



# Red Hat Enterprise Linux 7

## High Availability Add-On リファレンス

High Availability Add-On の設定および管理のためのリファレンスガイド



# Red Hat Enterprise Linux 7 High Availability Add-On リファレンス

---

High Availability Add-On の設定および管理のためのリファレンスガイド

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

## 法律上の通知

Copyright © 2023 | You need to change the HOLDER entity in the en-US/High\_Availability\_Add-On\_Reference.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

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<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

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

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

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

Node.js<sup>®</sup> 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<sup>®</sup> 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

Red Hat High Availability Add-On リファレンス は、Red Hat Enterprise Linux 7 向けの Red Hat High Availability Add-On をインストール、設定、および管理するための参考情報を提供します。

## 目次

<b>第1章 RED HAT HIGH AVAILABILITY ADD-ON 設定および管理リファレンスの概要</b>	<b>7</b>
1.1. 新機能と変更点	7
1.1.1. Red Hat Enterprise Linux 7.1 の新機能および変更された機能	7
1.1.2. Red Hat Enterprise Linux 7.2 の新機能および変更された機能	7
1.1.3. Red Hat Enterprise Linux 7.3 の新機能および変更された機能	8
1.1.4. Red Hat Enterprise Linux 7.4 の新機能および変更された機能	9
1.1.5. Red Hat Enterprise Linux 7.5 の新機能および変更された機能	9
1.1.6. Red Hat Enterprise Linux 7.8 の新機能および変更された機能	9
1.2. PACEMAKER 設定ツールのインストール	10
1.3. ファイアウォールでクラスターコンポーネントを許可する IPTABLES 設定	10
1.4. クラスターと PACEMAKER の設定ファイル	11
1.5. クラスター設定の注意事項	12
1.6. RED HAT ENTERPRISE LINUX HIGH AVAILABILITY クラスターの更新	12
1.7. RHEL クラスターでの VM のライブ移行についての問題	13
<b>第2章 PCSD WEB UI</b>	<b>14</b>
2.1. PCSD WEB UI の設定	14
2.2. PCSD WEB UI を用いたクラスターの作成	15
2.2.1. 高度なクラスター設定オプション	15
2.2.2. クラスター管理パーミッションの設定	17
2.3. クラスターコンポーネントの設定	18
2.3.1. クラスターノード	18
2.3.2. クラスターリソース	19
2.3.3. フェンスデバイス	19
2.3.4. ACL の設定	19
2.3.5. クラスターのプロパティー	20
2.4. 高可用性 PCSD WEB UI の設定	20
<b>第3章 PCS コマンドラインインターフェイス</b>	<b>22</b>
3.1. PCS コマンド	22
3.2. PCS の使用に関するヘルプ表示	22
3.3. RAW クラスター設定の表示	23
3.4. 設定の変更をファイルに保存	23
3.5. 状態の表示	23
3.6. 全クラスター設定の表示	24
3.7. 現在の PCS バージョンの表示	24
3.8. クラスター設定のバックアップおよび復元	24
<b>第4章 クラスターの作成と管理</b>	<b>25</b>
4.1. クラスターの作成	25
4.1.1. pcsd デーモンの開始	25
4.1.2. クラスターノードの認証	25
4.1.3. クラスターノードの設定と起動	26
4.2. クラスターのタイムアウト値の設定	26
4.3. 冗長リングプロトコル (RRP) の設定	27
4.4. クラスターノードの管理	27
4.4.1. クラスターサービスの停止	27
4.4.2. クラスターサービスの有効化および無効化	27
4.4.3. クラスターノードの追加	28
4.4.4. クラスターノードの削除	29
4.4.5. スタンバイモード	29
4.5. ユーザーのパーミッション設定	30

4.5.1. ネットワーク上でのノードアクセスのパーミッション設定	30
4.5.2. ACL を使用したローカルパーミッションの設定	30
4.6. クラスター設定の削除	32
4.7. クラスターの状態表示	32
4.8. クラスターメンテナンス	33
<b>第5章 フェンス機能: STONITH の設定</b>	<b>34</b>
5.1. STONITH (フェンス) エージェント	34
5.2. フェンスデバイスの一般的なプロパティ	34
5.3. デバイス固有のフェンスオプションの表示	35
5.4. フェンスデバイスの作成	36
5.5. フェンスデバイスの表示	36
5.6. フェンスデバイスの修正と削除	36
5.7. フェンスデバイスが接続されているノードの管理	37
5.8. その他のフェンス設定オプション	37
5.9. フェンスレベルの設定	42
5.10. 冗長電源のフェンシング設定	43
5.11. 統合フェンスデバイスで使用する ACPI の設定	44
5.11.1. BIOS で ACPI Soft-Off を無効化	45
5.11.2. logind.conf ファイルで ACPI Soft-Off の無効化	46
5.11.3. GRUB 2 ファイルでの ACPI の完全な無効化	46
5.12. フェンスデバイスのテスト	47
<b>第6章 クラスターリソースの設定</b>	<b>50</b>
6.1. リソースの作成	50
6.2. リソースのプロパティ	51
6.3. リソース固有のパラメーター	51
6.4. リソースのメタオプション	52
6.5. リソースグループ	55
6.5.1. グループオプション	56
6.5.2. グループの Stickiness (粘着性)	56
6.6. リソースの動作	56
6.6.1. リソース操作の設定	57
6.6.2. グローバルリソース操作のデフォルトの設定	58
6.7. 設定されているリソースの表示	59
6.8. リソースパラメーターの変更	60
6.9. 複数のモニターリング動作	60
6.10. クラスターリソースの有効化と無効化	61
6.11. クラスターリソースのクリーンアップ	61
<b>第7章 リソースの制約</b>	<b>62</b>
7.1. 場所の制約	62
7.1.1. 基本的な場所の制約	62
7.1.2. 高度な場所の制約	63
7.1.3. ルールを使用したリソースの場所の確定	64
7.1.4. 場所の制約ストラテジー	65
7.1.4.1. オプトインクラスターの設定	65
7.1.4.2. オプトアウトクラスターの設定	66
7.1.5. 現在のノードを優先させるリソースの設定	66
7.2. 順序の制約	67
7.2.1. 強制的な順序付け	67
7.2.2. 勧告的な順序付け	68
7.2.3. 順序付けされたリソースセット	68
7.2.4. 順序の制約からリソースを削除	69

7.3. リソースのコロケーション	69
7.3.1. 強制的な配置	70
7.3.2. 勧告的な配置	70
7.3.3. 複数リソースのコロケート	70
7.3.4. コロケーション制約の削除	71
7.4. 制約の表示	72
<b>第8章 クラスターリソースの管理</b>	<b>73</b>
8.1. リソースを手作業で移動する	73
8.1.1. 現在のノードからリソースを移動	73
8.1.2. リソースの優先ノードへの移動	74
8.2. 障害発生によるリソースの移動	74
8.3. 接続状態変更によるリソースの移動	75
8.4. クラスターリソースの有効化、無効化、および禁止	76
8.5. モニター操作の無効化	77
8.6. 管理リソース	77
<b>第9章 高度な設定</b>	<b>79</b>
9.1. リソースのクローン	79
9.1.1. クローンリソースの作成と削除	79
9.1.2. 制約のクローン作成	81
9.1.3. 粘着性のクローン作成	81
9.2. 多状態のリソース: 複数モードのリソース	82
9.2.1. 多状態リソースの監視	82
9.2.2. 多状態制約	83
9.2.3. 多状態の粘着性 (Stickiness)	83
9.3. リソースとしての仮想ドメインの設定	83
9.4. PACEMAKER_REMOTE サービス	85
9.4.1. ホストとゲストの認証	87
9.4.2. ゲストノードリソースのオプション	87
9.4.3. リモートノードリソースのオプション	87
9.4.4. ポートのデフォルトの場所の変更	88
9.4.5. 設定の概要: KVM ゲストノード	88
9.4.6. 設定概要: リモートノード (Red Hat Enterprise Linux 7.4)	90
9.4.7. 設定概要: リモートノード (Red Hat Enterprise Linux 7.3 以前)	91
9.4.8. システムアップグレードおよび pacemaker_remote	92
9.5. DOCKER コンテナの PACEMAKER サポート (テクノロジープレビュー)	93
9.5.1. Pacemaker バンドルリソースの設定	94
9.5.1.1. Docker パラメーター	94
9.5.1.2. (バンドルネットワークパラメーター	95
9.5.1.3. バンドルストレージパラメーター	97
9.5.2. バンドルでの Pacemaker リソースの設定	98
9.5.2.1. ノード属性とバンドルリソース	98
9.5.2.2. メタデータ属性とバンドルリソース	99
9.5.3. Pacemaker バンドルの制限	99
9.5.4. Pacemaker バンドル設定の例	99
9.6. 使用と配置ストラテジー	101
9.6.1. 使用率属性	101
9.6.2. 配置ストラテジー	102
9.6.3. リソースの割り当て	102
9.6.3.1. ノードの優先順位	102
9.6.3.2. ノードの容量	103
9.6.3.3. リソースの割り当て設定	103

9.6.4. リソース配置ストラテジーガイドライン	104
9.6.5. NodeUtilization リソースエージェント (Red Hat Enterprise Linux 7.4 以降)	104
9.7. PACEMAKER で管理されていないリソースの依存関係の起動順の設定 (RED HAT ENTERPRISE LINUX 7.4 以降)	104
9.8. SNMP での PACEMAKER クラスターを照会 (RED HAT ENTERPRISE LINUX 7.5 以降)	105
9.9. クリーンノードのシャットダウンで停止するようにリソースを設定 (RED HAT ENTERPRISE LINUX 7.8 以降)	107
9.9.1. クリーンノードシャットダウンで停止するようにリソースを設定するためのクラスタープロパティ	107
9.9.2. shutdown-lock クラスタープロパティの設定	108
<b>第10章 クラスタークォーラム</b>	<b>111</b>
10.1. クォーラムオプションの設定	111
10.2. クォーラム管理コマンド (RED HAT ENTERPRISE LINUX 7.3 以降)	112
10.3. クォーラムオプションの変更 (RED HAT ENTERPRISE LINUX 7.3 以降)	112
10.4. クォーラムアンブロック (QUORUM UNBLOCK) コマンド	113
10.5. クォーラムデバイス	113
10.5.1. クォーラムデバイスパッケージのインストール	114
10.5.2. クォーラムデバイスの設定	114
10.5.3. クォーラムデバイスサービスの管理	118
10.5.4. クラスターでのクォーラムデバイス設定の管理	118
10.5.4.1. クォーラムデバイス設定の変更	119
10.5.4.2. クォーラムデバイスの削除	119
10.5.4.3. クォーラムデバイスの破棄	120
<b>第11章 PACEMAKER ルール</b>	<b>121</b>
11.1. ノード属性の式	121
11.2. 時刻と日付ベースの式	122
11.3. 日付の詳細	123
11.4. 期間	124
11.5. PCS でのルールの設定	124
<b>第12章 PACEMAKER クラスターのプロパティ</b>	<b>125</b>
12.1. クラスタープロパティとオプションの要約	125
12.2. クラスターのプロパティの設定と削除	128
12.3. クラスタープロパティ設定のクエリー	128
<b>第13章 クラスターイベントのスクリプトのトリガー</b>	<b>130</b>
13.1. PACEMAKER アラートエージェント (RED HAT ENTERPRISE LINUX 7.3 以降)	130
13.1.1. サンプルアラートエージェントの使用	130
13.1.2. アラートの作成	131
13.1.3. アラートの表示、編集、および削除	132
13.1.4. アラートの受信側	132
13.1.5. アラートメタオプション	133
13.1.6. アラート設定コマンドの例	133
13.1.7. アラートエージェントの作成	135
13.2. モニタリングのリソースを使ったイベント通知	137
<b>第14章 PACEMAKER を用いたマルチサイトクラスターの設定</b>	<b>140</b>
<b>付録A OCF 戻りコード</b>	<b>144</b>
<b>付録B RED HAT ENTERPRISE LINUX 6 および RED HAT ENTERPRISE LINUX 7 でのクラスターの作成</b>	<b>147</b>
B.1. クラスター作成 - RGMANAGER と PACEMAKER	147
B.2. RED HAT ENTERPRISE LINUX 6 および RED HAT ENTERPRISE LINUX 7 での PACEMAKER のインストール	



付録C 更新履歴 ..... 153

索引 ..... 153



# 第1章 RED HAT HIGH AVAILABILITY ADD-ON 設定および管理リファレンスの概要

本章では、Pacemaker を使用する Red Hat High Availability Add-On がサポートするオプションと機能について説明します。ステップごとの基本設定の例は『Red Hat High Availability Add-On の管理』を参照してください。

Red Hat High Availability Add-On クラスターを設定するには、**pcs** 設定インターフェイスまたは **pcsd** GUI インターフェイスを使用します。

## 1.1. 新機能と変更点

本セクションでは、Red Hat Enterprise Linux 7 の初回リリース以降に追加された Red Hat High Availability Add-On の新機能を取り上げます。

### 1.1.1. Red Hat Enterprise Linux 7.1 の新機能および変更された機能

Red Hat Enterprise Linux 7.1 には、ドキュメントや機能を対象とする以下の更新および変更が含まれています。

- 「[クラスターリソースのクリーンアップ](#)」に記載されているように、**pcs resource cleanup** コマンドで、すべてのリソースの状態と **failcount** をリセットできるようになりました。
- 「[リソースを手作業で移動する](#)」に記載されているように、**pcs resource move** コマンドの **lifetime** パラメーターを指定できるようになりました。
- Red Hat Enterprise Linux 7.1 以降では、**pcs acl** コマンドを使用してローカルユーザーのパーミッションを設定し、アクセス制御リスト (ACL) を使用してクラスター設定への読み取り専用または読み書きアクセスを許可できます。ACL の詳細は、「[ユーザーのパーミッション設定](#)」を参照してください。
- 「[順序付けされたリソースセット](#)」 および 「[リソースのコロケーション](#)」 が大幅に更新および明確化されました。
- 「[リソースの作成](#)」 に、**pcs resource create** コマンドの **disabled** パラメーターに関する内容が追加され、作成されたリソースは自動的に起動しないことが明記されました。
- 「[クォーラムオプションの設定](#)」 に、クォーラムの確立時にクラスターがすべてのノードを待たないようにする **cluster quorum unblock** 機能の説明が追加されました。
- 「[リソースの作成](#)」 に、リソースグループの順序付けを設定するために使用できる **pcs resource create** コマンドの **before** および **after** パラメーターの説明が追加されました。
- Red Hat Enterprise Linux 7.1 リリース以降ではクラスター設定を tarball にバックアップし、**pcs config** コマンドで **backup** および **restore** オプションを使用してバックアップからすべてのノードのクラスター設定ファイルを復元できるようになりました。この機能の詳細は「[クラスター設定のバックアップおよび復元](#)」を参照してください。
- 内容を明確にするため本書全体に小変更が加えられました。

