインドするために使用するセキュリティレベル。
java.naming.security.protocol	トランスポートプロトコル	指定されない場合は、プロバイダーによって決定されます。	SSL などの、セキュアアクセスに使用するトランスポートプロトコル。
java.naming.security.principal	文字列	なし	サービスに対する呼び出し元を認証するプリンシパルの名前。これは、以下に示された他のプロパティから構築されます。
java.naming.security.credentials	クレデンシャルタイプ	なし	認証スキームにより使用されるクレデンシャルのタイプ。一部の例には、ハッシュ化されたパスワード、クリアテキストパスワード、キー、または証明書が含まれます。このプロパティが指定されない場合は、動作がサービスプロバイダーにより決定されます。
principalDNPrefix	文字列		ユーザー DN を形成するユーザー名に追加される接頭辞。ユーザーにユーザー名の指定を要求したり、 principalDNPrefix および principalDNSuffix を使用して完全修飾 DN を構築したりできます。
principalDNSuffix	文字列		ユーザー DN を形成するユーザー名に追加されるサフィックス。ユーザーにユーザー名の指定を要求したり、 principalDNPrefix および principalDNSuffix を使用して完全修飾 DN を構築したりできます。

オプション	タイプ	デフォルト	説明
useObjectCredential	true または false	false	JAAS PasswordCallback を使用した char[] パスワードではなく Callback の org.jboss.security.auth.callback.ObjectCallback タイプを使用した不透明なオブジェクトとしてクレデンシャルを取得するかどうか。これにより、 char[] クレデンシャル情報を LDAP サーバーに渡すことができます。
rolesCtxDN	完全修飾 DN	なし	ユーザーロールを検索するコンテキストの完全修飾 DN。
userRolesCtxDNAttribute	attribute	なし	ユーザーロールを検索するコンテキストの DN を含むユーザーオブジェクトの属性。これは、 rolesCtxDN と異なるため、ユーザーのロールを検索するコンテキストは各ユーザーに対して一意になることがあります。
roleAttributeID	attribute	roles	ユーザーロールを含む属性の名前。
roleAttributesDN	true または false	false	roleAttributeID にロールオブジェクトの完全修飾 DN が含まれるかどうか。false の場合は、コンテキスト名の roleNameAttributeID 属性の値からロール名が取得されます。 Microsoft Active Directory などの特定のディレクトリースキーマでは、この属性を true に設定する必要があります。

オプション	タイプ	デフォルト	説明
roleNameAttributeID	attribute	name	ロール名を含む roleCtxDN コンテキスト内の属性の名前。 roleAttributeID プロパティが true に設定された場合、このプロパティはロールオブジェクトの名前属性を見つけるために使用されます。
uidAttributeID	attribute	uid	ユーザー ID に対応する UserRolesAttributeID の属性の名前。これは、ユーザーロールを見つけるために使用されます。
matchOnUserDN	true または false	false	ユーザーロールの検索でユーザーの完全識別 DN またはユーザー名のみに一致するかどうか。 true の場合、完全 userDN は一致する値として使用されます。 false の場合は、ユーザー名のみが uidAttributeName 属性に対して一致する値として使用されます。
allowEmptyPasswords	true または false	false	空白のパスワードを許可するかどうか。ほとんどの LDAP サーバーでは、空白のパスワードが匿名ログイン試行として扱われます。空のパスワードを拒否するには、これを false に設定します。

表12.19 LdapExtended

コード	LdapExtended
クラス	org.jboss.security.auth.spi.LdapExtLoginModule

説明	<p>検索を使用してバインドユーザーと関連するロールを見つける別の LDAP ログインモジュール実装。ロールクエリーは再帰的に DN に従い、階層ロール構造をナビゲートします。同じ java.naming オプションを Ldap モジュールとして使用し、Ldap モジュールの他のオプションの代わりに次のオプションを使用します。</p> <p>認証は 2 つの手順で行われます。</p> <ol style="list-style-type: none"> LDAP サーバーに対する初期バインドは、bindDN および bindCredential オプションを使用して行われます。bindDN は、baseCtxDN および rolesCtxDN ツリーの両方でユーザーとロールを検索できるユーザーです。認証するユーザー DN は、baseFilter 属性で指定されたフィルターを使用して問い合わせられます。 結果となるユーザー DN は、InitialLdapContext 環境 Context.SECURITY_PRINCIPAL としてユーザー DN を使用して LDAP サーバーにバインドすることにより認証されます。Context.SECURITY_CREDENTIALS プロパティは、コールバックハンドラーにより取得された String パスワードに設定されます。
----	---

表12.20 LdapExtended モジュールオプション

オプション	タイプ	デフォルト	説明
baseCtxDN	完全修飾 DN	なし	ユーザー検索を開始する最上位コンテキストの固定 DN。
bindCredential	文字列、オプションで暗号化	なし	詳細は JBoss EAP『セキュリティガイド』を参照してください。
bindDN	完全修飾 DN	なし	ユーザーおよびロールクエリーのために LDAP サーバーに対してバインドするために使用される DN。この DN は baseCtxDN および rolesCtxDN の値に対する読み取りおよび検索パーミッションを必要とします。

オプション	タイプ	デフォルト	説明
baseFilter	LDAP フィルター文字列	なし	認証するユーザーのコンテキストを見つけるために使用される検索フィルター。入力ユーザー名またはログインモジュールコールバックから取得された userDN が、 {0} 式が使用されたフィルターに置換されます。検索フィルターの一般的な例は (uid={0}) です。
rolesCtxDN	完全修飾 DN	なし	ユーザーロールを検索するコンテキストの固定 DN。これは、実際のロールが存在する DN ではなく、ユーザーロールを含むオブジェクトが存在する DN です。たとえば、Microsoft Active Directory サーバーでは、これは、ユーザーアカウントが存在する DN です。
roleFilter	LDAP フィルター文字列	なし	認証済みユーザーと関連付けられたロールを検索するために使用される検索フィルター。入力ユーザー名またはログインモジュールコールバックから取得された userDN が {0} 式が使用されたフィルターに置換されます。認証済み userDN は、 {1} が使用されたフィルターに置換されます。入力ユーザー名に一致する検索フィルター例は、 (member={0}) です。認証済み userDN に一致する他の例は (member={1}) です。
roleAttributelsDN	true または false	false	roleAttributeID にロールオブジェクトの完全修飾 DN が含まれるかどうか。false の場合は、コンテキスト名の roleNameAttributeID 属性の値からロール名が取得されます。Microsoft Active Directory などの特定のディレクトリースキーマでは、この属性を true に設定する必要があります。

オプション	タイプ	デフォルト	説明
defaultRole	ロール名	なし	認証された全ユーザーに対して含まれるロール
parseRoleNameFromDN	true または false	false	クエリによって返された DN に <code>roleNameAttributeID</code> が含まれるかどうかを示すフラグ。 true に設定された場合、DN は <code>roleNameAttributeID</code> に対してチェックされます。 false に設定された場合、DN は <code>roleNameAttributeID</code> に対してチェックされません。このフラグは LDAP クエリのパフォーマンスを向上できます。
parseUsername	true または false	false	DN がユーザー名に対して解析されるかどうかを示すフラグ。 true に設定された場合、DN はユーザー名に対して解析されます。 false に設定された場合、DN はユーザー名に対して解析されません。このオプションは <code>usernameBeginString</code> および <code>usernameEndString</code> と共に使用されます。
usernameBeginString	文字列	なし	ユーザー名を公開するため、DN の最初から削除される文字列を定義します。このオプションは usernameEndString と共に使用されます。
usernameEndString	文字列	なし	ユーザー名を公開するため、DN の最後から削除される文字列を定義します。このオプションは usernameBeginString と共に使用されます。
roleNameAttributeID	attribute	name	ロール名を含む roleCtxDN コンテキスト内の属性の名前。 roleAttributesDN プロパティが true に設定された場合、このプロパティはロールオブジェクトの名前属性を見つけるために使用されます。

オプション	タイプ	デフォルト	説明
distinguishedNameAttribute	attribute	distinguishedName	ユーザーの DN を含むユーザーエントリーの属性の名前。これは、ユーザー自身の DN に正しいユーザーマッピングを妨げる特殊文字 (バックスラッシュなど) が含まれる場合に、必要になることがあります。属性が存在しない場合は、エントリーの DN が使用されます。
roleRecursion	整数	0	ロール検索が一致するコンテキストで行われる再帰のレベル数。再帰を無効にするには、これを 0 に設定します。
searchTimeLimit	整数	10000 (10 秒)	ユーザーまたはロール検索のタイムアウト (ミリ秒単位)。
searchScope	OBJECT_SCOPE, ONELEVEL_SCOPE, SUBTREE_SCOPE のいずれか。	SUBTREE_SCOPE	使用する検索範囲。
allowEmptyPasswords	true または false	false	空白のパスワードを許可するかどうか。ほとんどの LDAP サーバーでは、空白のパスワードが匿名ログイン試行として扱われます。空のパスワードを拒否するには、これを false に設定します。

オプション	タイプ	デフォルト	説明
referralUserAttribute IDToCheck	attribute	なし	リファール (referral) を使用しない場合はこのオプションを無視してもかまいません。リファールを使用し、ロールオブジェクトがリファール内部にある場合、このオプションは特定のロール (例: member) に対して定義されたユーザーが含まれる属性名を示します。ユーザーはこの属性名の内容に対してチェックされます。このオプションが設定されていないとチェックは常に失敗するため、ロールオブジェクトはリファールツリーに保存できません。

表12.21 RoleMapping

コード	RoleMapping
クラス	org.jboss.security.auth.spi.RoleMappingLoginModule
説明	認証プロセスの結果であるロールを宣言ロールに対してマップします。このモジュールは、セキュリティドメインに追加する場合に optional とフラグ付けする必要があります。

表12.22 RoleMapping モジュールオプション

オプション	タイプ	デフォルト	説明
rolesProperties	プロパティファイルまたはリソースの完全修飾ファイルパスまたは完全修飾ファイル名。	none	ロールを置換ロールに対してマップするプロパティファイルまたはリソースの完全修飾ファイルパスまたはファイル名。形式は original_role=role1,role2,role3 になります。
replaceRole	true または false	false	現在のロールを追加するか、現在のロールをマップされたロールに置き換えるか。 true に設定された場合は、置き換えられます。



注記

rolesProperties モジュールオプションは RoleMapping に必要です。

表12.23 RunAs

コード	RunAs
クラス	org.jboss.security.auth.spi.RunAsLoginModule
説明	run as ロールを、認証のログイン段階の間スタックにプッシュし、コミットまたはアボート段階でスタックから run as ロールをポップするヘルパーモジュール。このログインモジュールは、セキュアな EJB にアクセスするログインモジュールなどの、認証を実行するためにセキュアなリソースにアクセスする必要がある他のログインモジュール用ロールを提供します。 run as ロールを確立する必要があるログインモジュールの前に、 RunAsLoginModule を設定する必要があります。

表12.24 RunAs オプション

オプション	タイプ	デフォルト	説明
roleName	ロール名	nobody	ログイン段階で run as ロールとして使用するロールの名前。
principalName	プリンシパル名	nobody	ログインフェーズ中に run as プリンシパルとして使用するプリンシパルの名前。指定がない場合、デフォルトの nobody が使用されます。
principalClass	完全修飾クラス名	なし	プリンシパル名に String 引数を取るコンストラクターを含む Principal 実装クラス。

表12.25 Simple

コード	Simple
クラス	org.jboss.security.auth.spi.SimpleServerLoginModule

説明	<p>テスト目的でセキュリティーを素早くセットアップするモジュール。次の単純なアルゴリズムが実装されます。</p> <ul style="list-style-type: none">● パスワードが null である場合、ユーザーを認証し、guest の ID と guest のロールを割り当てます。● パスワードが null でなくユーザーと同じ場合、ユーザー名と同じ ID、admin ロールおよび guest ロールを割り当てます。● それ以外の場合は認証に失敗します。
----	--

Simple モジュールオプション

Simpleモジュールにはオプションがありません。

表12.26 ConfiguredIdentity

コード	ConfiguredIdentity
クラス	org.picketbox.datasource.security.ConfigureIdentityLoginModule
説明	モジュールオプションで指定されたプリンシパルをモジュールに対して認証されたサブジェクトに関連付けます。使用される Principal クラスのタイプは org.jboss.security.SimplePrincipal です。

表12.27 ConfiguredIdentity モジュールオプション

オプション	タイプ	デフォルト	説明
username	文字列	なし	認証のユーザー名

オプション	タイプ	デフォルト	説明
password	暗号化文字列	""	<p>認証に使用するパスワード。パスワードを暗号化するには、コマンドラインで直接モジュールを使用します。</p> <pre>java org.picketbox.datasource.security.SecureIdentityLoginModule password_to_encrypt</pre> <p>このコマンドの結果をモジュールオプションの値フィールドに貼り付けます。デフォルトの値は空の文字列です。</p>
principal	プリンシパルの名前	none	モジュールに対して認証されるサブジェクトに関連付けられるプリンシパル。

表12.28 SecureIdentity

コード	SecureIdentity
クラス	org.picketbox.datasource.security.SecureIdentityLoginModule
説明	<p>このモジュールは、レガシー対応のために提供されます。このモジュールを使用すると、パスワードを暗号化し、暗号化されたパスワードを最適なプリンシパルで使します。アプリケーションが SecureIdentity を使用する場合は、パスワード vault メカニズムを代わりに使用することを検討してください。</p>

表12.29 SecureIdentity モジュールオプション

オプション	タイプ	デフォルト	説明
username	文字列	なし	認証のユーザー名

オプション	タイプ	デフォルト	説明
password	暗号化文字列	""	<p>認証に使用するパスワード。パスワードを暗号化するには、コマンドラインで直接モジュールを使用します。</p> <pre>java org.picketbox.datas ource.security.Secu reIdentityLoginMod ule password_to_encry pt</pre> <p>このコマンドの結果をモジュールオプションの値フィールドに貼り付けます。デフォルトの値は空の文字列です。</p>
managedConnectionFactoryName	JCA リソース	なし	データソースの JCA 接続ファクトリーの名前。

表12.30 PropertiesUsers

コード	PropertiesUsers
クラス	org.jboss.security.auth.spi.PropertiesUsersLoginModule
説明	認証用ユーザー名およびパスワードを格納するプロパティファイルを使用します。認証 (ロールマッピング) は提供されません。このモジュールは、テスト向けのみに限定されます。

表12.31 SimpleUsers

コード	SimpleUsers
クラス	org.jboss.security.auth.spi.SimpleUsersLoginModule
説明	このログインモジュールは module-option を使用してユーザー名とクリアテキストパスワードを保存します。 module-option の name および value 属性はユーザー名とパスワードを指定します。テストの目的でのみ含まれているため、本番環境

表12.32 LdapUsers

コード	LdapUsers
クラス	org.jboss.security.auth.spi.LdapUsersLoginModule
説明	LdapUsers モジュールは ExtendedLDAP および AdvancedLdap モジュールが導入されたため廃止になりました。

表12.33 Kerberos

コード	Kerberos
クラス	com.sun.security.auth.module.Krb5LoginModule 。IBM JDK でのクラス名は com.ibm.security.auth.module.Krb5LoginModule 。
説明	GSSAPI を使用して Kerberos ログイン認証を実行します。このモジュールは、Sun Microsystems により提供された API のセキュリティーフレームワークの一部です。詳細については、 http://docs.oracle.com/javase/7/docs/jre/api/security/jaas/spec/com/sun/security/auth/module/Krb5LoginModule.html を参照してください。このモジュールは、認証とロールのマッピングを処理する別のモジュールと組み合わせる必要があります。

表12.34 Kerberos モジュールオプション

オプション	タイプ	デフォルト	説明
storekey	true または false	false	KerberosKey をサブジェクトのプライベートクレデンシャルに追加するかどうか。
doNotPrompt	true または false	false	true に設定された場合、ユーザーはパスワードを要求されません。
useTicketCache	true または false のブール値	false	true の場合、TGT はチケットキャッシュから取得されます。 false の場合、チケットキャッシュは使用されません。

オプション	タイプ	デフォルト	説明
ticketcache	Kerberos チケットキャッシュを表すファイルまたはリソース。	<p>デフォルトは使用するオペレーティングシステムによって異なります。</p> <ul style="list-style-type: none"> ● Red Hat Enterprise Linux / Solaris の場合: /tmp/krb5cc_<i>uid</i>(オペレーティングシステムの UID 数値を使用します)。 ● Microsoft Windows Server の場合: Local Security Authority (LSA) API を使用してチケットキャッシュを見つけます。 	チケットキャッシュの場所。
useKeyTab	true または false	false	キーテーブルファイルからプリンシパルのキーを取得するかどうか。
keytab	Kerberos keytab を表すファイルまたはリソース。	オペレーティングシステムの Kerberos 設定ファイルの場所または /home/user/krb5.keytab	キーテーブルファイルの場所。
principal	文字列	なし	プリンシパルの名前。これは、 host/testserver.acme.com などの単純なユーザー名またはサービス名のいずれかになります。これは、キーテーブルからプリンシパルを取得する代わり、またはキーテーブルに複数のプリンシパルが含まれる場合に使用します。

オプション	タイプ	デフォルト	説明
useFirstPass	true または false	false	javax.security.auth.login.name および javax.security.auth.login.password をキーとして使用して、モジュールの共有状態からユーザー名とパスワードを取得するかどうか。認証が失敗した場合、再試行は行われません。
tryFirstPass	true または false	false	useFirstPass と同じです。ただし、認証が失敗した場合、モジュールは CallbackHandler を使用して新しいユーザー名とパスワードを取得します。2 番目の認証が失敗した場合、失敗は読み出し元アプリケーションに報告されます。
storePass	true または false	false	モジュールの共有状態でユーザー名とパスワードを格納するかどうか。これは、キーがすでに共有状態である場合、または認証に失敗した場合は、行われません。
clearPass	true または false	false	これを true に設定して、認証段階が完了した後に共有状態からユーザー名とパスワードを削除します。

表12.35 SPNEGO

コード	SPNEGO
クラス	org.jboss.security.negotiation.spnego.SPNEGOLoginModule
説明	Microsoft Active Directory サーバーまたは SPNEGO をサポートする他の環境に対して SPNEGO 認証を許可します。SPNEGO は Kerberos クレデンシャルを持つこともできます。このモジュールは、認証とロールのマッピングを処理する別のモジュールと組み合わせる必要があります。

表12.36 SPNEGO モジュールオプション

オプション	タイプ	デフォルト	説明
serverSecurityDomain	string	null	Kerberos ログインモジュールを介してサーバーサービスの ID を読み出すために使用されるドメインを定義します。このプロパティは必ず設定する必要があります。
removeRealmFromPrincipal	boolean	false	処理を続行する前に Kerberos レalmをプリンシパルから削除する必要があることを指定します。
usernamePasswordDomain	string	null	Kerberos が失敗したときにフェールオーバーログインとして使用する必要がある設定内の別のセキュリティードメインを指定します。

表12.37 AdvancedLdap

コード	AdvancedLdap
クラス	org.jboss.security.negotiation.AdvancedLdapLoginModule
説明	SASL や JAAS セキュリティードメインの使用など、追加機能を提供するモジュール。

表12.38 AdvancedLdap モジュールオプション

オプション	タイプ	デフォルト	説明
bindAuthentication	文字列	なし	ディレクトリーサーバーへのバインディングに使用する SASL 認証のタイプ。
java.naming.provider.url	string	java.naming.security.protocol の値が SSL の場合は ldap://localhost:686 で、その他の場合は ldap://localhost:389 。	ディレクトリーサーバーの URI。
baseCtxDN	完全修飾 DN	なし	検索の基盤として使用する識別名。

オプション	タイプ	デフォルト	説明
baseFilter	LDAP 検索フィルターを表す文字列。	なし	検索結果を絞り込むために使用するフィルター。
roleAttributeID	LDAP 属性を表す文字列値。	なし	承認ロールの名前が含まれる LDAP 属性。
roleAttributesDN	true または false	false	ロール属性が識別名 (DN) であるかどうか。
roleNameAttributeID	LDAP 属性を表す文字列。	なし	実際のロール属性が含まれる RoleAttributeID 内に格納された属性。
recurseRoles	true または false	false	ロールに対して RoleAttributeID を再帰的に検索するかどうか。
referralUserAttributeIDToCheck	attribute	なし	リファーラル (referral) を使用しない場合はこのオプションを無視してもかまいません。リファーラルを使用し、ロールオブジェクトがリファーラル内部にある場合、このオプションは特定のロール (例: member) に対して定義されたユーザーが含まれる属性名を示します。ユーザーはこの属性名の内容に対してチェックされます。このオプションが設定されていないとチェックは常に失敗するため、ロールオブジェクトはリファーラルツリーに保存できません。

表12.39 AdvancedADLdap

コード	AdvancedADLdap
クラス	org.jboss.security.negotiation.AdvancedADLoginModule
説明	このモジュールは AdvancedLdap ログインモジュールを拡張し、Microsoft Active Directory に関連する追加パラメーターを追加します。

表12.40 UsersRoles

コード	UsersRoles
クラス	org.jboss.security.auth.spi.UsersRolesLoginModul
説明	2つの異なるプロパティファイルに格納された複数のユーザーおよびユーザーロールをサポートする簡単なログインモジュール。

表12.41 UsersRoles モジュールオプション

オプション	タイプ	デフォルト	説明
usersProperties	ファイルまたはリソースへのパス。	users.properties	ユーザーからパスワードへのマッピングが含まれるファイルまたはリソース。ファイルの形式は username=password になります。
rolesProperties	ファイルまたはリソースへのパス。	roles.properties	ユーザーからロールへのマッピングが含まれるファイルまたはリソース。ファイルの形式は username=role1,role2,role3 になります。
password-stacking	useFirstPass または false	false	このログインモジュールが ID を見つけるため LoginContext に格納された情報を最初に探すことを示す useFirstPass の値。このオプションは、他のログインモジュールをこのモジュールとスタックする場合に使用できます。

オプション	タイプ	デフォルト	説明
hashAlgorithm	パスワードをハッシュ化するアルゴリズムを表す文字列。	none	パスワードをハッシュ化するために使用する java.security.MessageDigest アルゴリズムの名前。デフォルト値はないため、ハッシュを有効にするためにこのオプションを明示的に設定する必要があります。 hashAlgorithm が指定された場合は、 inputPassword 引数として UsernamePasswordLoginModule.validatePassword に渡す前に CallbackHandler から取得されたクリアテキストパスワードがハッシュ化されます。 users.properties ファイルに格納されたパスワードも、同様にハッシュ化する必要があります。
hashEncoding	base64 または hex	base64	hashAlgorithm も設定されている場合、ハッシュ化されたパスワードの文字列形式。
hashCharset	文字列	コンテナのランタイム環境に設定されるデフォルトのエンコーディング。	クリアテキストのパスワードをバイトアレイに変換するために使用されるエンコーディング。
unauthenticatedIdentity	プリンシパル名	なし	認証情報を含まない要求に割り当てるプリンシパル名を定義します。これにより、保護されていないサーブレットは特定のルールを必要としない EJB でメソッドを呼び出すことができますようになります。このようなプリンシパルは関連付けられたルールを持たず、 unchecked permission 制約に関連付けられたセキュアでない EJB または EJB メソッドにのみアクセスできます。

カスタム認証モジュール

認証モジュールは、**javax.security.auth.spi.LoginModule** の実装です。カスタム認証モジュールの作成の詳細については、API ドキュメンテーションを参照してください。

[Report a bug](#)

12.2. 含まれる承認モジュール

以下のモジュールは承認サービスを提供します。

コード	クラス
DenyAll	org.jboss.security.authorization.modules.AllDenyAuthorizationModule
PermitAll	org.jboss.security.authorization.modules.AllPermitAuthorizationModule
Delegating	org.jboss.security.authorization.modules.DelegatingAuthorizationModule
Web	org.jboss.security.authorization.modules.web.WebAuthorizationModule
JACC	org.jboss.security.authorization.modules.JACCAuthorizationModule
XACML	org.jboss.security.authorization.modules.XACMLAuthorizationModule

AllDenyAuthorizationModule

認証リクエストを常に拒否する単純な承認モジュールです。設定オプションはありません。

AllPermitAuthorizationModule

認証リクエストを常に許可する単純な承認モジュールです。設定オプションはありません。

DelegatingAuthorizationModule

意思決定を設定済みの delegate へ移譲するデフォルトの認証モジュールです。

WebAuthorizationModule

デフォルトの Tomcat 承認ロジック (permit all) を持つデフォルトの Web 認証モジュールです。

JACCAuthorizationModule

このモジュールは 2 つの delegate を使用して JACC セマンティックを強制します (Web コンテナ承認リクエストの WebJACCPolicyModuleDelegate と、EJB コンテナリクエストの EJBJACCPolicyModuleDelegate)。設定オプションはありません。

XACMLAuthorizationModule

このモジュールは Web コンテナと EJB コンテナの 2 つの delegate を使用して (WebXACMLPolicyModuleDelegate および EJBXACMLPolicyModuleDelegate) XACML 承認を強制します。このモジュールは登録されたポリシーを基に PDP オブジェクトを作成し、それに対して Web または EJB リクエストを評価します。

AbstractAuthorizationModule

これはオーバーライドされる必要があるベースの承認モジュールで、他の承認モジュールへ委譲するためのファシリティーを提供します。

[Report a bug](#)

12.3. 含まれるセキュリティーマッピングモジュール

以下のセキュリティーマッピングロールが JBoss EAP 6 で提供されます。

コード	クラス
PropertiesRoles	org.jboss.security.mapping.providers.role.Pr opertiesRolesMappingProvider
SimpleRoles	org.jboss.security.mapping.providers.role.Si mpleRolesMappingProvider
DeploymentRoles	org.jboss.security.mapping.providers.Deploy mentRolesMappingProvider
DatabaseRoles	org.jboss.security.mapping.providers.role.Da tabaseRolesMappingProvider
LdapRoles	org.jboss.security.mapping.providers.role.Ld apRolesMappingProvider
LdapAttributes	org.jboss.security.mapping.providers.attribut e.LdapAttributeMappingProvider

DeploymentRolesMappingProvider

jboss-web.xml および **jboss-app.xml** デプロイメント記述子で可能なプリンシパルからロールへのマッピングを考慮するロールマッピングモジュール。

例12.1 例

```
<jboss-web>
...
<security-role>
  <role-name>Support</role-name>
  <principal-name>Mark</principal-name>
  <principal-name>Tom</principal-name>
</security-role>
...
</jboss-web>
```

org.jboss.security.mapping.providers.DeploymentRoleToRolesMappingProvider

jboss-web.xml および **jboss-app.xml** デプロイメント記述子で指定できるプリンシパルからロールへのマッピングを考慮するロールツーロール (Role to Roles) マッピングモジュールです。この場合、principal-name は他のロールをマップするロールを示します。

例12.2 例

```
<jboss-web>
...
<security-role>
  <role-name>Employee</role-name>
  <principal-name>Support</principal-name>
  <principal-name>Sales</principal-name>
</security-role>
...
</jboss-web>
```

これは、Support または Sales をロールに持つ各プリンシパルには Employee ロールも割り当てられることを意味します。

org.jboss.security.mapping.providers.OptionsRoleMappingProvider

オプションからロールを選び、渡されたグループの前に追加するロールマッピングプロバイダー。ロール (値) のカンマ区切りリストが含まれるロール名 (キー) のプロパティースタイルマッピングを取ります。

org.jboss.security.mapping.providers.principal.SimplePrincipalMappingProvider

SimplePrincipal を取り、異なるプリンシパル名で SimplePrincipal を変換するプリンシパルマッピングプロバイダー。

DatabaseRolesMappingProvider

データベースからロールを読み取る MappingProvider。

オプション:

- **dsJndiName:** ロールをユーザーにマップするために使用されるデータソースの JNDI 名。
- **rolesQuery:** このオプションは「select RoleName from Roles where User=?」と同等の準備済みステートメントです。「?」は現在のプリンシパル名に置き換えられます。
- **suspendResume:** ブール値 - ロールの検索の実行中に、現在のスレッドに関連するトランザクションを停止し、その後再開します。
- **transactionManagerJndiName:** トランザクションマネージャーの JNDI 名 (デフォルトは java:/TransactionManager)。

LdapRolesMappingProvider

ロールを検索するため LDAP サーバーを使用してロールを割り当てるマッピングプロバイダー。

オプション:

- **bindDN:** ユーザーおよびロールクエリーのために LDAP サーバーに対してバインドするために使用される DN。この DN は baseCtxDN および rolesCtxDN の値では読み取りおよび検索パーミッションが必要です。

- **bindCredential**: bindDN のパスワード。これは、jaasSecurityDomain が指定された場合に暗号化できます。
- **rolesCtxDN**: ユーザーロールを検索するコンテキストの固定 DN。これは、実際のロールが存在する DN ではなく、ユーザーロールを含むオブジェクトが存在する DN です。たとえば、Microsoft Active Directory サーバーでは、これは、ユーザーアカウントが存在する DN です。
- **roleAttributeID**: 承認ロールの名前が含まれる LDAP 属性。
- **roleAttributesDN**: **roleAttributeID** にロールオブジェクトの完全修飾 DN が含まれるかどうか。false の場合は、コンテキスト名の **roleNameAttributeID** 属性の値からロール名が取得されます。Microsoft Active Directory などの特定のディレクトリースキーマでは、この属性を **true** に設定する必要があります。
- **roleNameAttributeID**: ロール名を含む **roleCtxDN** コンテキスト内の属性の名前。**roleAttributesDN** プロパティが **true** に設定された場合、このプロパティはロールオブジェクトの名前属性を見つけるために使用されます。
- **parseRoleNameFromDN**: クエリーによって返された DN に **roleNameAttributeID** が含まれるかどうかを示すフラグ。**true** に設定された場合、DN は **roleNameAttributeID** に対してチェックされます。**false** に設定された場合、DN は **roleNameAttributeID** に対してチェックされません。このフラグは LDAP クエリーのパフォーマンスを向上できます。
- **roleFilter**: 認証済みユーザーと関連付けられたロールを検索するために使用される検索フィルター。入力ユーザー名またはログインモジュールコールバックから取得された **userDN** が **{0}** 式が使用されたフィルターに置換されます。認証済み **userDN** は、**{1}** が使用されたフィルターに置換されます。入力ユーザー名に一致する検索フィルター例は、**(member={0})** です。認証済み **userDN** に一致する他の例は **(member={1})** です。
- **roleRecursion**: ロール検索が一致するコンテキストで行われる再帰のレベル数。再帰を無効にするには、これを **0** に設定します。
- **searchTimeLimit**: ユーザー/ロール検索のタイムアウト (ミリ秒単位)。デフォルトは 10000 (10 秒) です。
- **searchScope**: 使用する検索範囲。

PropertiesRolesMappingProvider

「username=role1,role2,...」という形式でプロパティファイルからロールを読み取る MappingProvider。

オプション:

- **rolesProperties**: プロパティでフォーマットされるファイル名。JBoss 変数の拡張は **\${jboss.variable}** という形式で使われます。

SimpleRolesMappingProvider

オプションマップからロールを読み取る簡単な MappingProvider。オプションの属性名はロールを割り当てるプリンシパルの名前、属性値はプリンシパルに割り当てるロール名のカンマ区切りリストです。

例12.3 例

```
<module-option name="JavaDuke" value="JBossAdmin,Admin"/>
<module-option name="joe" value="Users"/>
```

org.jboss.security.mapping.providers.attribute.DefaultAttributeMappingProvider

モジュールをチェックし、マッピングコンテキストからプリンシパル名を見つけ、principalName + ".email" という名前のモジュールオプションから属性電子メールアドレスを作成し、それを指定のプリンシパルへマップします。

LdapAttributeMappingProvider

属性を LDAP からサブジェクトへマップします。オプションには、ご使用の LDAP JNDI プロバイダーがサポートするオプションがすべて含まれます。

例12.4 標準のプロパティー名の例

```
Context.INITIAL_CONTEXT_FACTORY = "java.naming.factory.initial"
Context.SECURITY_PROTOCOL = "java.naming.security.protocol"
Context.PROVIDER_URL = "java.naming.provider.url"
Context.SECURITY_AUTHENTICATION = "java.naming.security.authentication"
```

オプション:

- **bindDN**: ユーザーおよびロールクエリーのために LDAP サーバーに対してバインドするために使用される DN。この DN は baseCtxDN および rolesCtxDN の値では読み取りおよび検索パーミッションが必要です。
- **bindCredential**: bindDN のパスワード。これは、jaasSecurityDomain が指定された場合に暗号化できます。
- **baseCtxDN**: ユーザー検索を開始するコンテキストの固定 DN。
- **baseFilter**: 認証するユーザーのコンテキストを見つけるために使用される検索フィルター。入力ユーザー名またはログインモジュールコールバックから取得された **userDN** が、**{0}** 式が使用されたフィルターに置換されます。この置換の挙動は標準の **__DirContext.search(Name, String, Object[], SearchControls cons)__** メソッドからになります。一般的な検索フィルターの例は **(uid={0})** です。
- **searchTimeLimit**: ユーザー/ロール検索のタイムアウト (ミリ秒単位)。デフォルトは 10000 (10 秒) です。
- **attributeList**: ユーザーの属性のカンマ区切りリスト (例: mail,cn,sn,employeeType,employeeNumber)。
- **jaasSecurityDomain**: **java.naming.security.principal** の復号化に使用する JaasSecurityDomain。パスワードの暗号化された形式は **JaasSecurityDomain#encrypt64(byte[])** によって返されます。**org.jboss.security.plugins.PBEUtils** を使用して暗号化された形式を生成することもできます。

[Report a bug](#)

12.4. 含まれるセキュリティー監査プロバイダーモジュール

JBoss EAP 6 はセキュリティー監査プロバイダーを 1 つ提供します。

コード	クラス
LogAuditProvider	org.jboss.security.audit.providers.LogAuditProvider

[Report a bug](#)

第13章 サブシステムの設定

13.1. サブシステム設定の概要

はじめに

JBoss EAP 6 は単純化された設定を使用します (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)。ドメインモードでは、各ホストコントローラーに対して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 を手動で編集する必要はありません。設定は、管理 API により自動的にスキャンされ、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により、JBoss EAP 6 の各側面を設定することができます。

JBoss EAP 6 は、モジュールベースのクラスローディングの概念に基づいて構築されています。プラットフォームによって提供される各 API やサービスはモジュールとして実装されます。このモジュールはオンデマンドでロード/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には **`EAP_HOME/domain/configuration/domain.xml`**、スタンドアロンサーバーの場合には **`EAP_HOME/standalone/configuration/standalone.xml`** という統合設定ファイルに保管されます。多くのサブシステムには、以前の JBoss EAP の旧バージョンでデプロイメント記述子に追加された設定詳細が含まれます。

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、ご使用のインストールの **`EAP_HOME/docs/schema/`** ディレクトリーにあります。

以下のサブシステムは、設定可能な属性や要素がないため簡易サブシステムと呼ばれ、通常は設定ファイルの最上部に記載されます。

簡易サブシステム

- **ee**– Java EE 6 API 実装
- **ejb**– Enterprise JavaBeans (EJB) サブシステム
- **jaxrs**– RESTeasy によって提供される JAX-RS API
- **sar**– サービスアーカイブをサポートするサブシステム
- **threads**– プロセススレッドをサポートするサブシステム
- **weld**– Weld によって提供される Contexts and Dependency Injection (コンテキストと依存性の注入) API

[Report a bug](#)

第14章 ロギングサブシステム

14.1. はじめに

14.1.1. 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ために設定可能な高度なロギング機能を提供します。ロギングサブシステムは JBoss LogManager を基盤とし、JBoss Logging 以外にも複数のサードパーティーアプリケーションのロギングフレームワークをサポートします。

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

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

[Report a bug](#)

14.1.2. 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>

[Report a bug](#)

14.1.3. ブートロギングの設定

ブートロギングは、サーバーの「ブート中」(起動中)に発生したイベントの記録です。

サーバー起動時に **logging.properties** ファイルが使用できる場合、これらのプロパティー設定はロギングサブシステムが初期化される前に発生したイベントを記録するために使用されます。この時点で、ロギングサブシステムがイベントの記録を引き継ぎます。

管理 CLI を使用するか、サーバー設定ファイルを手作業で編集して **logging** サブシステムを編集すると、**logging.properties** ファイルが更新されます。

インストールに **logging.properties** ファイルがない場合、ロギングサブシステムが初期化される前のブート中に通常表示されるログメッセージは失われます。ロギングシステムが初期化されると、メッセージが再度ログに表示されます。

**警告**

サーバーのブートに深刻な問題があり、ホストまたはプロセスコントローラーから追加のロギングが必要な場合のみ **logging.properties** ファイルを直接編集することが推奨されます。

[Report a bug](#)

14.1.4. 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分析を行うのに便利です。JBoss EAP 6.3 より、ガベッジコレクションロギングは IBMJDK を除くサポートされる設定の **standalone** モードでデフォルトで有効になっています。

ロギングはファイル **EAP_HOME/standalone/log/gc.log.*digit*** へ出力されます。ログローテーションは有効になっており、ログファイルの数は 5 に制限され、各ログファイルの最大サイズは 3 MiB に制限されています。

[Report a bug](#)

14.1.5. 暗黙的なロギング API の依存関係

JBoss EAP 6 のロギングサブシステムには、コンテナ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** 属性が含まれています。デフォルトでは、この属性は **true** に設定され、暗黙的なロギング API 依存関係はすべてデプロイメントへ追加されます。**false** に設定すると、暗黙的なロギング API 依存関係は追加されません。

管理 CLI を使用して **add-logging-api-dependencies** 属性を設定できます。例は次のとおりです。

```
/subsystem=logging:write-attribute(name=add-logging-api-dependencies, value=false)
```

[Report a bug](#)

14.1.6. デフォルトのログファイルの場所

これらは、デフォルトのロギング設定に対して作成されたログファイルです。デフォルトの設定では、周期的なログハンドラーを使用してサーバーログファイルが書き込まれます。

表14.1 スタンドアロンサーバーのデフォルトログファイル

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

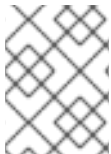
表14.2 管理対象ドメイン用のデフォルトログファイル

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。

[Report a bug](#)

14.1.7. ログインのフィルター式

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ロー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。この場合は、ハンドラーに配置されたフィルターよりもロガーフィルターが優先されます。



注記

ルートロガーに対して指定された **filter-spec** は他のロガーによって継承されません。ハンドラーごとに **filter-spec** を指定する必要があります。