### 1.1.2. Red Hat Enterprise Linux 7.2 の新機能および変更された機能

Red Hat Enterprise Linux 7.2 ではドキュメントと機能が以下のように更新/変更されています。

- **pcs resource relocate run** コマンドを使用して、現在のクラスター状態、制約、リソースの場所、およびその他の設定によって決定される優先ノードにリソースを移動できるようになりました。このコマンドの詳細は、「[リソースの優先ノードへの移動](#)」を参照してください。
- 外部プログラムを実行してクラスター通知の処理を判断するために **ClusterMon** リソースを設定する方法をより明確に説明するため、「[モニターリングのリソースを使ったイベント通知](#)」が変更および拡大されました。
- 冗長な電源供給用のフェンスを設定する場合に各デバイスを1度のみ設定する必要があり、ノードのフェンシングには両方のデバイスが必要になることを指定する必要があります。冗長な電源供給にフェンスを設定する方法の詳細は、「[冗長電源のフェンシング設定](#)」を参照してください。
- このドキュメントの「[クラスターノードの追加](#)」に、ノードを既存のクラスターに追加する手順が追加されました。
- [表7.1「簡単な場所の制約オプション」](#)の説明にあるように、新しい **resource-discovery** の場所の制約オプションにより、Pacemaker が指定されたリソースのノード上でリソースの検索を実行すべきかどうかを指定できるようになりました。
- ドキュメント全体にわたり、記載内容の明確化を図り、若干の修正を加えました。

### 1.1.3. Red Hat Enterprise Linux 7.3 の新機能および変更された機能

Red Hat Enterprise Linux 7.3 ではドキュメントと機能が以下のように更新、変更されています。

- このバージョンでは、「[pacemaker\\_remote サービス](#)」全体が書き直されました。
- アラートエージェントを使用して Pacemaker アラートを設定できます。アラートエージェントは、リソース設定と操作を処理するためにクラスター呼び出しのリソースエージェントと同様にクラスターが呼び出す外部プログラムです。Pacemaker アラートエージェントの説明は「[Pacemaker アラートエージェント \(Red Hat Enterprise Linux 7.3 以降\)](#)」を参照してください。
- 本リリースでは新しいクォーラム管理コマンドがサポートされ、クォーラムの状態を表示し、**expected\_votes** パラメーターを変更することができます。これらのコマンドの説明は「[クォーラム管理コマンド \(Red Hat Enterprise Linux 7.3 以降\)](#)」を参照してください。
- 「[クォーラムオプションの変更 \(Red Hat Enterprise Linux 7.3 以降\)](#)」の説明に従って、**pcs quorum update** コマンドを使用してクラスターの一般的なクォーラムオプションを変更できるようになりました。
- クラスターのサードパーティー判別デバイスとして動作する個別のクォーラムデバイスを設定できます。この機能は主に、標準のクォーラムルールが許可するよりも多くのノード障害をクラスターで維持できるようにするために使用されます。この機能はテクニカルレビューとしてのみ提供されます。クォーラムデバイスの説明は「[クォーラムデバイス](#)」を参照してください。
- Red Hat Enterprise Linux 7.3 には、Booth クラスターチケットマネージャーを使用して複数のサイトにまたがる高可用性クラスターを設定する機能が提供されます。この機能はテクニカルレビューとしてのみ提供されます。Booth クラスターチケットマネージャーの説明は [14 章 Pacemaker を用いたマルチサイトクラスターの設定](#) を参照してください。
- **pacemaker\_remote** サービスを実行している KVM ゲストノードを設定する場合、グループにゲストノードを含めることができます。これにより、ストレージデバイス、ファイルシステムおよび VM をグループ化できます。KVM ゲストノードの設定に関する詳細は「[設定の概要: KVM ゲストノード](#)」を参照してください。

さらに、ドキュメント全体にわたり記載内容の明確化を図り、若干の修正を加えました。

#### 1.1.4. Red Hat Enterprise Linux 7.4 の新機能および変更された機能

Red Hat Enterprise Linux 7.4 では、ドキュメントと機能が以下のように更新、変更されています。

- Red Hat Enterprise Linux 7.4 には、Booth クラスターチケットマネージャーを使用して複数のサイトにまたがる高可用性クラスターを設定する機能が完全に提供されます。Booth クラスターチケットマネージャーの説明は [14章Pacemaker を用いたマルチサイトクラスターの設定](#) を参照してください。
- Red Hat Enterprise Linux 7.4 は、個別のクォーラムを設定する機能に完全に対応しています。この機能は主に、標準のクォーラムルールが許可するよりも多くのノード障害をクラスターで維持できるようにするために使用されます。クォーラムデバイスの説明は [「クォーラムデバイス」](#) を参照してください。
- ノード名で適用した正規表現と、ノード属性とその値によってフェンシングトポロジでノードを指定できるようになりました。フェンシングレベルの説明は、[「フェンスレベルの設定」](#) を参照してください。
- Red Hat Enterprise Linux 7.4 は、**NodeUtilization** リソースエージェントに対応しています。これは、利用可能な CPU、ホストメモリの可用性、ハイパーバイザーメモリの可用性に関するシステムパラメーターを検出し、これらのパラメーターを CIB に追加します。このリソースエージェントの詳細は、[「NodeUtilization リソースエージェント \(Red Hat Enterprise Linux 7.4 以降\)」](#) を参照してください。
- Red Hat Enterprise Linux 7.4 では、**cluster node add-guest** と **cluster node remove-guest** コマンドは、**remote-node add** と **cluster remote-node remove** コマンドに代わります。**pcs cluster node add-guest** コマンドは、ゲストノードの **authkey** を設定し、**pcs cluster node add-remote** は、リモートノードの **authkey** を設定します。更新したゲストとリモートノード設定手順は、[「リソースとしての仮想ドメインの設定」](#) を参照してください。
- Red Hat Enterprise Linux 7.4 は **systemd resource-agents-deps** ターゲットに対応しています。これにより、[「Pacemaker で管理されていないリソースの依存関係の起動順の設定 \(Red Hat Enterprise Linux 7.4 以降\)」](#) で説明しているように、クラスターにより管理されない依存関係を持つリソースを含むクラスターに適切な起動順序を設定できるようになります。
- マスター/スレーブクローンとしてリソースを作成するコマンドの形式は、このリリースで変更されています。マスター/スレーブクローンの作成の説明は、[「多状態のリソース: 複数モードのリソース」](#) を参照してください。

#### 1.1.5. Red Hat Enterprise Linux 7.5 の新機能および変更された機能

Red Hat Enterprise Linux 7.5 では、ドキュメントと機能が以下のように更新、変更されています。

- Red Hat Enterprise Linux 7.5 では、**pcs\_snmp\_agent** デーモンを使用して、SNMP でデータについて Pacemaker クラスターを照会できます。SNMP でのクラスター照会は、[「SNMP での Pacemaker クラスターを照会 \(Red Hat Enterprise Linux 7.5 以降\)」](#) を参照してください。

#### 1.1.6. Red Hat Enterprise Linux 7.8 の新機能および変更された機能

Red Hat Enterprise Linux 7.8 では、ドキュメントと機能が以下のように更新、変更されています。

- Red Hat Enterprise Linux 7.8 以降では、ノードが正常にシャットダウンすると、ノードに接続されているリソースがノードにロックされ、シャットダウンしたノードがクラスターに再度参加するときに再び起動するまで、他の場所で起動できないように、Pacemaker を設定できま

す。これにより、ノードのリソースをクラスター内の他のノードにフェールオーバーせずに、サービスの停止が許容できるメンテナンスウィンドウ中にノードの電源を切ることができます。ノードの正常なシャットダウン時に停止したままになるようにリソースを設定する方法は、「[クリーンノードのシャットダウンで停止するようにリソースを設定 \(Red Hat Enterprise Linux 7.8 以降\)](#)」を参照してください。

## 1.2. PACEMAKER 設定ツールのインストール

以下の **yum install** コマンドを使って Red Hat High Availability Add-On ソフトウェアのパッケージおよび利用可能なフェンスエージェントを High Availability チャンネルからインストールします。

```
# yum install pcs pacemaker fence-agents-all
```

このコマンドの代わりに以下のコマンドを実行すると、Red Hat High Availability Add-On ソフトウェアパッケージと必要なフェンスエージェントのみをインストールできます。

```
# yum install pcs pacemaker fence-agents-model
```

以下のコマンドは、利用できるフェンスエージェントの一覧を表示します。

```
# rpm -q -a | grep fence
fence-agents-rhevm-4.0.2-3.el7.x86_64
fence-agents-ilo-mp-4.0.2-3.el7.x86_64
fence-agents-ipmilan-4.0.2-3.el7.x86_64
...
```

**lvm2-cluster** と **gfs2-utils** のパッケージは ResilientStorage チャンネルの一部になります。必要に応じて次のコマンドでインストールを行ってください。

```
# yum install lvm2-cluster gfs2-utils
```



### 警告

Red Hat High Availability Add-On パッケージをインストールしたら、自動的に何もインストールされないように、ソフトウェア更新設定を行う必要があります。実行中のクラスターにインストールすると、予期しない動作が発生する可能性があります。

## 1.3. ファイアウォールでクラスターコンポーネントを許可する IPTABLES 設定



## 注記

クラスターコンポーネントの理想的なファイアウォール設定は、ローカル環境によって異なります。ここでは、ノードに複数のネットワークインターフェイスがあるかどうか、またはオフホストのファイアウォールがあるかどうかを検討しないといけない場合があります。この例では、Pacemaker クラスターで通常必要となるポートを開きますが、ローカル条件に合わせて変更する必要があります。

表1.1「[High Availability Add-On で有効にするポート](#)」では、Red Hat High Availability Add-On で有効にするポートを示し、ポートの使用目的を説明します。また、以下のコマンドを実行して **firewalld** デーモンですべてのポートを有効化することもできます。

```
# firewall-cmd --permanent --add-service=high-availability
# firewall-cmd --add-service=high-availability
```

表1.1 High Availability Add-On で有効にするポート

ポート	必要になる場合
TCP 2224	すべてのノードで必須 ( <b>pcs</b> Web UI で必要となり、ノード間通信に必須)  任意のノードの <b>pcs</b> が、それ自体も含め、クラスター内のすべてのノードに通信できる方法でポート 2224 を開くことは重要です。Booth クラスターチケットマネージャーまたはクォーラムデバイスを使用する場合は、Booth Arbiter、クォーラムデバイスなどのすべての関連ホストで、ポート 2224 を開く必要があります。
TCP 3121	クラスターに Pacemaker リモートノードがある場合に、すべてのノードで必須です。  完全なクラスターノード上の Pacemaker の <b>cmd</b> デーモンは、ポート 3121 で Pacemaker リモートノードの <b>pacemaker remotd</b> デーモンに通信できます。クラスター通信に別のインターフェイスを使用する場合は、そのインターフェイスでポートを開くことのみが必要になります。少なくとも、ポートは、Pacemaker リモートノードの全クラスターノードに対して開いている必要があります。ユーザーは完全なノードとリモートノード間でホストを変換する可能性があるか、またはホストのネットワークを使用してコンテナ内でリモートノードを実行する可能性があるため、すべてのノードに対してポートを開くことは役に立ちます。ノード以外のホストにポートを開く必要はありません。
TCP 5403	<b>corosync-qnetd</b> で、クォーラムデバイスを使用するクォーラムデバイスホストで必須です。デフォルト値は、 <b>corosync-qnetd</b> コマンドの <b>-p</b> オプションで変更できます。
UDP 5404	<b>corosync</b> がマルチキャスト UDP に設定されている場合は、corosync ノードで必須
UDP 5405	すべての corosync ノードで必須 ( <b>corosync</b> で必要)
TCP 21064	DLM を必要とするリソースがクラスターに含まれる場合は、すべてのノードで必須です (例: <b>clvm</b> または <b>GFS2</b> )。
TCP 9929、UDP 9929	Booth チケットマネージャーを使用してマルチサイトクラスターを確立するときに、すべてのクラスターノード、および同じノードのいずれかからの接続に対して Booth arbitrator ノードで開いている必要があります。

## 1.4. クラスターと PACEMAKER の設定ファイル



Red Hat High Availability Add-On の設定ファイルは、**corosync.conf** および **cib.xml** です。

**corosync.conf** は、Pacemaker が構築されているクラスターマネージャーの **corosync** によって使用されるクラスターパラメーターを与えます。一般的に、**corosync.conf** は直接編集せずに、**pcs** または **pcsd** インターフェイスを使用してください。ただし、このファイルを直接編集する必要がある状況も考えられます。**corosync.conf** ファイルの編集は、[Editing the corosync.conf file in Red Hat Enterprise Linux 7](#)を参照してください。

**cib.xml** ファイルは、クラスターの設定、およびクラスターの全リソースの現在の状態を表す XML ファイルです。このファイルは Pacemaker のクラスター情報ベース (CIB) により使用されます。CIB の内容は、自動的にクラスター全体に同期されます。**cib.xml** ファイルは直接編集せず、代わりに **pcs** または **pcsd** インターフェイスを使用してください。

## 1.5. クラスター設定の注意事項

Red Hat High Availability Add-On クラスターの設定時、以下の注意事項を考慮する必要があります。

- RHEL 7.7 以降、Red Hat はノード数が 32 個を超えるクラスターデプロイメントをサポートしていません。しかし、**pacemaker\_remote** サービスを実行しているリモートノードを使用すると、この制限を超えた拡張が可能になります。**pacemaker\_remote** サービスの説明は「[pacemaker\\_remote サービス](#)」を参照してください。
- DHCP (Dynamic Host Configuration Protocol) を使用して **corosync** デーモンによって使用されるネットワークインターフェイスで IP アドレスを取得することはサポートされません。アドレスの更新中、DHCP クライアントは割り当てられたインターフェイスに対して定期的に IP アドレスを削除および再追加することができます。これにより、**corosync** によって接続障害が検出され、クラスターの他のノードからのフェンシングアクティビティによってハートビート接続性に **corosync** が使用されます。

## 1.6. RED HAT ENTERPRISE LINUX HIGH AVAILABILITY クラスターの更新

RHEL High Availability Add-On および Resilient Storage Add-On を設定するパッケージを、個別または一括で更新するには、以下に示す一般的な方法のいずれかを使用できます。

- **ローリング更新** - サービスからノードを、一度に1つずつ削除し、そのソフトウェアを更新してから、そのノードをクラスターに戻します。これにより、各ノードの更新中も、クラスターがサービスの提供とリソースの管理を継続できます。
- **クラスター全体の更新** - クラスター全体を停止し、更新をすべてのノードに適用してから、クラスターのバックアップを開始します。



### 警告

Red Hat Enterprise Linux の High Availability クラスターおよび Resilient Storage クラスターのソフトウェア更新手順を実行する場合は、更新を開始する前に、更新を行うノードがクラスターのアクティブなメンバーではないことを確認する必要があります。



これらの各方法の詳細な説明および更新手順は[Recommended Practices for Applying Software Updates to a RHEL High Availability or Resilient Storage Cluster](#)を参照してください。

## 1.7. RHEL クラスターでの VM のライブ移行についての問題

仮想化クラスターメンバーとの RHEL 高可用性クラスターのサポートポリシーの説明は、[Support Policies for RHEL High Availability Clusters - General Conditions with Virtualized Cluster Members](#) を参照してください。前述のように、Red Hat は、ハイパーバイザーまたはホスト全体のアクティブなクラスターノードのライブマイグレーションはサポート対象外です。ライブマイグレーションを実行する必要がある場合は、まず仮想マシンでクラスターサービスを停止して、クラスターからノードを削除し、移行後にクラスターを起動する必要があります。

以下の手順では、クラスターから仮想マシンを削除し、仮想マシンを移行し、クラスターに仮想マシンを復元する手順の概要を説明します。



### 注記

この手順を実行する前に、クラスターノードを削除するクラスタークォーラム (定足数) への影響を考慮してください。3つのノードクラスターがあり、1つのノードを削除する場合、クラスターが耐えられるのは、あと1つのノードエラーのみです。3つのノードクラスターの1つが既にダウンしている場合は、2つ目のノードを削除すると、クォーラムが失われます。

1. 移行する仮想マシンで実行しているリソースやソフトウェアの停止または移動を行う前に準備を行う必要がある場合は、以下の手順を実行します。
2. 管理リソースを VM から移動します。リソースの割り当てに関する特定の要件や条件がある場合は、正しいノードにリソースを配置するための新しい場所の制約を作成することを考慮してください。
3. VM をスタンバイモードにして、サービスで考慮されていないことや、残りのリソースが別の場所に再配置され、停止されるようにします。

```
# pcs cluster standby VM
```

4. 仮想マシンで以下のコマンドを実行して、仮想マシン上のクラスターソフトウェアを停止します。

```
# pcs cluster stop
```

5. 仮想マシンのライブマイグレーションを実行します。
6. 仮想マシンでクラスターサービスを起動します。

```
# pcs cluster start
```

7. VM をスタンバイモードから解除します。

```
# pcs cluster unstandby VM
```

8. VM をスタンバイモードにする前に一時的な場所の制約を作成した場合、これらの制約を調整または削除して、リソースが通常の優先場所に戻れるようにします。

## 第2章 PCSD WEB UI

本章では、**pcsd** Web UI を用いた Red Hat High Availability クラスターの設定について説明します。

### 2.1. PCSD WEB UI の設定

**pcsd** Web UI を使用してクラスターを設定するようシステムを設定するには、以下の手順に従います。

1. 「[Pacemaker 設定ツールのインストール](#)」の説明に従って Pacemaker 設定ツールをインストールします。
2. クラスターの一部である各ノードで、**passwd** コマンドを使用してユーザー **hacluster** のパスワードを設定します。各ノードに同じパスワードを使用してください。
3. 各ノードの **pcsd** デーモンを開始し、有効にします。

```
# systemctl start pcsd.service  
# systemctl enable pcsd.service
```

4. クラスターの1つのノードで、以下のコマンドを使用してクラスターを設定するノードを認証します。このコマンドを実行すると、**Username** と **Password** を指定するよう要求されます。**Username** には **hacluster** を指定してください。

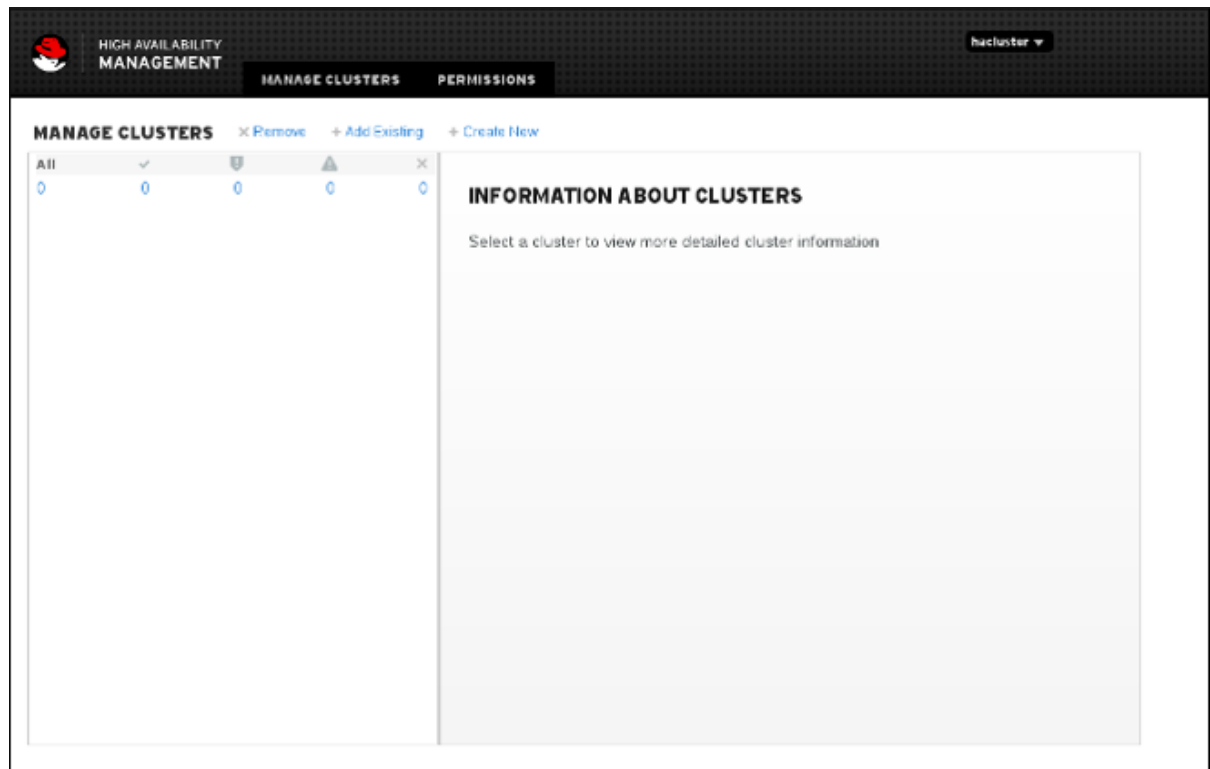
```
# pcs cluster auth node1 node2 ... nodeN
```

5. いずれかのシステムで、以下の URL をブラウザで開き、承認したノードの1つを指定します (**https** プロトコルを使用することに注意してください)。指定すると **pcsd** Web UI のログイン画面が表示されます。

```
https://nodename:2224
```

6. ユーザー **hacluster** としてログインします。これにより、[図2.1「クラスターの管理ページ」](#) に示されるように、**Manage Clusters** ページが表示されます。

図2.1 クラスターの管理ページ



[D]

## 2.2. PCSD WEB UI を用いたクラスターの作成

Manage Clusters ページでは新しいクラスターを作成できます。また、既存クラスターを Web UI へ追加したり、クラスターを Web UI から削除したりすることができます。

- クラスターを作成するには、**Create New** をクリックし、作成するクラスターとクラスターを設定するノードの名前を入力します。また、この画面では「[高度なクラスター設定オプション](#)」に記載されているクラスター通信のトランスポートメカニズムなどの高度なクラスターオプションを設定することもできます。クラスター情報の入力後、**Create Cluster** をクリックします。
- 既存のクラスターを Web UI に追加するには、**Add Existing** をクリックし、Web UI で管理したいクラスターのノードのホスト名または IP アドレスを入力します。

クラスターを作成または追加すると、**Manage Cluster** ページにクラスター名が表示されます。クラスターを選択すると、クラスターに関する情報が表示されます。



### 注記

**pcsd** Web UI を使用してクラスターを設定する場合、多くのオプションを説明するテキストの上にマウスを移動すると、**tooltip** 表示としてそのオプションの詳細を表示することができます。

### 2.2.1. 高度なクラスター設定オプション

クラスターの作成時、[図2.2「クラスターページの作成」](#)の説明のように **Advanced Options** をクリックすると追加のクラスターオプションを設定できます。表示されるオプションのテキスト上にマウスを移動すると、そのオプションの情報を確認できます。

各ノードのインターフェイスを指定すると、クラスターに冗長リングプロトコル (Redundant Ring Protocol) を設定できます。クラスターのトランスポートメカニズムのデフォルト値である **UDPU** の代わりに **UDP** を選択すると、冗長リングプロトコル (Redundant Ring Protocol) 設定の表示が変更されます。

図2.2 クラスターページの作成

Create Cluster

×

Enter the hostnames of the nodes you would like to use to create a cluster:

Cluster Name:

Node 1:

Node 2:

Node 3:

[More nodes...](#)

▼ Advanced Options:

Transport:

Wait for All: ☐

Auto Tie Breaker: ☐

Last Man Standing: ☐

Last Man Standing Window:  ms

Use IPv6: ☐

Token Timeout:  ms

Token Timeout Coefficient:  ms

Join Timeout:  ms

Consensus Timeout:  ms

Missed Messages Count:

Failures Count:

Redundant Ring Protocol settings for UDPU transport:

Node 1 (Ring 1):

Node 2 (Ring 1):

Node 3 (Ring 1):

Create Cluster

Cancel

[D]

## 2.2.2. クラスター管理パーミッションの設定

ユーザーに付与できるクラスターパーミッションには、以下の2つのセットがあります。

- Web UI を使用してクラスターを管理するためのパーミッション。ネットワーク経由でノードに接続する **pcs** コマンドを実行するパーミッションも付与されます。本セクションでは、Web UI でこのパーミッションを設定する方法を説明します。
- ACL を使用し、クラスター設定への読み取り専用アクセスまたは読み書きアクセスをローカルユーザーに許可するパーミッション。Web UI で ACL を設定する方法は、「[ACL の設定](#)」を参照してください。

ユーザーのパーミッションの詳細は、「[ユーザーのパーミッション設定](#)」を参照してください。

グループ **haclient** にユーザーを追加することで、ユーザー **hacluster** 以外の特定のユーザーにパーミッションを付与し、Web UI でクラスターを管理し、ネットワーク経由でノードに接続する **pcs** コマンドを実行できます。次に、**Manage Clusters** ページの **Permissions** タブをクリックし、表示された画面でパーミッションを設定することで、グループ **haclient** の個別のメンバーにパーミッションセットを設定できます。この画面では、グループのパーミッションを設定することもできます。

以下のパーミッションを付与できます。

- 読み取りパーミッション (クラスター設定の表示)
- 書き込みパーミッション (パーミッションおよび ACL を除くクラスター設定の変更)
- 付与パーミッション (クラスターパーミッションおよび ACL の変更)
- フルパーミッション (ノードの追加や削除などのクラスターへの無制限アクセス、およびキーや証明書へのアクセス)

## 2.3. クラスターコンポーネントの設定

クラスターのコンポーネントと属性を設定するには、**Manage Clusters** 画面に表示されるクラスターの名前をクリックします。これにより、「[クラスターノード](#)」に示されるように、**Nodes** ページが表示されます。[図2.3 「クラスターコンポーネントのメニュー」](#)の説明どおり、このページの上部には以下のエントリーが含まれるメニューが表示されます。

- **Nodes** ([「クラスターノード」](#) を参照)
- **Resources** ([「クラスターリソース」](#) を参照)
- **Fence Devices** ([「フェンスデバイス」](#) を参照)
- **ACL** ([「ACL の設定」](#) を参照)
- **Cluster Properties** ([「クラスターのプロパティ」](#) を参照)

図2.3 クラスターコンポーネントのメニュー



[D]

### 2.3.1. クラスターノード

クラスター管理ページの上部にあるメニューから **ノード オプション** を選択すると、現在設定されてい

るノードと、現在選択されているノードの状態 (ノードで実行しているリソースや、リソースの場所設定など) が表示されます。このページは、**Manage Clusters** 画面でクラスターを選択すると表示されるデフォルトページです。

このページでノードを追加または削除することができ、ノードを起動、停止、および再起動することができます。スタンバイモードの詳細は、「[スタンバイモード](#)」を参照してください。

また、**Configure Fencing** を選択することで、「[フェンスデバイス](#)」で説明されているように、このページで直接フェンスデバイスを設定することもできます。

### 2.3.2. クラスターリソース

クラスター管理ページの上部にあるメニューから **Resources** オプションを選択すると、クラスターの現在設定されているリソースがリソースグループに応じて表示されます。グループまたはリソースを選択すると、そのグループまたはリソースの属性が表示されます。

この画面では、リソースの追加または削除、既存リソースの設定の編集、およびリソースグループの作成を行うことができます。

新しいリソースをクラスターに追加するには、**Add** をクリックします。**Add Resource** 画面が開きます。ドロップダウンの **タイプ** メニューからリソースタイプを選択すると、そのリソースに指定する必要がある引数がメニューに表示されます。**Optional Arguments** をクリックすると、定義するリソースに指定できる任意の引数を表示できます。作成するリソースのパラメーターを入力したら、**Create Resource** をクリックします。