表14.3 ログインのフィルター式

フィルタータイプ	説明	パラメーター
expression		
Accept accept	すべてのログメッセージを許可します。	accept
Deny deny	すべてのログメッセージを拒否します。	deny
Not 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"))

フィルタータイプ expression	説明	パラメーター
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"))
Any 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")
Level Range levelRange [minLevel,maxLevel]	指定のレベル範囲内でログメッセージをフィルターします。	<p>フィルター式では、[を使用して含まれる最小レベルが示され、]を使用して含まれる最大レベルが示されます。この代わりに、それぞれ ((または)) を使用して含まれるレベルを示すことが可能です。式の最初の引数が許可される最小レベルで、2つ目の引数が許可される最大レベルになります。</p> <p>以下に例を示します。</p> <ul style="list-style-type: none"> ● 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。

フィルタータイプ expression	説明	パラメーター
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")
Substitute 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")

[Report a bug](#)

14.1.8. ログレベル

ログレベルとは、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、そのメッセージを送信するために選択したログインフレームワークの適切なメソッドを使用して開発者が指定します。

JBoss EAP 6 は、サポートされるアプリケーションログイン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、ログレベルの低い順に **TRACE**、**DEBUG**、**INFO**、**WARN**、**ERROR** および **FATAL** となります。

ログレベルはログカテゴリ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** のレベルのメッセージのみを記録します。

[Report a bug](#)

14.1.9. サポート対象のログレベル

表14.4 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のみに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活動の進捗状況を表示する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のデバッグ時のみに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。将来的にエラーをもたらす可能性のある状況を示します。
WARNING	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。

[Report a bug](#)

14.1.10. 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。そのパッケージ内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 사용할 수 있습니다。

[Report a bug](#)

14.1.11. ルートロガーについて

ルートロガーは、サーバーに送信された (指定レベルの) ログメッセージの中でログカテゴリによってキャプチャーされない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単一または複数の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定期ログハンドラーを使用するように設定されています。定期ログハンドラーは、**server.log** ファイルに書き込むように設定されています。このファイルはサーバーログと呼ばれる場合もあります。

[Report a bug](#)

14.1.12. ログハンドラー

ログハンドラーは、キャプチャーされたログメッセージが JBoss EAP 6 によって記録される方法を定義します。設定可能なログハンドラーは、**Console**、**File**、**Periodic**、**Size**、**Async**、**Custom**、および **syslog** です。

[Report a bug](#)

14.1.13. 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 (stdout) 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で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は、コンソールログハンドラーからのメッセージは保存されません。

File

ファイルログハンドラーは、ログメッセージを指定のファイルに書き込む、最もシンプルなログハンドラーです。

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。その時間が経過した後は、指定のタイムスタンプが追記され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続けます。

Size

サイズログハンドラーは、指定のファイルが指定サイズに到達するまで、そのファイルにログメッセージを書き込みます。ファイルが指定したサイズに到達すると、名前に数値の接頭辞を追加して名前変更され、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続けます。各サイズログハンドラーは、このような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。

Async

Async ログハンドラーは、単一または複数のログハンドラーを対象とする非同期動作を提供するラッパーログハンドラーです。Async ログハンドラーは、待ち時間が長かったり、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にパフォーマンス上の問題があるログハンドラーに対して有効です。

Custom

Custom ログハンドラーにより、実装された新たな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` を拡張する Java クラスとして実装し、モジュール内に格納する必要があります。

syslog

syslog-handler 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。

[Report a bug](#)

14.1.14. 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。 `java.util.Formatter` クラスを基にし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` は次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

[Report a bug](#)

14.1.15. ログフォーマッター構文

表14.5 ログフォーマッター構文

記号	説明
<code>%c</code>	ロギングイベントのカテゴリ
<code>%p</code>	ログエントリーのレベル (情報やデバッグなど)
<code>%P</code>	ログエントリーのローカライズレベル
<code>%d</code>	現在の日付/時刻 (yyyy-MM-dd HH:mm:ss,SSS 形式)
<code>%r</code>	相対時間 (ログが初期化された以降のミリ秒単位の時間)
<code>%z</code>	タイムゾーン
<code>%k</code>	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)
<code>%m</code>	ログメッセージ (例外トレースを除外)
<code>%s</code>	単純なログメッセージ (例外トレースなし)
<code>%e</code>	例外スタックトレース (拡張モジュール情報なし)

記号	説明
%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)

[Report a bug](#)

14.2. 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については、「[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、以下の手順を実行します。

1. 管理コンソールへのログイン
2. ロギング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とで異なります。
 - **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** を展開してから **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** メニューの **Core** を展開してから **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。

[Report a bug](#)

14.3. CLI でのロギング設定

前提条件

管理 CLI が実行され、関係する JBoss EAP インスタンスに接続している必要があります。詳細については、「[管理 CLI の起動](#)」を参照してください。

[Report a bug](#)

14.3.1. 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。

管理対象ドメインの場合は、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/` を `/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14.1 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

次の構文で **read-resource** 操作を使用します。

```
/subsystem=logging/root-logger=ROOT:read-resource
```

例14.2 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つです。

```
/subsystem=logging/root-logger=ROOT:write-attribute(name="level", value="LEVEL")
```

例14.3 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14.4 ログハンドラーの削除

```
[standalone@localhost:9999 /] /subsystem=logging/root-logger=ROOT:remove-  
handler(name="FILE")  
{"outcome" => "success"}
```

[Report a bug](#)

14.3.2. 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で設定する場合、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。

管理対象ドメインの場合は、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/** を **/profile=NAME/subsystem=logging/** に置き換えます)。

ログカテゴリーの追加

次の構文で **add** 操作を使用します。CATEGORY は追加するカテゴリーに置き換えます。

```
/subsystem=logging/logger=CATEGORY:add
```

例14.5 新規カテゴリーの追加

```
[standalone@localhost:9999 /] /subsystem=logging/logger=com.company.accounts.rec:add  
{"outcome" => "success"}  
[standalone@localhost:9999 /]
```

ログカテゴリーの設定の表示

次の構文で **read-resource** 操作を使用します。CATEGORY はカテゴリー名に置き換えます。

```
/subsystem=logging/logger=CATEGORY:read-resource
```

例14.6 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
  }
}
[standalone@localhost:9999 /]
```

ログレベルの設定

次の構文で **write-attribute** 操作を使用します。CATEGORY の箇所はログカテゴリー名に、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14.7 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"}
[standalone@localhost:9999 /]
```

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ーの設定

次の構文で **write-attribute** 操作を使用します。CATEGORY はログカテゴリー名に置き換えます。root ロガーのハンドラーを使用するためのこのログカテゴリーには、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には false に置き換えてください。

```
/subsystem=logging/logger=CATEGORY:write-attribute(name="use-parent-handlers",
value="BOOLEAN")
```

例14.8 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"}
```

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

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 はカテゴリ一名に、HANDLER は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14.9 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 はカテゴリ一名に、HANDLER は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14.10 ログハンドラーの削除

```
[standalone@localhost:9999 /] /subsystem=logging/logger=jacorb:remove-  
handler(name="AccountsNFSAsync")  
{"outcome" => "success"}  
[standalone@localhost:9999 /]
```

カテゴリの削除

次の構文で **remove** 操作を使用します。CATEGORY は削除するカテゴリの名前に置き換えます。

```
/subsystem=logging/logger=CATEGORY:remove
```

例14.11 ログカテゴリの削除

```
[standalone@localhost:9999 /]  
/subsystem=logging/logger=com.company.accounts.rec:remove  
{"outcome" => "success"}  
[standalone@localhost:9999 /]
```

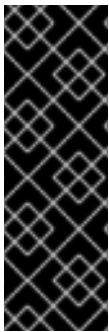
[Report a bug](#)

14.3.3. 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グインプロファイルで設定する場合、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。

管理対象ドメインの場合は、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/** を **/profile=NAME/subsystem=logging/** に置き換えます)。

Console ログハンドラーの追加

次の構文で **add** 操作を使用します。HANDLER は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14.12 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

次の構文で **read-resource** 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置き換えます。

```
/subsystem=logging/console-handler=HANDLER:read-resource
```

例14.13 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てください。

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="level", value="INFO")
```

例14.14 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 はコンソールログハンドラーの名前に置き換えます。TARGET は、システムエラーストリームの場合には **System.err** に、標準出力ストリームの場合には **System.out** に置き換えてください。

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="target", value="TARGET")
```

例14.15 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、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14.16 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 は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の文字列に置き換えます。

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="formatter",
value="FORMAT")
```

例14.17 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 は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直ちに書き込むようにするには、BOOLEAN を **true** に置き換えます。

```
/subsystem=logging/console-handler=HANDLER:write-attribute(name="autoflush",
value="BOOLEAN")
```

例14.18 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

次の構文で **remove** 操作を使用します。HANDLER は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14.19 Console ログハンドラーの削除

```
[standalone@localhost:9999 /] /subsystem=logging/console-handler=ERRORCONSOLE:remove
{"outcome" => "success"}
```

[Report a bug](#)

14.3.4. 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で設定する場合、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。

管理対象ドメインの場合は、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/** を **/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

次の構文で **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* をファイルを格納するディレクトリーの名前に置き換えます。 *DIR* の値はパス変数に指定できます。

```
/subsystem=logging/file-handler=HANDLER:add(file={"path"=>"PATH", "relative-to"=>"DIR"})
```

例14.20 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"}
[standalone@localhost:9999 /]
```

ファイルログハンドラー設定の表示

次の構文で **read-resource** 操作を使用します。HANDLER は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14.21 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,
    "encoding" => undefined,
    "file" => {
      "path" => "accounts.log",
      "relative-to" => "jboss.server.log.dir"
    },
    "filter" => undefined,
    "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
    "level" => 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 はファイルログハンドラーの名前に置き換えます。LOG_LEVEL_VALUE は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14.22 ログレベルの変更

```
/subsystem=logging/file-handler=accounts_log:write-attribute(name="level", value="DEBUG")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

追加動作の設定

次の構文で **write-attribute** 操作を使用します。HANDLER は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14.23 追加プロパティの変更

```
[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"
  }
}
[standalone@localhost:9999 /]
```

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 は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直ちに書き込むようにするには、BOOLEAN を **true** に置き換えます。

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="autoflush",
value="BOOLEAN")
```

例14.24 自動フラッシュプロパティの変更

```
[standalone@localhost:9999 /] /subsystem=logging/file-handler=accounts_log:write-
attribute(name="autoflush", value="false")
{
  "outcome" => "success",
  "response-headers" => {"process-state" => "reload-required"}
}
[standalone@localhost:9999 /]
```

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。

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 はファイルログハンドラーの名前に置き換えます。ENCODING は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14.25 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 をファイルを格納するディレクトリーの名前に置き換えます。DIR の値はパス変数を指定できます。

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="file", value={"path"=>"PATH",
"relative-to"=>"DIR"})
```

例14.26 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"}
[standalone@localhost:9999 /]
```

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* は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の文字列に置き換えます。

```
/subsystem=logging/file-handler=HANDLER:write-attribute(name="formatter", value="FORMAT")
```

例14.27 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"}
[standalone@localhost:9999 /]
```

File ログハンドラーの削除

次の構文で **remove** 操作を使用します。*HANDLER* は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14.28 File ログハンドラーの削除

```
[standalone@localhost:9999 /] /subsystem=logging/file-handler=accounts_log:remove
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

ログハンドラーは、ログカテゴリ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。

[Report a bug](#)

14.3.5. 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
- ハンドラーが **autoflush** を使用するかどうかを設定します。
- 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で設定する場合、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。

管理対象ドメインの場合は、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/** を **/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

次の構文で **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* をファイルが存在するディレクトリーの名前に置き換えます。 *DIR* の値をパス変数に指定できます。 *SUFFIX* を、使用するファイルローテーション接尾辞に置き換えます。

例14.29 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"}
[standalone@localhost:9999 /]
```

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

次の構文で **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14.30 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,
    "encoding" => undefined,
    "file" => {
      "path" => "daily-debug.log",
      "relative-to" => "jboss.server.log.dir"
    },
    "filter" => undefined,
    "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
    "level" => undefined
  }
}
```

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14.31 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14.32 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。このハンドラーがすぐに出力を書き込む場合は、*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14.33 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14.34 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* をファイルが存在するディレクトリーの名前に置き換えます。*DIR* の値をパス変数に指定できます。

例14.35 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* を必要なフォーマッター文字列に置き換えます。

例14.36 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"}
[standalone@localhost:9999 /]
```

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14.37

```
[standalone@localhost:9999 /] /subsystem=logging/periodic-rotating-file-
handler=HOURLY_DEBUG:write-attribute(name="suffix", value=".yyyy-MM-dd-HH")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

周期ログハンドラーの削除

次の構文で **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14.38 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"}  
[standalone@localhost:9999 /]
```

[Report a bug](#)

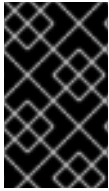
14.3.6. CLI でのサイズログハンドラーの設定

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の設定
- 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対するブート時のローテーションオプションの設定
- 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

ログハンドラーをログイン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。

新しいログハンドラーの追加

次の構文で **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* をファイルが存在するディレクトリの名前に置き換えます。 *DIR* の値をパス変数に指定できます。

例14.39 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

次の構文で **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14.40 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,
    "encoding" => undefined,
    "file" => {
      "path" => "accounts_trace.log",
      "relative-to" => "jboss.server.log.dir"
    },
    "filter" => undefined,
    "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
    "level" => undefined,
    "max-backup-index" => 1,
    "rotate-size" => "2m"
  }
}
```

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

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-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14.41 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"}
[standalone@localhost:9999 /]
```

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14.42 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"
  }
}
[standalone@localhost:9999 /]
```

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。このハンドラーがすぐに出力を書き込む場合は、*BOOLEAN* を **true** に置き換えます。

例14.43 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"}
[standalone@localhost:9999 /]
```

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14.44 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* をファイルが存在するディレクトリーの名前に置き換えます。*DIR* の値をパス変数に指定できます。

例14.45 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* を必要なフォーマッター文字列に置き換えます。

例14.46 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* を最大ファイルサイズに置き換えます。

例14.47 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"}
[standalone@localhost:9999 /]
```

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必要な数に置き換えます。

例14.48 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"}
[standalone@localhost:9999 /]
```

size-rotating-file-handler での rotate-on-boot (ブート時のローテーション) 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のみに使用できます。デフォルト値は **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。

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14.49 サーバーの再起動時に **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"}
[standalone@localhost:9999 /]
```

ログハンドラーの削除

次の構文で **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14.50 ログハンドラーの削除

```
[standalone@localhost:9999 /] /subsystem=logging/size-rotating-file-
handler=ACCOUNTS_TRACE:remove
{"outcome" => "success"}
```

[Report a bug](#)

14.3.7. 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で設定する場合、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。

管理対象ドメインの場合は、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/** を **/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

次の構文で **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要求の最大数に置き換えます。

例14.51

```
[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

次の構文で **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14.52

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:read-resource
{
  "outcome" => "success",
  "result" => {
    "encoding" => undefined,
    "filter" => undefined,
    "formatter" => "%d{HH:mm:ss,SSS} %-5p [%c] (%t) %s%E%n",
    "level" => undefined,
    "overflow-action" => "BLOCK",
    "queue-length" => "50",
    "subhandlers" => undefined
  }
}
[standalone@localhost:9999 /]
```


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14.53

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:write-
attribute(name="level", value="INFO")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

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要求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14.54

```
[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* を *DISCARD* または *BLOCK* に置き換えます。

例14.55

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:write-attribute(name="overflow-action", value="DISCARD")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

サブハンドラーの追加

次の構文で **size-rotating-file-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14.56

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:add-handler(name="NFS_FILE")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

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14.57

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:remove-handler(name="NFS_FILE")
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

非同期ログハンドラーの削除

次の構文で **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14.58

```
[standalone@localhost:9999 /] /subsystem=logging/async-handler=NFS_LOGS:remove
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

[Report a bug](#)

14.3.8. syslog-handler の設定

JBoss EAP 6 の logmanager に syslog-handler が含まれるようになりました。syslog-handler 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のログイン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。ここでは、管理 CLI を使用したハンドラーの作成および設定方法と、利用可能な設定オプションについて取り上げます。

- 管理 CLI のアクセスおよび適切なパーミッション。

手順14.1 syslog-handler の追加

- 以下のコマンドを実行し、syslog-handler を追加します。

```
/subsystem=logging/syslog-handler=HANDLER_NAME:add
```

手順14.2 syslog-handler の設定

- 以下のコマンドを実行し、syslog-handler 属性を設定します。

```
/subsystem=logging/syslog-handler=HANDLER_NAME:write-attribute(name=ATTRIBUTE_NAME,value=ATTRIBUTE_VALUE)
```

手順14.3 syslog-handler の削除

- 以下のコマンドを実行し、既存の syslog-handler を削除します。

```
/subsystem=logging/syslog-handler=HANDLER_NAME:remove
```

表14.6 syslog-handler 設定属性

属性	説明	デフォルト値
port	syslog サーバーがリッスンするポート。	514
app-name	メッセージを RFC5424 形式でフォーマットするときに使用されるアプリケーション名。	null

属性	説明	デフォルト値
enabled	true に設定されると、ハンドラーが有効になり、通常どおり機能します。false に設定されると、ログメッセージの処理時にハンドラーが無視されます。	true
level	ログに記録されるメッセージレベルを指定するログレベル。指定値よりも低いメッセージレベルは破棄されます。	ALL
facility	RFC-5424 および RFC-3164 の定義どおり。	user-level
server-address	syslog サーバーのアドレス。	localhost
hostname	メッセージ送信元のホスト名。	null
syslog-format	RFC 仕様にしたがってログメッセージをフォーマットします。	RFC5424

[Report a bug](#)

14.3.9. 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](#)」のログフォーマッター構文の他に、ログハンドラーと使用するカスタムのログフォーマッターを作成できます。この手順例では、コンソールログハンドラーに対して 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、コンソールログハンドラーを使用します。

手順14.4 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、次のコマンドは **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)
```

- 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。コンソールログへの出力は XML でフォーマットされます。

```
<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>
```

[Report a bug](#)

14.4. デプロイメントごとのロギング

14.4.1. 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挙動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は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は読み取り専用であるため、起動時に変更できないことです。

[Report a bug](#)

14.4.2. デプロイメントごとのロギングの無効化

手順14.5 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つの方法はすべてのバージョンの JBoss EAP 6 で使用できますが、もう1つの方法は JBoss EAP 6.3 およびそれ以降のバージョンでのみ使用できます。

- JBoss EAP 6 (全バージョン)
システムプロパティを追加します。

```
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ロギングサブシステムを除外します。詳細については、『開発ガイド』の「サブシステムをデプロイメントから除外する」を参照してください。

[Report a bug](#)

14.5. ロギングプロファイル

14.5.1. 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 およびそれ以降のバージョンでのみ使用できます。管理コンソールを使用して設定できません。

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られる独立したロギング設定のセットです。ロギングプロファイルはハンドラー、カテゴリーおよびルートロガーを通常のロギングサブシステム同様に定義できますが、他のプロファイルやメインのロギングサブシステムを参照できません。ロギングプロファイルは設定を容易にするため、ロギング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することができます。各プロファイルはサーバー設定に定義されるため、影響するアプリケーションを再デプロイする必要はなく、ロギング設定を変更できます。

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。

- 一意の名前 (必須)
- 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。

[Report a bug](#)

14.5.2. 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てください。

```
/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。

[Report a bug](#)

14.5.3. 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. ルートの設定パスは **/subsystem=logging/logging-profile=NAME** です。
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14.59 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. プロファイルの作成

```
/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")
```

[Report a bug](#)

14.5.4. 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。

前提条件:

1. 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。使用するプロファイルの名前はサーバー管理者に問い合わせてください。

手順14.6 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は、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てください。

```
Manifest-Version: 1.0  
Logging-Profile: NAME
```

アプリケーションに **MANIFEST.MF** ファイルがある場合は、以下の行を追加し、NAME を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、MANIFEST.MF ファイルを **src/main/resources/META-INF/** に置き、次の設定を **pom.xml** ファイルに追加できます。

```
<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。

[Report a bug](#)

14.5.5. 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れを使用するアプリケーションを表しています。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** ファイルを示します。

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。

- 名前は **accounts-app-profile** です。
- ログカテゴリは **com.company.accounts.ejbs** です。
- ログレベルは **TRACE** です。
- ログハンドラーはファイル **ejb-trace.log** を使用するファイルハンドラーです。

例14.60 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"}

[standalone@localhost:9999 /]
```

例14.61 XML 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14.62 アプリケーションの MANIFEST.MF ファイル

Manifest-Version: 1.0
Logging-Profile: accounts-app-profile

[Report a bug](#)

14.6. ロギング設定プロパティ

14.6.1. ルートロガーのプロパティ

表14.7 ルートロガーのプロパティ

プロパティ	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 <code>not(match("JBAS.*"))</code> はパターンに一致しないログエントリを除外するフィルターを定義します。



注記

ルートロガーに対して指定された **filter-spec** は他のハンドラーによって継承されません。ハンドラーごとに **filter-spec** を指定する必要があります。

[Report a bug](#)

14.6.2. ログカテゴリーのプロパティ

表14.8 ログカテゴリーのプロパティ

プロパティ	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[]	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。
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 <code>not(match("JBAS.*"))</code> はパターンに一致しないフィルターを定義します。

[Report a bug](#)

14.6.3. コンソールログハンドラーのプロパティ

表14.9 コンソールログハンドラーのプロパティ

プロパティ	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。
target	文字列	ログハンドラーの出力先となるシステム出力ストリーム。これはシステムエラーストリームの場合は System.err、標準出力ストリームの場合は System.out とすることができます。
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。

[Report a bug](#)

14.6.4. ファイルログハンドラープロパティ

表14.10 ファイルログハンドラープロパティ

プロパティ	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。
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 に設定された場合、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 に対する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。

プロパティ	データタイプ	説明
autoflush	ブール値	true に設定された場合は、受信直後にハンドラーにより割り当てられたファイルに送信されます。 autoflush に対する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。
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。このオブジェクトには、 relative-to と path の 2 つの設定プロパティが含まれます。
relative-to	文字列	ファイルオブジェクトのプロパティであり、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/ ディレクトリーを示します。
path	文字列	ファイルオブジェクトのプロパティであり、ログメッセージが書き込まれるファイルの名前です。これは、完全パスを決定するために、 relative-to プロパティの値に追加されます。
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。

[Report a bug](#)

14.6.5. 周期ログハンドラープロパティ

表14.11 周期ログハンドラープロパティ

プロパティ	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 に設定された場合、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 に対する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 に設定された場合は、受信直後にハンドラーにより割り当てられたファイルに送信されます。 autoflush に対する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。
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	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。
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。このオブジェクトには、 relative-to と path の 2 つの設定プロパティがあります。
relative-to	文字列	ファイルオブジェクトのプロパティであり、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。これは、完全パスを決定するために、 relative-to プロパティの値に追加されます。
suffix	文字列	<p>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、ローテーションの周期を決定するために使用されます。この接尾辞の形式では、ドット (.) の後に、SimpleDateFormat クラスで解析できる date String が指定されます。ログは接尾辞で定義された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単位は分であることに注意してください。</p> <p>たとえば、suffix 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が存在し、suffix 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。周期ログハンドラーの場合、接尾辞には最小時間単位の前の値が含まれます。この例では、dd の値は 19 (前日) です。</p>
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。

[Report a bug](#)

14.6.6. サイズログハンドラープロパティ

表14.12 サイズログハンドラープロパティ

プロパティ	データタイプ	説明
append	ブール値	<p>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 に設定された場合、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 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。</p>

プロパティ	データタイプ	説明
autoflush	ブール値	true に設定された場合は、受信直後にハンドラーにより割り当てられたファイルに送信されます。autoflush 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。
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。
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。このオブジェクトには、 relative-to と path の 2 つの設定プロパティがあります。
relative-to	文字列	ファイルオブジェクトのプロパティであり、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、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/ ディレクトリーを示します。
path	文字列	ファイルオブジェクトのプロパティであり、ログメッセージが書き込まれるファイルの名前です。これは、完全パスを決定するために、 relative-to プロパティの値に追加されます。
rotate-size	Integer	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。数字に追加された単一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。
max-backup-index	Integer	保持されるローテーションログの最大数です。この数字に達すると、最も古いログが再利用されます。
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。デフォルト値は false です。

[Report a bug](#)

14.6.7. 同期ログハンドラープロパティ

表14.13 同期ログハンドラープロパティ

プロパティ	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。
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	キューの長さを越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グイン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、ログイン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。
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 <code>not(match("JBAS.*"))</code> はパターンに一致しないフィルターを定義します。

[Report a bug](#)

14.7. ログイン用 XML 設定例

14.7.1. 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>
```

[Report a bug](#)

14.7.2. 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>
```

[Report a bug](#)

14.7.3. 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>
```

[Report a bug](#)

14.7.4. 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>
```

[Report a bug](#)

14.7.5. 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>
```

[Report a bug](#)

14.7.6. 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>
```

[Report a bug](#)

14.7.7. 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"/>
</async-handler>
```



```
<subhandlers>  
  <handler name="FILE"/>  
  <handler name="accounts-record"/>  
</subhandlers>  
</async-handler>
```

[Report a bug](#)

第15章 INFINISPAN

15.1. INFINISPAN

Infinispan は Java のデータグリッドプラットフォームで、[JSR-107](#) 準拠のキャッシュインターフェースを提供し、キャッシュされたデータを管理します。

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ビーンクラスタリング)
- **hibernate** (エンティティキャッシング)
- **singleton** (シングルトンキャッシング)

各キャッシュコンテナは repl と dist キャッシュを定義します。ユーザーアプリケーションがこれらのキャッシュを直接使用しないようにしてください。



重要

ユーザーはキャッシュコンテナとキャッシュをさらに追加し、JNDI より参照できますが、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については、[Infinispan のドキュメント](#) を参照してください。

[Report a bug](#)

15.2. クラスタリングモード

JBoss EAP 6 で Infinispan を使用すると、2つの方法でクラスタリングを設定できます。ご使用のアプリケーションに使用する適切な方法は、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、およびスケーラビリティ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しい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、通常レプリケート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

ディストリビューションモードでは、Infinispan はクラスターを線形にスケールできます。ディストリビューション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、データの永続性、およびパフォーマンスにはトレードオフが生じます。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を損失す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を配置し、ネットワークトラフィックを軽減します。

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的に実行でき、選択されるモードは要件やアプリケーションによって異なります。同期レプリケーションでは、ユーザーの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終了す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た場合のみ応答がクライアントに送信され、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します。しかし、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、バックグラウンドスレッドがレプリケーションを実行するために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またはキューのサイズによって引き起こされます。レプリケーションキューを使用すると、クラスターノード間の対話が発生しないため、パフォーマンスが向上します。非同期レプリケーションの欠点は、正確性が低いことです。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。

[Report a bug](#)

15.3. キャッシュコンテナ

キャッシュコンテナ

キャッシュコンテナ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です。Infinispan ではデフォルトのキャッシュコンテナは設定 xml ファイル (standalone-ha.xml、standalone-full-ha.xml、domain.xml) に定義されます。1つのキャッシュがデフォルトとして定義され、そのキャッシュがクラスタリングに使用されます。

例15.1 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に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で **repl** キャッシュがデフォルトとして定義されています。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15.1 管理コンソールでの Infinispan キャッシュコンテナの設定

1. 画面の上部にある **Configuration** タブを選択します。
2. ドメインモードの場合は、右上のドロップダウンメニューより **ha** または **full-ha** を選択します。
3. **Subsystems** メニューを展開し、**Infinispan** 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を作成するには、**Cache Containers** テーブルの **Add** をクリックします。
 - b. キャッシュコンテナを削除するには、**Cache Containers** テーブルのキャッシュコンテナをクリックします。**Remove** 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名を指定して **Save** をクリックして確定します。
6. キャッシュコンテナの属性を追加および更新するには、**Cache Containers** テーブルのキャッシュコンテナを選択します。画面の **Details** エリアにある **Attributes**、**Transport**、および **Aliases** タブの1つを選択し、**Edit** をクリックします。**Attributes**、**Transport**、および **Aliases** タブの内容に関するヘルプは、**Need Help?** をクリックします。

手順15.2 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を追加、削除、および更新できます。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が使用されることを確認してください。

- 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。入力中にタブ補完を使用するとコマンドの文字列を補完し、使用可能な属性を表示できます。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)
```

[Report a bug](#)

15.4. キャッシュストア

キャッシュストアはキャッシュにあるデータの外部ストレージです。JBoss EAP 6 に最も適した外部データストアタイプはファイルベース、JDBC ベース、またはリモート Infinispan/JDG ストアです。

ファイルベースのキャッシュストアでは、クラスターの各ノードは通常独自のファイルシステムを持つため、独自のファイルベースキャッシュストアになります。適切なファイルロッキングを実装せず、データが破損するおそれがあるため、ファイルベースのキャッシュストアを共有ファイルシステム (NFS など) に配置しないことが推奨されます。

JDBC ベースのキャッシュストアでは、単一の SQL データベースをすべてのクラスターノードのキャッシュストアとすることが可能です。しかし、キャッシュストアを共有として設定する必要があります (JDBC ベースのキャッシュストアで **shared** 属性を true に設定します)。キャッシュストアが共有として定義されていないと、データベースのデッドロックやパフォーマンスに影響するその他の問題が発生する可能性があります。

[Report a bug](#)

15.5. INFINISPAN の統計

監視目的で、Infinispan キャッシュやキャッシュコンテナオブジェクトに関する実行時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します。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、キャッシュ、または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キャッシュコンテナのオプションをオーバーライドします。キャッシュコンテナ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のすべてのキャッシュが設定を継承します。1つのキャッシュコンテナのみで統計が有効になっていると、有用な統計を使用できます。

[Report a bug](#)

15.6. 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。

[Report a bug](#)

15.6.1. 起動設定ファイルでの Infinispan 統計収集の有効化

手順15.3 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15.2 cache の統計収集の有効化

```
<replicated-cache name="sso" mode="SYNC" batching="true" statistics-enabled="true"/>
```

例15.3 cache-container の統計収集の有効化

```
<cache-container name="singleton" aliases="cluster ha-partition" default-cache="default" statistics-enabled="true">
```

[Report a bug](#)

15.6.2. 管理 CLI での Infinispan 統計収集の有効化

手順15.4 管理 CLI での Infinispan 統計収集の有効化

この手順では、以下を前提とします。

- 推奨の **cache-container** は **CACHE_CONTAINER** です (例: **web**)。
- 推奨のキャッシュタイプは **CACHE_TYPE** です (例: **distributed-cache**)。
- CACHE** はキャッシュ名です (例: **dist**)。

1. 以下のコマンドを実行します。

```
/subsystem=infinispan/cache-  
container=CACHE_CONTAINER/CACHE_TYPE=CACHE: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)
```

[Report a bug](#)

15.6.3. Infinispan 統計収集の有効化を検証

手順15.5 Infinispan 統計収集の有効化を検証

以下のコマンドの1つを使用して統計収集が有効になっていることを確認します。**cache** または **cache-container** のどちらで有効であるかによって使用するコマンドが異なります。

- **cache** の場合

```
/subsystem=infinispan/cache-
container=CACHE_CONTAINER/CACHE_TYPE=CACHE:read-attribute(name=statistics-
enabled)
```

- **cache-container** の場合

```
/subsystem=infinispan/cache-container=CACHE_CONTAINER:read-
attribute(name=statistics-enabled)
```

[Report a bug](#)

15.7. JGROUPS

15.7.1. 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、クラスターのサーバーがお互いに通信しあう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、ユーザーデータグラムプロトコル) 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 TPC よりも高速ですが、信頼性は劣ります。
- **tcp** - クラスターのノードは TCP (Transmission Control Protocol、伝送制御プロトコル) を使用してお互いに通信します。通常、TCP は UDP よりも低速ですが、宛先にデータを配信する信頼性は勝ります。

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。

[Report a bug](#)

第16章 JVM

16.1. JVM

16.1.1. JVM 設定

Java Virtual Machine (JVM) の設定は、管理対象ドメインインスタンスとスタンドアロンサーバーインスタンスでは異なります。管理対象ドメインでは、JVM 設定が **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り決定されます。スタンドアロンサーバーインスタンスでは、サーバー起動プロセスで起動時にコマンドライン設定を渡すことができます。これらは、管理コンソールのコマンドラインまたは **System Properties** 画面で宣言できます。

管理対象ドメイン

管理対象ドメインの重要な機能は、JVM 設定を複数のレベルで定義できることです。サーバーグループまたはサーバーインスタンスによって、ホストレベルでカスタム JVM 設定を指定できます。特別な子要素は親設定よりも優先され、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、親設定は他のレベルで継承できます。

例16.1 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 512 メガバイトの最大ヒープサイズを宣言しています。このグループに属するすべてのサーバーは、これらの設定を継承します。これらの設定は、グループ全体、ホスト、または個別サーバーで変更できます。

例16.2 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、サーバーグループに対する 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>

```

このインスタンスでは、**server-two** という名前のサーバーが、**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。前の例では、**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した。これよりも小さい 256 メガバイトを最大ヒープサイズとして宣言すると、**server-two** が **domain.xml** 設定よりも優先され、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 設定を実行時に宣言するには、サーバーを起動する前に **JAVA_OPTS** 環境変数を設定します。以下は Linux のコマンドラインで **JAVA_OPTS** 環境変数を設定する一例です。

```
[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"
```

JVM に渡すオプションの例が格納されている **EAP_HOME/bin** フォルダにある **standalone.conf** ファイルに JVM 設定を追加することも可能です。



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 の起動が阻止される可能性があります。

[Report a bug](#)

16.1.2. 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に対し、Java 仮想マシン (JVM) の状態を管理コンソールに表示することができます。コンソールにはヒープ使用量、非ヒープ使用量、およびサーバーのスレッド使用量が表示されます。統計はリアルタイムで表示されませんが、コンソールの表示を更新すると最新の JVM リソースの概要が表示されます。

JVM の状態には次の値が表示されます。

表16.1 JVM 状態属性

タイプ	説明
Max	メモリー管理に使用できる最大メモリー容量。使用可能な最大メモリー容量は薄い灰色のバーで表示されます。
Used	使用中のメモリー容量。使用中のメモリー容量は濃い灰色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たメモリー容量。確保されたメモリーは濃い灰色で表示されます。
Init	メモリー管理のために Java 仮想マシンが最初にオペレーティングシステムからリクエストするメモリー容量。これは濃い灰色のバーで表示されます。

手順16.1 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**
画面の上部から **Runtime** タブを選択します。 **Status** メニューを展開したら **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**
画面の上部から **Runtime** タブを選択します。 **Server Status** メニューを展開したら **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した1つのサーバーのみを一度に表示できます。サーバーグループの他のサーバーの状態を表示するには、画面の左にある **Change Server** をクリックしてグループに表示されるホストと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すると、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。

[Report a bug](#)

16.1.3. 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16.3 <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"
    }
  }
}
}}
}}}
```

[Report a bug](#)

第17章 WEB サブシステム

17.1. WEB サブシステムの設定

Web ベース管理コンソールまたはコマンドライン管理 CLI を使用すると、Web サブシステムのほとんどの側面を設定できます。各設定は、管理コンソールに表示される順番で説明され、管理 CLI コマンドも提供されます。

管理コンソールを使用した Web サブシステムの表示

Web ベース管理コンソールを使用して Web サブシステムを設定するには、画面上部の **Configuration** タブをクリックします。**Subsystems** メニューを展開した後、**Web** メニューを展開します。Web サブシステムの設定可能な各部分が表示されます。



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** である場合、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** 設定については「[mod_cluster サブシステムの設定](#)」を参照してください。

JSP コンテナ、HTTP コネクタ、および仮想 HTTP サーバーの設定

JSP コンテナ、HTTP コネクタ、および仮想 HTTP サーバーを設定するには、**Servlet/HTTP** メニューエントリをクリックします。**Edit** ボタンをクリックして任意の値を変更します。**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。これらのオプションについては以下で説明されています。HTTP コネクタおよび仮想サーバーのオプションは、別の表で示されます。

表17.1 サブレット/HTTP 設定オプション

オプション	説明	CLI コマンド
Instance ID	負荷分散が使用される場合にセッションアフィニティを有効にするため使用される ID。この ID はクラスターのすべての 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 ID の最後に追加されます。これにより、フロントエンドプロキシによって特定のセッションが同じ JBoss EAP インスタンスへ転送されるようになります。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。	<code>/profile=full-ha/subsystem=web:write-attribute(name=instance-id,value=worker1)</code>
Disabled?	true の場合は、Java ServerPages (JSP) コンテナを無効にします。デフォルトで false に設定されます。これは、JSP を使用しない場合に役に立ちます。	<code>/profile=full-ha/subsystem=web/configuration=jsp-configuration/:write-attribute(name=disabled,value=false)</code>

オプション	説明	CLI コマンド
Development?	true の場合は、Development Mode を有効にします。この場合、より冗長なデバッグ情報が生成されます。デフォルトで false に設定されます。	<pre>/profile=full- ha/subsystem=web/configur ation=jsp- configuration/:write- attribute(name=development ,value=false)</pre>
Keep Generated?	Advanced をクリックしてオプションを確認します (非表示である場合)。 true の場合は、生成されたサーブレットを保持します。デフォルト値は true です。	<pre>/profile=full- ha/subsystem=web/configur ation=jsp- configuration/:write- attribute(name=keep- generated,value=true)</pre>
Check Interval?	Advanced をクリックしてオプションを確認します (非表示である場合)。バックグラウンドプロセスを使用して JSP の更新をチェックする頻度を定める値 (秒単位)。デフォルトで 0 に設定されます。	<pre>/profile=full- ha/subsystem=web/configur ation=jsp- configuration/:write- attribute(name=check- interval,value=0)</pre>
Display Source?	Advanced をクリックしてこのオプションを確認します (非表示である場合)。 true の場合、ランタイムエラーが発生すると、JSP ソース断片が表示されます。デフォルトで true に設定されます。	<pre>/profile=full- ha/subsystem=web/configur ation=jsp- configuration/:write- attribute(name=display- source- fragment,value=true)</pre>

AJP および HTTP コネクタは負荷分散と HA クラスタリングに **mod_cluster**、**mod_jk**、**mod_proxy**、**ISAPI**、および **NSAPI** を使用します。コネクタを設定するには **Connectors** タブを選択し、**Add** をクリックします。コネクタを削除するには、エントリーを選択し **Remove** をクリックします。コネクタを編集するには、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17.1 新しいコネクタの作成

```
/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)
```