リソースの引数を設定する際に、引数の簡単な説明がメニューに表示されます。カーソルをフィールドに移動すると、その引数のヘルプが表示されます。

リソースは、クローンされたリソースまたはマスター/スレーブリソースとして定義できます。これらのリソースタイプの詳細は、[9章 高度な設定](#) を参照してください。

少なくとも1つのリソースを作成したら、リソースグループを作成できます。リソースグループの詳細は「[リソースグループ](#)」を参照してください。

リソースグループを作成するには、グループの一部になるリソースを **Resources** から選択し、**Create Group** をクリックします。**Create Group** 画面が表示されます。グループ名を入力して、**Create Group** をクリックします。これにより、リソースのグループ名を表示する **リソース** 画面に戻ります。リソースグループを作成したら、追加のリソースを作成または変更する際に、グループ名をリソースパラメーターとして指定できます。

### 2.3.3. フェンスデバイス

クラスター管理ページの上部にあるメニューから **Fence Devices** オプションを選択すると、**Fence Devices** 画面が表示され、現在設定されているフェンスデバイスが表示されます。

新しいフェンスデバイスをクラスターに追加するには、**Add** をクリックします。すると、**Add Fence Device** 画面が表示されます。ドロップダウンメニューの **タイプ** からフェンスデバイスのタイプを選択すると、そのフェンスデバイスに指定する必要がある引数がメニューに表示されます。**Optional Arguments** をクリックして、定義するフェンスデバイスに指定できる追加の引数を表示できます。新しいフェンスデバイスのパラメーターを入力したら、**フェンスインスタンスの作成** をクリックします。

Pacemaker を用いたフェンスデバイスの設定の詳細は [5章 フェンス機能: STONITH の設定](#) を参照してください。

### 2.3.4. ACL の設定



クラスター管理ページの上部にあるメニューから **ACL** オプションを選択すると、ローカルユーザーのパーミッションを設定できる画面が表示され、アクセス制御リスト (ACL) を使用して、クラスター設定への読み取り専用アクセスまたは読み書きアクセスが可能になります。

ACL パーミッションを割り当てるには、ロールを作成し、そのロールのアクセスパーミッションを定義します。各ロールには、XPath クエリーまたは特定要素の ID のいずれかにパーミッション (読み取り/書き込み/拒否) をいくつでも適用できます。ロールを定義したら、既存のユーザーまたはグループに割り当てることができます。

### 2.3.5. クラスターのプロパティ

クラスター管理ページの上部にあるメニューから **クラスターのプロパティ** オプションを選択するとクラスタープロパティが表示され、そのプロパティの値をデフォルト値から変更できます。Pacemaker クラスタープロパティの詳細は [12章Pacemaker クラスターのプロパティ](#) を参照してください。

## 2.4. 高可用性 PCSD WEB UI の設定

**pcsd** Web UI を使用すると、クラスターのノードのいずれかに接続して、クラスター管理ページを表示できます。接続先のノードがダウンするか、使用できなくなった場合は、クラスターの別のノードを指定する URL でブラウザを開くと、クラスターに再接続できます。ただし、**pcsd** Web UI 自体を高可用性向けに設定することもできます。この場合、新しい URL を入力することなく継続してクラスターを管理できます。

高可用性に **pcsd** Web UI を設定するには、以下の手順を実行します。

1. `/etc/sysconfig/pcsd` 設定ファイルで **PCSD\_SSL\_CERT\_SYNC\_ENABLED** が **true** に設定されていることを確認します。これは、RHEL 7 でのデフォルト値です。証明書の同期を有効にすると、クラスターセットアップコマンドおよびノード追加コマンド用の **pcsd** 証明書の同期が、**pcsd** によって実行されます。
2. **pcsd** Web UI への接続に使用するフローティング IP アドレスである **IPAddr2** クラスターリソースを作成します。物理ノードに関連付けられている IP アドレスは使用できません。**IPAddr2** リソースの NIC デバイスを指定していない場合は、そのノードに静的に割り当てられている IP アドレスの 1 つと同じネットワークにフローティング IP が存在しないと、フローティング IP アドレスを割り当てる NIC デバイスが適切に検出されません。
3. **pcsd** を使用するためにカスタムの SSL 証明書を作成し、**pcsd** Web UI への接続に使用するノードのアドレスに対して有効であることを確認します。
  - a. カスタムの SSL 証明書を作成するには、ワイルドカード証明書を使用するか、SAN (Subject Alternative Name: サブジェクトの別名) 証明書の延長を使用できます。Red Hat 証明書システムに関する詳細は、[Red Hat Certificate System Administration Guide](#) を参照してください。
  - b. **pcs pcsd certkey** コマンドを使用して、**pcsd** 用のカスタム証明書をインストールします。
  - c. **pcs pcsd sync-certificates** コマンドを使用して、**pcsd** 証明書をクラスター内の全ノードと同期させます。
4. クラスターリソースとして設定したフローティング IP アドレスを使用して、**pcsd** Web UI に接続します。





## 注記

高可用性に **pcsd** Web UI を設定している場合でも、ユーザーが接続しているノードがダウンすると、再びログインするように求められます。

## 第3章 PCS コマンドラインインターフェイス

**pcs** コマンドラインインターフェイスは、インターフェイスを **corosync.conf** および **cib.xml** ファイルに提供し、**corosync** と Pacemaker を制御および設定します。

**pcs** コマンドの一般的な形式を以下に示します。

```
pcs [-f file] [-h] [commands]...
```

### 3.1. PCS コマンド

**pcs** コマンドを以下に示します。

- **cluster**

クラスターオプションおよびノードの設定を行います。**pcs cluster** コマンドの詳細は [4章 クラスターの作成と管理](#) を参照してください。

- **resource**

クラスターリソースの作成と管理を行います。**pcs cluster** コマンドの詳細は [6章 クラスターリソースの設定](#)、[8章 クラスターリソースの管理](#)、[9章 高度な設定](#)などを参照してください。

- **stonith**

Pacemaker との使用に備えてフェンスデバイスを設定します。**pcs stonith** コマンドの詳細は、[5章 フェンス機能: STONITH の設定](#)を参照してください。

- **constraint**

リソースの制約を管理します。**pcs constraint** コマンドの詳細は、[7章 リソースの制約](#)を参照してください。

- **プロパティ**

Pacemaker のプロパティを設定します。**pcs property** コマンドでプロパティを設定する方法については [12章 Pacemaker クラスターのプロパティ](#)を参照してください。

- **status**

現在のクラスターとリソースの状態を表示します。**pcs status** コマンドの詳細は [「状態の表示」](#) を参照してください。

- **config**

ユーザーが理解できる形式でクラスターの全設定を表示します。**pcs config** コマンドの詳細は [「全クラスター設定の表示」](#) を参照してください。

### 3.2. PCS の使用に関するヘルプ表示

**pcs** の **-h** オプションを使用すると **pcs** のパラメーターとその詳細を表示させることができます。例えば、次のコマンドでは **pcs resource** コマンドのパラメーターが表示されます。出力の一部だけが表示されます。

```
# pcs resource -h
```

```
Usage: pcs resource [commands]...
Manage pacemaker resources
Commands:
  show [resource id] [--all]
    Show all currently configured resources or if a resource is specified
    show the options for the configured resource. If --all is specified
    resource options will be displayed

  start <resource id>
    Start resource specified by resource_id

  ...
```

### 3.3. RAW クラスタ設定の表示

クラスタ設定ファイルは直接編集すべきではありませんが、**pcs cluster cib** コマンドを使用すると raw クラスタ設定を表示できます。

「[設定の変更をファイルに保存](#)」に記載されているように、**pcs cluster cib filename** コマンドを使うと raw クラスタ設定を指定のファイルに保存することができます。

### 3.4. 設定の変更をファイルに保存

**pcs** コマンドを使用する際、**-f** オプションを使うとアクティブの CIB に影響を与えずに設定変更をファイルに保存することができます。

クラスタを事前に設定していて、アクティブな CIB が存在する場合は、以下のコマンドを実行して、未編集の xml ファイルを保存します。

```
pcs cluster cib filename
```

たとえば、次のコマンドを使用すると、**testfile** という名前のファイルに、未編集の CIB の xml ファイルが保存されます。

```
# pcs cluster cib testfile
```

次のコマンドでは **testfile** ファイル内にリソースをひとつ作成しています。ただし、そのリソースは現在実行中のクラスタ設定には追加されません。

```
# pcs -f testfile resource create VirtualIP ocf:heartbeat:IPAddr2 ip=192.168.0.120 cidr_netmask=24
op monitor interval=30s
```

次のコマンドで **testfile** の現在のコンテンツを CIB にプッシュします。

```
# pcs cluster cib-push testfile
```

### 3.5. 状態の表示

次のコマンドで、クラスタおよびクラスタリソースのステータスを表示します。

```
pcs status commands
```

`commands` パラメーターを指定しないとクラスターおよびリソースの全情報が表示されます。**resources**、**groups**、**cluster**、**nodes**、**pcsd**などを指定すると特定のクラスターコンポーネントの状態のみを表示させることができます。

### 3.6. 全クラスター設定の表示

現在のクラスター設定をすべて表示する場合は、次のコマンドを実行します。

```
pcs config
```

### 3.7. 現在の PCS バージョンの表示

実行中の **pcs** の現行バージョンを表示します。

```
pcs --version
```

### 3.8. クラスター設定のバックアップおよび復元

Red Hat Enterprise Linux 7.1 リリース以降では、以下のコマンドを使用してクラスター設定を tarball にバックアップできます。ファイル名を指定しないと、標準出力が使用されます。

```
pcs config backup filename
```

以下のコマンドを使用して、バックアップからすべてのノードのクラスター設定ファイルを復元します。ファイル名を指定しないと、標準入力を使用されます。**--local** オプションは、現在のノードにあるファイルだけを復元します。

```
pcs config restore [--local] [filename]
```

## 第4章 クラスターの作成と管理

本章ではクラスターの作成、クラスターコンポーネントの管理、クラスターの状態表示など Pacemaker で行うクラスターの基本的な管理について見ていきます。

### 4.1. クラスターの作成

クラスターを作成するため次のステップを行って行きます。

1. クラスターの各ノードで **pcsd** を開始します。
2. クラスターを設定するノードを認証します。
3. クラスターノードの設定と同期を行います。
4. クラスターノードでクラスターサービスを起動します。

次のセクションでは、上記の手順で使用するコマンドについて詳しく見ていきます。

#### 4.1.1. pcsd デーモンの開始

以下のコマンドは **pcsd** サービスを開始し、システムの起動時に **pcsd** を有効にします。これらのコマンドはクラスターの各ノードで実行する必要があります。

```
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

#### 4.1.2. クラスターノードの認証

次のコマンドではクラスター内のノード上にある **pcs** デーモンに対して **pcs** の認証を行います。

- すべてのノードで **pcs** 管理者のユーザー名を **hacluster** にする必要があります。 **hacluster** ユーザーのパスワードは、各ノードで同じにすることが推奨されます。
- **username** や **password** を指定しないと、コマンドの実行時にノードごとにこれらのパラメーターを入力するよう要求されます。
- ノードを指定しないと、前回実行した **pcs cluster setup** コマンドで指定されているノードの **pcs** を認証することになります。

```
pcs cluster auth [node] [...] [-u username] [-p password]
```

たとえば、以下のコマンドは **z1.example.com** と **z2.example.com** の両方で設定されるクラスターのノード両方に対して **z1.example.com** のユーザー **hacluster** を認証します。このコマンドは、クラスターノードのユーザー **hacluster** に対するパスワードを要求します。

```
root@z1 ~]# pcs cluster auth z1.example.com z2.example.com
Username: hacluster
Password:
z1.example.com: Authorized
z2.example.com: Authorized
```

認証トークンが **~/.pcs/tokens** (または **/var/lib/pcsd/tokens**) ファイルに格納されます。

### 4.1.3. クラスターノードの設定と起動

次のコマンドでクラスター設定ファイルの設定、指定ノードに対する設定の同期を行います。

- **--start** オプションを使用すると指定ノードでクラスターサービスが起動されます。必要に応じて、別途 **pcs cluster start** コマンドを使ってクラスターサービスを起動させることもできます。

**pcs cluster setup --start** コマンドでクラスターを作成する場合、または **pcs cluster start** コマンドでクラスターサービスを開始する場合、クラスターが稼働するまでに若干の遅延が発生することがあります。クラスターおよび設定の作業を続行する前に、**pcs cluster status** コマンドを使用してクラスターが稼働していることを確認することが推奨されます。

- **--local** オプションを使用するとローカルノードでのみ変更を実行します。

```
pcs cluster setup [--start] [--local] --name cluster_name node1 [node2] [...]
```

次のコマンドは指定ノード (複数指定可) でクラスターサービスを起動します。

- **--all** オプションを使用すると全ノードでクラスターサービスを起動します。
- ノードを指定しないとクラスターサービスはローカルのノードでしか起動されません。

```
pcs cluster start [--all] [node] [...]
```

## 4.2. クラスターのタイムアウト値の設定

**pcs cluster setup** コマンドを使用してクラスターを作成する場合、クラスターのタイムアウト値はほとんどのクラスター設定に適するデフォルト値に設定されます。システムに他のタイムアウト値が必要な場合は、[表4.1「タイムアウトオプション」](#)に記載されている **pcs cluster setup** オプションを使用してデフォルト値を変更できます。

表4.1 タイムアウトオプション

オプション	説明
<b>--token timeout</b>	トークンを受信しなかった後にトークンの損失が宣言されるまでの時間をミリ秒単位で設定します (デフォルトは 1000 ms です)。
<b>--join timeout</b>	join メッセージの待ち時間をミリ秒単位で設定します (デフォルトは 50 ms です)。
<b>--consensus timeout</b>	新しいメンバーシップの設定を開始する前に合意が得られるまでの待ち時間をミリ秒単位で設定します (デフォルトは 1200 ms です)。
<b>--miss_count_const count</b>	再送信が行われる前に、トークンの受信時に再送信のメッセージがチェックされる最大回数を設定します。デフォルトは 5 (5 つのメッセージ) です。
<b>--fail_recv_const failures</b>	新しい設定の設定前に、受信しなければならないメッセージが発生する可能性がある場合、メッセージを受信せずにトークンをローテーションする回数を指定します (デフォルトの失敗数は 2500 です)。

たとえば、以下のコマンドは **new\_cluster** というクラスターを作成し、トークンのタイムアウト値を 10000 ミリ秒 (10 秒)、join タイムアウト値を 100 ミリ秒に設定します。

```
# pcs cluster setup --name new_cluster nodeA nodeB --token 10000 --join 100
```

### 4.3. 冗長リングプロトコル (RRP) の設定



#### 注記

Red Hat は、[Support Policies for RHEL High Availability Clusters - Cluster Interconnect Network Interfaces](#) の Redundant Ring Protocol (RRP) の項で説明している条件に依存する、クラスターでの冗長リングプロトコルに対応しています。

**pcs cluster setup** コマンドを使用してクラスターを作成する場合、各ノードの両方のインターフェイスを指定すると冗長リングプロトコル (RRP) を用いてクラスターを設定できます。デフォルトの udp トラフィックを使用する場合にクラスターノードを指定するには、リング 0 アドレス、**、**、リング 1 アドレスの順に指定します。

たとえば、以下のコマンドはノード A とノード B の 2 つのノードを持つ **my\_rrp\_clusterM** というクラスターを設定します。ノード B には **nodeA-0** と **nodeA-1** の 2 つのインターフェイスがあります。ノード A には **nodeB-0** と **nodeB-1** の 2 つのインターフェイスがあります。RRP を使用してこれらのノードをクラスターとして設定するには、以下のコマンドを実行します。

```
# pcs cluster setup --name my_rrp_cluster nodeA-0,nodeA-1 nodeB-0,nodeB-1
```

**udp** トラフィックを使用するクラスターに RRP を設定する場合の詳細は、**pcs cluster setup** コマンドのヘルプスクリーンを参照してください。

### 4.4. クラスターノードの管理

次のセクションではクラスターサービスの起動や停止、クラスターノードの追加や削除などクラスターノードの管理で使用するコマンドについて説明します。

#### 4.4.1. クラスターサービスの停止

次のコマンドで、指定ノード (複数指定可) のクラスターサービスを停止します。**pcs cluster start** と同様に **--all** オプションを使うと全ノードのクラスターサービスが停止されます。ノードを指定しない場合はローカルノードのクラスターサービスのみが停止されます。

```
pcs cluster stop [--all] [node] [...]
```

次のコマンドでローカルノードでのクラスターサービスの停止を強制することができます。このコマンドは **kill -9** コマンドを実行します。

```
pcs cluster kill
```

#### 4.4.2. クラスターサービスの有効化および無効化

指定ノード (複数指定可) の起動時にクラスターサービスが実行されるよう設定する場合は次のコマンドを使用します。

- **--all** オプションを使用すると、全ノードでクラスターサービスが有効になります。
- ノードを指定しないと、ローカルノードでのみクラスターサービスが有効になります。

```
pcs cluster enable [--all] [node] [...]
```

指定した1つまたは複数のノードの起動時に、クラスターサービスが実行されないよう設定する場合は、次のコマンドを使用します。

- **--all** オプションを指定すると、全ノードでクラスターサービスが無効になります。
- ノードを指定しないと、ローカルノードでのみクラスターサービスが無効になります。

```
pcs cluster disable [--all] [node] [...]
```

#### 4.4.3. クラスターノードの追加



##### 注記

運用保守期間中に、既存のクラスターにノードを追加することが強く推奨されます。これにより、新しいノードとそのフェンシング設定に対して、適切なリソースとデプロイメントのテストを実行できます。

既存クラスターに新しいノードを追加する場合は、以下の手順に従ってください。この例では、**clusternode-01.example.com**、**clusternode-02.example.com**、および **clusternode-03.example.com** が既存のクラスターノードになります。新たに追加するノードは **newnode.example.com** になります。

クラスターに追加する新しいノードで、以下の作業を行います。

1. クラスターパッケージをインストールします。クラスターで SBD、Booth チケットマネージャー、またはクォーラムデバイスを使用する場合は、対応するパッケージ (**sbd**、**booth-site**、**corosync-qdevice**) を、新しいノードにも手動でインストールする必要があります。

```
[root@newnode ~]# yum install -y pcs fence-agents-all
```

2. **firewalld** デーモンを実行している場合は、以下のコマンドを実行して Red Hat High Availability Add-On が必要とするポートを有効にします。

```
# firewall-cmd --permanent --add-service=high-availability
# firewall-cmd --add-service=high-availability
```

3. ユーザー ID **hacluster** のパスワードを設定します。クラスターの各ノードで、同じパスワードを使用することが推奨されます。

```
[root@newnode ~]# passwd hacluster
Changing password for user hacluster.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```



4. 次のコマンドを実行して **pcsd** サービスを開始し、システムの起動時に **pcsd** が有効になるようにします。

```
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

既存クラスターのノードの1つで、以下の作業を行います。

1. 新しいクラスターノードで **hacluster** ユーザーを認証します。

```
[root@clusternode-01 ~]# pcs cluster auth newnode.example.com
Username: hacluster
Password:
newnode.example.com: Authorized
```

2. 新しいノードを既存のクラスターに追加します。さらに、このコマンドは **corosync.conf** クラスター設定ファイルをクラスターのすべてのノード (追加する新しいノードを含む) に対して同期します。

```
[root@clusternode-01 ~]# pcs cluster node add newnode.example.com
```

クラスターに追加する新しいノードで、以下の作業を行います。

1. 新しいノードで、クラスターサービスを開始して有効にします。

```
[root@newnode ~]# pcs cluster start
Starting Cluster...
[root@newnode ~]# pcs cluster enable
```

2. 新しいクラスターノードに対して、フェンシングデバイスを設定してテストします。フェンスデバイスの設定は [5章 フェンス機能: STONITH の設定](#) を参照してください。

#### 4.4.4. クラスターノードの削除

以下のコマンドは指定のノードをシャットダウンし、クラスターのその他すべてのノードで **corosync.conf** クラスター設定ファイルから指定のノードを削除します。クラスターに関するすべての情報をクラスターノード全体で削除し、クラスターを完全に破棄する方法については、[「クラスター設定の削除」](#) を参照してください。

```
pcs cluster node remove node
```

#### 4.4.5. スタンバイモード

以下のコマンドは、指定ノードをスタンバイモードにします。指定ノードはリソースのホストが行えなくなります。ノードで現在アクティブなリソースは、すべて別のノードに移行されます。**--all** を指定すると、このコマンドはすべてのノードをスタンバイモードにします。

リソースのパッケージを更新する場合にこのコマンドを使用します。また、設定をテストして、ノードを実際にシャットダウンせずに復元のシミュレーションを行う場合にも、このコマンドを使用できます。

```
pcs cluster standby node | --all
```

次のコマンドは、指定したノードをスタンバイモードから外します。このコマンドを実行すると、指定ノードはリソースをホストできるようになります。**--all** を指定すると、このコマンドはすべてのノードをスタンバイモードから外します。

```
pcs cluster unstandby node | --all
```

**pcs cluster standby** コマンドを実行すると、指定したノードでのリソースの実行が阻止されます。**pcs cluster unstandby** コマンドを実行すると、指定したノードでのリソースの実行が可能になります。このコマンドを実行しても、リソースが必ずしも指定のノードに戻る訳ではありません。その時点でリソースが実行できる場所は、リソースを最初に設定した方法によって異なります。リソース制約の詳細は [7章 リソースの制約](#) を参照してください。

## 4.5. ユーザーのパーミッション設定

ユーザー **hacluster** 以外の特定のユーザーにもクラスターを管理するパーミッションを付与できます。個々のユーザーに付与できるパーミッションには、以下の2つのセットがあります。

- 「[ネットワーク上でのノードアクセスのパーミッション設定](#)」で説明しているように、個別のユーザーが Web UI からクラスターを管理でき、ネットワークからノードに接続できる **pcs** コマンドを実行可能なパーミッション。ネットワーク経由でノードに接続するコマンドには、クラスターを設定するコマンド、またはクラスターからノードを追加または削除するためのコマンドが含まれます。
- 「[ACL を使用したローカルパーミッションの設定](#)」で説明しているように、クラスター設定への読み込み専用または書き込み専用アクセスをローカルユーザーに許可するパーミッション。ネットワーク経由で接続する必要のないコマンドには、リソースの作成や制約の設定など、クラスター設定を編集するコマンドが含まれます。

両方のパーミッションセットが割り当てられている状況では、ネットワーク経由で接続するコマンドのパーミッションが最初に適用され、次にローカルノードのクラスター設定を編集するパーミッションが適用されます。多くの **pcs** コマンドは、ネットワークアクセスを必要とせず、ネットワークパーミッションが適用されません。

### 4.5.1. ネットワーク上でのノードアクセスのパーミッション設定

Web UI からクラスターを管理し、ネットワークからノードに接続する **pcs** コマンドを実行するために、特定のユーザーにパーミッションを付与するには、これらのユーザーをグループ **haclient** に追加します。「[クラスター管理パーミッションの設定](#)」で説明しているように、Web UI を使用することで、これらのユーザーにパーミッションを付与することができます。

### 4.5.2. ACL を使用したローカルパーミッションの設定

Red Hat Enterprise Linux 7.1以降では、**pcs acl** コマンドを使用してローカルユーザーのパーミッションを設定し、アクセス制御リスト (ACL) を使用してクラスター設定への読み取り専用または読み書きアクセスを許可できます。また、「[ACL の設定](#)」で説明しているように、**pcs** Web UI を使用して ACL を設定することも可能です。デフォルトでは、root ユーザーと、**haclient** グループのメンバーのユーザーは、クラスター設定への完全なローカル読み込み/書き込みアクセスを持ちます。

ローカルユーザーのパーミッションを設定するには、以下の2つの手順を実行します。

1. **pcs acl role create...** コマンドを実行して *role* を作成しそのロールのパーミッションを定義します。
2. **pcs acl user create** コマンドで作成したロールをユーザーに割り当てます。

以下の例では、**rouser** という名前のローカルユーザーに、クラスター設定に対する読み取り専用アクセスを提供します。

1. この手順では、**rouser** ユーザーがローカルシステムに存在し、**rouser** ユーザーが **haclient** グループのメンバーである必要があります。

```
# adduser rouser
# usermod -a -G haclient rouser
```

2. **enable-acl** クラスタープロパティを使って Pacemaker ACL を有効にします。

```
# pcs property set enable-acl=true --force
```

3. cib に対して読み取り専用のパーミッションが付与されている **read-only** という名前のロールを作成します。

```
# pcs acl role create read-only description="Read access to cluster" read xpath /cib
```

4. pcs ACL システムで **rouser** ユーザーを作成し、そのユーザーに **read-only** ロールを割り当てます。

```
# pcs acl user create rouser read-only
```

5. 現在の ACL を表示します。

```
# pcs acl
User: rouser
Roles: read-only
Role: read-only
Description: Read access to cluster
Permission: read xpath /cib (read-only-read)
```

次の例では **wuser** というローカルユーザーにクラスター設定に対する書き込みアクセスを与えています。

1. この手順では、**wuser** ユーザーがローカルシステムに存在し、**wuser** ユーザーが **haclient** グループのメンバーである必要があります。

```
# adduser wuser
# usermod -a -G haclient wuser
```

2. **enable-acl** クラスタープロパティを使って Pacemaker ACL を有効にします。

```
# pcs property set enable-acl=true --force
```

3. cib に対して書き込みパーミッションを持つ **write-access** という名前のロールを作成します。

```
# pcs acl role create write-access description="Full access" write xpath /cib
```

4. pcs ACL システム内に **wuser** というユーザーを作成し、**write-access** ロールを割り当てます。

■

```
# pcs acl user create wuser write-access
```

5. 現在の ACL を表示します。

```
# pcs acl
User: rouser
  Roles: read-only
User: wuser
  Roles: write-access
Role: read-only
  Description: Read access to cluster
  Permission: read xpath /cib (read-only-read)
Role: write-access
  Description: Full Access
  Permission: write xpath /cib (write-access-write)
```

クラスター ACL の詳細については **pcs acl** コマンドのヘルプ画面を参照してください。

## 4.6. クラスター設定の削除

クラスター設定ファイルをすべて削除し全クラスターサービスを停止、クラスターを永久的に破棄する場合は次のコマンドを使用します。



### 警告

作成したクラスター設定をすべて永久に削除します。先に **pcs cluster stop** を実行してからクラスターの破棄を行うことを推奨しています。

```
pcs cluster destroy
```

## 4.7. クラスターの状態表示

次のコマンドで現在のクラスターとクラスターリソースの状態を表示します。

```
pcs status
```

次のコマンドを使用すると現在のクラスターの状態に関するサブセット情報を表示させることができます。

このコマンドはクラスターの状態を表示しますが、クラスターリソースの状態は表示しません。