表17.2 コネクターオプション

オプション	説明	CLI コマンド
Name	表示目的のコネクターの一意的な名前。	<code>/profile=full- ha/subsystem=web/connecto r=ajp/:read- attribute(name=name)</code>
Socket Binding	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に対して設定されるか、管理対象ドメインのソケットバインディンググループを介して設定されます。ソケットバインディンググループはサーバーグループへ適用されます。	<code>/profile=full- ha/subsystem=web/connecto r=ajp/:write- attribute(name=socket- binding,value=ajp)</code>
Scheme	HTTP や HTTPS などの Web コネクタースキーム。	<code>/profile=full- ha/subsystem=web/connecto r=ajp/:write- attribute(name=scheme,valu e=http)</code>
Protocol	AJP や HTTP などの、使用する Web コネクタープロトコル。	<code>/profile=full- ha/subsystem=web/connecto r=ajp/:write- attribute(name=protocol,valu e=AJP/1.3)</code>
Enabled	Web コネクターが有効であるかどうか。	<code>/profile=full- ha/subsystem=web/connecto r=ajp/:write- attribute(name=enabled,valu e=true)</code>
Redirect Port	リダイレクトする場合に使用されるポート番号を指定するために使用されます。セキュア (https) または AJP コネクターが一般的です。	<code>/profile=full- ha/subsystem=web/connecto r=http:write- attribute(name=redirect- port,value=8443)</code>

オプション	説明	CLI コマンド
Redirect Binding	バインディングのリダイレクトは挙動の面ではポートのリダイレクトと似ていますが、「value」にポート番号ではなくソケットバインディング名を指定する必要があります。事前定義されたソケットバインディング (http、AJP など) を使用できるため、 redirect-binding を使用すると設定の柔軟性が高くなります。結果は redirect-port オプションと同じになります。	<pre>/profile=full- ha/subsystem=web/connecto r=http:write- attribute(name=redirect- binding,value=https)</pre>

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、エントリを選択し、**Edit** または **Remove** 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17.2 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)
```

表17.3 仮想サーバーオプション

オプション	説明	CLI コマンド
Name	表示目的の仮想サーバーの一意的な名前。	<pre>/profile=full- ha/subsystem=web/virtual- server=default-host/:read- attribute(name=name)</pre>
Alias	この仮想サーバーに一致する必要があるホスト名のリスト。管理コンソールで、1行あたり1つのホスト名を使用します。	<pre>/profile=full- ha/subsystem=web/virtual- server=default-host/:write- attribute(name=alias,value= ["localhost","example.com"])</pre>

オプション	説明	CLI コマンド
Default Module	Web アプリケーションをこの仮想サーバーのルートノードにデプロイする必要があるモジュールであり、ディレクトリーが HTTP 要求で提供されない場合に表示されます。	<pre>/profile=full- ha/subsystem=web/virtual- server=default-host/:write- attribute(name=default- web- module,value=ROOT.war)</pre>

[Report a bug](#)

17.2. 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、8080 番ポートでサーバーの URL を開くと表示される Welcome アプリケーションが含まれています。次の手順は、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ます。

手順17.1 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** を使用して次のコマンドを実行します。異なる管理対象ドメインプロファイルの変更が必要となる場合があります。スタンドアロンサーバーでは、コマンドの **/profile=default** の部分を削除する必要がある場合があります。

```
/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。

Web アプリケーションを設定してルートコンテキスト (/) を URL アドレスとして使用するには、**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>** ディレクティブを次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。

サーバーグループか最初に変更したサーバーにアプリケーションをデプロイします。アプリケーションは **http://SERVER_URL:PORT/** で使用できるようになります。

[Report a bug](#)

第18章 WEB サービスサブシステム

18.1. WEB サービスオプションの設定

Web サービスオプションを設定するには、**Web Services** メニュー項目をクリックします。オプションは、以下の表で説明されます。

表18.1 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。デフォルトで true に設定されます。	<pre>/profile=full- ha/subsystem=webservices/: write- attribute(name=modify- wsdl-address,value=true)</pre>
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 の場合は、 modify-wsdl-address が true に設定されない限り、上書きされません。<soap:address> の値が有効な URL の場合は、 wsdl-host の値と wsdl-port または wsdl-secure-port を使用して上書きされます。 wsdl-host が jbossws.undefined.host に設定されている場合は、<soap-address> が書き換えられたときに要求側のホストアドレスが使用されます。デフォルトで \${jboss.bind.address:127.0.0.1} に設定されます。JBoss EAP 6 が起動されたときにバインドアドレスが指定されてない場合は、 127.0.0.1 を使用します。	<pre>/profile=full- ha/subsystem=webservices/: write-attribute(name=wsdl- host,value=127.0.0.1)</pre>
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ないポート。これが 0 (デフォルト値) に設定された場合、ポートはインストール済みコネクタのリストを問い合わせることにより識別されます。	<pre>/profile=full- ha/subsystem=webservices/: write-attribute(name=wsdl- port,value=80)</pre>
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ポート。これが 0 (デフォルト値) に設定された場合、ポートはインストール済みコネクタのリストを問い合わせることにより識別されます。	<pre>/profile=full- ha/subsystem=webservices/: write-attribute(name=wsdl- secure-port,value=443)</pre>



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、以下のシステムプロパティを **standalone/domain.xml** ファイルに設定します。

```
<system-properties>
<property name="org.apache.cxf.logging.enabled" value="true"/>
</system-properties>
```

[Report a bug](#)

第19章 HTTP クラスターリングおよび負荷分散

19.1. はじめに

19.1.1. 高可用性および負荷分散クラスター

クラスターリングとは、サーバーなどの複数のリソースを単一のエンティティとして使用することです。クラスターリングの2つの主なタイプは**負荷分散 (LB)** と **高可用性 (HA)** です。LB クラスターでは、すべてのリソースが同時に実行され、管理レイヤーによってそれらのリソース全体で負荷が分散されます。

HA クラスターリングでは、1つのリソースが実行され、最初のリソースが利用できなくなった場合に別のリソースが利用可能になります。HA クラスターリングの目的は、ハードウェア、ソフトウェア、およびネットワークの停止による影響を減らすことです。

JBoss EAP 6 は、クラスター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 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ーリングが使用できるようになります。**ha** 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っています。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、**distributable** として設定されたアプリケーションがサーバー上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 でのキャッシュのクラスターリングおよびレプリケーションを管理します。

[Report a bug](#)

19.1.2. 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) を高可用性として設定できます。つまり、1つのインスタンスまたはメンバーが停止したり、クラスターから消去された場合、そのワークロードはピアに移行されます。負荷分散機能を提供するようワークロードを管理することもできるため、多くの優れたリソースを持つサーバーやサーバーグループは、より多くのワークロードを担当できます。また、負荷が高い間、処理能力を追加することも可能です。

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 サーバー自体も HA 用にクラスター化できます。**mod_cluster** コネクタが最も柔軟で、JBoss EAP 6 のコンテナと密接に統合されます。他にも、Apache の **mod_jk** または **mod_proxy** コネクタ、ISAPI および NSAPI コネクタなどがあります。

アプリケーション

Java Enterprise Edition 6 (Java EE 6) 仕様によって、デプロイされたアプリケーションを高可用化できます。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、作業に参与するノードがなくなった場合に他のノードによる引き継ぎが可能です。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。

[Report a bug](#)

19.1.3. HTTP コネクタの概要

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 HTTP コネクタを使用して外部 Web サーバーを通信します。これらの HTTP コネクタは 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 Enterprise Application Platform のワーカー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ています。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、JBoss Enterprise Application Platform の複数のサーバーノード全体でワークロードのバランスを取るよう設定されたり、障害時にワークロードを他のサーバーに移動するよう設定されます。これらの機能は *負荷分散* および *高可用性 (HA)* と呼ばれます。

JBoss EAP 6 は、複数の HTTP コネクタをサポートします。選択するコネクタは、使用している Web サーバーと必要な機能によって異なります。

以下の表は、JBoss EAP 6 と互換性がある HTTP コネクタの違いを表しています。HTTP コネクタのサポート対象設定の最新情報については、<https://access.redhat.com/site/articles/111663> を参照してください。

表19.1 HTTP コネクタ機能および制約

コネクタ	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	ステッキセッションのサポート

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	ステッキ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	可。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。	○
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。	○
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。	○
ISAPI	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。	○

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	ステッキセッションのサポート
NSAPI	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。	○

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 \(ISAPI\) HTTP コネクター](#)」
- 「[Netscape Server API \(NSAPI\) HTTP コネクター](#)」

JBoss EAP 6 によってサポートされる設定は <https://access.redhat.com/site/articles/111663> を参照してください。

[Report a bug](#)

19.1.4. ワーカーノード

HTTP コネクターノード

ワーカーノードは、1つまたは複数のクライアント向け Web サーバーから要求を許可する JBoss EAP 6 サーバーで、単にノードと呼ばれることもあります。JBoss EAP 6 は Apache HTTP Server、Microsoft IIS、Oracle iPlanet Web Server などの独自の Web サーバーからの要求を許可できます。

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については、「[xref linkend="Overview_of_HTTP_Connectors" />](#)」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サーバーのクラスターのメンバーです。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、中央マネージャーが状況固有の均等性の単位を使用してノード間で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処理を行い、他のノードはアクティブなノードのいずれかがクラスターから離れるまで待機します。

[Report a bug](#)

19.2. コネクター設定

19.2.1. JBoss EAP 6 にて HTTP コネクタのスレッドプールを定義

概要

エグゼキューターモデルを使用すると JBoss EAP 6 のスレッドプールを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) コネクタで共有できるだけでなく、エグゼキューター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。HTTP コネクタのスレッドプールを現在の Web パフォーマンスの要件に合わせることは容易ではなく、現在のスレッドプール、現在および予想される Web ロードの要求を綿密に監視する必要があります。このタスクでは、エグゼキューターモデルを使用して HTTP コネクタのスレッドプールを設定する方法について取り上げます。コマンドラインインターフェースを使用して設定する方法と、XML 設定ファイルを編集して設定する方法の両方を説明します。

手順19.1 HTTP コネクタ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編集する場合は **standalone.xml**、ドメインベースの設定に対して編集する場合は **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>
```

CLI を使用したい場合は、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. エグゼキューターの作成

6 つある組み込みエグ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グゼキューターとして動作させることができます。6 つのエグ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。最大数未満のスレッドが実行されている場合、新しいスレッドが開始され、送信されたタスクを実行します。それ以外の場合は、タスクは非有界の FIFO キューに置かれ、スレッドが利用可能になると実行されます。



注記

Executors.singleThreadExecutor() によって提供された単一スレッドエグゼキュータータイプは、スレッドの数が 1 に制限される非有界キューのエグゼキューターです。このタイプのエグゼキューターは **unbounded-queue-thread-pool-executor** 要素を使用してデプロイされます。

- **bounded-queue-thread-pool**: このタイプのエグゼキューターは固定長のキューと、**core** および **maximum** の 2 つのプールサイズを維持します。タスクが許可されると、実行中のプールスレッドの数が **core** サイズ未満である場合は新しいスレッドが起動されタスクが実行されます。キューが空いている場合はタスクはキューに置かれます。実行中のプールスレッドの数が **maximum** サイズ未満である場合は新しいスレッドが起動されタスクが実行されます。エグゼキューターでブロッキングが有効になっている場合は、キューに空きが

できるまで呼び出しスレッドがブロックします。ハンドオフエグゼキューターが設定されている場合はタスクがハンドオフエグゼキューターに委譲されます。それ以外の場合はタスクが拒否されます。

- **blocking-bounded-queue-thread-pool**: スレッドの送信タスクがブロックする可能性がある、有界キューを持つスレッドプールエグゼキューター。このようなスレッドプールには `core` および `maximum` サイズ、指定されたキューの長さがあります。タスクの送信時、実行中のスレッド数が `core` サイズ未満である場合は新しいスレッドが作成されます。キューが空いている場合はタスクはキューに置かれます。実行中のスレッド数が `maximum` サイズ未満である場合は新しいスレッドが作成されます。これ以外の場合は、キューが空くまで呼び出し側がブロックします。
- **queueless-thread-pool**: 場合によっては、タスクの完了に關与するキューがなく、スレッドを再使用し、個別のスレッドでタスクを実行するために簡単なスレッドプールが必要ながあります。タスクを許可した後に他の実行タスクが完了するまで実行を遅延するのではなく、タスクは常に許可直後に開始されるため、このプールタイプは長期実行タスクや I/O のブロックを使用するタスクを処理するのに適しています。このタイプのエグゼキューターは **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するキューを持たないスレッドプールエグゼキューター。タスクが送信されると、実行中のスレッド数が `maximum` サイズ未満の場合は新しいスレッドが作成されます。その他の場合は、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呼び出し側がブロックします。
- **scheduled-thread-pool**: `java.util.concurrent.ScheduledThreadPoolExecutor` クラスを基に特定の時間および時間間隔でタスクを実行する目的の特別なタイプのエグゼキューターです。このタイプのエグゼキューターは **scheduled-thread-pool-executor** 要素を用いて設定されます。

この例では、**unbounded-queue-thread-pool** を使用してエグゼキューターとして動作させます。サーバーの環境に合わせて **max-threads** および **keepalive-time** パラメーターの値を編集します。

```
<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>
```

CLI を使用する場合は、以下を実行します。

```
[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" />
```

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

変更を有効にするため、サーバー（スタンドアロンまたはドメイン）を再起動します。次の CLI コマンドを使用して、以前の手順で行った変更が有効になったことを確認します。

```
[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 コネクタが編集されます。

[Report a bug](#)

19.3. WEB サーバーの設定

19.3.1. スタンドアロン Apache HTTP Server

JBoss EAP 6 は 認定バージョンの Red Hat Enterprise Linux 6 に含まれている Apache HTTP サーバーでテストおよびサポートされています。Apache HTTP Server は、Microsoft Windows Server などの他のオペレーティングシステムでも使用できます。しかし、Apache HTTP Server は Apache Foundation によって作成された個別の製品であるため、使用するバージョンの Apache HTTP Server と JBoss EAP の互換性を確認するのが困難でした。

スタンドアロン Apache HTTP Server バンドルを JBoss EAP 6 の個別ダウンロードとして使用できるようになりました。これにより、Red Hat Enterprise Linux 以外の環境を使用する場合や、Apache HTTP Server が設定済みのシステムで Web アプリケーションに別のインスタンスを使用したい場合に、インストールと設定が簡素化されます。この Apache HTTP Server は個別ダウンロードとしてカスタマーサービスポータルよりダウンロードできます。カスタマーポータルでは、ご使用のインストールプラットフォームの利用可能な JBoss EAP 6 ダウンロードの下にリストされています。

[Report a bug](#)

19.3.2. JBoss EAP 6 に含まれる 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)
 - Red Hat Enterprise Linux 7 の場合は **apr-util-ldap** (LDAP 認証機能)
- Apache Portability Runtime (APR) がインストールされている必要があります。Red Hat Enterprise Linux では、**apr-util-devel** をインストールします。

Apache HTTP Server Zip アーカイブには、複数の Kerberos モジュールに対するシンボリックリンクが含まれます。このため、**mod_auth_kerb** パッケージが前提条件になります。Kerberos 機能が必要でない場合は、**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**)。

Microsoft Windows Server 環境での Apache HTTP Server のインストールに関しては『JBoss Enterprise Web Server 2 Installation Guide』の「Installing Enterprise Web Server on Windows」に記載されている「Configuring the Environment」を参照してください。

手順19.2 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 <https://access.redhat.com> にログインします。ダウンロードをクリックした後、**Product Downloads** のリストより **Red Hat JBoss Enterprise Application Platform** を選択します。**Version** ドロップダウンメニューから適切な JBoss EAP のバージョンを選択します。

2. 一覧より **httpd バイナリー** を選択します。

ご使用のオペレーティングシステムの **Apache HTTP Server** オプションを探します。**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が含まれる 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。

希望のサーバーの任意の場所に Zip ファイルを一時的に展開します。Zip ファイルの `jboss-ews-version-number` フォルダー下に **httpd** ディレクトリーが含まれています。**httpd** フォルダーをコピーし、JBoss EAP 6 をインストールしたディレクトリー (通常、`EAP_HOME` と呼ばれます) の中に置きます。

これで、Apache HTTP Server の場所が **`EAP_HOME/httpd/`** ディレクトリーになります。他の JBoss EAP 6 ドキュメントに記載されているとおり、この場所を `HTTPD_HOME` として使用できるようになります。

4. **ポストインストールスクリプト** を実行し、**Apache ユーザーおよびグループのアカウント** を作成します。

ターミナルエミュレーター上で root ユーザーアカウントになり、**`EAP_HOME/httpd`** ディレクトリーへ移動し、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、**apache** というユーザーがシステム上に存在するか確認します。

```
id apache
```

ユーザー **apache** が存在しない場合は、そのユーザーと適切なユーザーグループを追加する必要があります。これには、以下のコマンドを実行します。

```
/usr/sbin/groupadd -g 91 -r apache 2> /dev/null || :  
/usr/sbin/useradd -c "Apache" -u 48 -g 91 -s /sbin/nologin -r apache 2>  
/dev/null || :
```

コマンドの完了後、**apache** ユーザーが **httpd** サービスを実行する場合、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実行されたことを確認するには、**apache** ユーザーが Apache HTTP Server のインストールパスへの実行権限を持っているか確認します。

```
ls -l
```

出力は以下のようになるはずです。

```
drwxrwxr-- 11 apache apache 4096 Feb 14 06:52 httpd
```

5. Apache HTTP Server を設定します。

以下のコマンドを使用して、新しいユーザーアカウントに切り替わります。

```
sudo su apache
```

apache ユーザーとして、組織の必要性に合わせて Apache HTTP Server を設定します。一般的な手順は、Apache Foundation <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。

```
EAP_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。

```
EAP_HOME/httpd/sbin/apachectl stop
```

[Report a bug](#)

19.3.3. 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

- root 権限。
- 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る必要があります (`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) からのインストール。JBoss EAP は独自のバージョンの Apache HTTP Server 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** チャンネルの有効なサブスクリプションが必要です。

手順19.3 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 に固有するバージョンの **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1つのみを実行して **httpd** パッケージをシステムに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グラフィカルツールから、**httpd** サービスの起動時の挙動を定義できます。挙動を定義するには、以下のコマンドを実行します。

```
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、表示されたウィンドウでサービス設定を変更します。

```
system-config-services
```

3. httpd の起動

以下のコマンドを使用して **httpd** を起動します。

```
service httpd start
```

4. httpd の停止

以下のコマンドを使用して **httpd** を停止します。

```
service httpd stop
```

手順19.4 Red Hat Enterprise Linux 7 への Apache HTTP Server のインストールおよび設定 (RPM)

1. httpd22 のインストール

JBoss EAP に固有するバージョンの **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起動時に **httpd22** サービスを開始します。

```
systemctl enable httpd22.service
```

3. httpd22 の起動

以下のコマンドを使用して **httpd22** を起動します。

```
systemctl start httpd22.service
```

4. httpd22 の停止

以下のコマンドを使用して **httpd22** を停止します。

```
systemctl stop httpd22.service
```

[Report a bug](#)

19.3.4. httpd 上の mod_cluster 設定

概要

mod_cluster は httpd 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要求を httpd 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 httpd 上の mod_cluster に設定できます。



注記

mod_cluser は JBossWEB に転送しなければならない 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19.2 mod_cluster の派生

派生	説明	値
CreateBalancers	バランサーが httpd VirtualHosts でどのように作成されるかを定義します。 ProxyPass /balancer://mycluster1/ のような派生を許可します。	0: httpd で定義された VirtualHosts をすべて作成する 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 1を値として使用する場合、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 sticky session 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るためです。

派生	説明	値
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
WaitForRemove	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	<p>ProxyPassMatch および ProxyPass は、!(バックエンド url の代わりに) の使用時にパスのリバースプロキシを防ぐ mod_proxy のディレクティブです。イメージなどの静的な情報に httpd が対応できるようにするため使用されます。例は次のとおりです。</p> <p>ProxyPassMatch ^(/.*\.(gif))\$!</p> <p>上記の例は httpd が直接 .gif ファイルに対応できるようにします。</p>	

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ロジックと似ています。

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 および Load: 0 で表示されます。詳細については、JBoss Enterprise Application Platform 『開発ガイド』の「Apache mod_cluster-manager アプリケーション」の項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。**server config** の内容は、ディレクティブが VirtualHost 設定の外部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、httpd が開始しません。

表19.3 mod_manager の派生

派生	説明	値
EnableMCPMReceive	VirtualHost がノードより 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、httpd 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生成に mod_manager が使用するベース名。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。 内容: サーバー設定	\$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。 内容: サーバー設定	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。 内容: サーバー設定	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。 内容: サーバー設定	デフォルト: 20

派生	説明	値
Maxsessionid	<p>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 sessionid の数。5 分以内に mod_cluster がセッションより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。</p> <p>内容: サーバー設定</p> <p>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。</p>	0: 論理はアクティベートされない。
MaxMCMPPMaxMessSize	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
ManagerBalancerName	Boss AS、JBossWeb、または Tomcat がバランサー名を提供しない場合に使用するバランサーの名前。	mycluster
PersistSlots	<p>ファイルのノード、別名、およびコンテキストを保持するよう mod_slotmem に伝えます。</p> <p>内容: サーバー設定</p>	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。	<p>on/off</p> <p>デフォルト: on - Nonce をチェック</p>
AllowDisplay	mod_cluster-manager メインページの追加表示を切り替えます。	<p>on/off</p> <p>デフォルト: off - バージョンのみを表示</p>
AllowCmd	mod_cluster-manager の URL を使用するコマンドを許可します。	<p>on/off</p> <p>デフォルト: on - コマンドを許可</p>
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。	<p>on/off</p> <p>デフォルト: off - 情報をすべて表示</p>

派生	説明	値
SetHandler mod_cluster-manager	<p>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。</p> <pre> <Location /mod_cluster-manager> SetHandler mod_cluster-manager Order deny,allow Allow from 127.0.0.1 </Location> </pre>	<p>on/off</p> <p>デフォルト: off</p>

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。

[Report a bug](#)

19.3.5. 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と、JBoss EAP 6 でサポートされるさまざまな HTTP コネクタの利点と欠点については、「[HTTP コネクタの概要](#)」を参照してください。状況によっては、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。

- [「JBoss EAP 6 に含まれる 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 リダイレクターを使用するよう設定」](#)

- [「Oracle Solaris での NSAPI コネクタの設定」](#)
- [「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」](#)

[Report a bug](#)

19.3.6. 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、要求を受け入れるプロキシを認識する必要がなく、検索するポートとプロトコルのみを認識する必要があります。これは、**mod_cluster** の場合は該当しません。この場合は、JBoss EAP 6 の設定に密接に関係します。ただし、**mod_jk**、**mod_proxy**、**ISAPI**、および **NSAPI** には以下のタスクが有効です。例にあるプロトコルとポートを設定が必要なものに置き換えます。

mod_cluster に対して JBoss EAP 6 を設定する場合は [「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。タスクの実際の手順では、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19.5 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を設定します。

デフォルトでは、jvmRoute はサーバー名と同じ値に設定されています。カスタマイズする必要がある場合は、次のようなコマンドを使用できます。使用するプロファイルやスタンドアロンサーバーの使用の有無によって、コマンドの **/profile=ha** 部分を置き換えまたは削除します。文字列 **CUSTOM_ROUTE_NAME** はカスタム jvmRoute の名前に置き換えます。

```
[user@localhost:9999 /] /profile=ha/subsystem=web:write-attribute(name="instance-id",value="CUSTOM_ROUTE_NAME")
```

2. Web サブシステムで利用可能なコネクタをリストします。



注記

この手順は、管理対象ドメインのスタンドアロンサーバーまたは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た独自のコネクタが必要です。

現在利用可能なコネクタをリストするには、以下のコマンドを実行します。

```
[standalone@localhost:9999 /] /subsystem=web:read-children-names(child-type=connector)
```

必要なコネクタ (HTTP、HTTPS、AJP) を示す行がない場合は、コネクタを追加する必要があります。

3. コネクタの設定を確認します。

コネクタ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 コネクタの設定を読み取ります。他のコネクタの設定出力も同様になります。

```
[standalone@localhost:9999 /] /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. 必要なコネクタを Web サブシステムに追加します。

コネクタを 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。以下の手順では、サーバーグループが **server-group-one** であり、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。

```
[standalone@localhost:9999 /] /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。

```
[standalone@localhost:9999 /] /subsystem=web/connector=ajp:add(socket-binding=ajp,
protocol="AJP/1.3", enabled=true, scheme="http")
```

[Report a bug](#)

19.4. クラスターリング

19.4.1. クラスター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他のクラスターの状態を監視します

が、TCP のみが許可されているネットワークもあります。このような場合、設定に **TCPPING** プロトコルスタックを追加し、デフォルトのメカニズムとして使用できます。このような設定オプションはコマンドラインベースの管理 CLI で使用可能です。

mod_cluster サブシステムもデフォルトで UDP 通信を使用しますが、TCP の使用を選択できます。

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」](#)

[Report a bug](#)

19.4.2. 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。次の手順を用いて JGroups サブシステムがユニキャスト TCP を使用するよう設定します。

TCP も使用するよう **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異なる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** 部分を削除します。以下のように、コマンドの最下部にリストされたプロパティーを変更します。これらは任意のプロパティーです。

initial_hosts

既知と見なされたホストのカンマ区切りのリストであり、初期メンバーシップを検索するために使用できます。

port_range

必要な場合は、ポート範囲を割り当てることができます。2 のポート範囲を割り当て、初期ポートが 7600 である場合は、TCPING がポート 7600 - 7601 の各ホストと通信しようとします。これは任意のプロパティーです。

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

クラスターが完了したと見なされるまでのノード数。これは任意のプロパティーです。

```
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_SOCKET,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。

ターミナルエミュレーターで、**jboss-cli.sh** スクリプトが含まれるディレクトリーに移動し、以下のコマンドを実行します。

```
./jboss-cli.sh -c --file=SCRIPT_NAME
```

SCRIPT_NAME はスクリプトが含まれる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す。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。**mod_cluster** サブシステムが TCP も使用するよう設定するには、「[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。

[Report a bug](#)

19.4.3. 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が、アドバタイズを無効にすることも可能です。次の手順を用いてこの挙動を設定します。

手順19.6

1. httpd 設定を変更します。

サーバーアドバタイジングを無効にし、代わりにプロキシリストを使用するよう httpd 設定を変更します。プロキシリストはワーカーで設定され、ワーカーが対話できる **mod_cluster** が有効な Web サーバーがすべて含まれます。

Web サーバーの **mod_cluster** 設定は、通常は httpd インストール内の **/etc/httpd/** または **etc/httpd/** ディレクトリーにあります (標準以外の場所にインストールされた場合)。そのファイルの詳細については、[「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を含むファイルを開き (**EnableMCPMReceive** ディレクティブを使用)、次のように **ServerAdvertise** ディレクティブを変更してサーバーアドバタイジングを無効にします。

ServerAdvertise Off

2. JBoss EAP 6 の **mod_cluster** サブシステム内でアドバタイジングを無効にし、プロキシ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の管理 CLI を使用して、**mod_cluster** サブシステムのアドバタイジングを無効にし、プロキシのリストを提供できます。アドバタイジングが無効な場合は、**mod_cluster** サブシステムがプロキシを自動的に検出できないため、プロキシ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** が有効になっているプロファイル (**ha** プロファイルや **full-ha** プロファイルなど) のみ **mod_cluster** を設定できます。

1. 管理コンソールにログインし、画面上部にある **Configuration** ラベルを選択します。
管理対象ドメインを使用する場合は、左上の **Profiles** 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** を展開した後に **Web** メニューを展開し、**mod_cluster** を選択します。
3. **mod_cluster** の **Advertising** タブ下にある **Edit** をクリックします。アドバタイジングを無効にするには、**Advertise** の横にあるチェックボックスのチェックマークを削除し、**Save** をクリックします。

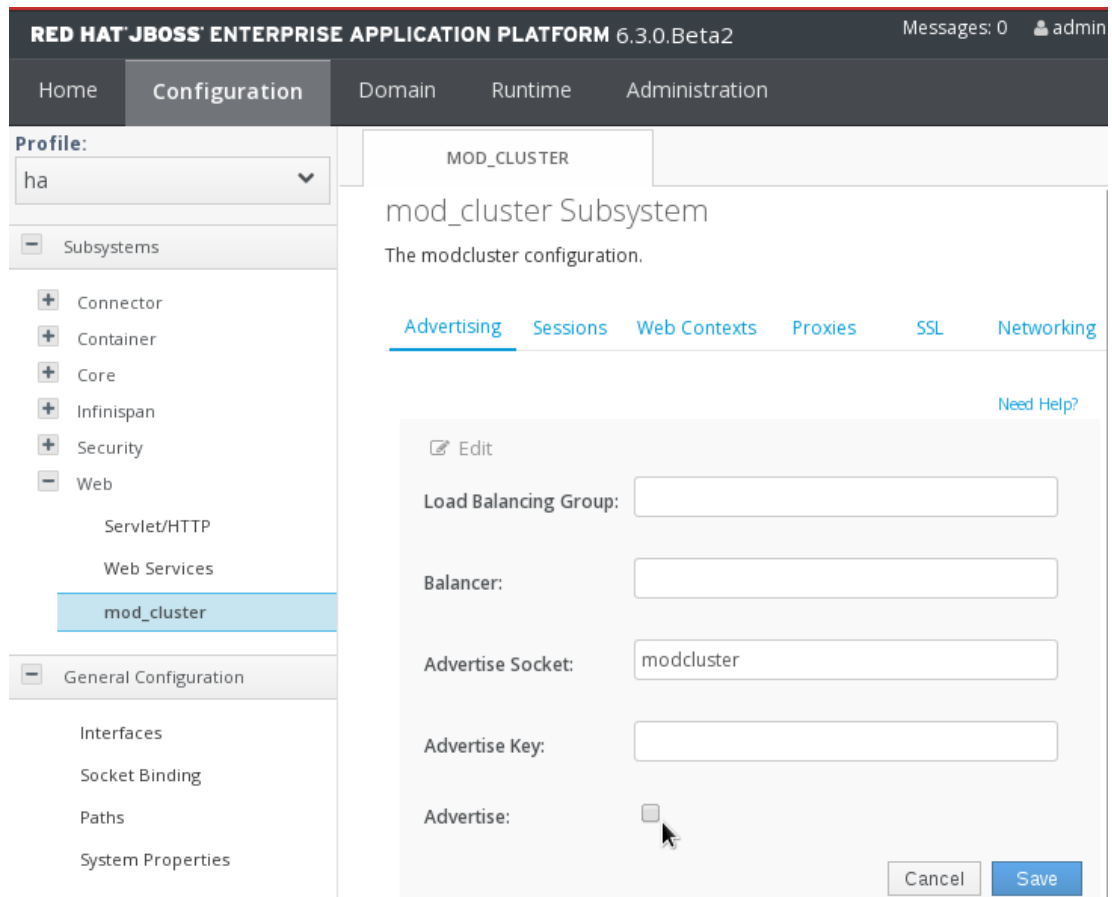


図19.1 mod_cluster アドバタイジング設定画面

4. **Proxies** 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サーバーのリストを入力します。正しい構文は、以下のような **HOSTNAME:PORT** 文字列のカンマ区切りリストになります。

```
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

次の2つの管理 CLI コマンドは、上記の管理コンソールの指示と同じ設定を作成します。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することが前提となります。異なる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** 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

httpd 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せず、UDP マルチキャストが使用されないようになります。

[Report a bug](#)

19.4.4. 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)。

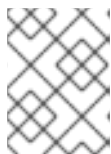
```
<socket-binding name="messaging-group" port="0" multicast-  
address="${jboss.messaging.group.address:231.7.7.7}" multicast-  
port="${jboss.messaging.group.port:9876}"/>
```

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

必要の場合は、host または port に対してシステムプロパティ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に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

自身への接続を持つノードを設定しないでください。誤設定と見なされます。

[Report a bug](#)

19.5. WEB、HTTP コネクター、および HTTP クラスターリング

19.5.1. `mod_cluster` HTTP コネクター

`mod_cluster` モジュールは、JBoss Web コンテナで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 \(ISAPI\) HTTP コネクター」](#)
- [「Netscape Server API \(NSAPI\) HTTP コネクター」](#)

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` が有効な Apache HTTP Server との間の追加的な接続です。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 サーバーへ再び送信するために 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` がある 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 サーバーに転送し、指定コンテキストのルーティング要求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SPORTを使用できます。

[Report a bug](#)

19.5.2. mod_cluster サブシステムの設定

管理コンソールでは、**mod_cluster** オプションを Web サブシステム設定エリアで使用できます。**Configuration** タブをクリックします。監理対象ドメインを使用する場合は、設定するプロファイルを上にある **Profile** 選択ボックスから選択します。デフォルトでは、**ha** および **full-ha** プロファイルで **mod_cluster** サブシステムが有効になっています。スタンドアロンサーバーを使用する場合は、**standalone-ha** または **standalone-full-ha** プロファイルを使用してサーバーを起動する必要があります。**Web** メニューを展開し、**mod_cluster** を選択します。オプションの説明は下表を参照してください。最初に設定全体が示され、次にセッション、Web コンテキスト、プロキシ、SSL、およびネットワークワーキングの設定が示されます。これらのタブは **mod_cluster** 設定画面内に表示されます。



注記

mod_cluster 設定ページは、**ha** プロファイルと **full-ha** プロファイルに対してのみ表示可能です。管理対象ドメインの場合、これらのプロファイルは **ha** と **full-ha** であり、スタンドアロンサーバーの場合は **standalone-ha** と **standalone-full-ha** です。

表19.4 mod_cluster 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)	これが 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。負荷分散グループを使用しない場合は、これを空白のままにしてください。これはデフォルトでは設定されません。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=load-balancing-group,value=myGroup)</pre>
バランサー (Balancer)	バランサーの名前。httpd プロキシの設定と一致する必要があります。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=balancer,value=myBalancer)</pre>
ソケットのアドバタイズ (Advertise Socket)	クラスタ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。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=advertise-socket,value=modcluster)</pre>

オプション	説明	CLI コマンド
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。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=advertise-security-key,value=myKey)</pre>
アドバタイズ (Advertise)	アドバタイズが有効かどうかを指定します。デフォルト値は true です。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=advertise,value=true)</pre>

表19.5 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)	要求にスティッキーセッションを使用するかどうかを指定します。つまり、クライアントが特定のクラスターノードに接続すると、クラスターノードが利用不可能でない限り、それ以降の通信は同じノードにルーティングされます。デフォルト値および推奨される設定値は true です。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=sticky-session,value=true)</pre>
スティッキーセッションの強制 (Sticky Session Force)	true を設定すると、最初のノードが利用不可能になった場合に要求は新しいクラスターノードにリダイレクトされず、失敗します。デフォルト値は false です。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=sticky-session-force,value=false)</pre>
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。デフォルト値は false です。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=sticky-session-remove,value=false)</pre>

表19.6 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
コンテキストの自動有効化 (Auto Enable Contexts)	デフォルトで mod_cluster に新しいコンテンツを追加するかどうかを指定します。このデフォルト値は true です。デフォルト値を変更し、コンテキストを手動で有効にする必要がある場合は、Web アプリケーションで enable() MBean メソッドまたは mod_cluster マネージャー (httpd の設定で指定された名前付き仮想ホストまたはポートの httpd プロキシで実行される Web アプリケーション) を使用してコンテキストを有効にできます。	<pre>/subsystem=modcluster/mod-cluster- config=configuration/:write- attribute(name=auto- enable-contexts,value=true)</pre>
除外されたコンテキスト (Excluded Contexts)	mod_cluster が無視する必要がある、コンテキストのカンマ区切りリスト。ホストが指定されない場合、ホストは localhost と見なされます。 ROOT は 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。	<pre>/subsystem=modcluster/mod-cluster- config=configuration/:write- attribute(name=excluded- contexts,value="ROOT,invok- er,jbossws,juddi,console")</pre>

表19.7 mod_cluster プロキシ設定オプション

オプション	説明	CLI コマンド
プロキシ URL (Proxy 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。	<pre>/subsystem=modcluster/mod-cluster- config=configuration/:write- attribute(name=proxy- url,value=myhost)</pre>
プロキシリスト (Proxy List)	httpd プロキシアドレスのカンマ区切りリスト (hostname:port 形式)。これは、 mod_cluster プロセスが最初に通信しようとするリストを示します。	<pre>/subsystem=modcluster/mod-cluster- config=configuration/:write- attribute(name=proxy- list,value="127.0.0.1,127.0.0.2")</pre>

mod_cluster に対する SSL 通信の設定

デフォルトでは、**mod_cluster** 通信は暗号化されていない HTTP リンクを介して行われます。コネクタースキームを **HTTPS** (表19.5 「**mod_cluster セッション設定オプション**」を参照) に設定する場合、以下の設定では、**mod_cluster** に、接続を暗号化する情報を見つける場所が通知されます。

表19.8 mod_cluster SSL 設定オプション

オプション	説明	CLI コマンド
ssl	SSL を有効にするかどうか。デフォルトは false です。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=ssl,value=true)</pre>
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。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=key-alias,value=jboss)</pre>
キーストア (Key Store)	クライアント証明書が含まれるキーストアの場所。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=key-store,value=System.getProperty("user.home") + "/.keystore")</pre>
キーストアタイプ (Key Store Type)	キーストアのタイプ	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=key-store-type,value=JKS)</pre>
キーストアプロバイダー (Key Store Provider)	キーストアプロバイダー。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/ssl=configuration/:write-attribute(name=key-store-provider,value=IBMJCE)</pre>

オプション	説明	CLI コマンド
パスワード	証明書が作成された時に選択されたパスワード。	<pre>/subsystem=modcluster/mod-cluster- config=configuration/ssl=configuration/:write- attribute(name=password,value=changeit)</pre>
トラストアルゴリズム (Trust Algorithm)	トラストマネージャーファクトリーのアルゴリズム。	<pre>/subsystem=modcluster/mod-cluster- config=configuration/ssl=configuration/:write- attribute(name=trust-algorithm,value=PKIX)</pre>
証明書ファイル (Cert File)	証明書ファイルの場所。	<pre>/subsystem=modcluster/mod-cluster- config=configuration/ssl=configuration/:write- attribute(name=ca-certificate-file,value=\${user.home}/jboss.crt)</pre>
CRL ファイル (CRL File)	証明書失効リスト (CRL) ファイル。	<pre>/subsystem=modcluster/mod-cluster- config=configuration/ssl=configuration/:write- attribute(name=ca-crl-file,value=\${user.home}/jboss.crl)</pre>
証明書の最大長 (Max Certificate Length)	トラストストアの保持される証明書の最大長。デフォルトは 5 です。	<pre>/subsystem=modcluster/mod-cluster- config=configuration/ssl=configuration/:write- attribute(name=trust-max-cert-length,value=5)</pre>

オプション	説明	CLI コマンド
キーファイル (Key File)	証明書のキーファイルの場所。	<pre>/subsystem=modcluster/mod-cluster- config=configuration/ssl=configuration/:write- attribute(name=certificate-key-file,value=\${user.home}/.key-store)</pre>
暗号スイート (Cipher Suite)	許可された暗号スイート。	<pre>/subsystem=modcluster/mod-cluster- config=configuration/ssl=configuration/:write- attribute(name=cipher-suite,value=ALL)</pre>
証明書エンコーディングアルゴリズム (Certificate Encoding Algorithms)	キーマネージャーファクトリーのアルゴリズム。	<pre>/subsystem=modcluster/mod-cluster- config=configuration/ssl=configuration/:write- attribute(name=encoding-algorithms,value=ALL)</pre>
失効 URL (Revocation URL)	証明局失効リストの URL。	<pre>/subsystem=modcluster/mod-cluster- config=configuration/ssl=configuration/:write- attribute(name=ca-revocation-url,value=jboss.crl)</pre>

オプション	説明	CLI コマンド
プロトコル	<p>有効 な SSL プロトコル。</p> <p>また、プロトコルの組み合わせをコンマで区切って指定することもできます (例: TLSv1, TLSv1.1, TLSv1.2)。</p> <div>  <p>警告</p> <p>Red Hat は、影響す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 を明示的に無効化することを推奨しています。</p> </div>	<pre>/subsystem=modcluster/mod-cluster- config=configuration/ssl=configuration/:write-attribute(name=protocol,value="TLSv1, TLSv1.1,TLSv1.2")</pre>

mod_cluster ネットワーキングオプションの設定

利用可能な **mod_cluster** ネットワーキングオプションは、**mod_cluster** サービスが通信する異なるタイプのサービスのさまざまなタイムアウト動作を制御します。

表19.9 mod_cluster ネットワーキング設定オプション

オプション	説明	CLI コマンド
ノードタイムアウト (Node Timeout)	<p>ノードに対するプロキシ接続のタイムアウト (秒単位)。これは、mod_cluster がバックエンド応答を待機する時間です。この時間が経過するとエラーが返されます。これはワーカー mod_proxy ドキュメンテーションのタイムアウトに対応します。値が -1 の場合は、タイムアウトがないことを示します。mod_cluster は要求を転送する前に cping/cpong を常に使用し、mod_cluster により使用される connectiontimeout 値は ping 値であることに注意してください。</p>	<pre>/subsystem=modcluster/mod-cluster- config=configuration/:write-attribute(name=node-timeout,value=-1)</pre>

オプション	説明	CLI コマンド
ソケットタイムアウト (Socket Timeout)	MCMP コマンドに対する httpd プロキシからの応答を待機する時間 (ミリ秒単位)。この時間が経過すると、タイムアウトになり、プロキシでエラーが発生していると示されます。	<code>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=socket-timeout,value=20)</code>
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)。	<code>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=stop-context-timeout,value=10)</code>
セッション排出ストラテジ (Session Draining Strategy)	<p>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。</p> <p>DEFAULT</p> <p>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。</p> <p>ALWAYS</p> <p>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。</p> <p>NEVER</p> <p>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)。</p>	<code>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=session-draining-strategy,value=DEFAULT)</code>
最大試行回数 (Max Attempts)	httpd プロキシが該当する要求を送信しようとする回数。最小値は、 1 であり、試行回数は1回だけです。 mod_proxy のデフォルト値も1であり、再試行は行われません。	<code>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=max-attempts,value=1)</code>

オプション	説明	CLI コマンド
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を指定します。デフォルト値は false です。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=flush-packets,value=false)</pre>
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時間 (秒単位)。デフォルト値は -1 です。値が -1 の場合は、パケットをフラッシュするまで永遠に待機します。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=flush-wait,value=-1)</pre>
Ping	ping に対するクラスターノードからの応答を待つ時間 (秒数単位)。デフォルト値は 10 秒です。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=ping,value=10)</pre>
SMAX	ソフトリミットのアイドル接続最大数 (ワーカー mod_proxy ドキュメントの smax と同じ)。httpd スレッド設定により最大値が異なり、 ThreadsPerChild または 1 になります。	<pre>profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=smax,value=ThreadsPerChild)</pre>
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。 nodeTimeout が定義されていない場合は、 ProxyTimeout ディレクティブの Proxy が使用されます。 ProxyTimeout が定義されていない場合は、サーバータイムアウト Timeout が使用されます。このデフォルト値は 300 秒です。 nodeTimeout 、 ProxyTimeout 、および Timeout はソケットレベルで設定されます。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=ttl,value=-1)</pre>

オプション	説明	CLI コマンド
ノードタイムアウト (Node Timeout)	外部 Web サーバーの利用可能なワーカプロセスが要求を処理するまで待機する時間 (秒数単位)。デフォルト値は -1 で、mod_cluster は httpd ワーカーが要求を処理するまで永遠に待機します。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=node-timeout,value=-1)</pre>

mod_cluster ロードプロバイダー設定オプション

以下の **mod_cluster** 設定オプションは管理コンソールで利用できません。管理 CLI を使用した場合のみ設定できます。

動的なロードプロバイダーが存在しない場合は、簡単なロードプロバイダーが使用されます。これは、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ワーク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。

```
/subsystem=modcluster/mod-cluster-config=configuration/simple-load-provider:add
```

次のリクエストを受信するクラスターノードを決定するためにさまざまなアルゴリズムの組み合わせを使用するよう動的ロードプロバイダーを設定できます。ロードプロバイダーを作成して環境に適した設定を行い、CLI を使用して複数の負荷メトリックを追加し、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 **busyness** を使用します。以下に、動的ロードプロバイダーのオプションや可能な負荷メトリックを示します。

表19.10 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。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/:write-attribute(name=decay,value=2)</pre>
履歴 (History)	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。	<pre>/subsystem=modcluster/mod-cluster-config=configuration/dynamic-load-provider=configuration/:write-attribute(name=history,value=9)</pre>

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 busyness で、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ノードの負荷を計算します。このメトリックの容量を設定できます (実際の負荷をこの容量で割ります: 計算された負荷/容量)。動的ロードプロバイダー内に複数の負荷メトリックを設定できます。	<pre>/subsystem=modcluster/mod-cluster- config=configuration/dynamic-load- provider=configuration/load-metric=busyness/:write-attribute(name=capacity,value=1.0)</pre> <pre>/subsystem=modcluster/mod-cluster- config=configuration/dynamic-load- provider=configuration/load-metric=busyness/:write-attribute(name=type,value=busyness)</pre> <pre>/subsystem=modcluster/mod-cluster- config=configuration/dynamic-load- provider=configuration/load-metric=busyness/:write-attribute(name=weight,value=1)</pre>

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クラスターノードを決定します。

mem

mem 負荷メトリックはネイティブメモリーの空き容量を負荷係数として使用します。このメトリックの使用は推奨されません。これは、バッファとキャッシュを含む値が提供され、メモリー管理が優れたシステムでは値が常に非常に小さくなるためです。

heap

heap 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クラスターを決定します。

sessions

session 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

requests 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取得するクラスターノードを決定します。たとえば、容量 1000 の場合、1000 要求数/秒はフルロードと見なされます。

send-traffic

send-traffic 負荷メトリックは、ワーカーノード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、デフォルト容量が 512 の場合は、平均送信トラフィックが 512 KB/秒以上のとき、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ノードに送信されるトラフィックの量を使用します。たとえば、デフォルト容量が 1024 の場合は、平均受信トラフィックが 1024 KB/秒以上のとき、ノード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19.1 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19.2 既存メトリックの値設定

既存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19.3 既存メトリックの値変更

既存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19.4 既存メトリックの削除

既存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)
```

19.5.3. 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 (Red Hat Enterprise Linux 6 Deployment Guide)』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、「JBoss EAP 6 に含まれる 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 (JBoss Enterprise Web Server Installation Guide)』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とアーキテクチャー用の **Webserver Connector 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。アーカイブの展開後に、モジュールは **EAP_HOME/modules/system/layers/base/native/lib/httpd/modules** ディレクトリーに配置されます。
- etc/** ディレクトリーには、いくつかのサンプル設定ファイルが含まれ、**share/** ディレクトリーには、いくつかの補足ドキュメントが含まれます。
- 管理 (root) 権限を使用してログインする必要があります。



注記

64 ビットのシステムを使用している場合、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。モジュールにアクセスするときは常にこのパスを使用する必要があります。

手順19.7 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 にダウンロード可能な個別コンポーネントとして同梱されているスタンドアロン Apache HTTP Server を使用しているか、または 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。設定場所は以下の 3 つのオプションの 1 つであり、本タスクでは **HTTPD_HOME** と表記します。

- 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 (拡張子が **.so** のファイル) 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_HOME/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 を使用する場合は、**mod_proxy_balancer** モジュールがデフォルトで有効になります。これには **mod_cluster** との互換性がありません。これを無効にするには、**HTTPD_HOME/conf/httpd.conf** を編集し、モジュールをロードする行の前に **#** (ハッシュ) 記号を置い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たり、コメントありで表示されたりします (以下参照)。

```
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、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 **mod_cluster.conf** ファイル (**EAP_HOME/modules/system/layers/base/native/etc/httpd/conf**) が含まれます。このファイルをコピーおよび編集し、**HTTPD_HOME/httpd/conf.d/JBoss_HTTP.conf** ファイルを作成できます。



注記

JBoss_HTTP.conf という名前を使用することは、本書では任意の慣例です。**conf.d/** ディレクトリーに保存され、**.conf** 拡張子を持つ設定ファイルは、名前に関係なくロードされます。

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ロードします。

5. プロキシサーバーリスナーを作成します。
HTTPD_HOME/httpd/conf.d/JBoss_HTTP.conf の編集を続行し、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 On

ManagerBalancerName mycluster
ServerAdvertise On

</VirtualHost>
```

これらのディレクティブにより、**IP_ADDRESS:PORT** でリッスンする新しい仮想サーバーが作成され、**MYDOMAIN.COM** からの接続が許可され、仮想サーバー自体が **mycluster** という名前のバランサーとしてアドバタイズされます。これらのディレクティブの詳細については、Apache Web Server 向けドキュメントに記載されています。**ServerAdvertise** および **EnableMCPMReceive** ディレクティブの詳細と、サーバーアドバタイジングの影響については、「[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の設定](#)」を参照してください。

ファイルを保存し、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 つのいずれかの方法を選択します。

- **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 が実行されている **mod_cluster** サブシステムと連携できます。JBoss EAP 6 が **mod_cluster** を認識するように設定する場合は「[mod_cluster ワーカーノードの設定](#)」を参照してください。

[Report a bug](#)

19.5.4. 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 (Red Hat Enterprise Linux 6 Deployment Guide)』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、『JBoss EAP 6 に含まれる 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 (JBoss Enterprise Web Server Installation Guide)』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 チャンネルへの有効なサブスクリプションを持っている必要があります。

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 2.2.15:

- 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** 行をコメントアウトして、デフォルトでロードされる **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
```

[Report a bug](#)

19.5.5. 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の設定

概要

Web サーバーを mod_cluster ロードバランサーと対話させる設定手順については、「[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。設定を行うにはサーバーアドバタイズメントの詳細を知る必要があります。

サーバーアドバタイズメントがアクティブな場合は、web サーバーが mod_cluster 仮想ホストで指定された 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可能でない場合、またはプロキシサーバーの静的リストでワーカーノードを設定する場合は、サーバー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できます。ワーカーノードの設定については、「[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する **httpd.conf** に加える必要があります。通常、このファイルは Red Hat Enterprise Linux の **/etc/httpd/conf/httpd.conf** にありますが、スタンドアロン Apache HTTP Server インスタンスの **etc/** ディレクトリーにあることもあります。

手順19.8 httpd.conf ファイルを編集し、変更を実装する

- 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) 記号を追加して、コメントアウトします。この値は **5** ではないことがあります。

```
AdvertiseFrequency 5
```

- 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> ステートメント内部に以下のディレクティブを追加してサーバーアドバタイズメントを無効にします。

```
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 On
```

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

```
C:\> net service http
C:\> net service httpd start
```

結果

Web サーバーが mod_cluster プロキシの IP アドレスとポートをアドバタイズしなくなります。繰り返すには、ワーカーノードが静的アドレスとポートを使用してプロキシと通信するよう設定する必要があります。詳細については、「[mod_cluster ワーカーノードの設定](#)」を参照してください。

[Report a bug](#)

19.5.6. 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 サーバーから構成されます。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。JBoss EAP 6 内では、クラスターのすべてのノードを管理する別のプロセスが実行されます。これはマスターと呼ばれます。ワーカーノードの概念の詳細については、「[ワーカーノード](#)」を参照してください。Web サーバーのロードバランシングの概要については、「[HTTP コネクターの概要](#)」を参照してください。

負荷分散 web サーバーは **mod_cluster** サブシステムより設定されます。**mod_cluster** サブシステムを設定する場合は『管理および設定ガイド』の「mod_cluster サブシステムの設定」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されているワーカーノードは個別に設定されますが、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** プロファイルと、**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 コマンドが提供された場合は、管理対象ドメインが使用されると見なされます。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** 部分を削除します。

手順19.9 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に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 は固定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. **EAP_HOME/bin/jboss-cli.sh** コマンド (Linux の場合) または **EAP_HOME\bin\jboss-cli.bat** コマンド (Microsoft Windows Server の場合) を使用して管理 CLI を起動します。**connect** と入力して localhost 上のドメインコントローラーに接続するか、**connect IP_ADDRESS** と入力してリモートサーバー上のドメインコントローラーに接続します。
- c. 以下のコマンドを入力し、**management**、**public**、および **unsecure** インターフェースの外部 IP アドレスを編集します。必ずコマンドの **EXTERNAL_IP_ADDRESS** をホストの実際の外部 IP アドレスと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するマスターでないホストの場合、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となる必要があり、クラスターに対する識別のために使用されます。そのため、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
```

例を以下に示します。

```
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)
```

例を以下に示します。

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

次の結果が表示されるはずです。

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

これは、以下のように **host-slave01.xml** ファイルの XML を変更します。

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

- e. 管理対象ドメインに参加する必要がある新たに設定されたホストの場合は、**local** 要素を削除し、ドメインコントローラーを示す **host** 属性の **remote** 要素を追加する必要があります。この手順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
```

例を以下に示します。

```
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")
```

例を以下に示します。

```
/host=host-slave01/:write-remote-domain-  
controller(host="192.168.1.200",port=${jboss.domain.master.port:9999},security-  
realm="ManagementRealm")
```

次の結果が表示されるはずです。

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

これは、以下のように **host-slave01.xml** ファイルの XML を変更します。

```
<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では、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。同じユーザー名を持つユーザーをスレーブとして **ManagementRealm** に追加します。このユーザーが外部 JBoss AS インスタンスに対して認証する必要があるかどうか尋ねられた場合は、**yes** と回答します。パスワード **changeme** を使用した、**slave1** という名前のスレーブに対するコマンドの入出力例は以下のとおりです。

```
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. add-user.sh の出力から Base64 でエンコードされた<secret> 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したい場合、**add-user.sh** の出力の最終行より **<secret>** 要素値をコピーします。次の手順でこの値が必要になります。

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. **EAP_HOME/bin/jboss-cli.sh** コマンド (Linux の場合) または **EAP_HOME/bin\jboss-cli.bat** コマンド (Microsoft Windows Server の場合) を使用して管理 CLI を起動します。**connect** と入力して localhost 上のドメインコントローラーに接続するか、**connect IP_ADDRESS** と入力してリモートサーバー上のドメインコントローラーに接続します。

- b. 以下のコマンドを入力して秘密の値を指定します。必ず **SECRET_VALUE** を前の手順の **add-user** 出力から返された秘密の値に置き換えてください。

```
/core-service=management/security-realm=ManagementRealm/server-identity=secret:add(value="SECRET_VALUE")
: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"
```

○ 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。次のような文字列が生成されます:
VAULT::secret::password::ODVmYmJjNGMtZDU2ZC00YmNILWE4ODMtZjQ1NWNmNDU4ZDc1TElORV9CUkVBS3ZhdWx0.

vault に関する詳細は、「機密性の高い文字列のパスワード vault」の項にある「[パスワード vault システム](#)」を参照してください。

- b. **EAP_HOME/bin/jboss-cli.sh** コマンド (Linux の場合) または **EAP_HOME\bin\jboss-cli.bat** コマンド (Microsoft Windows Server の場合) を使用して管理 CLI を起動します。**connect** と入力して localhost 上のドメインコントローラーに接続するか、**connect IP_ADDRESS** と入力してリモートサーバー上のドメインコントローラーに接続します。
- c. 以下のコマンドを入力して秘密の値を指定します。必ず **SECRET_VALUE** を前の手順で生成されたマスクされたパスワードに置き換えてください。

```
/core-service=management/security-realm=ManagementRealm/server-identity=secret:add(value="${VAULT::secret::password::SECRET_VALUE}")
: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"
```



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を指定します。
- i. **EAP_HOME/bin/jboss-cli.sh** コマンド (Linux の場合) または **EAP_HOME\bin\jboss-cli.bat** コマンド (Microsoft Windows Server の場合) を使用して管理 CLI を起動します。**connect** と入力して localhost 上のドメインコント

ローラーに接続するか、**connect IP_ADDRESS** と入力してリモートサーバー上のドメインコントローラーに接続します。

- ii. 以下のコマンドを入力し、システムプロパティを使用する秘密のアイデンティティーを設定します。

```
/core-service=management/security-realm=ManagementRealm/server-identity=secret:add(value="${server.identity.password}")
: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"
```

- b. システムプロパティ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、このパスワードを確認できます。

- パスワードをプロパティファイルに格納し、プロパティファイルの 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ファイルに追加します。例は次のとおりです。

```
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サーバーに複製され、外部の Web サーバーまたはロードバランサーから要求を受け入れることができます。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。自動検出と **mod_cluster** サブシステムの他の固有設定値を設定するには、「[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。

[Report a bug](#)

19.5.7. 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、アップグレード処理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したいことがあります。ここでは、停止時間やダウンタ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

- 新しいクラスター設定が必要です。「[mod_cluster サブシステムの設定](#)」を参照してください。この新しいクラスターは Cluster New と呼びます。
- 不要となった古いクラスターの設定 (このクラスターを Cluster OLD とします)。

手順19.10 クラスターのアップグレード処理

1. 前提条件に従って、新しいクラスターを設定します。
2. Cluster NEW および Cluster OLD の両方で、設定オプション **sticky-session** が **true** に設定されているようにしてください (デフォルト値は **true** です)。このオプションを有効にすると、どちらかのクラスターのクラスターノードへの新しい要求はすべてそのノードへ送信されます。

```
/profile=full-ha/subsystem=modcluster/mod-cluster-config=configuration/:write-attribute(name=sticky-session,value=true)
```

3. 「[mod_cluster ワーカーノードの設定](#)」のプロセスを使用して、Cluster NEW のノードを mod_cluster 設定に1つずつ追加します。
4. ロードバランサー (mod_cluster) を設定し、Cluster OLD の各コンテキストを停止します。Cluster OLD のコンテキストを停止 (無効ではなく) すると、各コンテキストが正常にシャットダウンされます (最終的にノード全体がシャットダウンされます)。既存のセッションはノードが対応しますが、新しいセッションはこれらのノードに転送されません。コンテキストの停止に数分から数時間かかることがあります。

次の CLI コマンドを使用してコンテキストを停止できます。パラメーターの値をご使用の環境に適した値に置き換えてください。

```
[standalone@localhost:9999 subsystem=modcluster] :stop-context(context=/myapp, virtualhost=default-host, waittime=50)
```

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。

[Report a bug](#)

19.6. APACHE MOD_JK

19.6.1. Apache mod_jk HTTP コネクター

Apache **mod_jk** は互換性を維持するために必要となるユーザー向けに提供される HTTP コネクターで

す。Apache **mod_jk** は JBoss Web Container に含まれる **jboss-eap-native-webserver-connectors** の一部で、ロードバランシングを提供します。サポートされるプラットフォームについては <https://access.redhat.com/site/articles/111663> を参照してください。**mod_jk** コネクタは Apache によって維持管理され、ドキュメントは <http://tomcat.apache.org/connectors-doc/> にあります。

JBoss EAP 6 は Apache HTTP プロキシサーバーからのワークロードを許可します。プロキシサーバーは Web フロントエンドからのクライアント要求を許可し、ワークを参加する JBoss EAP 6 サーバーへ渡します。スティッキーセッションが有効な場合、同じクライアント要求は常に同じ JBoss EAP 6 サーバーに送信されます (同じサーバーが使用できない場合を除く)。

JBoss **mod_cluster** HTTP コネクタとは異なり、Apache **mod_jk** HTTP コネクタはサーバーまたはサーバーグループ上のデプロイメントの状態を認識しないため、状況に応じてワークを送信する環境に対応できません。

mod_jk は **mod_cluster** と同様に AJP 1.3 プロトコル上で通信します。



注記

mod_cluster は **mod_jk** よりも高度なロードバランサーです。**mod_cluster** は **mod_jk** の全機能と追加機能を提供します。**mod_cluster** の詳細については、「[mod_cluster HTTP コネクタ](#)」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)」

[Report a bug](#)

19.6.2. JBoss EAP 6 が Apache Mod_jk と通信するよう設定

概要

mod_jk HTTP コネクタには、Web サーバーによってロードされる単一のコンポーネント **mod_jk.so** モジュールがあります。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 (この場合は JBoss EAP 6) に転送します。また、これらの要求を許可し、返信を 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 Server の設定は、「[Apache HTTP Server への mod_jk モジュールのインストール \(ZIP\)](#)」で説明しています。

JBoss EAP 6 が Apache HTTP Server と通信するには、AJP/1.3 コネクタが有効になっている必要があります。このコネクタがデフォルトで存在する設定は次のとおりです。

- 管理対象ドメインでは、**ha** および **full-ha** プロファイル、**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** サーバーグループはデフォルトのインストールで正しく設定されます。
- スタンドアロンサーバーでは、クラスター化された設定向けに **standalone-ha** および **standalone-full-ha** プロファイルが設定されます。どちらかのプロファイルを使用してスタンドアロンサーバーを起動するには、**EAP_HOME/**ディレクトリーより以下のコマンドを実行します。プロファイル名は適切な名前に置き換えてください。

```
[user@host bin]$ ./bin/standalone.sh --server-config=standalone-ha.xml
```

[Report a bug](#)

19.6.3. 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 にインストールされた Apache HTTP Server を使用する必要があります。JBoss Enterprise Web Server にインストールされた Apache HTTP 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。
- 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 (Red Hat Enterprise Linux Deployment Guide)』に記載された手順を実行します。
- 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 (JBoss Enterprise Web Server Installation Guide)』に記載された手順を実行します。
- Apache HTTP Server を使用する場合は、ご使用のプラットフォーム向けの JBoss EAP 6 ネイティブコンポーネントパッケージを Red Hat カスタマーサービスポータル (<https://access.redhat.com>) からダウンロードします。このパッケージには、Red Hat Enterprise Linux 向けにプリコンパイルされた **mod_jk** および **mod_cluster** バイナリーが含まれます。JBoss Enterprise Web Server を使用している場合は、すでに **mod_jk** のバイナリーは含まれています。
- 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。

手順19.11 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** という名前の新しいファイルを作成し、以下の内容をそのファイルに追加します。



注記

JkMount ディレクティブは、Apache 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 は受け取った URL を正しいサーブレットコンテナに転送します。

静的なコンテンツを直接提供し、Java アプリケーションにのみロードバランサーを使用するには、URL パスが **/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>
```

値を見て、セットアップに適切であることを確認します。適切な場合は、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 ディレクティブに加えて、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. 以下の内容を **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/uriworkermmap.properties
```

- ii. 照合される各 URL パターンに対する行を含む

HTTPD_HOME/conf/uriworkermmap.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、ファイルの構文の例を示しています。

■

```
# 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 Aache HTTP Server の **modules/** ディレクトリーに **mod_jk.so** がない場合のみ必要となります。JBoss EAP 6 のダウンロードとして含まれている Apache HTTP Server 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 ビットの場合は **EAP_HOME/modules/system/layers/base/native/lib/httpd/modules/** ディレクトリー、64 ビットの場合は

EAP_HOME/modules/system/layers/base/native/lib64/httpd/modules/ ディレクトリーにある **mod_jk.so** ファイルを見つけます。

このファイルを **HTTPD_HOME/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_HOME/conf/workers.properties** という名前の新しいファイルを作成します。以下の例を土台として使用し、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 ファイルの構文と高度な設定オプションの詳細については、[「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 サーバーの再起動の方法は、Red Hat Enterprise Linux の Apache HTTP Server を使用しているか、または 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1つを選択します。

- **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 サーバーを再起動する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 をインストールした場合は、**apachectl** コマンドを使用して Web サーバーを再起動します。*EWS_HOME* は、JBoss Enterprise Web Server Apache HTTP Server を展開したディレクトリーに置き換えてください。

```
[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。

```
C:\> net stop Apache2.2  
C:\> net start Apache2.2
```

- **Solaris**

コマンドプロンプトにて、以下のコマンドを管理者権限で実行します。*EWS_HOME* は、JBoss Enterprise Web Server Apache HTTP Server を展開したディレクトリーに置き換えてください。

```
[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す。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。

[Report a bug](#)

19.6.4. 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、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 にインストールされた Apache HTTP Server を使用する必要があります。JBoss Enterprise Web Server にインストールされた Apache HTTP 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。
- Apache HTTP Server をインストールする必要がある場合は、<https://access.redhat.com/site/documentation/> で入手可能な『Red Hat Enterprise Linux デプロイメントガイド (Red Hat Enterprise Linux Deployment Guide)』に記載された手順を実行します。
- JBoss Enterprise Web Server をインストールする必要がある場合は、<https://access.redhat.com/site/documentation/> で入手可能な『JBoss Enterprise Web Server インストールガイド (JBoss Enterprise Web Server Installation Guide)』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19.12 Red Hat Enterprise Linux 5: mod_jk と Apache HTTP Server 2.2.3

1. mod_jk-ap22 1.2.37 と、依存関係の mod_perl を **jbappplatform-6-*-server-5-rpm** チャンネルよりインストールします。

```
yum install mod_jk
```

2. **任意設定:** 使用するサンプル設定ファイルをコピーします。

```
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 モジュールがロードされた後に mod_jk モジュールがロードされるようにするため、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19.13 Red Hat Enterprise Linux 5: mod_jk と JBoss EAP Apache HTTP Server 2.2.26

1. 以下のコマンドを実行して、**jbappplatform-6*-server-5-rpm** チャンネルより 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. **任意設定:** 使用するサンプル設定ファイルをコピーします。

```
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19.14 Red Hat Enterprise Linux 6: mod_jk と JBoss EAP Apache HTTP Server 2.2.26

1. mod_jk-ap22 1.2.37 および Apache HTTP Server 2.2.26 httpd パッケージを、**jbappplatform-6*-server-6-rpm** チャンネルより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. **任意設定:** 使用するサンプル設定ファイルをコピーします。

```
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19.15 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. **任意設定:** 使用するサンプル設定ファイルをコピーします。

```
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19.16 Red Hat Enterprise Linux 7: mod_jk と JBoss EAP Apache HTTP Server 2.2.26

1. mod_jk-ap22 1.2.37 および Apache HTTP Server 2.2.26 httpd22 パッケージを、**jbappplatform-6*-server-6-rpm** チャンネルより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. **任意設定:** 使用するサンプル設定ファイルをコピーします。

```
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
```

[Report a bug](#)

19.6.5. 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ノードの動作を定義しま

す。Red Hat Enterprise Linux では、このファイルは `/etc/httpd/conf/workers.properties` にあります。**workers.properties** ファイルは、異なるサブレットコンテナが存在する場所と、負荷をコンテナ全体で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。最初のセクションは、すべてのワーカーノードに適用されるグローバルプロパティを扱います。2番目のセクションには、特定のワーカーに適用される設定が含まれます。3番目のセクションには、ワーカーで調整される特定のノードに設定が含まれます。

プロパティの一般的な構造は **worker.WORKER_NAME.DIRECTIVE** です。`WORKER_NAME` はワーカーの一意的な名前、`DIRECTIVE` はワーカーに適用される設定になります。

Apache mod_jk ワーカーの設定リファレンス

ノードテンプレートは、デフォルトの各ノードの設定を指定します。ノード設定内のテンプレート自体を上書きできます。ノードテンプレートの例は、[例19.5「workers.properties ファイルの例」](#)で参照できます。

表19.11 グローバルプロパティ

プロパティ	説明
worker.list	mod_jk で使用されるワーカー名のリスト。これらのワーカーは要求を受信できます。

表19.12 各ワーカーのプロパティ

プロパティ	説明
type	ワーカーのタイプ。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、 http://tomcat.apache.org/connectors-doc/ajp/ajpv13a.html の Apache Tomcat Connector AJP Protocol Reference を参照してください。
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対してディレクティブを複数回使用できます。ディレクティブは、カンマで区切られたワーカー名のリストから構成されます。これはワーカーごとに設定され、ノードごとには設定されません。このワーカータイプにより調整されるすべてのノードが影響を受けます。
sticky_session	同じセッションからの要求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。デフォルト値は 0 であり、スティッキーセッションが無効になります。スティッキーセッションを有効にするには、 1 に設定します。すべての要求が実際にステートレスでない限り、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、ワーカーごとに設定され、ノードごとに設定されません。そのワーカータイプにより調整されるすべてのノードが影響を受けます。

表19.13 各ノードのプロパティ

プロパティ	説明
host	ワーカーのホスト名または IP アドレス。ワーカーノードは 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。

プロパティ	説明
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、AJP13 ワーカーのデフォルトリッスンポートである 8009 です。AJP14 ワーカーのデフォルト値は 8011 です。
ping_mode	<p>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 は、C、P、I、および A の任意の組み合わせです。</p> <ul style="list-style-type: none"> ● C - Connect (接続)。サーバーへの接続後に 1 回だけ接続をプローブします。connect_timeout 値を使用してタイムアウトを指定します。指定しないと、値 ping_timeout が使用されます。 ● P - Prepost (プレポスト)。各要求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。指定しないと、値 ping_timeout が使用されます。 ● I - Interval (間隔)。connection_ping_interval (存在する場合) で指定された間隔で接続をプローブします。指定しないと、値 ping_timeout 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	各ワーカ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にのみ適用します。これは、より強力なサーバーにより多くの負荷を割り当てる場合に役に立ちます。ワーカーにデフォルトの 3 倍の負荷を割り当てるには、これを 3 に設定します (worker.my_worker.lbfactor=3)。

例19.5 workers.properties ファイルの例

```

worker.list=node1, node2, node3

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=C
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
```

Apache mod_jk の設定の詳細については、本書の範囲外です。詳細については、Apache ドキュメンテーション (<http://tomcat.apache.org/connectors-doc/>) を参照してください。

[Report a bug](#)

19.7. APACHE MOD_PROXY

19.7.1. Apache mod_proxy HTTP コネクター

Apache は、httpd に **mod_proxy** と **mod_jk** の 2 つのプロキシおよびロードバランシングモジュールを提供します。**mod_jk** の詳細については、「[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使用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 をより密接に結合するため、推奨される HTTP コネクターになります。利点と欠点を含むサポート対象 HTTP コネクターの概要については、「[HTTP コネクターの概要](#)」を参照してください。

mod_jk とは異なり、**mod_proxy** は、HTTP および HTTPS プロトコルを介して接続をサポートします。これらは 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指定でき、スティッキーセッションをサポートします。

mod_proxy モジュールを使用するには、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となります。Web サブシステムの設定については「[Web サブシステムの設定](#)」を参照してください。

[Report a bug](#)

19.7.2. 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 により提供される負荷分散モジュールです。このタスクは、基本的な設定を提供します。高度な設定または詳細については、Apache の **mod_proxy** ドキュメンテーション (https://httpd.apache.org/docs/2.2/mod/mod_proxy.html) を参照してください。JBoss EAP 6 の観点からの **mod_proxy** の詳細については、「[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 の httpd または Apache HTTP Server のどちらか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 の JBoss EAP 6 のダウンロードエリアから個別にダウンロードできます。この Apache HTTP Server を使用したい場合は「[JBoss EAP 6 に含まれる Apache HTTP Server のインストール \(Zip\)](#)」の詳細情報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 Server は、通常、すでに同梱されている **mod_proxy** モジュールで提供されます (Red Hat Enterprise Linux と JBoss Enterprise Web Server に含まれる Apache HTTP Server の場合)。
- Apache HTTP Server の設定を変更するには、**root** または管理者権限が必要です。
- ここで使用する例では JBoss EAP 6 が HTTP または HTTPS Web コネクタで設定されていることを仮定しています。Web サブシステムの設定については「[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_HOME/conf/httpd.conf ファイルで次の行を探します。これらの行が存在しない場合は、行を最下部に追加します。これらの行が存在し、行がコメント (#) 文字始まる場合は、この文字を削除します。編集後、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を追加します。

以下の設定を、他の **<VirtualHost>** ディレクトリーの直下にある

HTTPD_HOME/conf/httpd.conf ファイルに追加します。値をセットアップ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. 負荷分散プロキシを追加します。

mod_proxy をロードバランサーとして使用し、複数の JBoss EAP 6 サーバーにワークを送信するには、以下の設定を **HTTPD_HOME/conf/httpd.conf** ファイルに追加します。IP アドレスの例には架空の値が使用されています。これらの値を環境に適した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>
```

上記の例では、すべて HTTP プロトコルを使用して通信します。適切な **mod_proxy** モジュールをロードする場合は、AJP または HTTPS プロトコルを代わりに使用できます。詳細については、Apache の **mod_proxy** ドキュメンテーション (http://httpd.apache.org/docs/2.2/mod/mod_proxy.html) を参照してください。

4. スティックセッションを有効にします。

スティッキーセッションを使用すると、クライアント要求が特定の JBoss EAP 6 ノードに送信された場合に、ノードが利用不可能にならない限り、すべての将来的な要求が同じノードに送信されます。これは、ほとんどの場合で適切な挙動です。

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、使用できる他のいくつかのパラメーターも示されます。詳細については、Apache の **mod_proxy** ドキュメンテーション (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

標準またはロードバランシング設定で **mod_proxy** を使用してクライアント要求を JBoss EAP 6 サーバーまたはクラスターに送信するよう Apache HTTP Server が設定されます。JBoss EAP 6 がこれらの

要求に応答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。

[Report a bug](#)

19.8. MICROSOFT ISAPI

19.8.1. インターネットサーバー API (ISAPI) HTTP コネクター

Internet Server API (ISAPI) は、Microsoft の Internet Information Services (IIS) Web サーバー向けの HTTP コネクターです。IIS クラスター内では、JBoss EAP 6 をワーカーノードとして使用できます。

JBoss EAP 6 が IIS クラスターに参加するよう設定するには、「[Microsoft IIS が ISAPI リダイレクターを使用するよう設定](#)」を参照してください。ISAPI の詳細については、[http://msdn.microsoft.com/en-us/library/ms524911\(v=VS.90\).aspx](http://msdn.microsoft.com/en-us/library/ms524911(v=VS.90).aspx) を参照してください。

[Report a bug](#)

19.8.2. 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>) にアクセスします。
2. **ダウンロード** の **Red Hat JBoss Middleware** をクリックし、**Product** ドロップダウンリストから **Enterprise Application Platform** を選択します。
3. **バージョン** 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、**Download** オプションの **Red Hat JBoss Enterprise Application Platform 6.3.0 Webserver Connector Natives for Windows Server 2008 x86_64** または **Red Hat JBoss Enterprise Application Platform 6.3.0 Webserver Connector Natives for Windows Server 2008 i686** を選択します。
5. zip ファイルを開き、**jboss-eap-6.3/modules/system/layers/base/native/sbin** ディレクトリーの内容をサーバー上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。

[Report a bug](#)

19.8.3. Microsoft IIS が ISAPI リダイレクターを使用するよう設定

必読トピック:

- [「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リストは <https://access.redhat.com/site/articles/111663> を参照してください。

手順19.17 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** をクリックし、**inetmgr** と入力して、IIS マネージャーを開きます。

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** ウィンドウが開きます。
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。
 - **Alias:** 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** 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** ウィ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で、**IIS 7** アイテム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** を見つけます。**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** ウィンドウを閉じます。

結果

Microsoft IIS が ISAPI リダイレクターを使用するよう設定されます。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](#)」を行い、「[ISAPI リダイレクターがクライアント要求を JBoss EAP 6 に送信するよう設定](#)」または「[ISAPI がクライアント要求を複数の JBoss EAP 6 サーバーで分散するよう設定](#)」を行います。

[Report a bug](#)

19.8.4. ISAPI リダイレクターがクライアント要求を 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 リダイレクターから要求を受け入れるよう設定します。負荷分散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 がクライアント要求を複数の 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 がすでに「[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](#)」どおりに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 リダイレクターを使用するよう設定](#)」

手順19.18 プロパティ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 Redirector
# Extension uri definition
extension_uri=/jboss/isapi_redirect.dll
```

```
# Full path to the log file for the ISAPI Redirector
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 uriworkermmap.properties file
worker_mount_file=c:\connectors\uriworkermmap.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。詳細については、[ステップ 5](#) を参照してください。

3. **uriworkermmap.properties** ファイルを作成します。

uriworkermmap.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。**uriworkermmap.properties** ファイルを **C:\connectors** に格納してください。

```
# 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** ディレクトリーに格納してください。

```
# 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** ディレクトリーに格納してください。

```
#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 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。

[Report a bug](#)

19.8.5. ISAPI がクライアント要求を複数の JBoss EAP 6 サーバーで分散するよう設定

概要

この設定により、クライアント要求は指定した JBoss EAP 6 サーバーで分散されます。クライアント要求を、デプロイメントごとに、特定の JBoss EAP 6 サーバーに送信する場合は、[「ISAPI リダイレクターがクライアント要求を 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 がすでに [「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」](#) どおりに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 リダイレクターを使用するよう設定」](#)