```
pcs cluster status
```

クラスターリソースの状態は次のコマンドで表示させます。

```
pcs status resources
```

## 4.8. クラスターメンテナンス

クラスターのノードでメンテナンスを実行するには、そのクラスターで実行しているリソースおよびサービスを停止するか、または移行する必要がある場合があります。または、サービスを変更しない状態で、クラスターソフトウェアの停止が必要になる場合があります。Pacemaker は、システムメンテナンスを実行するための様々な方法を提供します。

- クラスターの別のノードでサービスが継続的に実行している状態で、クラスター内のノードを停止する必要がある場合は、そのクラスターノードをスタンバイモードにすることができます。スタンバイノードのノードは、リソースをホストすることができなくなります。ノードで現在アクティブなリソースは、別のノードに移行するか、(他のノードがそのリソースを実行できない場合は) 停止します。

スタンバイモードの詳細は、[「スタンバイモード」](#) を参照してください。

- リソースを停止せずに、実行しているノードから個別のリソースを移行する必要がある場合、**pcs resource move** コマンドを使用してリソースを別のノードに移行できます。**pcs resource move** コマンドの詳細は、[「リソースを手作業で移動する」](#) を参照してください。

**pcs resource move** コマンドを実行すると、現在実行しているノードでそれが実行されないように、制約がリソースに追加されます。リソースを戻す準備ができたなら、**pcs resource clear** または **pcs constraint delete** コマンドを実行すると、制約を削除できます。ただし、このコマンドを実行しても、リソースが必ずしも元のノードに戻る訳ではありません。その時点でリソースが実行できる場所は、リソースを最初に設定した方法によって異なるためです。[「現在のノードからリソースを移動」](#) で説明しているように、**pcs resource relocate run** コマンドを実行すると、リソースを指定のノードに移動できます。

- 完全にリソースの実行を停止して、クラスターが再び起動しないようにするには、**pcs resource disable** コマンドを使用します。**pcs resource disable** コマンドの詳細は、[「クラスターリソースの有効化、無効化、および禁止」](#) を参照してください。
- Pacemaker がリソースに対するアクションを実行するのを防ぐ必要がある場合 (たとえば、リソースに対するメンテナンスの実行中に復元アクションを無効にする必要がある場合や、**/etc/sysconfig/pacemaker** 設定をリロードする必要がある場合)、[「管理リソース」](#) で説明しているように **pcs resource unmanage** コマンドを使用します。Pacemaker Remote 接続リソースは、非管理モードにしないでください。
- クラスターを、サービスの開始や停止が行われない状態にする必要がある場合は、**maintenance-mode** クラスタープロパティを設定できます。クラスターをメンテナンスモードにすると、すべてのリソースが自動的に非管理モードになります。クラスターのプロパティの詳細は [表12.1「クラスターのプロパティ」](#) を参照してください。
- Pacemaker リモートノードでメンテナンスを実行する必要がある場合、[「システムアップグレードおよび pacemaker\\_remote」](#) で説明しているように、リモートノードリソースを無効にすることで、ノードをクラスターから削除できます。

## 第5章 フェンス機能: STONITH の設定

STONITH は Shoot The Other Node In The Head の頭字語で、不安定なノードや同時アクセスによるデータの破損を防ぐことができます。

ノードが無反応だからと言ってデータにアクセスしていないとは限りません。STONITH を使ってノードを排他処理することが唯一 100% 確実にデータの安全を確保する方法になります。排他処理することによりそのノードを確実にオフラインにしてから、別のノードに対してデータへのアクセスを許可することができます。

STONITH はクラスター化したサービスを停止できない場合にも役に立ちます。この場合は、クラスターが STONITH を使用してノード全体を強制的にオフラインにし、その後サービスを別の場所で開始すると安全です。

フェンシングの概要と、Red Hat High Availability クラスターにおけるフェンシングの重要性は[RHEL 高可用性クラスターでフェンシングが重要なのはなぜですか？](#)を参照してください。

### 5.1. STONITH (フェンス) エージェント

以下のコマンドは、利用可能な STONITH エージェントを一覧表示します。フィルターを指定するとフィルターに一致する STONITH エージェントのみを表示します。

```
pcs stonith list [filter]
```

### 5.2. フェンスデバイスの一般的なプロパティ

クラスターノードは、フェンスリソースが開始しているかどうかに関わらず、フェンスデバイスでその他のクラスターノードをフェンスできます。以下の例外を除き、リソースが開始しているかどうかは、デバイスの定期的なモニターのみを制御するものとなり、使用可能かどうかは制御しません。

- フェンスデバイスは、**pcs stonith disable stonith\_id** コマンドを実行して無効にできます。これにより、ノードがそのデバイスを使用できないようにすることができます。
- 特定のノードがフェンスデバイスを使用できないようにするには、**pcs constraint location ... avoids** コマンドで、フェンスリソースの場所制約を設定できます。
- stonith-enabled=false** を設定すると、フェンシングがすべて無効になります。ただし、実稼働環境でフェンシングを無効にすることは適していないため、フェンシングが無効になっている場合は、Red Hat ではクラスターがサポートされないことに注意してください。

表5.1「フェンスデバイスの一般的なプロパティ」は、フェンスデバイスに設定できる一般的なプロパティを説明します。特定のフェンスデバイスに設定できるフェンスプロパティについては「[デバイス固有のフェンスオプションの表示](#)」を参照してください。



#### 注記

より高度なフェンス設定プロパティについては「[その他のフェンス設定オプション](#)」を参照してください。

表5.1 フェンスデバイスの一般的なプロパティ

フィールド	タイプ	デフォルト	説明
<b>pcmk_host_map</b>	文字列		ホスト名を、ホスト名に対応していないデバイスのポート番号へマッピングします。たとえば、 <b>node1:1;node2:2,3</b> の場合は、node1 にはポート 1 を使用し、node2 にはポート 2 と 3 を使用するようにクラスターに指示します。
<b>pcmk_host_list</b>	文字列		このデバイスで制御するマシンの一覧です ( <b>pcmk_host_check=static-list</b> 以外は任意)。
<b>pcmk_host_check</b>	文字列	dynamic-list	デバイスで制御するマシンを指定します。使用できる値は、 <b>dynamic-list</b> (デバイスへの問い合わせ)、 <b>static-list</b> ( <b>pcmk_host_list</b> 属性の確認)、なし (すべてのデバイスで全マシンのフェンスが可能と見なされる) です。

### 5.3. デバイス固有のフェンスオプションの表示

指定した STONITH エージェントのオプションを表示するには、次のコマンドを使用します。

```
pcs stonith describe stonith_agent
```

次のコマンドでは Telnet または SSH 経由の APC 用フェンスエージェントのオプションを表示します。

```
# pcs stonith describe fence_apc
Stonith options for: fence_apc
ipaddr (required): IP Address or Hostname
login (required): Login Name
passwd: Login password or passphrase
passwd_script: Script to retrieve password
cmd_prompt: Force command prompt
secure: SSH connection
port (required): Physical plug number or name of virtual machine
identity_file: Identity file for ssh
switch: Physical switch number on device
inet4_only: Forces agent to use IPv4 addresses only
inet6_only: Forces agent to use IPv6 addresses only
ipport: TCP port to use for connection with device
action (required): Fencing Action
verbose: Verbose mode
debug: Write debug information to given file
version: Display version information and exit
help: Display help and exit
separator: Separator for CSV created by operation list
power_timeout: Test X seconds for status change after ON/OFF
shell_timeout: Wait X seconds for cmd prompt after issuing command
login_timeout: Wait X seconds for cmd prompt after login
```



power\_wait: Wait X seconds after issuing ON/OFF  
 delay: Wait X seconds before fencing is started  
 retry\_on: Count of attempts to retry power on



#### 警告

**method** オプションを提供するフェンスエージェントでは **cycle** がサポートされず、データの破損が生じる可能性があるため、この値は指定できません。

## 5.4. フェンスデバイスの作成

次のコマンドで stonith デバイスを作成します。

```
pcs stonith create stonith_id stonith_device_type [stonith_device_options]
```

```
# pcs stonith create MyStonith fence_virt pcmk_host_list=f1 op monitor interval=30s
```

1つのノードのみをフェンスできるフェンスデバイスや、複数のノードをフェンスできるデバイスもあります。フェンスデバイスの作成時に指定するパラメーターは、フェンスデバイスが対応しているか、必要としているかにより異なります。

- フェンスデバイスの中には、フェンスできるノードを自動的に判断できるものがあります。
- フェンスデバイスの作成時に **pcmk\_host\_list** パラメーターを使用すると、フェンスデバイスで制御されるすべてのマシンを指定できます。
- フェンスデバイスによっては、フェンスデバイスが理解する仕様へのホスト名のマッピングが必要となるものがあります。フェンスデバイスの作成時に、**pcmk\_host\_map** パラメーターを使用して、ホスト名をマッピングできます。

**pcmk\_host\_list** パラメーターおよび **pcmk\_host\_map** パラメーターの詳細は、[表5.1「フェンスデバイスの一般的なプロパティ」](#) を参照してください。

フェンスデバイスを設定したら、デバイスをテストして正しく機能していることを確認してください。フェンスデバイスのテストの詳細は「[フェンスデバイスのテスト](#)」を参照してください。

## 5.5. フェンスデバイスの表示

以下のコマンドは現在設定されているフェンスデバイスをすべて表示します。*stonith\_id* が指定されていると、指定された stonith デバイスのオプションのみが表示されます。**--full** オプションが指定されていると、設定された stonith のオプションがすべて表示されます。

```
pcs stonith show [stonith_id] [--full]
```

## 5.6. フェンスデバイスの修正と削除

現在設定されているフェンスデバイスのオプションを修正したり、新たにオプションを追加する場合は次のコマンドを使用します。



```
pcs stonith update stonith_id [stonith_device_options]
```

現在の設定からフェンスデバイスを削除する場合は次のコマンドを使用します。

```
pcs stonith delete stonith_id
```

### 5.7. フェンスデバイスが接続されているノードの管理

次のコマンドで、ノードを手動でフェンスできます。**--off** を指定すると、stonith に API コールの **off** を使用し、ノードをオフにします (再起動はしません)。

```
pcs stonith fence node [--off]
```

ノードがアクティブでない場合でも、そのノードを stonith デバイスがフェンスできない状況では、そのノードのリソースをクラスターが復旧できない可能性があります。この場合は、ノードの電源が切れたことを手動で確認した後、次のコマンドを入力して、ノードの電源が切れたことをクラスターに確認し、そのリソースを回復のために解放できます。



**警告**

指定したノードが実際にオフになっていない状態で、クラスターソフトウェア、または通常クラスターが制御するサービスを実行すると、データ破損またはクラスター障害が発生します。