手順19.19 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。

この手順は、JBoss EAP 6 のインストールディレクトリーに格納された `connectors` ディレクトリーを使用します。

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 Redirector
# Extension uri definition
extension_uri=/jboss/isapi_redirect.dll

# Full path to the log file for the ISAPI Redirector
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。詳細については、[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要求を **router** という名前の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については、[ステップ 4](#) を参照してください。

uriworkermap.properties ファイルを **C:\connectors** に格納してください。

```
# 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、ファイルの最後で設定され、ワーカー **worker01** および **worker02** から構成されます。**workers.properties** ファイルは Apache mod_jk 設定に使用されるのと同じファイルの構文に従います。**workers.properties** ファイルの構文の詳細については、「[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** ディレクトリーに格納してください。

```
# 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** ディレクトリーに格納してください。

```
#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 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 サーバーが、**workers.properties** ファイルで参照された JBoss EAP 6 サーバーにクライアント要求を送信し、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。

[Report a bug](#)

19.9. ORACLE NSAPI

19.9.1. Netscape Server API (NSAPI) HTTP コネクター

Netscape Server API (NSAPI) は、JBoss EAP 6 がノードとして Oracle iPlanet Web Server (旧名 Netscape Web Server) に参加することを許可する HTTP コネクターです。このコネクターを設定するには、「[NSAPI を負荷分散クラスターとして設定](#)」を参照してください。

[Report a bug](#)

19.9.2. 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。
- NSAPI コネクターを除き、Intel アーキテクチャーでは Oracle iPlanet Web Server 7.0.15 以上のバージョン、SPARC アーキテクチャーでは 7.0.14 以上のバージョンがインストールおよび設定されている必要があります。
- JBoss EAP 6 は、ワーカーノードとして機能する各サーバー上でインストールおよび設定されます。「[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 ZIP パッケージをカスタマーサービスポータル (<https://access.redhat.com>) からダウンロードする必要があります。

手順19.20 NSAPI コネクターの抽出および設定

- JBoss ネイティブコンポーネントパッケージを抽出します。
この手順の残りでは、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ます。この手順の残りでは、`/opt/oracle/webserver7/config` ディレクトリーは `IPLANET_CONFIG` と示されます。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。
- 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、お使いの設定に合うよう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、ワーカーノードとプロパティがリストされた **workers.properties** ファイルの場所を定義します。

```
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 を基本的な HTTP コネクターとして設定」
- 「NSAPI を負荷分散クラスターとして設定」

19.9.3. NSAPI を基本的な HTTP コネクターとして設定

概要

このタスクでは、NSAPI コネクターが負荷分散または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に設定します。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。負荷分散設定については「[NSAPI を負荷分散クラスターとして設定](#)」を参照してください。

前提条件

- 現在のタスクを継続する前に、「[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19.21 基本的な HTTP コネクターの設定

- 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>
```

- 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>
```

上記の例は、URL パス **/status** を、**worker01** という名前のワーカーにリダイレクトし、**/nc/** 以下のすべての 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 番目の行は、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** という名前の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については、「[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。

[Report a bug](#)

19.9.4. NSAPI を負荷分散クラスターとして設定

概要

このタスクでは、NSAPI コネクターが負荷分散設定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。NSAPI 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 を基本的な HTTP コネクターとして設定](#)」を参照してください。

前提条件

- 現在のタスクを継続する前に、「[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19.22 負荷分散のためコネクターを設定する

- 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>**) のすぐ後に、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 **jknsapi** オブジェクトは、**default** オブジェクトの **name="jknsapi"** 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。以下の内容をファイルに貼り付け、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については、[「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。

実行する iPlanet Web Server のバージョンに応じて、以下のいずれかを選択します。

- iPlanet Web Server 6.1 の場合

```
IPLANET_CONFIG/./stop
IPLANET_CONFIG/./start
```

- iPlanet Web Server 7.0 の場合

```
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。

[Report a bug](#)

第20章 メッセージング

20.1. はじめに

20.1.1. 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、自動クライアントフェイルオーバーとともに高可用性 (HA) を提供し、サーバー障害時にメッセージの信頼性を保証します。また、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をサポートします。

HornetQ は JBoss EAP 6 の Java Message Service (JMS) プロバイダーで、**Messaging Subsystem** として設定されます。

[Report a bug](#)

20.1.2. 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することが可能となる上、信頼性が向上します。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) パターンをベースとするシステムとは異なり、メッセージングシステムは要求と応答の間に密な関係のない非同期メッセージパッシングパターンを主に使用します。大半のメッセージングシステムでは、要求-応答モードをサポートしますが、これはメッセージングシステムの主要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とメッセージの送信元を切り離します。メッセージの送信元とコンシューマーは完全に独立し、お互いについては何も知りません。多くの場合、大型のエンタープライズ環境では、メッセージングシステムを使用して、異種システムを疎結合するメッセージバスを実装しています。メッセージバスは通常 Enterprise Service Bus (ESB) の中核を形成します。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により、システムを拡張して、より容易に適応させることができます。また、脆弱な相互依存関係がないため、新規システムの追加や旧システムのリタイアを行う柔軟性も向上します。

[Report a bug](#)

20.1.3. 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。メッセージがキューに入ると、通常は永続化されて、配信が保証されます。キューを通過したメッセージは、メッセージングシステムによりメッセージコンシューマーに配信されます。メッセージが処理されると、メッセージコンシューマーはメッセージが配信されたことを確認応答します。

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 パターンでは、サーバー上の単一のエンティティに対して複数の送信者がメッセージを送信できます。このエンティティは、「トピック」と呼ばれます。各トピックには、「サブスクリプション」と呼ばれる複数のコンシューマーが参加できます。

各サブスクリプションは、トピックによって送信された全メッセージのコピーを受信します。これは、各メッセージを消費するのが単一のコンシューマーのみである 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、トピックに送信された各メッセージをサブスクライバーが消費するまで、そのコピーを保持します。このようなコピーは、サーバーの再起動時にも維持されます。非永続的なサブスクリプションは、そのサブスクリプションを作成した接続が有効な間のみ継続します。

[Report a bug](#)

20.2. トランスポートの設定

20.2.1. 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 の略語であり、クライアントとサーバーが同じ JVM で実行されているときに使用できます。

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、サーバーのいずれかのアクセプターと互換性があるコネクターを使用する必要があります。Invm コネクターは 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サーバーの **standalone.xml** と **domain.xml** で設定されます。これらは、管理コンソールまたは管理 CLI のいずれかを使用して定義できます。

[Report a bug](#)

20.2.2. Netty TCP の設定

Netty TCP は暗号化されていない TCP ソケットをベースとした簡単なトランスポートです。Netty

TCP を設定すると旧式の ブロッキング Java IO またはノンブロッキング Java NIO を使用できます。同時接続が多い場合にスケーラビリティが高いためサーバー側には Java NIO の使用が推奨されます。同時接続の数が少ない場合は、旧式の Java IO の待ち時間は NIO よりも短くなります。

Netty TCP は暗号化されないため、信頼されないネットワーク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20.1 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>
```

この設定例は、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によって異なり、特定のホストとポートを宣言する必要があります。

下表は Netty TCP 設定プロパティの説明になります。

表20.1 Netty TCP 設定プロパティ

プロパティ	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 を設定すると batch-delay に設定されたミリ秒を上限として書き込みを一括処理できます。メッセージ転送の平均待ち時間が長くなり、大変小さなメッセージのスループットの合計が増加します。

プロパティ	デフォルト	説明
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たとき、デフォルトではメッセージが到達した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適切な待ち時間が発生しますが、スループットや待ち時間が少なくなります。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 コネクタでは、リモートアドレスへ接続するとき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は、コネクタは使用できるローカルアドレスを使用します。
local-port	0	Netty コネクタでは、リモートアドレスへ接続するとき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) が指定された場合、システムが一時的なポートを選択します。有効なポートは 0 から 65535 です。
nio-remoting-threads	-1	NIO を使用するよう設定された場合、デフォルトでは受信パケットの処理に <code>Runtime.getRuntime().availableProcessors()</code> によって報告されたコア (またはハイパースレッド) の数の 3 倍にあたるスレッド数が HornetQ によって使用されます。この値を上書きするには、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。このアルゴリズムは、ネットワーク上で送信されるパケットの数を削減し、TCP/IP ネットワークの効率を向上します。

プロパティ	デフォルト	説明
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 Java NIO が使用されます。false に設定された場合、旧式のブロッキング Java IO が使用されます。サーバーに多くの同時接続を処理させる必要がある場合はノンブロッキング Java NIO を使用します。これ以外の場合は旧式の (ブロッキング) IO を使用するとよいでしょう。



注記

Netty TCP プロパティはすべてのタイプのトランスポートに対して有効です。

[Report a bug](#)

20.2.3. 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 TCP ソケットをベースとした簡単なトランスポートです。Netty SSL は Netty TCP と似ていますが、セキュアソケットレイヤー (SSL) を使用して TCP 接続を暗号化して、セキュリティを強化します。



警告

Red Hat は、影響す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 を明示的に無効化することを推奨しています。

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と使用できますが、一部のパラメーターはアクセプターを使用した場合のみ動作します。パラメーターの詳細には、これらのパラメーターをコネクターで使った場合とアクセプターで使った場合の違いが説明されています。

```

<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20.2 Netty SSL 設定プロパティ

プロパティ名	デフォルト	説明
ssl-enabled	true	SSL を有効にします。
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た場合はサーバー側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た場合はクライアント側キーストアのパスワードになります。双方向 SSL (相互認証) を使用する場合はコネクターのみに関係します。この値はサーバー上で設定できますが、クライアントによってダウンロードおよび使用されます。
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た場合は、サーバーが信用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 SSL キーストアへのパスになります。双方向 SSL (相互認証など) を使用する場合はアクセプターのみが関係します。コネクターで使用された場合は、クライアントが信用するすべてのサーバーの公開鍵を保持するクライアント側 SSL キーストアへのパスになります。このパスはサーバー上で設定されますが、クライアントによってダウンロードおよび使用されます。

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た場合はサーバー側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た場合はクライアント側トラストストアのパスワードになります。これは、一方方向および双方向 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、クライアントによってダウンロードおよび使用されます。
- **trust-store-path**: アクセプターで使用された場合は、サーバーが信用するクライアントすべてのキーの証明書を保持するサーバー側 SSL トラストストアへのパスになります。コネクターで

使用された場合は、クライアントが信用するサーバーすべての公開鍵の証明書を保持するクライアント側 SSL トラストストアへのパスになります。これは、一方向および双方向 SSL のコネクタに関連します。このパスはサーバー上で設定できますが、クライアントによってダウンロードおよび使用されます。

[Report a bug](#)

20.2.4. 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 トラフィックの通過のみを許可する場合に便利です。Netty HTTP は Netty TCP と同じプロパティを使用し、以下のプロパティの一部を追加で使用します。



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と使用できます。Netty HTTP トランスポートは標準 HTTP ポート (デフォルトでは 8080) の再使用を許可しません。標準 HTTP ポートを使用すると例外が発生します。HornetQ の接続を標準 HTTP ポートを通じてトンネリングするには「[Netty サブレットの設定](#)」(Netty サブレットトランスポート)を使用します。

```
<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の説明になります。

表20.3 Netty HTTP 設定プロパティ

プロパティ名	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)。

プロパティ名	デフォルト	説明
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。

[Report a bug](#)

20.2.5. Netty サブレットの設定

サブレットトランスポートを使用すると、HTTP 上で HornetQ トラフィックをサブレットエンジンで稼働しているサブレットヘトンネリングできます。サブレットエンジンは HornetQ トラフィックを in-VM HornetQ サーバーへリダイレクトします。Netty HTTP トランスポートは、特定ポート上で HTTP トラフィックをリスンする Web サーバーとして動作します。サ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があるサブレットエンジンを介してルーティングされます。

Netty サブレットトランスポートが動作するようにサ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。

- サブレットをデプロイします。以下の例はサ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>
```

初期化パラメーター **endpoint** 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上で挿入します。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**) でクライアントの接続を定義します。

```
<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"/>
</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。

[Report a bug](#)

20.3. JAVA NAMING AND DIRECTORY INTERFACE (JNDI)

Java Naming and Directory Interface (JNDI) は、ネーミングサービスやディレクトリーサービス向けの標準 Java API です。JNDI により、Java ベースの技術で分散されたコンピューティング環境におかれている名前付きのコンポーネントを検出および整理できます。

[Report a bug](#)

20.4. デッド接続の検出

20.4.1. 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ことにより、アプリケーションがリソースを自動的に閉じるよう設定できます。

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 パケットを受け取らない場合にサーバーで接続をアライブ状態に維持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回復した時にクライアントが再接続できるようになります。

HornetQConnectionFactory インスタンスで **connection-ttl** パラメーターを指定することにより、JMS クライアントに対して接続 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で **connection-ttl-override** パラメーターを定義してすべての値をオーバーライドできます。**connection-ttl-override** パラメーターはデフォルトで無効になり、値は -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なかった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。

[Report a bug](#)

20.4.2. 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自動的に ping パケットをサーバーに送信します。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かぎり、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場合、クライアントは通信が失敗したと見なします。次に、クライアントはフェールオーバーを開始するか、**FailureListener** インスタンスを呼び出します。

JMS クライアントの場合、**HornetQConnectionFactory** インスタンスで **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。サーバーサイドで 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 **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 3000 ミリ秒です。値が -1 の場合は、サーバーからデータを受信されないときにクライアントで接続が閉じられません。トランジションが失敗した場合にクライアントが再接続できるように、クライアント障害チェック期間の値は接続 TTL よりも大幅に小さくなります。

非同期接続実行の設定

デフォルトでは、サーバーサイドで受信されたパケットはリモートスレッドで実行されます。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ことにより、リモートスレッドを解放できます。非同期接続実行は、**standalone.xml** および **domain.xml** サーバー設定ファイルで **async-connection-execution-enabled** パラメーターを使用して設定できます。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場合は、若干のレイテンシーが追加されます。パフォーマンス上の理由により、リモートスレッドでは短い実行中の操作が常に処理されます。

[Report a bug](#)

20.5. サイズの大きなメッセージの処理

20.5.1. 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。サイズが大きなメッセージは、そのままストリームしたり、効率的に転送する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 がこの **InputStream** を読み取り、データを断片的にサーバーへ送信します。

クライアントやサーバーは、大きなメッセージのボディー部分を完全な形で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ムします。

[Report a bug](#)

20.5.2. 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。

[Report a bug](#)

20.5.3. 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。デフォルトでは、大きなメッセージの最小サイズ (min-large-message-size) は 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。デフォルトでは、大きなメッセージの最小サイズ (min-large-message-size) は 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指定する必要があります。

大きなメッセージを迅速および効率的に送信するには、大きなメッセージの圧縮を選択できます。圧縮および展開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のサイズが **min-large-message-size** 未満であった場合、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) を使用する場合、サーバーロケーターまたは ConnectionFactory でブール値プロパティーである **compress-large-messages** を true に設定すると、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>

```

[Report a bug](#)

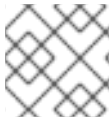
20.6. ページング

20.6.1. ページング

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同時にメモリーに保存するのは困難です。

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。

[Report a bug](#)

20.6.2. ページファイル

ファイルシステム上の各アドレスに対して個別のファイルがあり、メッセージを複数のファイルに保存します。メッセージを保存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**) 以下のメッセージが含まれます。

システムは必要に応じてページファイルを検索し、ページのすべてのメッセージがクライアントによって受信されると、ページファイルを削除します。

セクターを持つコンシューマーもページファイルを検索し、基準に見合わないメッセージを無視します。

[Report a bug](#)

20.6.3. 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。ページングディレクトリー/フォルダーの場所は、**paging-directory** パラメーターを使用して設定できます。

```

<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/ホ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対して1つの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、スタンドアロンモードの場合は **EAP_HOME/standalone/data/messagingpaging**、ドメインモードの場合は **EAP_HOME/domain/servers/SERVERNAME/data/messagingpaging** になります。

[Report a bug](#)

20.6.4. 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。



注記

ページングはアドレスごとに個別に行われます。アドレスに対して **max-size-bytes** を設定すると、一致する各アドレスには指定した最大サイズが適用されます。しかし、一致するすべてのアドレスの合計サイズが **max-size-bytes** に限定されるわけではありません。

サーバー設定ファイル (**standalone.xml** および **domain.xml**) で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>
```

下表はアドレス設定のパラメーターについて説明しています。

表20.4 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	ページングシステムで使用する各ページファイルのサイズを指定します。

要素	デフォルト値	説明
address-full-policy	PAGE	ページングの決定に使用されます。この属性には、以下の値のいずれかを指定できます。 PAGE : ページングが有効になり、設定された制限を越えたメッセージはディスクにページングされます。 DROP : 設定された制限を越えたメッセージは通知なしでドロップされます。 FAIL : メッセージがドロップされ、クライアントメッセージプロデューサーへ例外が送信されます。 BLOCK : クライアントメッセージプロデューサーによって設定された制限を越えるメッセージが送信されると、クライアントメッセージプロデューサーがブロックされます。
page-max-cache-size	5	ページング検索中に入出力を最適化するため、 page-max-cache-size 以下のページファイルがメモリーに保持されます。



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 **DROP** (メッセージをドロップする)、**FAIL** (クライアント側で例外を発生してメッセージをドロップする)、または **BLOCK** (今後メッセージを送信しないようプロデューサーをブロックする) に設定します。デフォルトの設定では、アドレスが **max-size-bytes** に達した後、メッセージをページングするようにすべてのアドレスが設定されています。

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た場合、メッセージが1つのみメモリーに格納されます。各キューは、このメッセージへの参照のみを処理します。そのため、このメッセージを参照するキューすべてがメッセージを送信したときのみメモリーに空きができます。



注記

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出入力のパフォーマンスが減少することがあります。これは、ページングシステム上の追加ストレージより送信されたメッセージがすべてのキューにあるからです。

[Report a bug](#)

20.7. 迂回

Diverts 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 (メッセージのルーティング先) から別のアドレスに迂回する場合に役に立ちます。Diverts 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、メッセージのコピーが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せん。

迂回は **Transformer** とオプションのメッセージフィルターを適用するよう設定できます。オプションのメッセージフィルターを使用すると、指定したフィルターに一致するメッセージのみを迂回できます。トランスフォーマーは、メッセージを別の形式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た場合は、迂回されたすべてのメッセージが **Transformer** 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、以下に示されたパターンに従ってください。

- 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、メッセージを別のサーバーのアドレスに迂回するブリッジをセットアップします

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。

[Report a bug](#)

20.7.1. 特別な迂回

特別な迂回により、すべてのメッセージが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。特別な迂回は、**standalone.xml** および **domain.xml** サーバー設定ファイルで **exclusive** 属性を **true** に設定することにより有効にできます。

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。

- **address**: 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が迂回されます。他のすべてのメッセージは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本文またはプロパティーを変更できるようになります
- **exclusive**: 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

[Report a bug](#)

20.7.2. 特別でない迂回

特別でない迂回により、元のメッセージのコピーが新しいアドレスに転送されます。元のメッセージは引き続き古いメッセージに届きます。特別でない迂回は、**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** に設定することにより有効にできます。

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る各メッセージのコピーが作成され、**jms.topic.spyTopic** アドレスに送信されます。

[Report a bug](#)

20.8. 設定

20.8.1. JMS サーバーの設定

HornetQ 向けに 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、またはスタンドアロンの **EAP_HOME/standalone/configuration/standalone-full.xml** ファイルに含まれています。

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 設定が含まれています。JNDI に必要な JMS の **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。
<extensions> 要素内に以下の行が存在し、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該当する **<profile>** を探し、**<subsystems>** タグを見つけます。
 - b. **<profile>** タグのすぐ後に以下の XML を追加します。

```
<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> タグの後の空行に以下の 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 の値が **min-large-message-size** の値 (デフォルトでは 100KiB) 以上でないと、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** オブジェクトを使い、サーバーへの接続を確立します。JMS 接続ファクトリーオブジェクトを HornetQ に追加するには、次のように接続ファクトリごとに、単一の **<jms-connection-factories>** タグと **<connection-factory>** 要素が含まれるようにします。

```
<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 コネクタおよびアクセプターを設定します。

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。コネクタオブジェクトは、HornetQ サーバーへ接続するため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 サーバーによって許可される接続のタイプを識別します。

netty コネクタを設定するには、以下の設定が含まれるようにします。


```

<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、以下の設定が含まれるようにします。

```

<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>
  </hornetq-server>
</subsystem>

```

```

<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、**messaging** および **messaging-throughput** ソケットバインディングを参照します。**messaging** 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。**<socket-binding-group>** タグはサーバー設定ファイルの別のセクションにあります。以下のソケットバインディングが **<socket-binding-groups>** 要素に存在するようにしてください。

```

<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、サーバーを **Message-Enabled** モードで起動する必要があります。これには、**-c** オプションを使用し、**standalone-full.xml** (スタンドアロンサーバー向け) 設定ファイルの使用を強制します。たとえば、スタンドアロンモードでは以下を使用するとサーバーをメッセージ有効モードで起動できます。

```
./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、**Configuration** タブを選択します。**Subsystems** メニューを展開した後、**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 **View** 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

最初に、JMS 設定ファイル *example-jms.xml* を作成します。以下のエントリーを追加し、値は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。

standalone-full.xml を例として使用し、このファイルでメッセージングサブシステムを見つけます。

```
<subsystem xmlns="urn:jboss:domain:messaging:1.4">
```

再度、以下のエントリーを追加し、例の値は独自の値に置き換えます。これらのエントリーは `</jms-connection-factories>` 終了タグと `</hornetq-server>` 要素の間に追加する必要があります。

```
<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>
```

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** DTD を確認します。

[Report a bug](#)

20.8.2. 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配信タイミング、配信試行回数、メッセージの有効期限などの側面を制御する複数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

アドレスワイルドカードを使用すると単一のステートメントで複数の似たアドレスを一致させることができます。これは、システムがアスタリスク (*****) 文字を使用して1回の検索で複数のファイルや文字列を一致させることと似ています。以下の文字はワイルドカードステートメントでは特別な意味を持っています。

表20.5 JMX のワイルドカード構文

文字	説明
. (ピリオド1つ)	ワイルドカード表現で単語の間のスペースを意味します。
# (シャープまたはハッシュマーク)	ゼロ以上の単語シーケンスと一致します。
* (アスタリスク)	1つの単語と一致します。

表20.6 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 とは一致しません。
news.*	news.europe と一致しますが news.europe.sport とは一致しません。
news.*.sport	news.europe.sport と news.usa.sport とは一致しますが、 news.europe.politics とは一致しません。

例20.2 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20.7 JMS アドレス設定の説明

要素	説明	デフォルト値	タイプ
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	無効なレターアドレスが指定された場合、 max-delivery-attempts 配信試行が失敗すると、メッセージが無効なレターアドレスに移動されます。それ以外の場合、未配達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	期限切れの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	キューで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	dead-letter-address に送信する前、または破棄する前にメッセージを再配信する最大試行回数。	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

要素	説明	デフォルト値	タイプ
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
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 に設定された場合は、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

- **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 を使用してアドレスを設定します。

- a. **新しいパターンの追加**

add 操作を使用して、新しいアドレス設定を作成します (必要な場合)。このコマンドは、管理 CLI セッションのルートから実行できます。この場合、以下の例では、*patternname* というタイトルの新しいパターンが作成され、**max-delivery-attempts** 属性が 5 として宣言されます。**full** プロファイルのスタンドアロンおよび管理対象ドメインの編集例を以下に示します。

```
[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**

write 操作を使用して新しい値を属性に書き込みます。タブ補完を使用して、入力するコマンド文字列を補完したり、利用可能な属性を公開したりできます。以下の例は、**max-delivery-attempts** 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**

include-runtime=true パラメーターを用いて **read-resource** 操作を実行し、値の変更を確認します。サーバーモデルでアクティブな現在の値をすべて表示します。

```
[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
```

o **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。ドメインモードでは、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 **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 プロバイダーのリストが表示されます。デフォルトの設定では、**default** という名前の1つのプロバイダーだけが表示されます。**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、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。また、**Dead Letter Address**、**Expiry Address**、**Redelivery Delay**、および **Max Delivery Attempts** を編集することもできます。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。

[Report a bug](#)

20.8.3. HornetQ でのメッセージングの設定

JBoss EAP 6 でのメッセージングの設定では、管理コンソールまたは管理 CLI の使用が推奨されます。どちらの管理ツールでも、**standalone.xml** や **domain.xml** 設定ファイルを手作業で編集せずに永続的な変更を行うことができますが、デフォルト設定ファイルのメッセージングコンポーネントについて理解できると便利です。デフォルトの設定ファイルでは、管理ツールを使用するドキュメントサンプルによって参考用の設定ファイルスニペットが提供されます。

[Report a bug](#)

20.8.4. 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**) を手作業で編集します。
- CLI を使用してサーバー設定ファイルを編集します。

手順20.1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. 編集するサーバー設定ファイルを開きます (例: **standalone-full.xml** および **standalone-full-ha.xml**)。
2. ファイルのロギングサブシステム設定を見つけます。デフォルトの設定は次のようになります。

```
<logger category="com.arjuna">
  <level name="TRACE"/>
</logger>
...
<logger category="org.apache.tomcat.util.modeler">
  <level name="WARN"/>
</logger>
....
```

3. 下例のように、**org.hornetq** ロガーカテゴリーと指定する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、サーバー設定ファイルに **org.hornetq** ロガーカテゴリーと指定するロギングレベルを追加することもできます。詳細については、「[CLI でのログカテゴリー設定](#)」を参照してください。

[Report a bug](#)

20.8.5. HornetQ Core Bridge の設定

例20.3 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20.8 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	これは必須のパラメーターで、ブリッジが消費するローカルキューの一意名になります。ブリッジのインスタンスが起動時に作成されるまでにキューがすでに存在する必要があります。
forwarding-address	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。転送先のアドレスが指定されないと、メッセージの元のアドレスが保持されます。
ha	任意のパラメーターで、このブリッジが高可用性をサポートすべきかどうかを決定します。 true の場合は、クラスターの利用可能なサーバーへ接続し、フェイルオーバーをサポートします。デフォルト値は false です。
reconnect-attempts	任意のパラメーターで、試行を停止してシャットダウンする前にブリッジが再接続を試みる合計回数を決定します。値が -1 の場合は、試行回数が無制限になります。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 プロパティを自動的に挿入するかどうかを決定します。
static-connectors	static-connectors は 別の場所に定義されたコネクタ要素を示す connector-ref 要素のリストです。コネクタは使用するトランスポート (TCP、SSL、HTTP など) や、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。

[Report a bug](#)

20.8.6. 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、これらのメッセージを通常別のサーバーにあるターゲットキューまたはトピックへ送信します。

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と共にまたはJBoss AS インスタンス内に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の場所は同じ仮想マシンでも別の仮想マシンでもかまいません。

例20.4 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>
```

表20.9 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。

属性	説明
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 はターゲットの 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 はターゲットの Destination を作成します。
quality-of-service	このパラメーターはサービスモードの必要なクオリティーを表します。可能な値は AT_MOST_ONCE、DUPLICATES_OK、および 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。
max-retries	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再接続を試行する回数を表します。この回数接続を試行した後、ブリッジは接続を断念します。-1 を指定すると永久に接続を試行します。
max-batch-size	メッセージを一括してターゲットの宛先に送る前にソースの宛先から消費するメッセージの最大数を表します。値は1以上である必要があります。
max-batch-time	消費されたメッセージの数が MaxBatchSize に到達しない場合でも、ターゲットへバッチを送信する前に待機する最大期間 (ミリ秒単位) を表します。-1 (永久に待機) または1以上 (実際の時間) を値として指定する必要があります。
add-messageID-in-header	<p>true の場合、元メッセージのメッセージ ID が宛先に送信されたメッセージのヘッダー HORNETQ_BRIDGE_MSG_ID_LIST に追加されます。メッセージが2回以上ブリッジングされる場合は、各メッセージ ID が追加されます。これは、分散された要求/応答パターンが使用されるようにします。</p> <p>メッセージを受信するとき、最初のメッセージ ID の関連 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。</p>

詳細については、「[JMS ブリッジの作成](#)」を参照してください。

[Report a bug](#)

20.8.7. 遅延再配信の設定

はじめに

遅延再配信は、Java Messaging Service (JMS) のサブシステム設定の **<address-setting>** 設定要素の子要素である **<redelivery-delay>** 要素に定義されます。

```
<!-- 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。**<redelivery-delay>** を **0** に設定すると、再配信の遅延はありません。**<address-match>** 要素の **match** 属性にアドレスワイルドカードを使用すると、ワイルドカードに一致するアドレスに対して再配信の遅延を設定することができます。

[Report a bug](#)

20.8.8. 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) の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と、**<max-delivery-attempts>** で指定される回数配信を試みた後、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することができ、特定アドレスの **<max-delivery-attempts>** を **-1** に設定し、このアドレスのみ再配信が無限に行われるようにすることが可能です。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可能です。

[Report a bug](#)

20.8.9. 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** 設定に定義されています。例は次のとおりです。

```
<!-- 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>
```

メッセージの有効期限が切れ、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より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特定範囲の期限切れアドレスを設定することも可能です。JMX のワイルドカード構文とその例については「[JMS アドレスの設定](#)」を参照してください。

[Report a bug](#)

20.8.10. HornetQ 設定属性のリファレンス

HornetQ の JBoss EAP 6 実装では、設定の以下の属性が公開されます。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公開できます。

例20.5 例

```
[standalone@localhost:9999 /] /subsystem=messaging/hornetq-server=default:read-resource
```

表20.10 HornetQ 属性

属性	デフォルト値	タイプ	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。
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。

属性	デフォルト値	タイプ	説明
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	CHANGE ME!!	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 ディレクトリを作成するかどうか。
create-journal-dir	true	BOOLEAN	起動時にサーバーが journal ディレクトリ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。

属性	デフォルト値	タイプ	説明
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。
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 より公開するかどうか。これらの MBean にアクセスすると設定の一貫性が失われることがあるため、この設定は推奨されません。
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)。

属性	デフォルト値	タイプ	説明
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)。
journal-max-io	1	INT	一度に AIO キューに置ける最大書き込み要求数。ASYNCIO ジャーナルが使用されると、デフォルト値が 500 に変更されます。
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 を値として取ります。
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接続するために使用されるコネクターの名前。このサーバーがシェアードナッシング 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	

属性	デフォルト値	タイプ	説明
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。
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	

属性	デフォルト値	タイプ	説明
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 に指定すると、メッセージがキャンセルされた後のみ発生します。
persist-id-cache	true	BOOLEAN	ID がジャーナルへ永続化されるかどうか。
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化する速さに対して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判断するのに便利です。
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。

属性	デフォルト値	タイプ	説明
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	other	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。
security-enabled	true	BOOLEAN	セキュリティが有効かどうか。
security-setting			セキュリティ設定は、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	基本のランタイム情報をサーバーログにダンプする頻度。1未満の数字を指定するとこの機能は無効になります。
shared store	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 の値が、サーバーへ送信されたメッセージのサイズよりも大きくないと、サーバーはメッセージを格納できません。

[Report a bug](#)

20.8.11. 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が指定期間 (ミリ秒単位) 後にコンシューマーへ配達されなかった場合、サーバー側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 を使用して、期限切れの時間を直接メッセージに設定できます。例は次のとおりです。

```
// 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。

- `_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。

Time-To-Live パラメーターは、JMS プロデューサーで設定する以外に、メッセージごとに設定することもできます。これは、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することにより実現できます。

```
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。値は 0 から 9 までの数字を指定する必要があり、最も優先度が高いのは 9 になります。デフォルト値は 3 です。

[Report a bug](#)

20.9. メッセージのグループ化

20.9.1. 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。そのため、これらのメッセージは共通のグループプロパティで特定できます。
- 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され、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。

[Report a bug](#)

20.9.2. クライアント側での HornetQ Core API の使用

プロパティ `_HQ_GROUP_ID` 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。無作為で一意のメッセージグループ ID を選択するには、**autogroup** プロパティを `SessionFactory` で `true` に設定します。

[Report a bug](#)

20.9.3. Java Messaging Service (JMS) クライアントのサーバー設定

プロパティ **JMSXGroupID** 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。複数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には、これらのメッセージに同じ **JMSXGroupID** を設定します。

■

```

Message message = ...
message.setStringProperty("JMSXGroupID", "Group-0");
producer.send(message);

message = ...
message.setStringProperty("JMSXGroupID", "Group-0");
producer.send(message);

```

autogroup プロパティを `HornetQConnectonFactory` で `true` に設定する方法もあります。設定後、`HornetQConnectionFactory` は一意なメッセージグループ ID を無作為に選出します。以下のように **autogroup** プロパティをサーバー設定ファイル (**standalone.xml** および **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>
  <autogroup>true</autogroup>
</connection-factory>

```

前述の 2 つの方法の他に、接続ファクトリーを介して特定のメッセージグループ ID を設定する方法もあります。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で、**JMSXGroupID** プロパティが指定の値に設定されます。この設定ファクトリーで特定のメッセージグループ ID 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** プロパティを次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>

```

[Report a bug](#)

20.9.4. クラスター化されたグルーピング

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方法が異なります。クラスターでは、特定のグループ ID を持つメッセージグループはいずれかのノードに到達できます。どのグループ ID がどのノードでどのコンシューマーにバインドされるかを判断することが重要になります。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、グループ ID を処理するコンシューマーを持つノードへメッセージグループをルーティングする必要があります。

この状況はグルーピングハンドラーによって対処されます。各ノードにグルーピングハンドラーがあり、このグルーピングハンドラー (他のハンドラーとともに) によってメッセージグループが正しいノードへルーティングされます。グルーピングハンドラーには **local** と **remote** の 2 つがあります。

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めます。リモートハンドラーはローカルハンドラーと通信し、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を持つ必要があり、他のすべてのノードはリモートハンドラーが必要になります。

以下のように local および remote グルーピングハンドラーをサーバー設定ファイル (**standalone.xml** および **domain.xml**) で設定できます。

```
<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、指定時間内にルーティングが決定されるようにします。この時間内にルーティングが決定しないと、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が、通常のクラスタールーティング条件 (ラウンドロビン方式のキューの可用性) を基にルーティングを決定します。ノードはこの決定に対応するグルーピングハンドラーへ提案し、提案が許可された場合はメッセージを提案したキューへ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、グルーピングハンドラーは他のルートを提案し、その提案に従ってルーティングが実行されます。他のノードもそれに従い、選択されたキューへ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へ固定されます。

[Report a bug](#)

20.9.5. 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ピングのベストプラクティスの一部は次のとおりです。

- 頻繁にコンシューマーを作成および閉じる場合は、コンシューマーが異なるノードで均等に分散されるようにします。キューが固定されると、カスタマーの削除に関係なく、メッセージは自動的にそのキューへ転送されます。
- 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るようにします。これにより、キューの削除後に別のノードが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。

[Report a bug](#)

20.10. 複製メッセージの検出

20.10.1. 複製メッセージの検出

複製メッセージの検出は、アプリケーション内で複製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。複製メッセージの検出は HornetQ で設定できます。

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正常に行われ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) または接続に異常が発生することがあります。このような場合、送信側 (クライアント) はメッセージが正常に受信側へ送信されたかどうかを判断するのが大変難しくなります。

メッセージが正常に送信されたかどうかは、ターゲットの受信側または接続に異常が発生したタイミングによります (メッセージの送信前または送信後)。送信側 (クライアント/サーバー) が最後のメッセージの再送信を決定した場合、アドレスに複製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。

[Report a bug](#)

20.10.2. メッセージ送信への複製メッセージ検出の使用

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、メッセージ上で特別なプロパティーを一意的な値に設定する必要があります。好きな値を使用できますが、値が一意的となる必要があります。

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にその値がある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に同じ値があるメッセージを見つけると、クライアントによって送信され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の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1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複製を検出した場合、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`** の値によって付与され、その値は **`_HQ_DUPL_ID`** になります。このプロパティーの値は、コア API のタイプ **`byte[]`** または **`SimpleString`** になります。JMS (Java Messaging Service) クライアントでは、一意的な値を持つタイプ **`String`** である必要があります。簡単に一意的な ID を生成するには、UUID を生成します。

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);

...
```


[Report a bug](#)

20.10.3. 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されています。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、キャッシュに保存された $(n + 1)$ 個目の ID が 0 個目の要素を上書きします。

ディスクへの永続化を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します。この値を `true` に設定すると、受信時に各 ID が永久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

キャッシュに保存されている以前送信されたメッセージが、再送信されたメッセージによって上書きされないようにするため、複製 ID キャッシュのサイズには大きな値を設定してください。

[Report a bug](#)

20.10.4. ブリッジおよびクラスター接続での複製検出の使用

メッセージをターゲットに転送する前に一意の複製 ID 値 (メッセージにこの値が存在しない場合) を自動的に追加するよう、コアブリッジを設定できます。複製メッセージの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 **`use-duplicate-detection`** プロパティを `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でコアブリッジを使用し、クラスターのノード間でメッセージを移動します。複製メッセージ検出の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 **`use-duplicate-detection`** プロパティを `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。

[Report a bug](#)

20.11. JMS ブリッジ

20.11.1. ブリッジ

ブリッジの機能は、ソースキューからの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ない接続を処理し、接続が再び利用可能になった場合に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して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、含まれる HornetQ が専用のバックアップとして設定されている EAP 6 サーバーにはデプロイできません。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 のライブサーバーで起動されたトランザクションをリカバーできないからです。

[Report a bug](#)

20.11.2. 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20.2 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

サーバータイプに見合った手順を使用して、JMS ブリッジをソースサーバーに設定します。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』のトピック「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 およびそれ以降のバージョンでは、JMS ブリッジを使用して JMS 1.1 に準拠する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 モジュールでバンドルされる必要があります。次の手順では、例として架空の「MyCustomMQ」メッセージブローカーが使用されています。

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. 新しいモジュール向けに **EAP_HOME/modules/system/layers/base/** 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 と **module.xml** ファイルを格納します。**EAP_HOME/modules/system/layers/base/org/mycustommq/main/** は 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になります。
- ii. **main/** サブディレクトリー内に、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 メッセージプロバイダー用に作成された **module.xml** の例は次のとおりです。

```
<?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 to the 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へコピーします。MyCustomMQ モジュールのディレクトリー構造は次のようになるはずです。

```

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** サブシステムに JMS ブリッジを設定します。

- i. 設定を行う前に、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てください。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** ファイルをバックアップします。管理対象ドメインを実行している場合は、**EAP_HOME/domain/configuration/domain.xml** ファイルおよび **EAP_HOME/domain/configuration/host.xml** ファイルを両方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** サブシステムへ追加します。**source** 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。**user** および **password** クレデンシャルが指定されいると、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用に設定された **jms-bridge** 要素の例は次のとおりです。

```

<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>
    </source>
  </jms-bridge>
</subsystem>

```

```

    <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 プロパティは **source** の **context** 要素に定義されています。上記の **target** の例のように、**context** 要素が省略されると、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。

[Report a bug](#)

20.12. 永続性

20.12.1. HornetQ の永続性

HornetQ は独自の永続性を処理します。HornetQ には、メッセージング固有のユースケースに対して最適化された高パフォーマンスな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に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を向上します。HornetQ ジャーナルは、ディスク上のファイルのセットで構成されます。これらのファイルは当初、固定サイズで事前に作成され、パディングが含まれています。サーバーの操作 (add message、delete message、update message など) が実行されると、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、満杯になると次のジャーナルファイルが使用されます。

高度なガベージコレクションアルゴリズムは、ジャーナルファイルのすべてのデータが削除されたときに、ジャーナルファイルを回収および再使用できるかどうかを決定します。圧縮アルゴリズムはジャーナルファイルからデッドスペースを削除し、データを圧縮します。

また、ジャーナルはローカルおよび XA トランザクションの両方を完全サポートします。

ジャーナルのほとんどは Java で記述されますが、さまざまなプラグ可能な実装を許可するため、ファイルシステムとの対話は抽象化されます。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。大変優れたパフォーマンスを実現し、Java 6 またはそれ以降のランタイムの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、Linux 非同期 IO ライブラリー (AIO) と対話します。データが永続化されると、HornetQ は AIO を用いてメッセージを受信します。これにより、明示的な同期化の必要がなくなります。通常、AIO のパフォーマンスは Java NIO よりも優れていますが、Linux カーネル 2.6 (またはそれ以降) と `libaio` パッケージが必要になります。

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。

- **bindings journal**

サーバーおよび属性にデプロイされたキューのセットを含む、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。メッセージジャーナルよりも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。

このジャーナル上のファイルには、`hornetq-bindings` という接頭辞が付けられます。各ファイルには `bindings` 拡張子が付けられます。ファイルサイズは 1048576 バイトで、`bindings` フォルダーに格納されます。

- **JMS journal**

JMS キュー、トピックスまたは接続ファクトリー、これらリソースの 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 リソースはすべてこのジャーナルで永続化されますが、設定ファイルで設定されたリソースは永続化されません。このジャーナルは JMS が使用される場合のみ作成されます。

- **message journal**

メッセージ自体および `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 が使用できない場合は、自動的に NIO へフォールバックします。

大型のメッセージはメッセージジャーナル外部で永続化されます。メモリー不足の場合は、ディスクへメッセージを呼び出すよう HornetQ を設定します。永続化が必要ない場合は、データを永続化しないよう HornetQ を設定できます。

[Report a bug](#)

20.13. HORNETQ クラスタリング

HornetQ クラスターは、HornetQ サーバーのグループを作成し、メッセージ処理の負荷を分配するために使用されます。クラスターのアクティブなノードは、独立した HornetQ サーバーとして動作し、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、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`) で設定パラメーターを使用して他のノードへのクラスター設定を宣言します。

クラスタリングでは、クラスター間でメッセージをブリッジング/ルーティングするため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、同じまたは別のクラスターにあるターゲット HornetQ サーバー (ノード) へそ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、内部にコアブリッジを作成します。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って、異なるクラスターにあるノード間でメッセージが送信されるときにメッセージ処理の負荷が分散されま

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、この設定を別のノードにコピーして対称のクラスターを作成できます。しかし、サーバー設定ファイルをコピーするときに注意する必要があります。HornetQ データ (バインディング、ジャーナル、大きなメッセージのディレクトリーなど) をあるノードから別のノードへコピーしてはなりません。ノードが初めて開始されると、クラスターを適切に形成するために必要となる一意な識別子をジャーナルディレクトリーへ永続化します。

[Report a bug](#)

20.13.1. サーバーディスカバリー

サーバーは「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、コネクタのリストを提供して接続を確立します。

[Report a bug](#)

20.13.2. ブロードキャストグループ

コネクタはクライアントで使用され、クライアントがサーバーへ接続する方法を定義します。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上でコネクタ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のペアのセットを取り、ネットワーク上でブロードキャストします。コネクタのペアにはライブおよびバックアップサーバーの接続設定が含まれています。

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** 要素で指定できます。1つの HornetQ サーバーは複数のブロードキャストグループを持つことができます。UDP (ユーザーデータグラムプロトコル) または JGroups のブロードキャストグループを定義できます。

[Report a bug](#)

20.13.2.1. 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 属性は廃止されました。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20.11 UDP ブロードキャストグループパラメーター

属性	説明
name attribute	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。ブロードキャストグループの名前は一意名である必要があります。
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ために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（カーネルによって無作為に生成されたアドレス）へバインドします。
local-bind-port	[廃止済み] データグラムソケットのバインド先とな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。デフォルト値は -1 で、匿名ポートの使用が指定されます。
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 までになります。IP アドレス 224.0.0 は予約済みで使用できません。

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間の時間 (ミリ秒単位) を指定します。設定は任意です。
connector-ref	ブロードキャストされるコネクタを参照します。

[Report a bug](#)

20.13.2.2. 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* の 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: この属性は、**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表します。
- ***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。

[Report a bug](#)

20.13.3. ディスカバリーグループ

ブロードキャストグループは、ネットワーク上でコネクタをブロードキャストするために使用されます。ディスカバリーグループは、コネクタの情報がどのように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) によって受信されるかを定義します。ディスカバリーグループは、各サーバーのブロードキャストごとにコネクタペアのリストを 1 つ維持します。

ディスカバリーグループが、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、特定サーバーのリストにあるコネクタペアのエントリを更新します。特定サーバーから長期間ブロードキャストを受信しないと、リストからサーバーのエントリを削除します。

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。

[Report a bug](#)

20.13.3.1. 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 属性は廃止されました。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 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20.12 UDP ディスカバリーグループパラメーター

属性	説明
name attribute	この属性はディスカバリーグループの名前を表します。ディスカバリー名はサーバーごとに一意の名前となる必要があります。
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。

属性	説明
group-address	[廃止済み] 必須の UDP 固有の属性です。グループのマルチキャスト IP アドレ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。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] 必須の 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。
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	任意の UDP 固有の属性です。最後のブロードキャストをサーバーから受信した後、削除する前にディスカバリー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。ブロードキャスト処理が継続されているときにサーバーが即座にリストから削除されないように、 refresh-timeout の値を broadcast-period の値よりもはるかに大きくする必要があります。この属性のデフォルト値は 10000 秒です。

[Report a bug](#)

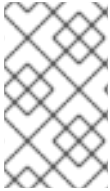
20.13.3.2. サーバー上での JGroups ディスカバリーグループの設定

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: この属性は、**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表します。
- **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 チャンネルが接続するチャンネルを表します。



注記

JGroups 属性と UDP 固有の属性は排他的です。ディスカバリーグループまたはブロードキャストグループの設定には JGroups または UDP のどちらかの属性セットを使用できます。

[Report a bug](#)

20.13.3.3. 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

ディスカバリーグループは JMS およびコアクライアント向けに設定できます。JMS 接続ファクトリーに使用するディスカバリーグループはサーバー設定ファイル (**standalone.xml** および **domain.xml**) で指定できます。

```
<connection-factory name="ConnectionFactory">
  <discovery-group-ref discovery-group-name="my-discovery-group"/>
  <entries>
    <entry name="ConnectionFactory"/>
  </entries>
</connection-factory>
```

discovery-group-ref 要素を使用してディスカバリーグループの名前を指定します。クライアントアプリケーションがこの接続ファクトリーを JNDI (Java Naming and Directory Interface) からダウンロードし、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ディスカバリーグループがリスンし維持されるすべてのサーバー全体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();
```

セッターメソッド **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。最初の接続を確立する前に接続ファクトリーが特定期間待機するようにするには、**DiscoveryGroupConfiguration** でセッターメソッド **setDiscoveryInitialWaitTimeout()** を使用します。

この設定を行うと、接続ファクトリーがクラスターのすべてのノードからブロードキャストを受信するために十分な時間が与えられます。このパラメーターのデフォルト値は 10000 ミリ秒です。

[Report a bug](#)

20.13.3.4. コア API のディスカバリー設定

コア API を使用して直接 **ClientSessionFactory** インスタンスを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();
```

セッターメソッド **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。セッションの作成前にセッションファクトリーが特定期間待機するようにするには、`DiscoveryGroupConfiguration` でセッターメソッド **setDiscoveryInitialWaitTimeout()** を使用します。

[Report a bug](#)

20.13.4. サーバー側の負荷分散

重要なクラスタートポロジは次のとおりです。

- 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にある他のすべてのノードへ直接接続されます。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、クラスターのすべてのノードがクラスター接続を定義するようにします。



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。そのため、各ノードはメッセージの負荷を分散する方法と再配布する方法を決定できます。

[Report a bug](#)

20.13.4.1. クラスター接続の設定

クラスター接続はサーバー設定ファイル (**standalone.xml** および **domain.xml**) の **cluster-connection** 要素に設定されます。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20.13 クラスター接続の設定可能な属性

属性	説明	デフォルト
<i>address</i>	各クラスター接続は、この値で始まるアドレスへ送信されたメッセージへのみ適用されます。アドレスはどの値でもよく、異なるアドレスの値を持つ多くのクラスター接続を維持し、同時にこ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へ分散します。ワイルドカードの一致は使用されません。	
<i>connector-ref</i>	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を参照する必須の属性です。	
<i>check-period</i>	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
<i>connection-ttl</i>	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
<i>min-large-message-size</i>	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
<i>call-timeout</i>	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

属性	説明	デフォルト
<i>retry-interval</i>	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、または再起動された場合、他のノードからのクラスター接続はターゲットノードが稼働するまで接続を再試行します。 <i>retry-interval</i> パラメーターは再試行の間隔 (ミリ秒単位) を定義します。	500 ミリ秒
<i>retry-interval-multiplier</i>	再試行が行われるたびに <i>retry-interval</i> をインクリメントするために使用されます。	1
<i>max-retry-interval</i>	再試行の最大遅延 (ミリ秒単位) を表します。	2000 ミリ秒
<i>reconnect-attempts</i>	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)
<i>use-duplicate-detection</i>	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ます。ブリッジを設定して転送された各メッセージに複製 ID プロパティを追加できます。ブリッジのターゲットノードがクラッシュした後にリカバリーすると、メッセージがソースノードから再送信されることがあります。複製検出を有効にすると、複製されたメッセージはフィルターされ、ターゲットノードが受信すると無視されます。	True
<i>forward-when-no-consumers</i>	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
<i>max-hops</i>	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
<i>confirmation-window-size</i>	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ィンドウのサイズ (バイト単位)。	1048576

属性	説明	デフォルト
<i>call-failover-timeout</i>	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)
<i>notification-interval</i>	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
<i>notification-attempts</i>	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
<i>discovery-group-ref</i>	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。	

クラスター接続を形成するためにクラスターのノード間で接続を作成するとき、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。

[Report a bug](#)

20.14. 高可用性

20.14.1. 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後も引き続き機能する能力をサポートします。この一部は、ライブサーバーの障害時に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ールオーバーサポートを介して実現されます。バックアップサーバーを最新状態にするため、共有ストアとレプリケーションの2つのストラテジーによって、メッセージはライブサーバーからバックアップサーバーへ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。最初のサーバーではHornetQはライブサーバーとして設定されます。2つ目のサーバーでは、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のコンテナとしてのみ動作します。このサーバーはアクティブではなく、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バックアップ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)に障害が発生すると、ただちにバックアップのHornetQサーバーが引き継ぎし、アクティブになります。フェイルバックが原因でバックアップの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および接続ファクトリーのJNDI(Java Naming and Directory Interface)へのバインドが削除されます。

JNDI(Java Naming and Directory Interface)は別のライブHornetQサーバー(別のライブ/バックアップペアの一部)と共有されます。宛先と接続ファクトリーのJNDIへ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削除されます。



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

以下の情報は **standalone-full-ha.xml** を参照します。設定の変更は、**standalone-full-ha.xml** やこのファイルから派生する設定ファイルに適用できます。

[Report a bug](#)

20.14.2. HornetQ Shared の共有ストア

共有ストアを使用する場合、ライブサーバーとバックアップサーバーの両方が共有のファイルシステムを使用して同じデータディレクトリ全体を共有します。これには、ページングディレクトリ、ジャーナルディレクトリ、大型のメッセージ、およびバインディングジャーナルが含まれます。フェー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より永続ストレージをロードします。その後、クライアントの接続が可能になります。

このような形式の高可用性はデータレプリケーションとは異なります。これは、ライブおよびバックアップノードの両方が共有ファイルシステムにアクセスできないためです。通常は、高性能な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の間でレプリケーションが発生しないことです。そのため、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ってパフォーマンスが劣化しません。

共有ストアのレプリケーションの難点は、共有ファイルシステムが必要となり、バックアップサーバーが有効になると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。ストアに格納されているデータの量によってはロードに時間がかかることがあります。

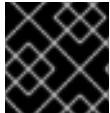
通常の操作中に最良のパフォーマンスが必要で、高速の SAN にアクセスでき、若干速度が遅いフェールオーバーを許可する (データの量に応じて) 場合は、共有ストアの高可用性が推奨されます。

[Report a bug](#)

20.14.3. HornetQ ストレージの設定

HornetQ は、共有ストアに対する 2 つの設定をサポートします。

- SAN 上の GFS2、ASYNCIO ジャーナルタイプを使用。
- NFSv4、ASYNCIO または NIO ジャーナルタイプを使用。



重要

NFS は以下の厳格な設定ガイドラインに従ってサポートされます。

Red Hat Enterprise Linux の 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) およびカーネルベースの非同期 I/O の両方をサポートします。これらの機能が両方あるため、厳格な設定ルール下で NFS を共有ストレージオプションとして使用可能です。

- Red Hat Enterprise Linux NFS クライアントのキャッシュは無効にする必要があります。



注記

上記の規定下で NFS を使用する場合、高可用性の NFS 設定を使用することが推奨されます。

[Report a bug](#)

20.14.4. 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 ととも呼ばれる Linux 非同期 IO ライブラリー (AIO) 周囲の薄い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 よりもパフォーマンスが良くなります。このジャーナルタイプは Red Hat Enterprise Linux 上でのみサポートされ、JBoss EAP 6 が実行されている場所に **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。Native Components パッケージのインストール手順については『インストールガイド』を参照してください。



重要

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るようにするため、JBoss EAP 6 が起動した後にサーバーログをチェックしてください。ネイティブライブラリーのロードに失敗した場合、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。このジャーナルタイプは大変優れたパフォーマンスを実現し、サポートされるすべてのプラットフォームで実行できます。

HornetQ ジャーナルタイプを指定するには、**<journal-type>** パラメーターを **Messaging** サブシステムに設定します。

[Report a bug](#)

20.14.5. 共有ストアを持つ専用トポロジ向けの HornetQ の設定

専用トポロジの共有ストア向けにライブおよび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="${shared.directory}/journal"/>
<bindings-directory path="${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20.14 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。デフォルト値は false です。
paging-directory path	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、パスは同じになります。
bindings-directory path	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、パスは同じになります。
journal-directory path	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、パスは同じになります。
large-messages-directory path	大型メッセージの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、パスは同じになります。
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独自の設定属性は **allow-failback** です。これは、元のライブサーバーが復旧した場合にバックアップサーバーを自動的にシャットダウンするかどうかを指定します。

[Report a bug](#)

20.14.6. 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。非永続メッセージはフェールオーバー後に維持されません。

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現されます。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る場合、2つのサーバー間ですべてのジャーナルがレプリケートされます。ライブサーバーによって受信されるすべての永続データトラフィックはバックアップサーバーへレプリケートされます。

バックアップサーバーがオンラインになると、同期するためにライブサーバーを見つけ、接続します。同期中はバックアップサーバーとして使用できません。同期するデータの量やネットワークの速度によっては、同期化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状態になり、使用できるライブサーバーがない場合、クラスターでライブサーバーが使用できるようになるまでバックアップサーバーは待機します。

データをレプリケートするようサーバーを有効にするには、**standalone-full-ha.xml** ファイルでサーバー間の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のみとレプリケートします。グループ名は、各サーバーの **standalone-full-ha.xml** ファイルにある **backup-group-name** パラメーターに定義する必要があります。

ライブサーバーに障害が発生すると、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引き継ぎます。ライブサーバーに障害が発生し、バックアップサーバーがクラスターにある他のサーバーの半分以上に接続できる場合のみバックアップサーバーは有効になります。クラスター内の他のサーバーが半分以上が応答しない場合は一般的なネットワーク障害と判断され、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ため待機します。

フェー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。その後、元のライブサーバーが再度有効な状態になるよう、バックアップサーバーをシャットダウンできます。**allow-failback** 属性が **true** に設定されていると、これが自動的に行われます。

[Report a bug](#)

20.14.7. 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ションの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>

```

表20.15 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。デフォルト値は 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	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チェックするため、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確認するかどうか。
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。デフォルト値は 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20.16 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。デフォルト値は true です。
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 である場合のみこの属性が必要となります。デフォルト値は 2 で、ライブサーバーからジャーナルをレプリケートし、再度バックアップとなるようにするため、2 回目のフェイルバック後にバックアップサーバーが再起動されます。

[Report a bug](#)

20.14.8. 高可用性 (HA) フェールオーバー

高可用性フェールオーバーは、ライブバックアップの仕組みにより、自動クライアントフェールオーバーまたはアプリケーションレベルフェールオーバーのいずれかを用いて利用できます。各ライブサーバーはバックアップサーバーを持ちます。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、フェールオーバーが発生した場合のみ引き継ぎします。**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起動した後に再度ライブサーバーとして稼働します。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ションするため HA クラスターの各ノードは他のノードすべてへのクラスター接続を持つ必要があるためです。

高可用性クラスタートポロジは、ライブおよびバックアップサーバーが IP マルチキャストを使用して接続の詳細情報を送信することで実現されます。IP マルチキャストが使用できない場合、最初の接続の静的接続を使用することも可能です。最初の接続後、クライアントはトポロジの情報を受け取ります。現在の接続が陳腐化すると、クライアントは別のノードへ新しい接続を確立します。

ライブサーバーに障害が発生し、バックアップサーバーが引き継ぎした後、ライブサーバーを再起動し、クライアントをフェイルバックする必要があります。これを行うには、元のライブサーバーを再起動し、新しいライブサーバーをキルします。これには、プロセス自体をキルするか、サーバー自体がクラッシュするまで待ちます。また、**standalone.xml** 設定ファイルで **failover-on-shutdown** プロパティを `true` に設定すると、通常のサーバーのシャットダウンでフェールオーバーが発生するようになります。

```
<failover-on-shutdown>true</failover-on-shutdown>
```

デフォルトでは、**failover-on-shutdown** プロパティは `false` に設定されています。

また、**standalone.xml** 設定ファイルで **allow-failback** プロパティを `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のシャットダウンが強制され、元のライブサーバーへ自動的に引き継ぎされます。

```
<allow-failback>true</allow-failback>
```

レプリケーション HA モードで、元のライブサーバーが復旧したときに新しいサーバーのシャットダウンを強制するには、**standalone.xml** 設定ファイルで **check-for-live-server** プロパティを `true` に設定します。

```
<check-for-live-server>true</check-for-live-server>
```

[Report a bug](#)

20.14.9. 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ている 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てはなりません。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)

などのデプロイメントやサーブレットが含まれます。

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。

[Report a bug](#)

第21章 TRANSACTION サブシステム

21.1. トランザクションサブシステムの設定

21.1.1. 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」](#)

[Report a bug](#)

21.1.2. トランザクションマネージャーの設定

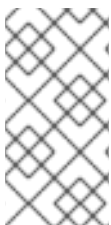
トランザクションマネージャー (TM) は、Web ベースの管理コンソールまたはコマンドラインの管理 CLI を使用して設定できます。各コマンドやオプションでは、JBoss EAP 6 を管理対象ドメインとして実行していると仮定します。スタンドアロンサーバーを使用する場合や **default** 以外のプロファイルを修正したい場合は、以下の方法で手順とコマンドを修正する必要があることがあります。

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のコンソールログイン時に選択されるものです。異なる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修正する必要がある場合は、**default** の代わりに使用しているプロファイルを選択してください。

同様に、例の CLI コマンドの **default** プロファイルを使用しているプロファイルに置き換えてください。

- スタンドアロンサーバーを使用する場合、存在するプロファイルは1つのみです。特定のプロファイルを選択する手順は無視してください。CLI コマンドでは、例のコマンドの **/profile=default** 部分を削除してください。



注記

TM オプションが管理コンソールまたは管理 CLI で表示されるようにするには、**transactions** サブシステムが有効でなくてはなりません。これは、デフォルトで有効になっており、他の多くのサブシステムが適切に機能するために必要なため、無効にする可能性は大変低くなります。

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、左上にある **Profile** 選択ボックスから適切なプロファイルを選択します。**Container** メニューを展開して、**Transactions** を選択します。

トランザクションマネージャーの設定ページには、さらなるオプションが表示されます。**Recovery** オ

プションはデフォルトでは表示されません。**Recovery** ヘッダーをクリックしてリカバリーのオプションを表示します。オプションを編集するには、**Edit** ボタンをクリックします。変更は直ちに反映されます。

インラインヘルプを表示するには、**Need Help?** ラベルをクリックします。

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。プロファイル **default** の管理対象ドメインの場合、コマンドはすべて **/profile=default/subsystem=transactions/** で始まり、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

HornetQ では、複数のインスタンスはメッセージログストアを共有できません。HornetQ のインスタンスを複数設定する場合、インスタンスごとに独自のメッセージストアが必要になります。

表21.1 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を指定します。統計は Runtime タブの Subsystem Metrics セクションにある管理コンソールで閲覧できます。	/profile=default/subsystem=transactions/:write-attribute(name=enable-statistics,value=true)
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です。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、 jboss.server.data.dir プロパティーの値です。デフォルト値は、管理対象ドメインの場合は EAP_HOME/domain/data/ 、スタンドアロンサーバーインスタンスの場合は EAP_HOME/standalone/data/ です。オブジェクトストアの 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を使用する場合に、トランザクションマネージャーの回復およびトランザクション識別子の生成に使用するソケットバインディングの名前を指定します。一意の識別子を生成する詳しい情報は、 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。	<code>/profile=default/subsystem=transactions/:write-attribute(name=socket-binding,value=txn-recovery-environment)</code>
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スンするかどうかを指定します。デフォルトは false です。	<code>/profile=default/subsystem=transactions/:write-attribute(name=recovery-listener,value=false)</code>

以下は、高度なオプションで、管理 CLI を用いて変更する必要があります。デフォルト設定の変更は注意して行ってください。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21.2 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。	<code>/profile=default/subsystem=transactions/:write-attribute(name=jts,value=false)</code>

オプション	説明	CLI コマンド
node-identifier	<p>トランザクションマネージャーのノード識別子です。このオプションは以下の場合に必要です。</p> <ul style="list-style-type: none"> ● 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 <p>リカバリー中にデータの整合性を強制するのに必要なため、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。</p>	/profile=default/subsystem=transactions/:write-attribute(name=node-identifier,value=1)
process-id-socket-max-ports	<p>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。</p> <p>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用に使用されます。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 が失敗するまでに試行するソケットの最大値を意味します。デフォルト値は 10 です。</p>	/profile=default/subsystem=transactions/:write-attribute(name=process-id-socket-max-ports,value=10)
process-id-uuid	<p>true に設定すると、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。</p>	/profile=default/subsystem=transactions/:write-attribute(name=process-id-uuid,value=true)

オプション	説明	CLI コマンド
use-hornetq-store	トランザクションログ用に、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 パフォーマンスが向上します。別々のトランザクションマネージャーで JTS トランザクションを使用することは推奨されません。このオプションの変更を反映するには shutdown コマンドを使用してサーバーを再起動する必要があります。	<code>/profile=default/subsystem=transactions/:write-attribute(name=use-hornetq-store,value=false)</code>

[Report a bug](#)

21.1.3. 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、データソースまたはリソースの文書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21.1 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*** にあります。Microsoft Windows サーバーの場合は ***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の設定ファイルは、Red Hat Enterprise Linux の場合は ***EAP_HOME/standalone/configuration/standalone.xml*** にあります。Microsoft Windows サーバーの場合は ***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>` タグを探します。

このタスクを完了するには、[JBoss EAP 6 の管理インターフェース](#) の [データソースの作成](#) ページを参照してください。

データソースの **jndi-name** 属性には作成時に指定した属性が設定されます。たとえば、ExampleDS データソースは次のようになります。

```
<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。
上記のように、**jta="true"** を **<datasource>** タグの内容に追加します。
4. **設定ファイルを保存**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。

[Report a bug](#)

21.1.4. XA Datasource の設定

前提条件

XA Datasource を追加するには、管理コンソールにログインする必要があります。詳細については、「[管理コンソールへのログイン](#)」を参照してください。

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。「[管理インターフェースによる非 XA データソースの作成](#)」の手順に従いますが、上部の **XA Datasource** タブをクリックしてください。
2. **必要に応じて他のプロパティを設定**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](#)」にリストされています。

結果

XA Datasource が設定され、使用する準備ができます。

[Report a bug](#)

21.1.5. 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。詳細なデバッグでは **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については、「[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。一般的に表示されるメッセージの一部は次のとおりです。他のメッセージが表示されることもあります。

表21.3 トランザクション状態の変更

トランザクションの開始	<p>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。</p> <pre>com.arjuna.ats.arjuna.coordinator.BasicAction: :Begin:1342 tsLogger.logger.trace("BasicAction::Begin() for action-id "+ get_uid());</pre>
トランザクションのコミット	<p>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。</p> <pre>com.arjuna.ats.arjuna.coordinator.BasicAction: :End:1342 tsLogger.logger.trace("BasicAction::End() for action-id "+ get_uid());</pre>
トランザクションのロールバック	<p>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。</p> <pre>com.arjuna.ats.arjuna.coordinator.BasicAction: :Abort:1575 tsLogger.logger.trace("BasicAction::Abort() for action-id "+ get_uid());</pre>
トランザクションのタイムアウト	<pre>com.arjuna.ats.arjuna.coordinator.Transaction Reaper::doCancellations:349 tsLogger.logger.trace("Reaper Worker " + Thread.currentThread() + " attempting to cancel " + e._control.get_uid());</pre> <p>その後、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確認できます。</p>

[Report a bug](#)

21.1.6. トランザクションサブシステムのログ設定

概要

JBoss EAP 6 の他のログ設定に依存せずにトランザクションログの情報量を制御する手順を説明します。主に Web ベースの管理コンソールを用いた手順を説明し、管理 CLI コマンドはその後で説明します。

手順21.2 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したいサーバープロファイルを選択します。

Core メニューを展開して、**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** セクションの **Edit** をクリックします。ここにクラス固有のログ情報を追加できます。**com.arjuna** クラスはすでに存在します。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。

ログのレベル

デフォルトのログレベルは **WARN** です。トランザクションはログを大量に出力できるため、標準的なログレベルの意味は、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重要度が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

ロガーがログ出力を親ロガーに送信するかどうかを指定します。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。

[Report a bug](#)

21.2. トランザクション管理

21.2.1. 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 と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、待機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** と呼ばれる API 操作はトランザクションログを読み取り、各ログに対してノードを作成します。**probe** コマンドは、**log-store** を更新する必要があるときに、いつでも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表示された後すぐに消去されるのは普通です。

■

例21.1 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21.2 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21.1「[ログストアの更新](#)」を参照)、ファイルシステムの **ls** コマンドに似た機能を持つ次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的 ID 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** (参加者) と呼ばれる子要素が含まれます。トランザクションの参加者を確認するには、この要素に対して **read-resource** CLI コマンドを使用します。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ステータスは **HEURISTIC** であり、復元可能です。詳細については、[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。

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ログは、**:recover** CLI コマンドを使用したリカバリーをサポートします。

ヒューリスティックなトランザクションと参加者のリカバリー

- トランザクションの状態が **HEURISTIC** である場合は、リカバリー操作によって、状態が **PREPARE** に変わり、リカバリーがトリガーされます。
- 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** 操作を再実行しようとします。成功した場合、トランザクションログから参加者が削除されます。これを確認するには、**log-store** 上で **:probe** 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。これが最後の参加者の場合は、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、トランザクションのリカバリーが必要である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。

統計は、管理コンソールまたは管理 CLI より表示できます。管理コンソールでは、トランザクション統計情報は **Runtime** → **Status** → **Subsystems** → **Transactions** を選択して取得できます。管理対象ドメインの各サーバーのトランザクション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、左側のメニューにある **Change Server** を選択し、そのリストからサーバーを選択します。

以下の表は、利用可能な各統計情報、その説明、および統計情報を表示する 管理 CLI コマンドを示しています。

表21.4 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
----	----	----------

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。	<pre>/host=master/server=server - one/subsystem=transactions/ :read- attribute(name=number-of- transactions,include- defaults=true)</pre>
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。	<pre>/host=master/server=server - one/subsystem=transactions/ :read- attribute(name=number-of- committed- transactions,include- defaults=true)</pre>
Aborted	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。	<pre>/host=master/server=server - one/subsystem=transactions/ :read- attribute(name=number-of- aborted- transactions,include- defaults=true)</pre>
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。	<pre>/host=master/server=server - one/subsystem=transactions/ :read- attribute(name=number-of- timed-out- transactions,include- defaults=true)</pre>
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。	<pre>/host=master/server=server - one/subsystem=transactions/ :read- attribute(name=number-of- heuristics,include- defaults=true)</pre>

統計	説明	CLI コマンド
フライト状態のトランザクション	管理コンソールでは使用できません。開始した未終了のトランザクションの数。	<pre>/host=master/server=server - one/subsystem=transactions/ :read- attribute(name=number-of- inflight-transactions,include- defaults=true)</pre>
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。	<pre>/host=master/server=server - one/subsystem=transactions/ :read- attribute(name=number-of- application- rollbacks,include- defaults=true)</pre>
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。	<pre>/host=master/server=server - one/subsystem=transactions/ :read- attribute(name=number-of- resource-rollbacks,include- defaults=true)</pre>
参加者 ID	参加者の ID。	<pre>/host=master/server=server - one/subsystem=transactions/ log-store=log- store/transactions=0\::ffff7f00 0001\:- b66efc2\:4f9e6f8f\:9:read- children-names(child- type=participants)</pre>
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。	<pre>/host=master/server=server - one/subsystem=transactions/ log-store=log-store:read- children-names(child- type=transactions)</pre>

[Report a bug](#)

21.3. トランザクションに関するリファレンス

21.3.1. 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については、<http://docs.oracle.com/javaee/6/api/javax/transaction/UserTransaction.html> の『UserTransaction API』の仕様を参照してください。

[Report a bug](#)

21.3.2. JTA トランザクションの制限

JTA トランザクションは、複数の JBoss EAP 6 インスタンス全体でディストリビューションに対応できません。そのため、JTS トランザクションを使用します。

JTS トランザクションを使用するには、JacORB サブシステムでのトランザクションの有効化が含まれる ORB を設定し、JTS サブシステムを設定する必要があります。

- 「JTS トランザクション用 ORB の設定」

[Report a bug](#)

21.4. ORB 設定

21.4.1. 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することを可能にする標準です。CORBA 要求は *Object Request Broker (ORB)* というサーバーサイドコンポーネントにより JacORB コンポーネントを使用して処理されます。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、ユーザー独自のアプリケーションが使用することもできます。

[Report a bug](#)

21.4.2. 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。ORB は、コマンドライン管理 CLI を使用して有効にすることができます。



注記

管理対象ドメインでは、JacORB サブシステムが **full** および **full-ha** プロファイルでのみ利用可能です。スタンドアロンサーバーでは、**standalone-full.xml** または **standalone-full-ha.xml** 設定で利用可能です。

手順21.3 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、左上にある選択ボックスから **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 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 用 ORB を有効にするには、**Transaction Interceptors** 値をデフォルトの **spec** ではなく **on** に設定します。

これらの値に関する詳細な説明については、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、パラメーターに関する詳細な情報とともに **Need Help?** リンクが含まれます。

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、管理コンソールに対するイニシャライザーに上記の手順と同じ値を設定します。これは、JTS と使用する ORB の最小設定です。

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、設定する必要がある管理対象ドメインに合わせてプロファイルを変更します。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** 部分を省略します。

例21.3 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21.4 JacORB サブシステムでのトラザクションの有効化

```
/profile=full/subsystem=jacorb/:write-attribute(name=transactions,value=on)
```

例21.5 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。

[Report a bug](#)

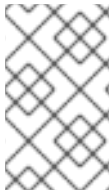
21.5. JDBC オブジェクトストアのサポート

21.5.1. トランザクションの JDBC ストア

必読トピック:

- 「[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 データストアをオブジェクトストアとして使用できます。使用されるデータベースがフェールオーバーおよび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のディスクスペースを使用するよりも JDBC データストアをオブジェクトストアとして使用した方がよいことがあります。raw JDBC オブジェクトストアは特別なオブジェクトストアで、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よりもパフォーマンスが劣るため、使用の長所と短所を比較する必要があります。



注記

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する場合、サーバーの設定ファイルの **datasource** セクションに **jta="false"** を指定する必要があります。

手順21.4 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21.5 トランザクション JDBC ストアのプロパティ

プロパティ	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。

プロパティ	説明
jdbc-action-store-drop-table	起動時にアク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。任意のプロパティで、デフォルト値は 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。任意のプロパティです。
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。任意のプロパティで、デフォルト値は false です。
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。任意のプロパティです。
jdbc-state-store-drop-table	起動時にステートストアテーブルがドロップおよび再作成されます。任意のプロパティで、デフォルト値は false です。
jdbc-state-store-table-prefix	ステートストアテーブル名の接頭辞。任意のプロパティです。

関連トピック:

- [「JTA Transaction API を使用するようデータソースを設定」](#)

[Report a bug](#)

第22章 メールサブシステム

22.1. 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** です。この属性は、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、ホスト設定に複数のホストが使用されるため、ユーザーによっては最も効率的な方法ではありません。この設定は標準的な JavaMail によってサポートされないため、カスタムのメールトランスポートを実装する必要がある場合があります。

このような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使用できます。

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22.1

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** を示す **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. **outbound-socket-binding-ref** を用いて、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際、ピリオド (.) が含まれるプロパティ名は完全修飾名とみなされ、指定どおりに渡されます。他の形式 (*my-property* など) 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>
```

[Report a bug](#)

第23章 ENTERPRISE JAVABEANS

23.1. はじめに

23.1.1. Enterprise JavaBeans の概要

Enterprise JavaBeans (EJB) 3.1 は、エンタープライズ Bean と呼ばれるサーバーサイドコンポーネントを使用してセキュアでポータブルな分散 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、再利用を促進する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します。Enterprise JavaBeans 3.1 は、Java EE 仕様 JSR-318 としてドキュメント化されています。

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。

[Report a bug](#)

23.1.2. 管理者向け Enterprise JavaBeans の概要

JBoss 管理者は、JBoss EAP 6 で Enterprise 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に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。XML サーバー設定ファイルを編集して変更を適用することは可能ですが、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. ドメインモードで稼働している場合は、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。

[Report a bug](#)

23.1.3. エンタープライズ Bean

Enterprise JavaBeans (EJB) 3.1 仕様、JSR-318 に定義されているように、エンタープライズ Bean はサーバー側のアプリケーションコンポーネントのことです。エンタープライズ Bean は疎結合方式でアプリケーションのビジネスロジックを実装し、再利用ができ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。アプリケーションサーバーに独自のアーカイブ (JAR ファイル) でデプロイするか、Java EE アプリケーションの一部としてデプロイすることが可能です。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、並行処理管理などのサービスを提供します。

エンタープライズ Bean は、ビジネスインターフェースをいくつでも定義できます。ビジネスインターフェースは、クライアントが使用できる Bean のメソッドに対して優れた制御機能を提供し、リモート JVM で実行され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 Bean の 3 種類があります。



重要

エンティティ Bean は EJB 3.1 で廃止されました。Red Hat は代わりに JPA エンティティの使用を推奨します。Red Hat はレガシーシステムで後方互換性に対応する場合のみエンティティ Bean の使用を推奨します。

[Report a bug](#)

23.1.4. セッション Bean

セッション Bean は、関連の業務プロセスやタスクのセットをカプセル化し、要求したクラスにインジェクトするエンタープライズ Bean です。セッション Bean には、ステートレス、ステートフル、シングルトンの 3 種類があります。

[Report a bug](#)

23.1.5. メッセージ駆動型 Bean

メッセージ駆動型 Bean (MDB) は、アプリケーション開発に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されませんが、Java Messaging Service (JMS) サーバーなどのメッセージングサービスからメッセージを受け取ることによってトリガーされます。Java EE 6 仕様では JMS がサポートされている必要がありますが、他のメッセージングシステムをサポートすることもできます。

[Report a bug](#)

23.2. BEAN プールの設定

23.2.1. Bean プール

JBoss EAP 6 はさらに高速なパフォーマンスを提供するため、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で保持します。この技術は Bean プーリングと呼ばれます。Bean が必要なとき、アプリケーションサーバーは新しい Bean をインスタンス化せずに、すでに存在する Bean の適切なプールから Bean を 1 つ取ります。Bean が不要になったら、再使用するために Bean をプールに戻します。

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。

[Report a bug](#)

23.2.2. 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。

Bean プールは、テキストエディターを用いて必要な Bean プール設定をサーバー設定ファイルに追加して作成することもできます。[例23.2「XML 設定の例」](#) は設定例になります。

手順23.1 管理コンソールを使用した Bean プールの作成

1. 管理コンソールにログインします。「[管理コンソールへのログイン](#)」を参照してください。
2. 画面上部にある **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、**Name**、**Max Pool Size**、**Timeout** の値、および **Timeout** の単位を指定します。
5. **Save** ボタンをクリックして終了します。

手順23.2 CLI を使用した Bean プールの作成

1. CLI ツールを起動して、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 **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 プールの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* を置き換えます。
- *UNIT* を必要な時間単位に置き換えます。使用可能な値は **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23.1 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"}
[standalone@localhost:9999 /]
```

例23.2 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>

```

[Report a bug](#)

23.2.3. 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。

前提条件:

- 使用中の Bean プールを削除することはできません。「[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、Bean プールが使用されていないことを確認してください。