```
pcs stonith confirm node
```

### 5.8. その他のフェンス設定オプション

フェンスデバイスに設定できるその他のプロパティは [表5.2「フェンスデバイスの高度なプロパティ」](#) にまとめられています。これらのオプションは高度な設定を行う場合にのみ使用されます。

表5.2 フェンスデバイスの高度なプロパティ

フィールド	タイプ	デフォルト	説明
pcmk_host_argument	文字列	port	port の代替パラメーターです。デバイスによっては、標準の port パラメーターに対応していない場合や、そのデバイス固有のパラメーターも提供している場合があります。このパラメーターを使用して、フェンスするマシンを示すデバイス固有の代替パラメーターを指定します。クラスターが追加パラメーターを提供しないようにする場合は、 <b>none</b> 値を使用します。

フィールド	タイプ	デフォルト	説明
<b>pcmk_reboot_action</b>	文字列	reboot	<b>reboot</b> の代替コマンドです。標準的なコマンドに対応していないデバイスや、別のコマンドを提供しているデバイスがあります。このパラメーターを使用して、再起動を実行するデバイス固有の代替コマンドを指定します。
<b>pcmk_reboot_timeout</b>	時間	60s	<b>stonith-timeout</b> の代替コマンドで、再起動にタイムアウトを指定します。再起動が完了するまでに通常より長い時間を要するデバイスもあれば、通常より短い時間で完了するデバイスもあります。このパラメーターを使用して、再起動にデバイス固有のタイムアウトを指定します。
<b>pcmk_reboot_retries</b>	整数	2	タイムアウト期間内に、 <b>reboot</b> コマンドを再試行する回数の上限です。複数の接続に対応していないデバイスもあります。デバイスが別のタスクでビジー状態になると操作が失敗する場合があるため、タイムアウトに達していなければ、Pacemaker が操作を自動的に再試行します。Pacemaker による再起動の動作の再試行回数を変更する場合に使用します。
<b>pcmk_off_action</b>	文字列	off	<b>off</b> の代替コマンドです。標準的なコマンドに対応していないデバイスや、別のコマンドを提供しているデバイスがあります。このような場合は、このパラメーターを使用して、オフ操作を実行するデバイス固有のコマンドを指定します。
<b>pcmk_off_timeout</b>	時間	60s	<b>stonith-timeout</b> の代替コマンドで、オフ操作にタイムアウトを指定します。デバイスによって、この操作が完了するのにかかる時間が、通常と大きく異なる場合があります。このパラメーターを使用して、オフ操作にデバイス固有のタイムアウトを指定します。
<b>pcmk_off_retries</b>	整数	2	タイムアウト期間内に、off コマンドを再試行する回数の上限です。複数の接続に対応していないデバイスもあります。デバイスが別のタスクでビジー状態になると操作が失敗する場合があるため、タイムアウトに達していなければ、Pacemaker が操作を自動的に再試行します。Pacemaker によるオフ動作の再試行回数を変更する場合に使用します。

フィールド	タイプ	デフォルト	説明
<b>pcmk_list_action</b>	文字列	list	<b>list</b> の代替コマンドです。標準的なコマンドに対応していないデバイスや、別のコマンドを提供しているデバイスがあります。このような場合は、このパラメーターを使用して、list 操作を実行するデバイス固有のコマンドを指定します。
<b>pcmk_list_timeout</b>	時間	60s	<b>stonith-timeout</b> の代わりに list の動作に対して使用する代替タイムアウトです。デバイスによって、この操作が完了するのにかかる時間が、通常と大きく異なる場合があります。このパラメーターを使用して、list 操作にデバイス固有のタイムアウトを指定します。
<b>pcmk_list_retries</b>	整数	2	タイムアウト期間内に、 <b>list</b> コマンドを再試行する回数の上限です。複数の接続に対応していないデバイスもあります。デバイスが別のタスクでビジー状態になると操作が失敗する場合があるため、タイムアウトに達していなければ、Pacemaker が操作を自動的に再試行します。Pacemaker による list 動作の再試行回数を変更する場合に使用します。
<b>pcmk_monitor_action</b>	文字列	monitor	<b>monitor</b> の代替コマンドです。標準的なコマンドに対応していないデバイスや、別のコマンドを提供しているデバイスがあります。このような場合は、このパラメーターを使用して、監視操作を実行するデバイス固有のコマンドを指定します。
<b>pcmk_monitor_timeout</b>	時間	60s	<b>stonith-timeout</b> の代替コマンドで、監視にタイムアウトを指定します。デバイスによって、この操作が完了するのにかかる時間が、通常と大きく異なる場合があります。このパラメーターを使用して、監視操作にデバイス固有のタイムアウトを指定します。
<b>pcmk_monitor_retries</b>	整数	2	タイムアウト期間内に、 <b>monitor</b> コマンドを再試行する回数の上限です。複数の接続に対応していないデバイスもあります。デバイスが別のタスクでビジー状態になると操作が失敗する場合があるため、タイムアウトに達していなければ、Pacemaker が操作を自動的に再試行します。Pacemaker による監視操作の再試行回数を変更する場合に使用します。

フィールド	タイプ	デフォルト	説明
<b>pcmk_status_action</b>	文字列	status	<b>status</b> の代替コマンドです。標準的なコマンドに対応していないデバイスや、別のコマンドを提供しているデバイスがあります。このような場合は、このパラメーターを使用して、status 操作を実行するデバイス固有のコマンドを指定します。
<b>pcmk_status_timeout</b>	時間	60s	<b>stonith-timeout</b> の代替コマンドで、status 操作にタイムアウトを指定します。デバイスによって、この操作が完了するのにかかる時間が、通常と大きく異なる場合があります。このパラメーターを使用して、status 操作にデバイス固有のタイムアウトを指定します。
<b>pcmk_status_retries</b>	整数	2	タイムアウト期間内に、status コマンドを再試行する回数の上限です。複数の接続に対応していないデバイスもあります。デバイスが別のタスクでビジー状態になると操作が失敗する場合があるため、タイムアウトに達していなければ、Pacemaker が操作を自動的に再試行します。Pacemaker による status 動作の再試行回数を変更する場合に使用します。
<b>pcmk_delay_base</b>	時間	0s	<p>stonith 操作のベース遅延を有効にし、ベース遅延の値を指定します。ノードの数が偶数になるクラスターでは、遅延を設定すると、均等の分割時に同時にノードが互いにフェンシングするのを回避できます。ランダムな遅延は、すべてのノードに同じフェンスデバイスが使用されている場合に役に立つことがあります。また、静的遅延を変更すると、各ノードで異なるデバイスが使用される場合に各フェンシングデバイスで役に立つことがあります。全体の遅延は、合計が最大遅延を下回るように、ランダムな遅延値に静的遅延を加算します。<b>pcmk_delay_base</b> を設定し、<b>pcmk_delay_max</b> を設定しない場合は、遅延にランダムなコンポーネントはなく、<b>pcmk_delay_base</b> の値となります。</p> <p>各フェンスエージェントには delay パラメーターが実装されています。これは、<b>pcmk_delay_*</b> プロパティーで設定された遅延の影響は受けません。この遅延の両方が設定されている場合は、その両方が一緒に追加されるため、通常は併用されません。</p>

フィールド	タイプ	デフォルト	説明
<b>pcmk_delay_max</b>	時間	0s	<p>stonith 動作のランダムな遅延を有効にし、ランダムな遅延の最大値を指定します。ノードの数が偶数になるクラスターでは、遅延を設定すると、均等の分割時に同時にノードが互いにフェンシングするのを回避できます。ランダムな遅延は、すべてのノードに同じフェンスデバイスが使用されている場合に役に立つことがあります。また、静的遅延を変更すると、各ノードで異なるデバイスが使用される場合に各フェンシングデバイスで役に立つことがあります。全体の遅延は、合計が最大遅延を下回るように、このランダムな遅延値に静的遅延を加算します。<b>pcmk_delay_max</b> を設定し、<b>pcmk_delay_base</b> を設定しない場合は、静的なコンポーネントが遅延に含まれません。</p> <p>各フェンスエージェントには delay パラメーターが実装されています。これは、<b>pcmk_delay_*</b> プロパティで設定された遅延の影響は受けません。この遅延の両方が設定されている場合は、その両方が一緒に追加されるため、通常は併用されません。</p>
<b>pcmk_action_limit</b>	整数	1	<p>このデバイスで並行して実行できる操作の上限です。最初に、クラスタープロパティの <b>concurrent-fencing=true</b> を設定する必要があります。値を -1 にすると無制限になります。</p>
<b>pcmk_on_action</b>	文字列	on	<p>高度な使用のみ - <b>on</b> の代替コマンドです。標準的なコマンドに対応していないデバイスや、別のコマンドを提供しているデバイスがあります。このような場合は、このパラメーターを使用して、<b>on</b> 操作を実行するデバイス固有のコマンドを指定します。</p>
<b>pcmk_on_timeout</b>	時間	60s	<p>高度な使用のみ - <b>stonith-timeout</b> の代替コマンドで、<b>on</b> 操作にタイムアウトを指定します。デバイスによって、この操作が完了するのにかかる時間が、通常と大きく異なる場合があります。このパラメーターを使用して、<b>on</b> 操作にデバイス固有のタイムアウトを指定します。</p>
<b>pcmk_on_retries</b>	整数	2	<p>高度な使用のみ - タイムアウト期間内に、<b>on</b> コマンドを再試行する回数の上限です。複数の接続に対応していないデバイスもあります。デバイスが別のタスクでビジー状態になると操作が失敗する場合があるため、タイムアウトに達していなければ、Pacemaker が操作を自動的に再試行します。Pacemaker による <b>on</b> 動作の再試行回数を変更する場合に使用します。</p>

フィールド	タイプ	デフォルト	説明
-------	-----	-------	----

表12.1「クラスターのプロパティ」で説明しているように、**fence-reaction** クラスタープロパティを設定することにより、独自のフェンシングの通知を受信した場合にクラスターノードがどのように反応するかを決定できます。クラスターノードは、フェンシングの設定が間違っている場合に独自のフェンシングの通知を受信するか、またはファブリックフェンシングがクラスター通信を遮断しない状態である可能性があります。このプロパティのデフォルト値は **stop** (Pacemaker をただちに停止して停止状態を維持しようと試みる) ですが、この値に対する最も安全な選択肢は **panic** (ローカルノードをただちに再起動しようと試みる) です。停止動作を希望する場合は、おそらくファブリックフェンシングと併用する場合は、明示的に指定することが推奨されます。

## 5.9. フェンスレベルの設定

Pacemaker は、フェンストポロジと呼ばれる機能を用いて、複数デバイスでのノードのフェンシングに対応します。トポロジを実装するには、通常の方法で各デバイスを作成し、設定のフェンストポロジセクションでフェンスレベルを1つ以上定義します。

- レベルは、1から昇順で試行されていきます。
- デバイスに障害が発生すると、現在のレベルの処理が終了します。同レベルのデバイスには試行されず、次のレベルが試行されます。
- すべてのデバイスのフェンシングが正常に完了すると、そのレベルが継承され、他のレベルは試行されなくなります。
- いずれかのレベルで成功するか、すべてのレベルが試行され失敗すると、操作は終了します。

ノードにフェンスレベルを追加する場合は、次のコマンドを使用します。デバイスは、stonith ID をコマで区切って指定します。stonith ID が、指定したレベルで試行されます。

```
pcs stonith level add level node devices
```

次のコマンドを使用すると現在設定されている全フェンスレベルが表示されます。

```
pcs stonith level
```

以下の例では、ノード **rh7-2** に、2つのフェンスデバイス (iilo フェンスデバイス **my\_ilo** と、apc フェンスデバイス **my\_apc**) が設定されています。このコマンドはフェンスレベルを設定し、デバイス **my\_ilo** に障害が発生し、ノードがフェンスできない場合に、Pacemaker がデバイス **my\_apc** の使用を試行できるようにします。この例では、レベル設定後の **pcs stonith level** コマンドの出力も表示されています。

```
# pcs stonith level add 1 rh7-2 my_ilo
```

```
# pcs stonith level add 2 rh7-2 my_apc
# pcs stonith level
Node: rh7-2
Level 1 - my_ilo
Level 2 - my_apc
```

次のコマンドは、指定したノードおよびデバイスのフェンスレベルを削除します。ノードやデバイスを指定しないと、指定したフェンスレベルがすべてのノードから削除されます。

```
pcs stonith level remove level [node_id] [stonith_id] ... [stonith_id]
```

以下のコマンドを使用すると、指定したノードや stonith id のフェンスレベルが削除されます。ノードや stonith id を指定しないと、すべてのフェンスレベルが削除されます。

```
pcs stonith level clear [node|stonith_id(s)]
```

複数の stonith ID を指定する場合はコンマで区切って指定します。空白は入力しないでください。以下に例を示します。

```
# pcs stonith level clear dev_a,dev_b
```

次のコマンドは、フェンスレベルで指定されたフェンスデバイスとノードがすべて存在することを確認します。

```
pcs stonith level verify
```

Red Hat Enterprise Linux 7.4 では、フェンストポロジのノードは、ノード名に適用する正規表現と、ノードの属性 (およびその値) で指定できます。たとえば、次のコマンドでは、ノード **node1**、**node2**、および **node3** で、フェンスデバイス **apc1** および **apc2** を使用するように設定し、ノード **node4**、**node5**、および **node6** には、フェンスデバイス **apc3** および **apc4** を使用するように設定します。

```
pcs stonith level add 1 "regexp%node[1-3]" apc1,apc2
pcs stonith level add 1 "regexp%node[4-6]" apc3,apc4
```

次のコマンドでは、ノード属性のマッチングを使用して、同じように設定します。

```
pcs node attribute node1 rack=1
pcs node attribute node2 rack=1
pcs node attribute node3 rack=1
pcs node attribute node4 rack=2
pcs node attribute node5 rack=2
pcs node attribute node6 rack=2
pcs stonith level add 1 attrib%rack=1 apc1,apc2
pcs stonith level add 1 attrib%rack=2 apc3,apc4
```

## 5.10. 冗長電源のフェンシング設定

冗長電源にフェンシングを設定する場合は、ホストを再起動するときに、クラスターが、最初に両方の電源をオフにしてから、いずれかの電源をオンにするようにする必要があります。

ノードの電源が完全にオフにならないと、ノードがリソースを解放しない場合があります。このとき、解放できなかったリソースに複数のノードが同時にアクセスして、リソースが破損する可能性があります。

Red Hat Enterprise Linux 7.2 より古いバージョンでは、オンまたはオフのアクションを使用するデバイスで異なるバージョンを明示的に設定する必要がありました。Red Hat Enterprise Linux 7.2 では、以下の例のように各デバイスを1度定義して、ノードのフェンシングに両方が必要であることを指定するだけで済むようになりました。

```
# pcs stonith create apc1 fence_apc_snmp ipaddr=apc1.example.com login=user
passwd='7a4D#1j!pz864' pcmk_host_map="node1.example.com:1;node2.example.com:2"

# pcs stonith create apc2 fence_apc_snmp ipaddr=apc2.example.com login=user
passwd='7a4D#1j!pz864' pcmk_host_map="node1.example.com:1;node2.example.com:2"

# pcs stonith level add 1 node1.example.com apc1,apc2
# pcs stonith level add 1 node2.example.com apc1,apc2
```

## 5.11. 統合フェンスデバイスで使用する ACPI の設定

クラスターが統合フェンスデバイスを使用する場合は、即時かつ完全なフェンシングを実行できるように、ACPI (Advanced Configuration and Power Interface) を設定する必要があります。

クラスターノードが統合フェンスデバイスでフェンシングされるように設定されている場合は、そのノードの ACPI Soft-Off を無効にします。ACPI Soft-Off を無効にすることにより、統合フェンスデバイスは、クリーンシャットダウンを試行する代わりに、ノードを即時に、かつ完全にオフにできます (例: **shutdown -h now**)。それ以外の場合は、ACPI Soft-Off が有効になっていると、統合フェンスデバイスがノードをオフにするのに 4 秒以上かかることがあります (以下の注記部分を参照してください)。さらに、ACPI Soft-Off が有効になっていて、ノードがシャットダウン時にパニック状態になるか、フリーズすると、統合フェンスデバイスがノードをオフにできない場合があります。このような状況では、フェンシングが遅延するか、失敗します。したがって、ノードが統合フェンスデバイスでフェンシングされ、ACPI Soft-Off が有効になっている場合は、クラスターが徐々に復元します。または管理者の介入による復旧が必要になります。



### 注記

ノードのフェンシングにかかる時間は、使用している統合フェンスデバイスによって異なります。統合フェンスデバイスの中には、電源ボタンを押し続けるのと同じ動作を実行するものもあります。この場合は、ノードがオフになるのに 4 秒から 5 秒かかります。また、電源ボタンを押してすぐ離すのと同様の動作を行い、ノードの電源をオフにする行為をオペレーティングシステムに依存する統合フェンスデバイスもあります。この場合は、ノードがオフになるのにかかる時間は 4~5 秒よりも長くなります。

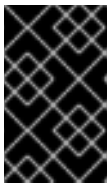
- ACPI Soft-Off を無効にする場合は、BIOS 設定を instant-off、またはこれに類似する設定に変更することが推奨されます。これにより、[「BIOS で ACPI Soft-Off を無効化」](#) で説明しているように、ノードは遅延なくオフになります。

システムによっては、BIOS で ACPI Soft-Off を無効にできません。お使いのクラスターでは、BIOS で ACPI Soft-Off を無効にできない場合に、以下のいずれかの方法で ACPI Soft-Off を無効にできます。

- [「logind.conf ファイルで ACPI Soft-Off の無効化」](#) で説明しているように、**HandlePowerKey=ignore** を **/etc/systemd/logind.conf** ファイルに設定し、ノードがフェンシングされるとすぐにオフになることを確認します。これが、ACPI Soft-Off を無効にする 1 つ目の代替方法です。



- 「GRUB 2 ファイルでの ACPI の完全な無効化」 で説明しているように、カーネル起動コマンドラインに **acpi=off** を追加します。これは、ACPI Soft-Off を無効にする 2 つ目の代替方法です。この方法の使用が推奨される場合、または 1 つ目の代替方法が利用できない場合に使用してください。



**重要**

この方法は、ACPI を完全に無効にします。コンピューターの中には、ACPI が完全が無効になるとシステムが正しく起動しないものもあります。お使いのクラスターに適した方法が他にない場合に **限り**、この方法を使用してください。

5.11.1. BIOS で ACPI Soft-Off を無効化

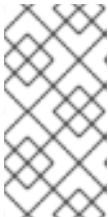
以下の手順で、各クラスターノードの BIOS を設定して、ACPI Soft-Off を無効にできます。



**注記**

BIOS で ACPI Soft-Off を無効にする手順は、サーバーシステムにより異なる場合があります。この手順は、お使いのハードウェアのドキュメントで確認する必要があります。

1. ノードを再起動して **BIOS CMOS Setup Utility** プログラムを起動します。
2. **電源** メニュー (または同等の電源管理メニュー) に移動します。
3. **電源** メニューで、**Soft-Off by PWR-BTTN** 機能 (または同等) を **Instant-Off** (または、遅延なく電源ボタンでノードをオフにする同等の設定) に設定します。例5.1「**BIOS CMOS Setup Utility: Soft-Off by PWR-BTTN を to Instant-Off に設定**」は、**ACPI Function** が **Enabled** に設定され、**Soft-Off by PWR-BTTN** が **Instant-Off** に設定されている **電源** メニューを示しています。



**注記**

**ACPI Function**、**Soft-Off by PWR-BTTN**、**Instant-Off** と同等の機能は、コンピューターによって異なることがあります。ただし、この手順の目的は、電源ボタンを使用して遅延なしにコンピューターをオフにするように BIOS を設定することです。

4. **BIOS CMOS Setup Utility** プラグラムを終了します。BIOS 設定が保存されます。
5. ノードがフェンシングされるとすぐにオフになることを確認します。フェンスデバイスのテストの詳細は「**フェンスデバイスのテスト**」を参照してください。

例5.1 BIOS CMOS Setup Utility: Soft-Off by PWR-BTTN を to Instant-Off に設定