手順23.3 管理コンソールを使用した Bean プールの削除

1. 管理コンソールにログインします。「[管理コンソールへのログイン](#)」を参照してください。
2. 画面上部にある **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. **Confirm** をクリックして確認します。

手順23.4 CLI を使用した Bean プールの削除

1. CLI ツールを起動して、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 **remove** 操作を使用します。

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:remove
```

- *BEANPOOLNAME* を Bean プールの必要な名前に置き換えます。

例23.3 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"}
[standalone@localhost:9999 /]

```

[Report a bug](#)

23.2.4. 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23.5 管理コンソールを使用した Bean プールの編集

1. 管理コンソールにログインします (「[管理コンソールへのログイン](#)」)。
2. 画面上部にある **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. **編集** 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** の値、および **Timeout Unit** が変更可能です。
6. **Save** をクリックして終了します。

手順23.6 CLI を使用した Bean プールの編集

1. CLI ツールを起動して、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 プールの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。このように編集できる属性は、 **max-pool-size**、 **timeout**、 および **timeout-unit** です。
 - *VALUE* を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 プールの変更を確認します。

```
/subsystem=ejb3/strict-max-bean-instance-pool=BEANPOOLNAME:read-resource
```

例23.4 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"}
[standalone@localhost:9999 /]
```

[Report a bug](#)

23.2.5. 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。Bean プールは、管理コンソールまたは CLI ツールを使用して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 に対する **slsb-strict-max-pool** とメッセージ駆動型 Bean に対する **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場合は「[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。

手順23.7 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. 管理コンソールにログインします (「[管理コンソールへのログイン](#)」)。
2. 画面上部にある **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. **編集** 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23.8 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 ツールを起動して、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** または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23.5 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"}
[standalone@localhost:9999 /]
```

例23.6 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>

```

[Report a bug](#)

23.3. EJB スレッドプールの設定

23.3.1. 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。

この技術はスレッドプーリングと呼ばれ、スレッド作成のオーバーヘッドを削減して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。

[Report a bug](#)

23.3.2. 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23.9 管理コンソールを使用した EJB スレッドプールの作成

1. 管理コンソールにログインします (「[管理コンソールへのログイン](#)」)。
2. 画面上部にある **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. **Add** クリックします。 **Add EJB3 Thread Pools** ダイアログが表示されます。
4. 必要な詳細、 **Name**、 **Max. Threads**、 **Keep-Alive Timeout** の値を指定します。
5. **Save** をクリックして終了します。

手順23.10 CLI を使用したスレッドプールの作成

1. CLI ツールを起動して、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 **add** 操作を使用します。

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:add(max-threads=MAXSIZE, keepalive-time={"time"=>"TIME", "unit"=>"UNIT"})
```

- *THREADPOOLNAME* をスレッドプールの必要な名前に置き換えます。
- *MAXSIZE* をスレッドプールの最大サイズに置き換えます。
- *UNIT* を必要な keep-alive 時間に使用される時間単位に置き換えます。使用可能な値は **NANOSECONDS**、**MICROSECONDS**、**MILLISECONDS**、**SECONDS**、**MINUTES**、**HOURS**、**DAYS** です。
- *TIME* を必要な keep-alive 時間の整数値に置き換えます。この値は *UNIT* の数になります。

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/strict-max-bean-instance-pool=THREADPOOLNAME:read-resource
```

例23.7 CLI を使用したスレッドプールの作成

```
[standalone@localhost:9999 /] /subsystem=ejb3/thread-pool=testmepool:add(max-threads=50,
keepalive-time={"time"=>"150", "unit"=>"SECONDS"})
{"outcome" => "success"}
[standalone@localhost:9999 /]
```

例23.8 XML 設定の例

```
<subsystem xmlns="urn:jboss:domain:ejb3:1.2">

  <thread-pools>
    <thread-pool name="default" max-threads="20" keepalive-time="150"/>
  </thread-pools>

</subsystem>
```

[Report a bug](#)

23.3.3. 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

- 使用されているスレッドプールは削除できません。次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てください。
 - [「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23.11 管理コンソールを使用した EJB スレッドプールの削除

1. 管理コンソールにログインします (「[管理コンソールへのログイン](#)」)。
2. 画面上部にある **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. **OK** をクリックして確定します。

手順23.12 CLI を使用したスレッドプールの削除

1. CLI ツールを起動して、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 **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23.9 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"}
[standalone@localhost:9999 /]
```

[Report a bug](#)

23.3.4. 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23.13 管理コンソールを使用したスレッドプールの編集

1. 管理コンソールにログインします (「[管理コンソールへのログイン](#)」)。
2. 画面上部にある **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. **編集** をクリックします。
5. 変更したい詳細を編集します。**Thread Factory**、**Max Threads**、**Keepalive Timeout**、**Keepalive Timeout Unit** の値のみが編集可能です。

6. **Save** をクリックして終了します。

手順23.14 CLI を使用したスレッドプールの編集

1. CLI ツールを起動して、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。このように編集できる属性は、**keepalive-time**、**max-threads**、および **thread-factory** です。
 - *VALUE* を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。構文は次のとおりです。

```
/subsystem=ejb3/thread-pool=THREADPOOLNAME:write-attribute(name="keepalive-
time", value={"time" => "VALUE", "unit" => "UNIT"})
```

例23.10 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"}
[standalone@localhost:9999 /]
```

例23.11 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"}
[standalone@localhost:9999 /]
```

[Report a bug](#)

23.4. セッション BEAN の設定

23.4.1. 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アクセスタイムアウトの値が指定されています。この値は、タイムアウトする前にセッション Bean メソッドへの要求をブロックできる期間になります。

メソッド上に `@javax.ejb.AccessTimeout` アノテーションを用いて、使用するタイムアウトの値と時間の単位を指定できます。セッション Bean (すべての Bean のメソッドに適用) や特定のメソッドに指定し、Bean の設定を上書きできます。

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。

<http://docs.oracle.com/javaee/6/api/javax/ejb/AccessTimeout.html> にある `AccessTimeout` の Javadocs を参照してください。

[Report a bug](#)

23.4.2. 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。管理コンソールまたは CLI を使用するとデフォルトのタイムアウト値を変更できます。デフォルト値は 5000 ミリ秒です。

手順23.15 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. 管理コンソールにログインします。「[管理コンソールへのログイン](#)」を参照してください。
2. 画面上部にある **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** のフィールドが編集可能になります。
4. **Stateful Access Timeout** または **Singleton Access Timeout** 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23.16 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 ツールを起動して、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** または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23.12 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"}
[standalone@localhost:9999 /]
```

例23.13 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>
```

[Report a bug](#)

23.5. メッセージ駆動型 BEAN の設定

23.5.1. 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 の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することができます。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 のデフォルトのリソースアダプターは **hornetq-ra** です。

手順23.17 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. 管理コンソールにログインします (「[管理コンソールへのログイン](#)」)。
2. 画面上部にある **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** のフィールドが編集可能になります。
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23.18 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 ツールを起動して、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 を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23.14 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23.15 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>
```

[Report a bug](#)

23.6. EJB3 タイマーサービスの設定

23.6.1. 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 より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、指定の時間にコールバックのメソッドをスケジュールできます。メソッドのコールバックは、特定時、一定期間の後、繰り返しの間隔、またはカレンダーのスケジュールでの発生が可能です。

[Report a bug](#)

23.6.2. EJB3 タイマーサービスの設定

JBoss の管理者は JBoss EAP 6 の管理コンソールに EJB3 タイマーサービスを設定できます。設定可能な機能は、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、タイマーサービスデータが保存されるディレクトリーです。

手順23.19 EJB3 タイマーサービスの設定

1. 管理コンソールにログインします。「[管理コンソールへのログイン](#)」を参照してください。
2. 画面上部にある **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 Services** をクリックします。
3. **Edit** をクリックします。**Details** のフィールドが編集可能になります。
4. 追加のスレッドプールが設定されている場合はタイマーサービスに使用される別の EJB3 スレッドプールを選択し、タイマーサービスデータの保存に使用されるディレクトリを変更することができます。タイマーサービスデータディレクトリの設定は、データが保存されるディレクトリである **Path** と、**Path** が含まれるディレクトリである **Relative To** の2つの値で構成されます。デフォルトでは **Relative To** はファイルシステムの Path 変数に設定されています。
5. **Save** をクリックして終了します。

[Report a bug](#)

23.7. EJB3 非同期呼び出しサービスの設定

23.7.1. 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スレッド数 (スレッドプール) を維持管理します。

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。

[Report a bug](#)

23.7.2. 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23.20 EJB3 非同期呼び出しサービスのスレッドプールの設定

1. 管理コンソールにログインします。「[管理コンソールへのログイン](#)」を参照してください。
2. 画面上部にある **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. **編集** をクリックします。
4. 使用する EJB3 スレッドプールを一覧より選択します。スレッドプールがすでに作成済みである必要があります。
5. **Save** をクリックして終了します。

[Report a bug](#)

23.8. EJB3 リモート呼び出しサービスの設定

23.8.1. 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。

[Report a bug](#)

23.8.2. EJB3 リモートサービスの設定

JBoss の管理者は JBoss EAP 6 の管理コンソールに EJB3 リモートサービスを設定できます。設定可能な機能は、リモートの Bean 呼び出しに使用されるスレッドプールと、EJB3 リモートリングチャンネルが登録されるコネクタです。

手順23.21 EJB3 リモートサービスの設定

1. 管理コンソールにログインします。「[管理コンソールへのログイン](#)」を参照してください。
2. 画面上部にある **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. **編集** 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リモートサービスに使用される別の EJB3 スレッドプールを選択することができます。EJB リモートリングチャンネルの登録に使用されるコネクタを変更できます。
5. **Save** をクリックして終了します。

[Report a bug](#)

23.9. EJB 2.X エンティティ Bean の設定

23.9.1. EJB エンティティ Bean

EJB エンティティ Bean はEJB仕様のバージョン 2.x からのエンタープライズ bean の一種で、データベースで保持された永続データを表します。エンティティ Bean より JPA エンティティが優先され、今後の仕様バージョンから削除 (プルーニング) される公式一覧に挙げられています。Red Hat は後方互換性を維持する以外の目的でエンティティ Bean を使用することを推奨しません。

JBoss EAP 6 ではエンティティ Bean のサポートはデフォルトで無効になっています。

[Report a bug](#)

23.9.2. 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 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。

[Report a bug](#)

23.9.3. 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や **standalone-full.xml** などのスタンドアロンサーバーの完全設定ではデフォルトで有効になっています。

他の設定で 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>
```

拡張が追加されたら、<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"/>
```

サーバー設定の 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。

[Report a bug](#)

23.9.4. EJB 2.x のコンテナ管理による永続性の設定

EJB 2.x のコンテナ管理による永続性 (CMP) サブシステムが任意の数のキージェネレーターを指定するよう設定できます。キージェネレーターは、CMP サービスにより永続化された各エンティティを識別する一意のキー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、UUID を使用してキーを生成します。UUID キージェネレーターは、一意の名前のみを持つ必要があり、他の設定は持ちません。

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23.16 UUID キージェネレーターの追加

名前が **uuid_identities** の 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 ID キーを作成および格納します。HiLo キージェネレーターは、一意の名前を持つ必要があり、データと、キーを格納するテーブルおよび列の名前を格納するために使用されるデータソースを指定するプロパティで設定されます。

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23.17 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、エンティティの作成に失敗します。

注記

一貫性を保つため、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キング機能をサポートしません。

23.9.5. HiLo キージェネレーター用の CMP サブシステムプロパティ

表23.1 HiLo キージェネレーター用の CMP サブシステムプロパティ

プロパティ	データ型	説明
block-size	long	ブロックサイズ。
create-table	boolean	TRUE に設定すると、 table-name というテーブルが見つからない場合に create-table-ddl の内容を使用して table-name テーブルが作成されます。
create-table-ddl	文字列	table-name で指定されたテーブルが見つからず、かつ create-table が TRUE に設定されている場合に、その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	文字列	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。
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。

[Report a bug](#)

第24章 JAVA CONNECTOR ARCHITECTURE (JCA)

24.1. はじめに

24.1.1. Java EE Connector API (JCA)

JBoss EAP 6 は、Java EE Connector API (JCA) 1.6 仕様の完全なサポートを提供します。JCA 仕様の詳細については、[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。リソースアダプターはデータソースオブジェクトと似ていますが、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 (ERP) システムなどの幅広い異種システムへエンタープライズ情報システム (EIS) から接続を提供します。

[Report a bug](#)

24.1.2. 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 (ERP) システム、メインフレーム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 (<http://jcp.org/en/jsr/detail?id=313>) として開発されました。

[Report a bug](#)

24.1.3. 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。

[Report a bug](#)

24.2. 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24.1 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。

- アーカイブ検証が有効な状態で、アーカイブが Java EE Connector Architecture 仕様を正しく実装しない場合、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。例は次のとおりです。

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

- アーカイブ検証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 の検証

- 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24.2 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 検証が指定されていない場合は、Bean 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。

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24.3 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。カスタムワークマネージャーは必須になります。
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24.4 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。

属性	説明
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より小さくなければなりません。
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24.5 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24.6 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24.1 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、サーバーがどのように実行されているかに応じて、管理コンソールの若干異なる場所に存在します。JCA 設

定オプションは、どのようにサーバーが実行されているかに応じて、管理コンソールでの場所が若干異なります。

1. 画面上部にある **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 **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. 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、キャッシュ済み接続マネージャー、アーカイブ検証、および Bean 検証 (JSR-303) の設定が含まれます。また、これらの各設定は独自のタブに含まれます。これらの設定を変更するには、適切なタブを開き、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。

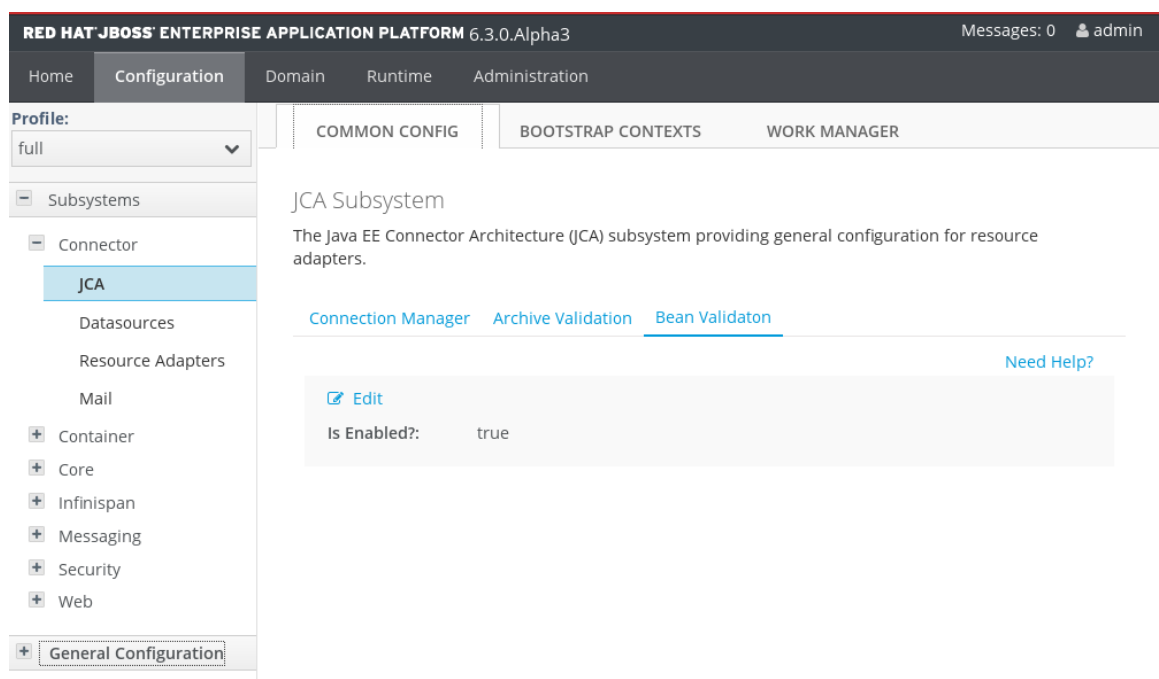


図24.1 JCA 共通設定

b. ワークマネージャー

Work Manager タブには、設定され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。各ワークマネージャー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。

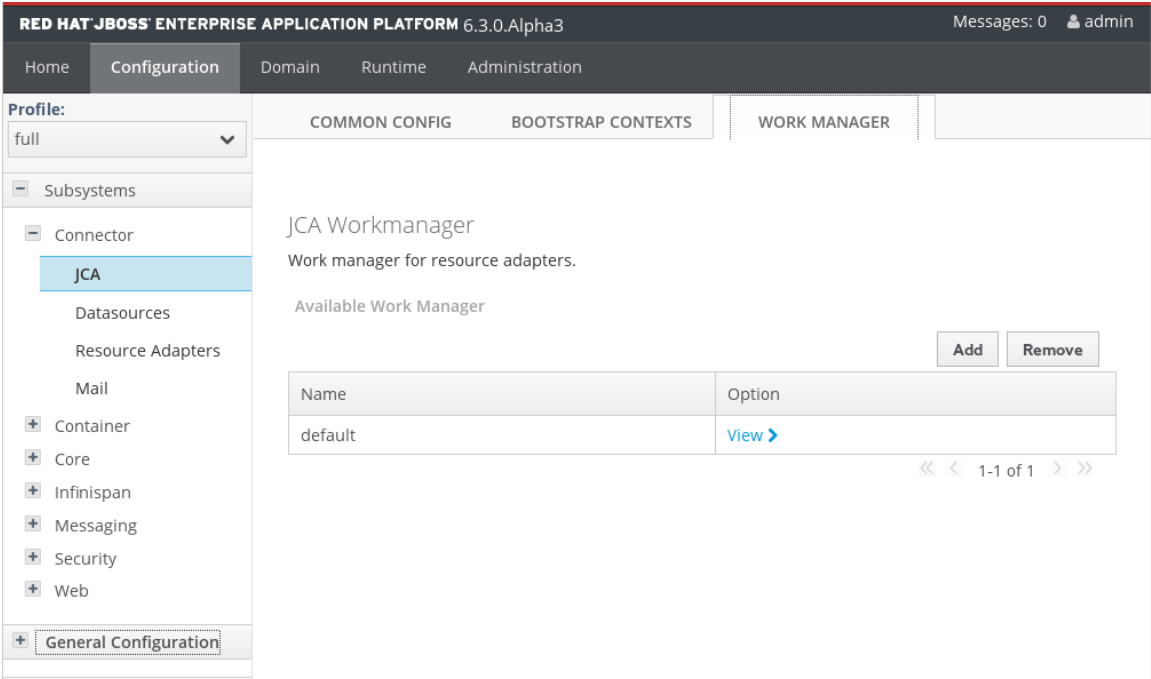


図24.2 ワークマネージャー

選択したリソースアダプターで **View** をクリックすると、スレッドプール属性を設定できます。

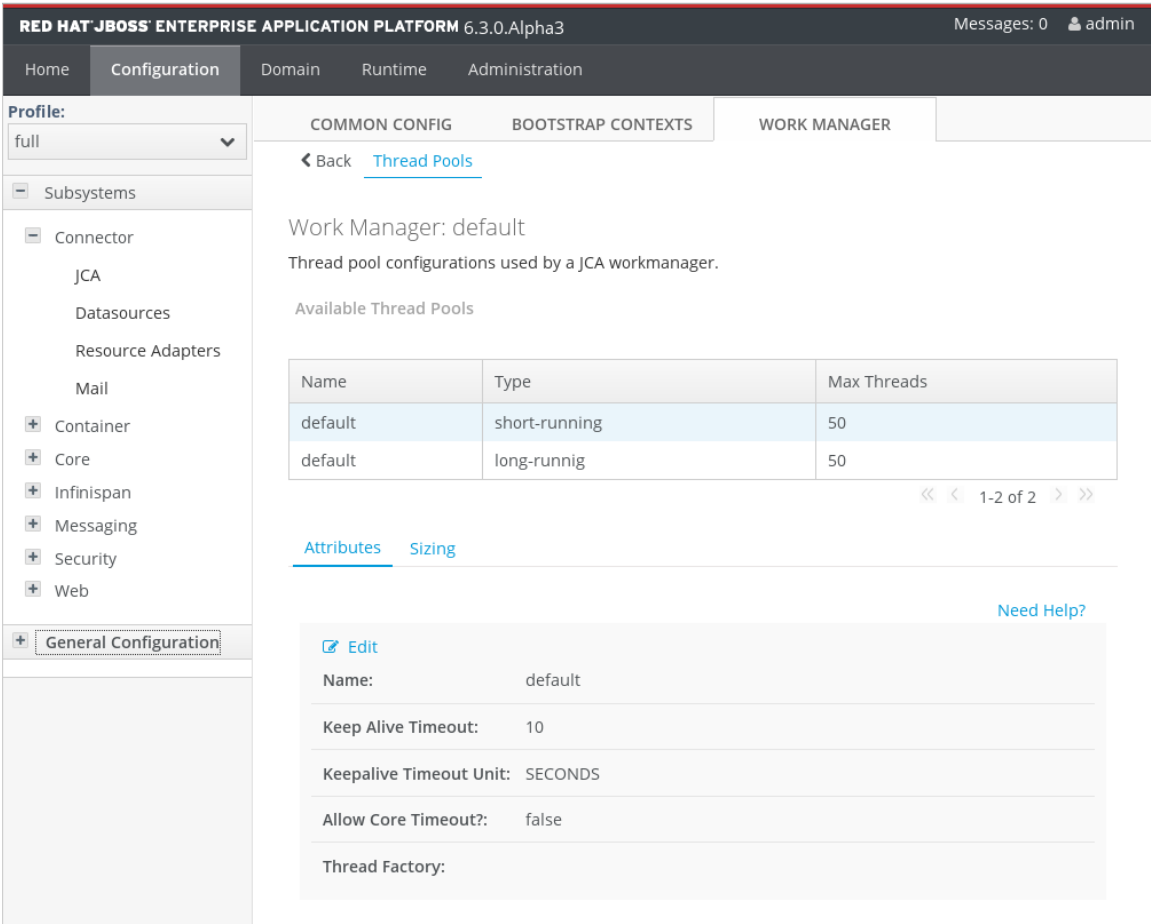


図24.3 ワークマネージャースレッドプール

- c. ブートストラップコンテキスト
- Bootstrap Contexts** 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をワークマネージャーに割り当てる必要があります。

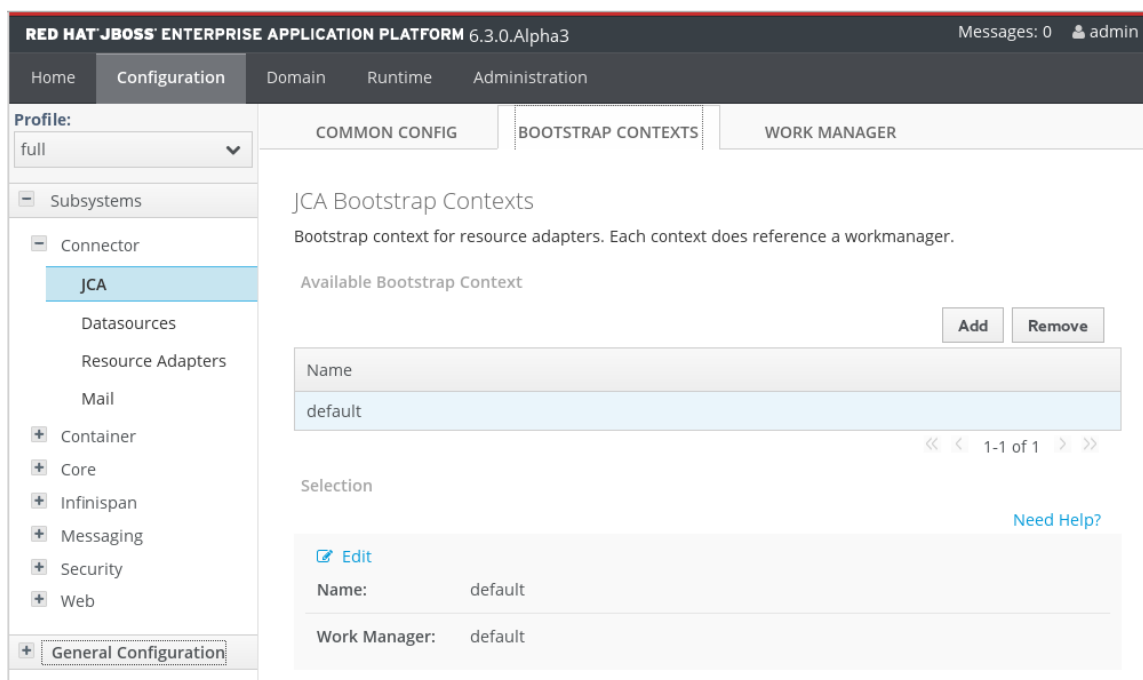


図24.4 ブートストラップコンテキスト

[Report a bug](#)

24.3. 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 ベース管理コンソールを使用して JBoss EAP 6 にデプロイできます。また、ファイルを手作業でコピーしてデプロイすることも可能です。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24.2 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24.3 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. 管理コンソールにログインします。「[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た後、**Next** をクリックします。
4. デプロイ名を検証した後、**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 **Assign** 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、リストからアプリケーションコンポーネントを選択します。**En/Disable** をクリックします。**Are You Sure?** ダイアログの **Confirm** をクリックし、コンポーネントを有効にします。

手順24.4 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、リソースアダプターアーカイブを **EAP_HOME/standalone/deployments/** ディレクトリーへコピーします。
 - 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。

[Report a bug](#)

24.4. 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

次の手順では、**[standalone@localhost:9999 /]** プロンプトの後にコマンドラインを入力する必要があります。波括弧の間にテキストを入力しないでください。コマンドを入力すると、**{"outcome" => "success"}** (出力例) のような出力が生成されます。

手順24.5 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. **server** 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. **port** 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. **name** 管理対象接続ファクトリーレベル <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24.6 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. 管理コンソールにログインします。「[管理コンソールへのログイン](#)」を参照してください。
2. 画面上部にある **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メニューから **Profile** を選択します。

追加 をクリックします。
3. アーカイブ名を入力し、**TX:** ドロップダウンボックスよりトランザクションタイプ **XATransaction** を選択した後、**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. **Name** に **server** を入力し、**Value** にホスト名 (例: **localhost**) を入力します。**Save** をクリックして終了します。
6. **Add** を再度クリックします。**Name** に **port** を入力し、**Value** にポート番号 (例: **9000**) を入力します。**Save** をクリックして終了します。
7. この時点で **server** および **port** プロパティが **Properties** パネルに表示されます。一覧表示されたリソースアダプターの **Option** カラム下にある **View** リンクをクリックし、**Connection Definitions** を表示します。

8. **Available Connection Definitions** の表の上にある **Add** をクリックし、接続定義を追加します。
9. **JNDI Name** と **Connection Class** の完全修飾クラス名を入力します。 **Save** をクリックして終了します。
10. 新しい接続定義を選択し、 **Properties** タブを選択します。 **Add** をクリックし、この接続定義の **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. これで接続の定義は終わりましたが、無効の状態になっています。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 名に対し、 **Really modify Connection Definition?** (本当に接続定義を編集しますか) という内容のダイアログが表示されます。 **Confirm** をクリックします。この時点で 接続定義の状態が **Enabled** と表示されるはずです。
13. ページ上部にある **Admin Objects** をクリックし、管理オブジェクトを作成および設定します。その後、 **Add** をクリックします。
14. 管理オブジェクトの **JNDI Name** と完全修飾 **Class Name** を入力します。入力後、 **Save** をクリックします。
15. **Properties** タブを選択した後、 **Add** をクリックして管理オブジェクトプロパティを追加します。
16. **Name** フィールドに管理オブジェクト設定プロパティ (例: **threshold**) を入力します。 **Value** フィールドに設定プロパティ値 (例: **10**) を入力します。すべて入力後、 **Save** をクリックしてプロパティを保存します。
17. これで管理オブジェクトは完了しましたが、無効の状態になっています。 **Enable** をクリックして管理オブジェクトを有効にします。
18. JNDI 名に対し **Really modify Admin Object?** (本当に管理オブジェクトを編集しますか) という内容のダイアログが表示されます。 **Confirm** をクリックします。この時点で管理オブジェクトの状態が **Enabled** と表示されるはずです。
19. この処理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、左側のナビゲーションパネルにある **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オーバーします。 **Restart Group** を選択します。
 - スタンドアロンモードでは **Reload** 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 **Do you want to reload the server configuration?** (サーバー設定をリロードしますか) という内容のダイアログが表示されます。 **Confirm** をクリックします。これでサーバー設定が最新の状態になります。

手順24.7 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は **EAP_HOME/standalone/configuration/standalone.xml** ファイルになります。
 - 管理対象ドメインの場合は **EAP_HOME/domain/configuration/domain.xml** ファイルになります。
3. 設定ファイルで **urn:jboss:domain:resource-adapters** サブシステムを探します。
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"/>
```

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。前述の管理 CLI および Web ベース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 は次のとおりです。

```
<resource-adapter>
  <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。

[Report a bug](#)

24.5. 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24.7 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的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を指定します。
transaction-support	このリソースアダプターによりサポートされた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 、および 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24.8 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24.9 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 コンテキストが使用され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24.10 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24.11 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、接続の要求からサブジェクトが既知になるまで作成されません。このデフォルト値は 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。各サブプールで作成できる最大接続数は、max-pool-size の接続数です。このデフォルト値は 20 です。

要素	説明
prefill	接続プールを事前に満たすかどうかを指定します。デフォルトでは false です。
use-strict-min	min-pool-size を厳密と見なすかどうかを指定します。デフォルト値は false です。
flush-strategy	エラー発生時にプールをどのようにフラッシュするかを指定します。有効な値は FailingConnectionOnly (デフォルト)、 IdleConnections 、および EntirePool です。

表24.12 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、接続の要求からサブジェクトが既知になるまで作成されません。このデフォルト値は 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。各サブプールで作成できる最大接続数は、max-pool-size の接続数です。このデフォルト値は 20 です。
prefill	接続プールを事前に満たすかどうかを指定します。デフォルトでは false です。
use-strict-min	min-pool-size を厳密と見なすかどうかを指定します。デフォルト値は false です。
flush-strategy	エラー発生時にプールをどのようにフラッシュするかを指定します。有効な値は FailingConnectionOnly (デフォルト)、 IdleConnections 、および EntirePool です。
is-same-rm-override	is-same-rm-override 要素を使用すると、 <code>javax.transaction.xa.XAResource.isSameRM(XAResource)</code> が true または false を返す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ピ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 <code>org.jboss.tm.XAResourceWrapper</code> インスタンスにラップするかどうか

表24.13 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) を使用することを指定します。
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。

表24.14 タイムアウト要素

要素	説明
blocking-timeout-millis	blocking-timeout-millis 要素は、接続待機中にブロックする最大時間数 (ミリ秒単位) を指定します。この時間を超過すると、例外が発生します。これは、接続許可の待機中にブロックするだけで、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。デフォルト値は 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 (プールの最小 idle-timeout-minutes の 1/2) に基づきます。
allocation-retry	allocation-retry 要素は、接続の割り当てを再試行する回数を指定します。その回数を超えると、例外が発生します。デフォルト値は 0 です。
allocation-retry-wait-millis	allocation-retry-wait-millis 要素は、接続割り当ての再試行間の時間 (ミリ秒単位) を指定します。デフォルト値は 5000 (5 秒) です。
xa-resource-timeout	XAResource.setTimeout() に渡されます。デフォルト値はゼロで、セッターは呼び出されません。秒単位で指定されます。

表24.15 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)、プールですべての接続が使用されるまで試行を続けるか (false) を指定します。デフォルト値は false です。

表24.16 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を指定します。

表24.17 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の自動有効化は、**jboss-as-resource-adapters_1_0.xsd** と http://www.ironjacamar.org/doc/schema/ironjacamar_1_0.xsd に定義されています。

[Report a bug](#)

24.6. 定義された接続統計の表示

定義された接続の統計は **deployment=name.rar** サブツリーより確認できます。

統計はこのレベルで定義され、**/subsystem** レベルでは定義されていません。これは、**standalone.xml** または **domain.xml** ファイルの設定に定義されていない **rar** に対してアクセス可能にするためです。

例を以下に示します。

例24.1

```
/deployment=example.rar/subsystem=resource-adapters/statistics=statistics/connection-definitions=java\:VtestMe:read-resource(include-runtime=true)
```



注記

統計はすべてランタイムのみの情報で、デフォルト値は **false** であるため、必ず ***include-runtime=true*** 引数を指定するようにしてください。

[Report a bug](#)

24.7. 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24.18 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。
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。

[Report a bug](#)

24.8. WEBSphere MQ リソースアダプターのデプロイ

WebSphere MQ

WebSphere MQ は、分散型システム上のアプリケーションの相互通信を可能にする、IBM の Messaging Oriented Middleware (MOM) ソフトウェアです。この機能は、メッセージとメッセージキューを使用することによって実現されます。WebSphere MQ はメッセージキューへのメッセージ配信と、メッセージチャネルを使用したその他のキューマネージャーへのデータ転送を行います。WebSphere MQ の詳細については、[WebSphere MQ](#) を参照してください。

概要

本トピックでは、Red Hat JBoss Enterprise Application Platform 6 における WebSphere MQ Resource Adapter のデプロイと設定の手順について説明します。この作業は、設定ファイルを手動で編集する方法もしくは管理 CLI ツールや Web ベースの管理コンソールを使用する方法で行うことができます。

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_INVALID:
javax.transaction.xa.XAException: The method 'xa_recover' has failed with errorCode
'-5'.
```

そのため、バージョン 7.5.0.4 以上を使用することが推奨されます。この問題の詳細については、<http://www-01.ibm.com/support/docview.wss?uid=swg1lC97579> を参照してください。

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。**7.5.0.0** およびそれ以降のバージョンを使用する必要があります。
- 以下の WebSphere MQ 設定プロパティの値を知っておく必要があります。これらのプロパティに関する詳細については、WebSphere MQ 製品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に精通する必要があります。
 - :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

IBM によって提供されるデフォルト設定は次のとおりです。これらの設定は変更することがあります。詳細については、については、WebSphere MQ のドキュメンテーションを参照してください。

手順24.8 リソースアダプターの手動でのデプロイ

1. WebSphereMQ リソースアダプターにトランザクションサポートが必要な場合は、**wmq.jmsra-VERSION.rar** アーカイブを再パッケージ化し、**mqetclient.jar** が含まれるようにします。これには次のコマンドを使用できます。

```
[user@host ~]$ jar -uf wmq.jmsra-VERSION.rar mqetclient.jar
```

必ず **VERSION** を正しいバージョン番号に置き換えてください。

2. **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。
3. 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** サブシステムを探します。
 - 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"/>
```

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で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。
 - 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 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-object>
</admin-objects>
</resource-adapter>
```


必ず *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、**<!-- <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>
          <user-name>USER_NAME</user-name>
          <password>PASSWORD</password>
        </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>

```

必ず *VERSION* を RAR 名の実際のバージョンに置き換えてください。さらに、*USER_NAME* と *PASSWORD* も有効なユーザー名とパスワードに置き換える必要があります。



注記

トランザクションをサポートするため、<transaction-support> 要素が **XATransaction** に設定されました。XA リカバリーをサポートするため、<recovery> 要素が接続定義に追加されました。

- e. JBoss EAP 6 で 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 に変更するには、**urn:jboss:domain:ejb3:1.2** サブシステムを以下のように変更します。

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>

```

以下のように置き換えます。

```

<mdb>
  <resource-adapter-ref resource-adapter-name="wmq.jmsra-VERSION.rar"/>
  <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
</mdb>

```

必ず *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24.9 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。

- 次のように、MDB コードの `ActivationConfigProperty` および `ResourceAdapter` を設定します。

```

@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 {
    }

```

必ず `VERSION` を RAR 名の実際のバージョンに置き換えてください。

[Report a bug](#)

24.9. WEBSPPHERE MQ リソースアダプターのインストール

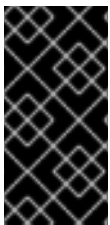
JBoss Active MQ (A-MQ) リソースアダプターを JBoss EAP 6 にインストールし、JBoss A-MQ 6.1.0 と動作するようにするには、https://access.redhat.com/site/documentation/en-US/Red_Hat_JBoss_Fuse/6.1/html/Deploying_into_a_Web_Server/files/DeployRar.html の手順に従います。

[Report a bug](#)

24.10. サードパーティー JMS プロバイダーを使用するよう汎用 JMS リソースアダプターを設定

概要

サードパーティー JMS プロバイダーを使用するよう JBoss EAP 6 を設定できますが、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統合するための JMS JCA リソースアダプターを作成するわけではありません。ここでは、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ための手順について取り上げます。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ていますが、他の JMS プロバイダーを使用する場合は、必要な設定が異なる場合があります。



重要

汎用 JMS JCA リソースアダプターは、JMS プロバイダーが独自のリソースアダプターを提供しない場合のみ使用するようにしてください。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、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てください。

前提条件

この手順では、JMS プロバイダーサーバーは設定済みで、使用できる状態であることを前提としています。プロバイダーの JMS 実装に必要なバイナリーがすべて必要になります。また、以下の JMS プロバイダープロパティの値を知っている必要があります。

- `PROVIDER_HOST:PROVIDER_PORT`: JMS プロバイダーサーバーのホスト名およびポート番号。

- `PROVIDER_CONNECTION_FACTORY`: JMS プロバイダーサーバーにデプロイされた接続ファクトリーの名前。XA である必要があります。
- `PROVIDER_QUEUE`、`PROVIDER_TOPIC`: 使用される JMS プロバイダーサーバーのキューおよびトピックの名前。

手順24.10 汎用 JMS リソースアダプターの設定

1. キューおよびトピックの JNDI バインディング向けに **ObjectFactory** 実装を作成します。
 - a. 以下のコードをテンプレートとして使用し、サーバーの詳細を実際の JMS プロバイダーサーバーの値に置き換えます。

```
import java.util.Hashtable;
import java.util.Properties;

public class RemoteJMSObjectFactory implements ObjectFactory {

    private Context context = null;

    public RemoteJMSObjectFactory() {
    }

    public Object getObjectInstance(Object obj, Name name, Context nameCtx,
        Hashtable<?, ?> environment) throws Exception {
        try {
            String jndi = (String) obj;

            final Properties env = new Properties();
            env.put(Context.INITIAL_CONTEXT_FACTORY,
                "com.tibco.tibjms.naming.TibjmsInitialContextFactory");
            env.put(Context.URL_PKG_PREFIXES, "com.tibco.tibjms.naming");
            env.put(Context.PROVIDER_URL, "tcp://TIBCO_HOST:TIBCO_PORT");

            context = new InitialContext(env);
            Object o = context.lookup(jndi);

            return o;
        } catch (NamingException e) {
            e.printStackTrace();
            throw e;
        }
    }
}
```

- b. 上記のコードをコンパイルし、結果となるクラスファイルを **remoteJMSObjectFactory.jar** という名前の JAR ファイルに保存します。
2. JBoss EAP 6 インスタンスの **genericjms** モジュールを作成します。
 - a. **EAP_HOME/modules/system/layers/base/org/jboss/genericjms/provider/main** というディレクトリ構造を作成します。
 - b. **remoteJMSObjectFactory.jar** ファイルを **EAP_HOME/modules/system/layers/base/org/jboss/genericjms/provider/main** にコピーします。

- c. 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。Tibco EMS では、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 **tibcrypt.jar** になります。
- d. 下記のように **module.xml** ファイルを **EAP_HOME/modules/system/layers/base/org/jboss/genericjms/provider/main** に作成し、以前の手順の JAR ファイルをリソースとしてリストします。

```
<module xmlns="urn:jboss:module:1.1" name="org.jboss.genericjms.provider">
  <resources>
    <resource-root path="tibjms.jar"/>
    <resource-root path="tibcrypt.jar"/>
    <resource-root path="remoteJMSObjectFactory.jar"/>
  </resources>

  <dependencies>
    <module name="javax.api"/>
    <module name="javax.jms.api"/>
  </dependencies>
</module>
```

3. グローバルモジュールとするすべてのデプロイメントに対し、汎用 JMS モジュールを依存関係として追加します。



注記

この手順では、**EAP_HOME/standalone/configuration/standalone-full.xml** が JBoss EAP 6 設定ファイルとして使用されます。

EAP_HOME/standalone/configuration/standalone-full.xml にて、**<subsystem xmlns="urn:jboss:domain:ee:1.1">** の下に以下を追加します。

```
<global-modules>
  <module name="org.jboss.genericjms.provider" slot="main"/>
  <module name="org.jboss.common-core" slot="main"/>
</global-modules>
```

4. デフォルトの HornetQ リソースアダプターを汎用リソースアダプターに置き換えます。

EAP_HOME/standalone/configuration/standalone-full.xml の **<subsystem xmlns="urn:jboss:domain:ejb3:1.4">** **<mdb>** を以下に置き換えます。

```
<mdb>
  <resource-adapter-ref resource-adapter-name="org.jboss.genericjms"/>
  <bean-instance-pool-ref pool-name="mdb-strict-max-pool"/>
</mdb>
```

5. 必要に応じて、JMS トピックおよびキューのバインディングをリモートオブジェクトとして追加します。

EAP_HOME/standalone/configuration/standalone-full.xml にて、**<subsystem xmlns="urn:jboss:domain:naming:1.3">** の下にバインディングを追加し、必要に応じて **PROVIDER_QUEUE** および **PROVIDER_TOPIC** を置き換えます。

```
<bindings>
  <object-factory name="PROVIDER_QUEUE" module="org.jboss.genericjms.provider"
class="org.jboss.qa.RemoteJMSObjectFactory"/>
  <object-factory name="PROVIDER_TOPIC" module="org.jboss.genericjms.provider"
class="org.jboss.qa.RemoteJMSObjectFactory"/>
</bindings>
```

6. **EAP_HOME/standalone/configuration/standalone-full.xml** にて、汎用リソースアダプターを **<subsystem xmlns="urn:jboss:domain:resource-adapters:1.1">** に追加します。

PROVIDER_CONNECTION_FACTORY、*PROVIDER_HOST*、および *PROVIDER_PORT* を JMS プロバイダーの値に置き換えます。

```
<resource-adapters>
  <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/PROVIDER_CONNECTION_FACTORY" pool-
name="PROVIDER_CONNECTION_FACTORY">
        <config-property name="JndiParameters">

          java.naming.factory.initial=com.tibco.tibjms.naming.TibjmsInitialContextFactory;java.naming.pro
vider.url=tcp://PROVIDER_HOST:PROVIDER_PORT
        </config-property>
        <config-property name="ConnectionFactory">
          PROVIDER_CONNECTION_FACTORY
        </config-property>
        <security>
          <application/>
        </security>
      </connection-definition>
    </connection-definitions>
  </resource-adapter>
</resource-adapters>
```

結果

汎用 JMS リソースアダプターが設定され、使用できる状態になりました。

新しいメッセージ駆動型 Bean (MDB) を作成するとき、以下のようなコードを使用し、リソースアダプターを使用します。 *PROVIDER_CONNECTION_FACTORY*、*PROVIDER_HOST*、および *PROVIDER_PORT* を JMS プロバイダーの値に置き換えます。

オプションとして、以下のように **user** および **password** プロパティを指定して (必要に応じてプロパティの値を置き換えます) Tibco EMS のセキュアな接続を設定することもできます。

```
@MessageDriven(activationConfig = {
  @ActivationConfigProperty(propertyName = "destinationType", propertyValue = "javax.jms.Queue"),
  @ActivationConfigProperty(propertyName = "jndiParameters", propertyValue =
"java.naming.factory.initial=com.tibco.tibjms.naming.TibjmsInitialContextFactory;java.naming.provider.ur
=tcp://PROVIDER_HOST:PROVIDER_PORT")
  @ActivationConfigProperty(propertyName = "destination", propertyValue = "PROVIDER_QUEUE"),
```

```
@ActivationConfigProperty(propertyName = "connectionFactory", propertyValue =
"PROVIDER_CONNECTION_FACTORY"),
@ActivationConfigProperty(propertyName = "user", propertyValue = "USER"),
@ActivationConfigProperty(propertyName = "password", propertyValue = "PASSWORD"),
})

@ResourceAdapter("org.jboss.genericjms")
public class SampleMdb implements MessageListener {
    @Override

    public void onMessage(Message message) {

    }

}
```

[Report a bug](#)

第25章 AMAZON EC2 での JBOSS EAP 6 のデプロイ

25.1. はじめに

25.1.1. Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) は、amazon.com により運用されているサービスであり、顧客に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。そのサービスを使用して Amazon Machine Image (AMI) を起動して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、インスタンスで必要な任意のソフトウェアをインストールし、使用した機能に応じて請求されます。Amazon EC2 は、柔軟に設計され、ユーザーがデプロイされたアプリケーションを素早くスケールすることを可能にします。

詳細については、Amazon EC2 の Web サイト (<http://aws.amazon.com/ec2/>) を参照してください。

[Report a bug](#)

25.1.2. Amazon Machine Instance (AMI)

Amazon Machine Image (AMI) は、EC2 仮想マシンインスタンス用のテンプレートです。ユーザーは、作成元の適切な AMI を選択して EC2 インスタンスを作成します。AMI の主要なコンポーネントは、インストールされたオペレーティングシステムと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。各 AMI では、さまざまな使用ケースに応じてさまざまなソフトウェアがインストールされます。Amazon EC2 では、amazon.com とサードパーティーが提供する多くの AMI から選択できます。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。

[Report a bug](#)

25.1.3. 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プロバイダーで JBoss EAP 6 をサポートする Red Hat サブスクリプション機能です。JBoss Cloud Access を使用すると、従来のサーバーのリソースとパブリッククラウドベースのリソース間で単純かつ費用効果がある方法でサブスクリプションを移行できます。

詳細については、<http://www.redhat.com/solutions/cloud/access/jboss/> を参照してください。

[Report a bug](#)

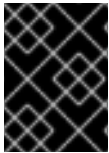
25.1.4. JBoss Cloud Access 機能

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。

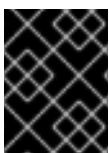
[Report a bug](#)

25.1.5. サポートされる Amazon EC2 インスタンスタイプ

JBoss Cloud Access は、次の 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については、『Amazon EC2 User Guide (Amazon EC2 ユーザーガイド)』(<http://docs.amazonwebservices.com/AWSEC2/latest/UserGuide/instance-types.html>) を参照してください。

表25.1 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。
高メモリーインスタンス	高メモリーインスタンスでは、標準インスタンスよりも多いメモリーが割り当てられます。高メモリーインスタンスは、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。
高 CPU インスタンス	高 CPU インスタンスでは、メモリーよりも多い CPU リソースが割り当てられるため、これはスループットが比較的低く、CPU リソースを大量に消費する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適していません。

[Report a bug](#)

25.1.6. サポート対象 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 (**6.3** など)。

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 (**6.2** など)。

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 が作成された日付 (YYYYMMDD 形式) です (**20120501** など)。

AMI 名の例: **RHEL-6.2-JBEAP-6.0.0-x86_64-20120501**。

[Report a bug](#)

25.2. AMAZON EC2 での JBOSS EAP 6 のデプロイ

25.2.1. Amazon EC2 での JBoss EAP 6 のデプロイ (概要)

JBoss EAP 6 は、Amazon EC2 AMI を使用してデプロイできます。AMI には、クラスターインスタンスと非クラスターインスタンスのデプロイメントに必要なものがすべて含まれます。

非クラスターインスタンスのデプロイは最も簡単なシナリオです。インスタンスの作成時は、アプリケーションデプロイメントを指定するためにいくつかの設定を変更する必要があります。

クラスター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仮想プライベートクラウドを作成することが推奨されます。mod_cluster プロキシとして動作するよう JBoss EAP インスタンスを使用することはオプションですが、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/JBoss_Enterprise_Application_Platform/

- Red Hat Enterprise Linux 6

https://access.redhat.com/documentation/Red_Hat_Enterprise_Linux/

- Amazon Web サービス

<http://aws.amazon.com/documentation/>

[Report a bug](#)

25.2.2. 非クラスター化の JBoss EAP 6

25.2.2.1. 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 が実行されている単一の 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。

[Report a bug](#)

25.2.2.2. 非クラスターインスタンス

25.2.2.2.1. 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 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グループ。

手順25.1 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」](#) および [「カスタムスクリプトパラメーター」](#) を参照してください。

例25.1 User Data フィールドの例

この例は、非クラスター JBoss EAP 6 インスタンス用の User Data フィールドを示します。ユーザー **admin** のパスワードは **adminpwd** に設定されます。

```
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

本番環境インスタンスの場合は、次の行を **User Data** フィールドの **USER_SCRIPT** 行の下に追加してセキュリティアップデートが起動時に適用されるようにします。

```
yum -y update
```



注記

yum -y update を定期的に実行して、セキュリティ修正と拡張を適用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。

[Report a bug](#)

25.2.2.2.2. 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

次の行を **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25.2 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 で提供された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 JBoss EAP 6 インスタンスの基本的な設定も含まれます。ユーザー **admin** のパスワードが **adminpwd** に設定されています。

```
JBOSASAS_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

次の行を **User Data** フィールドに追加し、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25.3 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** と呼ばれる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的な設定が含まれます。ユーザー **admin** のパスワードが **adminpwd** に設定されます。

```

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。

[Report a bug](#)

25.2.2.2.3. 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25.2 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 **Public DNS** を調べます。
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。ホームページが表示されない場合は「[Amazon EC2 のトラブルシューティング](#)」を参照してください。
4. **Admin Console** 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**
 - パスワード: 「[非クラスター化の JBoss EAP 6 インスタンスの起動](#)」の **User Data** フィールドに指定します。
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。この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](#)」を参照してください。

- 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。

[Report a bug](#)

25.2.2.3. 非クラスター化管理対象ドメイン

25.2.2.3.1. 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 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25.3 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

- Security Group 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。アクセスを制限したい場合は、Red Hat Enterprise Linux のビルトインファイアウォール機能を使用できます。
- VPC のパブリックサブネットを **running** に設定します。
- 静的 IP を選択します。
- 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](#)」および「[カスタムスクリプトパラメーター](#)」を参照してください。Amazon EC2 でのドメインコントローラー検出の詳細については、「[ドメインコントローラー検出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25.4 User Data フィールドの例

この例は、非クラスター化 JBoss EAP 6 管理対象ドメイン用の User Data フィールドを示しています。ユーザー **admin** のパスワードは **admin** に設定されます。

```

## 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,scheme=htt
p,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}$i" ;
done >> /etc/hosts

EOF

```

5. 本番稼働インスタンスの場合

※ 本番稼働インスタンスの場合、1 行の行を User Data フォーマットの USER_SCRIPT 行の下に

本番環境 1 ノスタンスの場合は、次の行を **User Data** フィールドの **USER_SCRIPT** 行の末尾に追加してセキュリティアップデートが起動時に適用されるようにします。

```
yum -y update
```



注記

yum -y update を定期的 to 実行して、セキュリティ修正と拡張を適用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。

[Report a bug](#)

25.2.2.3.2. 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

- 非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。「[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25.4 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-${SUBNET}/${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については、[「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

本番環境インスタンスの場合は、次の行を **User Data** フィールドの **USER_SCRIPT** 行の下に追加してセキュリティアップデートが起動時に適用されるようにします。

```
yum -y update
```



注記

yum -y update を定期的に行って、セキュリティ修正と拡張を適用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。

[Report a bug](#)

25.2.2.3.3. 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する必要があります。「[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。
- ホストコントローラーを設定および起動する必要があります。「[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25.5 Web サーバーのテスト

- ブラウザで `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25.6 ドメインコントローラーのテスト

- `http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console` へ移動します。
- ドメインコントローラーの User Data フィールドで指定されたユーザー名 **admin** とパスワードを使用してログインします。管理対象ドメインに対する管理コンソールランディングページ (`http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console/App.html#server-instances`) が表示されるはずです。
- 画面の右上にある **Server** ラベルをクリックし、画面の左上にある **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。
- 各ホストコントローラーに **server-one** と **server-two** の2つのサーバー構成があることを確認し、これら両方が **main-server-group** に属することを確認します。
- JBoss EAP 6 の管理コンソールからログアウトします。

手順25.7 ホストコントローラーのテスト

- `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。テキスト **Hello World!** がブラウザで表示されるはずです。

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。

- 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
```

- `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。

[Report a bug](#)

25.2.2.3.4. 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 で稼働している管理対象ドメインの場合、ホストコントローラーはドメインコントローラーを静的に検索するだけでなく、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索できます。Amazon S3 バケットへのアクセスに必要な情報を使用して、ホストコントローラーとドメインコントローラーを設定できます。

ドメインコントローラーの起動時にこの設定を使用すると、バケットの S3 ファイルへコントラクト情報を書き込みます。ホストコントローラーがドメインコントローラーへコンタクトしようとするたびに、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。

そのため、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 インスタンスが停止および起動すると EC2 インスタンスの IP アドレスが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。

JBOSS_DOMAIN_S3_ACCESS_KEY、**JBOSS_DOMAIN_S3_SECRET_ACCESS_KEY**、および **JBOSS_DOMAIN_S3_BUCKET** パラメーターを起動時に JBoss EAP 6 インスタンスへ渡すと自動的にドメインコントローラー検索を有効にできます。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。この代わりに、以下の設定を使用してドメイン検索を手動で設定することもできます。

手動のドメインコントローラー検索の設定は、以下のプロパティを使用して指定されます。

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になります。下例では、検索オプションが1つ示されていますが、静的検索オプションやS3 検索オプションをいくつでも設定できます。ドメイン検索やフェールオーバー処理の詳細については、「[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25.5 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 security-realm>
</domain-controller>
```

```

    </discovery-option>
  </discovery-options>
</remote>
</domain-controller>

```

例25.6 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>

```

[Report a bug](#)

25.2.3. クラスター化された JBoss EAP 6

25.2.3.1. 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 が実行されている Amazon EC2 インスタンスです。Apache HTTP サーバーが実行され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で使用する 2 つの設定ファイル (**standalone-ec2-ha.xml** と **standalone-mod_cluster-ec2-ha.xml**) が含まれます。Amazon EC2 はマルチキャストをサポートしないため、これらの各設定ファイルは、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、クラスター通信用 TCP ユニキャストと S3_PING を検出プロトコルとして使用して行われます。また、**standalone-mod_cluster-ec2-ha.xml** 設定は、mod_cluster プロキシを使用して簡単な登録を提供します。

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 と mod_cluster-ec2-ha の 2 つのプロファイルを提供します。

[Report a bug](#)

25.2.3.2. リレーショナルデータベースサービスデータベースインスタンスの作成

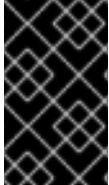
概要

このトピックでは、MySQL を例として使用して、リレーショナルデータベースサービスデータベースインスタンスを作成する手順について説明します。



警告

本番稼働環境に対してはバックアップおよび保守機能を有効にしたままにすることを強くお勧めします。



重要

また、データベースにアクセスする各アプリケーションに対して個別のユーザーとパスワードのペアを作成することをお勧めします。アプリケーションの要件に応じて他の設定オプションを調整します。

手順25.8 リレーショナルデータベースサービスデータベースインスタンスの作成

1. AWS コンソールの **RDS** をクリックします。
2. 必要の場合は、サービスをサブスクライブします。
3. **Launch DB instance** をクリックします。
4. **MySQL** をクリックします。
 - a. バージョンを選択します (**5.5.12** など)。
 - b. **small instance** を選択します。
 - c. **Multi-AZ Deployment** と **Auto upgrade** が **off** であることを確認します。
 - d. **Storage** を **5GB** に設定します。
 - e. データベース管理者のユーザー名とパスワードを定義し、**Next** をクリックします。
 - f. インスタンスで作成するデータベース名を選択し、**Next** をクリックします。
 - g. 必要の場合は、バックアップおよび保守を無効にします。
 - h. 設定を確認します。

結果

データベースが作成されます。数分後に、データベースは初期化され、使用できるようになります。

[Report a bug](#)

25.2.3.3. Virtual Private Cloud

Amazon Virtual Private Cloud (Amazon VPC) は、プライベートネットワークの AWS リソースのセットを隔離することを可能にする 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と設定は、必要に応じてカスタマイズできます。

詳細については、Amazon Virtual Private Cloud Web サイト (<http://aws.amazon.com/vpc/>) を参照してください。

[Report a bug](#)

25.2.3.4. 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、データベースに対する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の詳細については、Amazon の「RDS FAQ」を参照してください。



重要

VPC は、JBoss EAP 6 クラスターセットアップに対して推奨されます。VPC を使用すると、クラスターノード、JON サーバー、および mod_cluster プロキシ間のセキュアな通信が大幅に単純化されます。VPC がないと、これらの通信チャネルを暗号化し、認証する必要があります。

SSL の設定手順の詳細は、「[JBoss EAP 6 Web サーバーでの SSL 暗号化の実装](#)」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. 必要の場合は、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. **Security groups** 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**、**9443**、**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。

[Report a bug](#)

25.2.3.5. 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

- 「[リレーショナルデータベースサービスデータベースインスタンスの作成](#)」.
- 「[Virtual Private Cloud \(VPC\) の作成](#)」

手順25.9 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** を選択します。
5. 静的な IP (**10.0.0.4** など) 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。

```
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 On
</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. 稼働している 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。

[Report a bug](#)

25.2.3.6. 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、クラスターノードでは 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ためにデフォルトルートを設定する必要があります。

手順25.10 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 → route tables** 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. 新しい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。

[Report a bug](#)

25.2.3.7. Identity and Access Management (IAM)

Identity and Access Management (IAM) は、AWS リソースに対して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したり、IAM と独自の ID サービス間の ID フェデレーションを提供したりするよう設定できます。

詳細については、AWS Identity and Access Management Web サイト (<http://aws.amazon.com/iam/>) を参照してください。

[Report a bug](#)

25.2.3.8. 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 インスタンス用の IAM をセットアップするの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するために Amazon AWS アカウントアクセスと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索は 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** プロトコルと似ている) に対して認証するために Amazon AWS アカウントへのアクセスとシークレットキーが必要になります。S3 検索に使用される 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 検索とは別でなければなりません。

user-data フィールドで主要なアカウントクレデンシャルを入力し、これらをオンラインで格納したり、AMI に格納したりすることはセキュリティー上のリスクになります。この問題を回避するには、単一の S3 バケットへのアクセスのみを提供する Amazon IAM 機能を使用して個別アカウントを作成します。

手順25.11 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. **users** 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、**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、セキュア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** 値を書き留めます。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。

[Report a bug](#)

25.2.3.9. 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。バケットは任意のオブジェクトの任意の数を格納でき、Amazon S3 で識別する一意の名前を必要とします。

詳細については、Amazon S3 Web サイト (<http://aws.amazon.com/s3/>) を参照してください。

[Report a bug](#)

25.2.3.10. 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25.12 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。名前は再使用できません。

4. 新しいバケットを右クリックし、**Properties** を選択します。
5. パーミッションタブの **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」で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 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。

[Report a bug](#)

25.2.3.11. クラスターインスタンス

25.2.3.11.1. 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

- 「リレーショナルデータベースサービスデータベースインスタンスの作成」.
- 「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行ったり、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せん。ただし、デプロイメント前にテストする必要があります。特定のワークロードが必要な数のノードにスケールされ、パフォーマンスが、使用予定のインスタンスタイプの要件に見合う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、異なる EC2 クラウドリソースを受け取ります)。

さらに、インスタンスローカルティニー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ます。想定される実際のロードでテストし、予期しない状況を考慮するようにします。



警告

Amazon EC2 **scale-down** 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。一部のトランザクションは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) はフェールオーバーする時間が必要です。そのため、多くの場合でアプリケーションの使用に影響を与えます。

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、ユーザーに悪影響が出ないようにテストします。特定のワークロード、ロードバランサー、およびセットアップに対して追加措置が必要になることがあります。

手順25.13 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. **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。

例25.7 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=instancetype.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。

[Report a bug](#)

25.2.3.11.2. 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25.14 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 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。

2. クラスター化されたノードのテスト

- a. ブラウザーで http://ELASTIC_IP_OF_APACHE_HTTPD/cluster-demo/put.jsp にアクセスします。
- 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 http://ELASTIC_IP_OF_APACHE_HTTPD/cluster-demo/get.jsp にアクセスします。
- e. 示された時間が、手順 2-a で **put.jsp** を使用して示され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. 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

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。

[Report a bug](#)

25.2.3.12. クラスター化管理対象ドメイン

25.2.3.12.1. 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 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25.15 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. 静的な 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

本番環境インスタンスの場合は、次の行を **User Data** フィールドの **USER_SCRIPT** 行の下に追加してセキュリティーアップデートが起動時に適用されるようにします。

```
yum -y update
```



注記

yum -y update を定期的に実行して、セキュリティ修正と拡張を適用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。

[Report a bug](#)

25.2.3.12.2. 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する必要があります。「[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25.16 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

本番環境インスタンスの場合は、次の行を **User Data** フィールドの **USER_SCRIPT** 行の下に追加してセキュリティーアップデートが起動時に適用されるようにします。

```
yum -y update
```



注記

yum -y update を定期的に行って、セキュリティー修正と拡張を適用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。

[Report a bug](#)

25.2.3.12.3. 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する必要があります。「[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および起動する必要があります。「[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25.17 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** にアクセスし、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25.18 ドメインコントローラーのテスト

1. **http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console** にアクセスします。
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たユーザー名 **admin** とパスワードを使用してログインします。ログイン後に、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 (**http://ELASTIC_IP_OF_DOMAIN_CONTROLLER:9990/console/App.html#server-instances**) が表示されるはずです。
3. 画面の右上にある **Server** ラベルをクリックします。画面の左上にある **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** の2つのサーバー構成があることを確認し、これら両方が **other-server-group** に属することを確認します。

手順25.19 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` にアクセスします。
2. いずれかのホストコントローラーで次のメッセージがログに記録されていることを確認します。 **Putting date now.**
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` にアクセスします。
5. 示された時間が、手順 2 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。
6. 稼働している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 (セクション 2.2.2 「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。

[Report a bug](#)

25.3. JBOSS OPERATIONS NETWORK (JON) での監視の確立

25.3.1. 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部にあるため、サーバーと各エージェント間でセキュアな接続を確立する必要があります。2つのポート間での VPN の確立は、最も一般的なソリューションですが、必要なネットワーク設定が複雑になります。本章では、JON エージェントと JON サーバー間の通信を有効化する際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 ドキュメンテーションを参照してください。

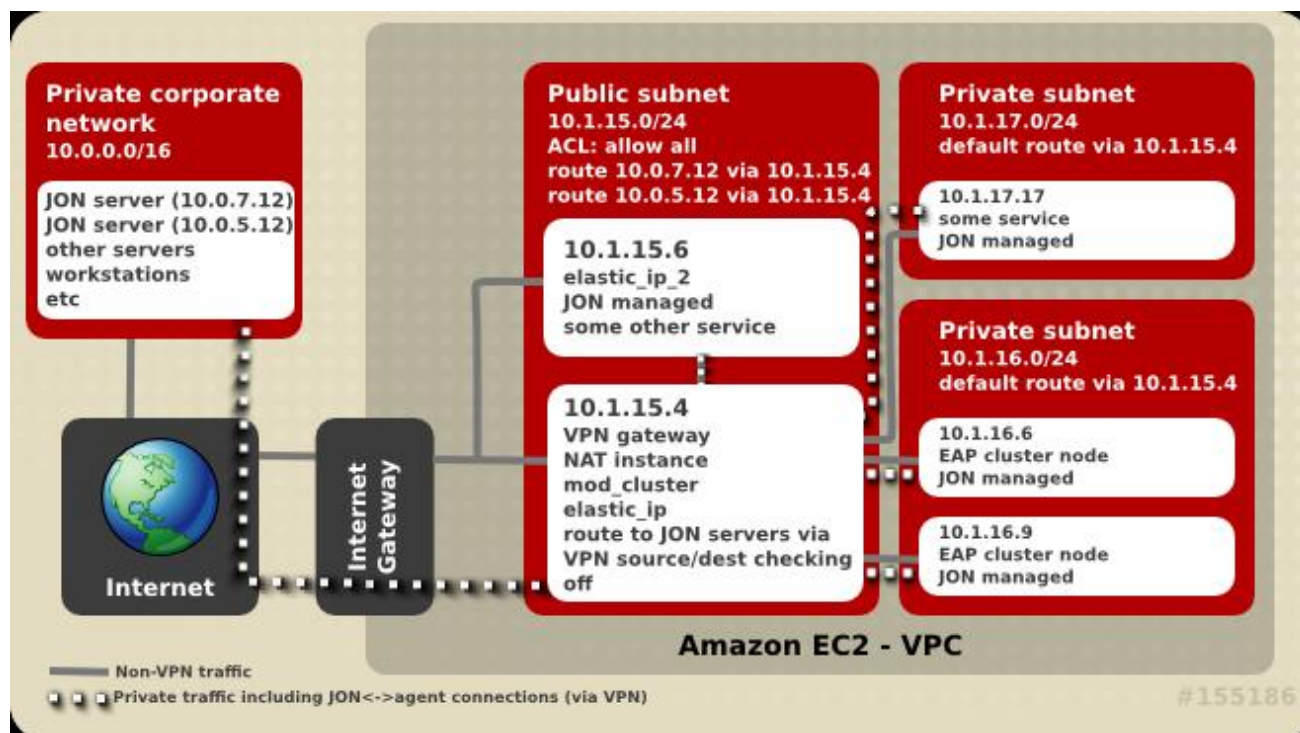


図25.1 JON サーバーの接続性

[Report a bug](#)

25.3.2. 接続要件

JON エージェントをサーバーで登録するには、エージェントとサーバー間で双方向通信が必要です。JON エージェントはすべての JON サーバーのポート **7080** にアクセスする必要があります (ただし、ポート **7443** が使用された SSL の場合を除く)。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。エージェントのポートは通常 **16163** です。

複数のクラスター 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 クラスター内のすべてのサーバーと通信できるようにします。登録するエージェントにより使用される JON サーバーは、初期化後に使用しようとするサーバーでないことがあります。

[Report a bug](#)

25.3.3. 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単純化されます。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。この場合は、各エージェントに対してポートフォワーディング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上にあるポートの JON エージェントアドレスに転送する必要があります。また、JON エージェントがサーバーに、転送されたポート番号と IP アドレスを通知するよう設定する必要があります。詳細については、**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。

[Report a bug](#)

25.3.4. Amazon EC2 および DNS

JON サーバーおよび JON エージェントは、お互いのホスト名を解決できる必要があります。DNS 解決は、VPN 設定では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つのオプションは、Amazon EC2 または企業ネットワークの DNS サーバーを使用することです。別のオプションは、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 設定を使用することです。

[Report a bug](#)

25.3.5. 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 **source/destination checking** ルーティング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送信先を持つサーバーに送信されるすべてのパケットを破棄します。JON サーバーにエージェントを接続するために選択された VPN ソリューションにルーターが含まれる場合は、サーバーをルーターまたは VPN ゲートウェイとして動作するためにこの機能を無効にする必要があります。この設定は、Amazon AWS コンソールを介してアクセスできます。また、**source/destination checking** は、Virtual Private Cloud (VPC) でも無効にする必要があります。

一部の VPN 設定では、デフォルトで一般的なインターネットトラフィックが企業 VPN を介してルーティングされます。場合によっては設定が低速かつ非効率になることがあるため、このルーティングを行わない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 に固有の問題ではありませんが、適切でないスキーマを使用すると、JON が影響を受けることがあります。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。通常、インスタンスはパブリック IP アドレスを持ちますが、同じ利用可能なゾーン内の内部 IP アドレスでのネットワークトラフィックのみが使用可能になります。プライベートアドレス指定で **10.0.0.0/8** ネットワークの使用しないようにするには、以下を考慮する必要があります。

- 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いサブネット (JON サーバーのみなど) にアクセスする場合は、これらのアドレスのみを VPN を介してルーティングする必要があります。これにより、セキュリティが強化され、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。

[Report a bug](#)

25.3.6. JON での終了と再起動

クラウド環境の利点の1つは、簡単にマシンインスタンスを終了および起動できることです。また、最初のインスタンスと同一のインスタンスを起動することもできます。これにより、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の名前を使用して JON サーバーで登録しようとする場合は、問題が発生することがあります。この問題が発生した場合、JON サーバーは、不明または一致しない ID トークンを用いた再接続をエージェントに許可しません。

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適切な ID トークンを指定します。

発生する可能性がある別の問題は、エージェントマシンに、JON 設定で記録されたアドレスに一致しなくなった新しい VPN IP アドレスが割り当てられることです。例には、再起動されたり、VPN 接続が終了されたりするマシンが含まれます。この場合は、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をお勧めします。接続が破棄された場合は、エージェントを停止できます。接続が復元された場合は、新しい IP アドレスを反映するよう `/etc/sysconfig/jon-agent-ec2` の `JON_AGENT_ADDR` を更新し、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、https://access.redhat.com/site/documentation/JBoss_Operations_Network/ の JON Servers and Agents Guide (JON サーバーおよびエージェントガイド) を参照してください。

大量のインスタンスが起動または終了される場合、それらのインスタンスを JON インベントリから手動で追加および削除することは実質的に不可能になることがあります。JON のスクリプト機能を使用すると、これらの手順を自動化できます。詳細については、JON ドキュメンテーションを参照してください。

[Report a bug](#)

25.3.7. 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、「[永続的な設定パラメーター](#)」の `JON_` で始まるパラメーターを参照してください。

[Report a bug](#)

25.4. ユーザースクリプト設定

25.4.1. 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。この内容は、`/etc/sysconfig/jbossas` と `/etc/sysconfig/jon-agent-ec2` に書き込まれます。

表25.2 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

名前	説明	デフォルト
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	<p>クラスターメンバーノードの ID。 クラスタリングの場合のみ使用 されます。許可される値は次のと おりです (順番どおり)。</p> <ul style="list-style-type: none"> ● 0 から 1023 までの範囲 にある有効なクラスター 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。 <p>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。</p>	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
JON_SERVER_ADDR	登録するのに使用する JON サー バーのホスト名または IP。これは 登録のためにのみ使用され、登録 後、エージェントは JON クラス ター内の他のサーバーと通信でき ます。	該当なし

名前	説明	デフォルト
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、サーバーが複数のパブリックアドレス (VPN など) を持つ場合に使用されます。	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。	該当なし
JBOSS_SERVER_CONFIG	<p>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。それ以外の場合は、次のようになります。</p> <ul style="list-style-type: none"> ● 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 が指定された場合は、standalone-mod_cluster-ec2-ha.xml が選択されます。 ● 最初の 2 つのオプションのどれも使用されない場合は、standalone.xml ファイルが使用されます。 ● standalone-full.xml に設定することも可能です。 	standalone.xml 、 standalone-full.xml 、 standalone-ec2-ha.xml 、 standalone-mod_cluster-ec2-ha.xml 、 domain-ec2.xml (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。

名前	説明	デフォルト
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)。これは一意である必要があります。	HOSTNAME 環境変数の値。
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。これが提供されない場合は、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依存します)。
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

25.4.2. カスタムスクリプトパラメーター

概要

User Data: フィールドのユーザーカスタマイズセクションでは、次のパラメーターを使用できます。

表25.3 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 (<code>/var/lib/jbossas/standalone/deployments/</code> など)。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、デプロイディレクトリーの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お勧めします。
JBOSS_CONFIG_DIR	アクティブプロファイルの設定ディレクトリー (<code>/var/lib/jbossas/standalone/configuration</code> など)。
JBOSS_HOST_CONFIG	アクティブホスト設定ファイルの名前 (<code>host-master.xml</code> など)。Red Hat は、設定ファイルを編集する代わりに、管理コンソールまたは CLI ツールを使用してサーバーを設定することをお勧めします。
JBOSS_SERVER_CONFIG	アクティブサーバー設定ファイルの名前 (<code>standalone-ec2-ha.xml</code> など)。Red Hat は、設定ファイルを編集する代わりに、管理コンソールまたは CLI ツールを使用してサーバーを設定することをお勧めします。
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)。

[Report a bug](#)

25.5. トラブルシューティング

25.5.1. Amazon EC2 のトラブルシューティング

EC2 では、各インスタンスに対して Alarm Status が提供され、インスタンスの深刻な誤動作が示されますが、このようなアラームがないと、インスタンスが正しく起動され、サービスが適切に実行される保証はありません。Amazon CloudWatch とそのカスタムメトリック機能を使用すると、インスタンスサービスの状態を監視できますが、企業向けの管理ソリューションの使用が推奨されます。

トラブルシューティングを簡易化するため、Red Hat は JBoss Operations Network (JON) を使用した EC2 インスタンスの管理を推奨します。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 の監視についての詳細については、「[AMI 監視](#)」を参照してください。

[Report a bug](#)

25.5.2. 診断情報

JBoss Operations Network、Amazon CloudWatch、または手動による検査で検出される問題の場合、診断情報の共通ソースは以下のとおりです。

- **/var/log/jboss_user-data.out** は、jboss-ec2-eap 初期化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。
- また、**/var/log** には、マシンの起動時に収集され、JBoss EAP、httpd、および他のほとんどサービスから収集されたすべてのログが含まれます。

これらのファイルへのアクセスは SSH セッションを介してのみ利用可能です。Amazon EC2 インスタンスを使用して 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。

[Report a bug](#)

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

- このタスクを始める前に、カスタマーポータルのアカウントを作成する必要があります。<https://access.redhat.com> へ移動し、右上端にある **登録** リンクをクリックしアカウントを作成してください。

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

- <https://access.redhat.com> へ移動し、右上の **ログイン** リンクをクリックします。認証情報を入力し、**ログイン** をクリックします。

結果

RHN へログインし、<https://access.redhat.com> のメイン Web ページに戻ります。

- ダウンロード ページへ移動します。**
以下のオプションのいずれかを使い、**ダウンロード ページ**へ移動します。
 - 上部のナビゲーションバーの**ダウンロード** リンクをクリックします。
 - <https://access.redhat.com/downloads/> へ直接アクセスします。
- 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。
- 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。**
選択した製品に従い、オペレーティングシステムやアーキテクチャー別に ZIP アーカイブ、RPM、ネイティブインストーラーを選んでいただけます。ファイル名あるいは、ダウンロードしたいファイルの右側にある **ダウンロード** 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。

[Report a bug](#)

A.2. RED HAT ENTERPRISE LINUX でのデフォルト JDK の設定

複数の Java Development Kits (JDK) を Red Hat Enterprise Linux システムにインストール可能です。このタスクでは、現在の環境で使用される JDK の指定方法を説明します。**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

この手順を必要としない場合もあります。Red Hat Enterprise Linux は OpenJDK 1.6 をデフォルトオプションとして使用します。OpenJDK 1.6 をデフォルトオプションとして使用し、システムが正常に動作している場合は、使用する JDK を手作業で指定する必要はありません。

前提条件

- このタスクの実行には、スーパーユーザー権限が必要です。スーパーユーザー権限を取得するには、直接ログインするか、**sudo** コマンドを使用します。

手順A.2 デフォルト JDK の設定

- 希望の **java** および **javac** バイナリのパスを決定します。

rpm -ql packagename |grep bin コマンドを使い、RPM からインストールしたバイナリの場所を検索します。Red Hat Enterprise Linux 32 ビットシステムでは、**java** および **javac** バイナリのデフォルトの場所は以下のとおりです。

表A.1 java および javac バイナリのデフォルトの場所

JDK	パス (Path)
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

- 個別に代替の JDK を設定します。

/usr/sbin/alternatives --config java または **/usr/sbin/alternatives --config javac** コマンドを実行し、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。画面の指示に従います。

- 任意の設定: **java_sdk_1.6.0** の代替を設定します。

Oracle JDK を使用する場合、**java_sdk_1.6.0** の代替を設定する必要もあります。コマンド **/usr/sbin/alternatives --config java_sdk_1.6.0** を使用します。通常、正しいパスは **/usr/lib/jvm/java-1.6.0-sun** となります。ファイルを一覧表示すると検証できます。

結果

代替の JDK が選択され、有効になります。

[Report a bug](#)

付録B 改訂履歴

改訂 6.3.0-51.2 翻訳ファイルを XML ソースバージョン 6.3.0-51 と同期	Tue Aug 20 2019	Ludek Janda
改訂 6.3.0-51.1 翻訳ファイルを XML ソースバージョン 6.3.0-51 と同期	Wed Jul 31 2019	Ludek Janda
改訂 6.3.0-51 Red Hat JBoss Enterprise Application Platform 6.3.0 連続リリース	Tuesday November 18 2014	Russell Dickenson
改訂 6.3.0-38 Red Hat JBoss Enterprise Application Platform 6.3.0.GA	Monday August 4 2014	Sande Gilda