```
+-----+-----+
| ACPI Function      [Enabled] | Item Help |
| ACPI Suspend Type  [S1(POS)] |-----|
| x Run VGABIOS if S3 Resume Auto | Menu Level * |
| Suspend Mode       [Disabled] |           |
| HDD Power Down     [Disabled] |           |
| Soft-Off by PWR-BTTN [Instant-Off |           |
| CPU THRM-Throttling [50.0%]   |           |
| Wake-Up by PCI card [Enabled]  |           |
| Power On by Ring    [Enabled]  |           |
```

```

| Wake Up On LAN      [Enabled] |
| x USB KB Wake-Up From S3  Disabled |
| Resume by Alarm      [Disabled] |
| x Date(of Month) Alarm    0 |
| x Time(hh:mm:ss) Alarm    0 : 0 :
| POWER ON Function      [BUTTON ONLY |
| x KB Power ON Password    Enter |
| x Hot Key Power ON      Ctrl-F1 |
|
+-----+

```

この例では、**ACPI Function** が **Enabled** に設定され、**Soft-Off by PWR-BTTN** が **Instant-Off** に設定されている状態を示しています。

### 5.11.2. logind.conf ファイルで ACPI Soft-Off の無効化

`/etc/systemd/logind.conf` ファイルで電源キーの処理を無効にするには、以下の手順を行います。

1. `/etc/systemd/logind.conf` ファイルに、以下の設定を定義します。

```
HandlePowerKey=ignore
```

2. **systemd** 設定をリロードします。

```
# systemctl daemon-reload
```

3. ノードがフェンシングされるとすぐにオフになることを確認します。フェンスデバイスのテストの詳細は「[フェンスデバイスのテスト](#)」を参照してください。

### 5.11.3. GRUB 2 ファイルでの ACPI の完全な無効化

ACPI Soft-Off は、カーネルの GRUB メニューエントリーに **acpi=off** を追加して無効にできます。



#### 重要

この方法は、ACPI を完全に無効にします。コンピューターの中には、ACPI が完全に無効になるとシステムが正しく起動しないものもあります。お使いのクラスターに適した方法が他にない場合に **限り**、この方法を使用してください。

以下の手順で、GRUB 2 ファイルで ACPI を無効にします。

1. 以下のように、**grubby** ツールで、**--args** オプションと **--update-kernel** オプションを使用して、各クラスターノードの **grub.cfg** ファイルを変更します。

```
# grubby --args=acpi=off --update-kernel=ALL
```

GRUB 2 の概要は、[システム管理者のガイド](#) の **GRUB 2 での作業** を参照してください。

2. ノードを再起動します。

3. ノードがフェンシングされるとすぐにオフになることを確認します。フェンスデバイスのテストの詳細は「[フェンスデバイスのテスト](#)」を参照してください。

## 5.12. フェンスデバイスのテスト

フェンシングは、Red Hat Cluster インフラストラクチャーの基本的な部分を設定しているため、フェンシングが適切に機能していることの確認またはテストを行うことは重要です。

以下の手順で、フェンスデバイスをテストします。

1. デバイスへの接続に使用する ssh、telnet、HTTP などのリモートプロトコルを使用して、手動でログインしてフェンスデバイスをテストしたり、出力される内容を確認します。たとえば、IPMI 対応デバイスのフェンシングを設定する場合は、**ipmitool** を使用してリモートでのログインを試行します。手動でログインする際に使用するオプションに注意してください。これらのオプションは、フェンスエージェントを使用する際に必要になる場合があります。

フェンスデバイスにログインできない場合は、そのデバイスが ping 可能であること、ファイアウォール設定フェンスデバイスへのアクセスを妨げていないこと、フェンスエージェントでリモートアクセスが有効になっていること、認証情報が正しいことなどを確認します。

2. フェンスエージェントスクリプトを使用して、フェンスエージェントを手動で実行します。フェンスエージェントを実行するのに、クラスターサービスが実行している必要はないため、デバイスをクラスターに設定する前にこのステップを完了できます。これにより、先に進む前に、フェンスデバイスが適切に応答することを確認できます。



### 注記

本セクションの例では、iLO デバイスの **fence\_ilo** フェンスエージェントスクリプトを使用します。実際に使用するフェンスエージェントと、そのエージェントを呼び出すコマンドは、お使いのサーバーハードウェアによって異なります。指定するオプションを確認するには、フェンスエージェントの man ページを参照してください。通常は、フェンスデバイスのログイン、パスワードなどの情報と、その他のフェンスデバイスに関する情報を把握しておく必要があります。

以下の例は、**-o status** パラメーターを指定して **fence\_ilo** フェンスエージェントスクリプトを実行する場合に使用する形式になります。ノードの再起動を試行する前にデバイスをテストして、動作させることができます。このコマンドを実行する際に、iLO デバイスの電源をオン/オフにするパーミッションを持つ iLO ユーザーの名前およびパスワードを指定します。

```
# fence_ilo -a ipaddress -l username -p password -o status
```

以下の例は、**-o reboot** パラメーターを指定して **fence\_ilo** フェンスエージェントスクリプトを実行するために使用する形式になります。このコマンドをノードで実行すると、フェンスエージェントを設定した別のノードが再起動します。

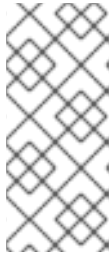
```
# fence_ilo -a ipaddress -l username -p password -o reboot
```

フェンスエージェントがステータス、オフ、オン、または再起動の動作を適切に実行しない場合は、ハードウェア、フェンスデバイスの設定、およびコマンドの構文を確認する必要があります。さらに、デバッグ出力を有効にした状態で、フェンスエージェントスクリプトを実行できます。デバッグ出力は、一部のフェンスエージェントで、フェンスデバイスにログインする際に、フェンスエージェントスクリプトに問題が発生しているイベントシーケンスの場所を確認するのに役に立ちます。

```
# fence_ilo -a ipaddress -l username -p password -o status -D
/tmp/$(hostname)-fence_agent.debug
```

発生した障害を診断する際に、フェンスデバイスに手動でログインする際に指定したオプションが、フェンスエージェントスクリプトでフェンスエージェントに渡した内容と同一であることを確認する必要があります。

フェンスエージェントが、暗号化した接続に対応する場合は、証明書の検証で障害が生じているためにエラーが出力される場合があります。ホストを信頼することや、フェンスエージェントの **ssl-insecure** パラメーターを使用することが求められます。同様に、ターゲットデバイスで SSL/TLS を無効にした場合は、フェンスエージェントに SSL パラメーターを設定する際に、これを考慮しないといけない場合があります。



### 注記

テストしているフェンスエージェントが **fence\_drac** または **fence\_ilo** の場合、もしくはその他の、継続して失敗しているシステム管理デバイスのフェンスエージェントの場合は、フォールバックして **fence\_ipmilan** を試行します。大半のシステム管理カードは IPMI リモートログインに対応しており、フェンスエージェントとしては **fence\_ipmilan** だけに対応しています。

- フェンスデバイスを、手動で機能したオプションと同じオプションでクラスターに設定し、クラスターを起動したら、以下の例にあるように、任意のノードから **pcs stonith fence** コマンドを実行してフェンシングをテストします (または複数のノードから複数回実行します)。**pcs stonith fence** コマンドは m クラスター設定を CIB から読み取り、フェンス動作を実行するように設定したフェンスエージェントを呼び出します。これにより、クラスター設定が正確であることが確認できます。

```
# pcs stonith fence node_name
```

**pcs stonith fence** コマンドに成功した場合は、フェンスイベントの発生時に、クラスターのフェンシング設定が機能します。このコマンドが失敗すると、クラスター管理が取得した設定でフェンスデバイスを起動することができません。以下の問題を確認し、必要に応じてクラスター設定を更新します。

- フェンス設定を確認します。たとえば、ホストマップを使用したことがある場合は、指定したホスト名を使用して、システムがノードを見つけられるようにする必要があります。
- デバイスのパスワードおよびユーザー名に、bash シェルが誤って解釈する可能性がある特殊文字が含まれるかどうかを確認します。パスワードとユーザー名を引用符で囲んで入力すると、この問題に対処できます。
- pcs stonith** コマンドで IP アドレスまたはホスト名を使用してデバイスに接続できるかどうかを確認してください。たとえば、stonith コマンドでホスト名を指定し、IP アドレスを使用して行ったテストは有効ではありません。
- お使いのフェンスデバイスが使用するプロトコルにアクセスできる場合は、そのプロトコルを使用してデバイスへの接続を試行します。たとえば、多くのエージェントが ssh または telnet を使用します。デバイスへの接続は、デバイスの設定時に指定した認証情報を使用して試行する必要があります。これにより、有効なプロンプトを取得し、そのデバイスにログインできるかどうかを確認できます。

すべてのパラメーターが適切であることが確認できたものの、フェンスデバイスには接続できない時に、フェンスデバイスでログ機能が使用できる場合は、ログを確認できます。これにより、ユーザーが接続したかどうかと、ユーザーが実行したコマンドが表示されま

す。`/var/log/messages` ファイルで stonith やエラーを確認すれば、発生している問題のヒントが得られる可能性もあります。また、エージェントによっては、より詳細な情報が得られる場合があります。

4. フェンスデバイステストに成功し、クラスターが稼働したら、実際の障害をテストします。このテストでは、クラスターで、トークンの損失を生じさせる動作を実行します。

- ネットワークを停止します。ネットワークの利用方法は、設定により異なります。ただし、多くの場合は、ネットワークケーブルまたは電源ケーブルをホストから物理的に抜くことができます。



#### 注記

ネットワークや電源ケーブルを物理的に切断せずに、ローカルホストのネットワークインターフェイスを無効にすることは、フェンシングのテストとしては推奨されません。実際に発生する障害を正確にシミュレートしていません。

- ローカルのファイアウォールを使用して、corosync の受信トラフィックおよび送信トラフィックをブロックします。

以下の例では corosync をブロックします。ここでは、デフォルトの corosync ポートと、ローカルのファイアウォールとして **firewalld** が使用されていることと、corosync が使用するネットワークインターフェイスがデフォルトのファイアウォールゾーンにあることが前提となっています。

```
# firewall-cmd --direct --add-rule ipv4 filter OUTPUT 2 -p udp --dport=5405 -j DROP
# firewall-cmd --add-rich-rule='rule family="ipv4" port port="5405" protocol="udp" drop'
```

- **sysrq-trigger** でクラッシュをシミュレートし、マシンをクラッシュします。ただし、カーネルパニックが発生させると、データが損失する可能性があることに注意してください。クラッシュする前に、クラスターリソースを無効にすることが推奨されます。

```
# echo c > /proc/sysrq-trigger
```

## 第6章 クラスターリソースの設定

本章ではクラスター内にリソースを設定する方法について説明していきます。

### 6.1. リソースの作成

次のコマンドを使用してクラスターリソースを作成します。

```
pcs resource create resource_id [standard:[provider:]]type [resource_options] [op operation_action
operation_options [operation_action operation_options]...] [meta meta_options] [clone
[clone_options] | master [master_options] | --group group_name [--before resource_id | --after
resource_id] | [bundle bundle_id] [--disabled] [--wait[=n]]
```

**--group** オプションを指定すると、名前付きのリソースグループにリソースが追加されます。グループが存在しない場合は作成され、そのグループにリソースが追加されます。リソースグループの詳細は「[リソースグループ](#)」を参照してください。

**--before** および **--after** オプションは、リソースグループに含まれるリソースを基準にして、追加するリソースの位置を指定します。

**--disabled** オプションは、リソースが自動的に起動しないことを示しています。

以下のコマンドは、仕様 **ocf**、プロバイダー **heartbeat**、およびタイプ **IPAddr2** で、リソース **VirtualIP** を作成します。このリソースのフローティングアドレスは 192.168.0.120 で、システムはリソースが 30 秒毎に実行されるかどうかをチェックします。

```
# pcs resource create VirtualIP ocf:heartbeat:IPAddr2 ip=192.168.0.120 cidr_netmask=24 op monitor
interval=30s
```

*standard* と *provider* のフィールドを省略して次のようにすることもできます。規格とプロバイダーはそれぞれ **ocf** と **heartbeat** にデフォルト設定されます。

```
# pcs resource create VirtualIP IPAddr2 ip=192.168.0.120 cidr_netmask=24 op monitor interval=30s
```

設定したリソースを削除する場合は、次のコマンドを実行します。

```
pcs resource delete resource_id
```

例えば、次のコマンドでは **VirtualIP** というリソース ID の既存リソースを削除しています。

```
# pcs resource delete VirtualIP
```

- `pcs resource create` コマンドのフィールド *resource\_id*、*standard*、*provider*、**type** については「[リソースのプロパティ](#)」を参照してください。
- リソースごとにパラメーターを指定する方法は「[リソース固有のパラメーター](#)」を参照してください。
- リソースの動作をクラスターが決定する場合に使用するリソースのメタオプションを定義する方法は「[リソースのメタオプション](#)」を参照してください。
- リソースで行う動作を定義する方法は「[リソースの動作](#)」を参照してください。

- **clone** オプションを指定すると、クローンリソースが作成されます。**master** オプションを指定すると、クローンリソースが作成されます。リソースのクローンや、複数モードのリソースに関する詳細は [9章 高度な設定](#) を参照してください。

## 6.2. リソースのプロパティ

リソースに定義するプロパティを使ってリソースに使用するスクリプト、スクリプトの格納先、準拠すべき標準をクラスターに指示します。[表6.1「リソースのプロパティ」](#) では、このプロパティを説明します。

表6.1 リソースのプロパティ

フィールド	説明
resource_id	リソースの名前
standard	スクリプトが準拠する規格。使用できる値は、 <b>ocf</b> 、 <b>service</b> 、 <b>upstart</b> 、 <b>systemd</b> 、 <b>lsb</b> 、 <b>stonith</b> です。
type	使用するリソースエージェントの名前 (例: <b>IPaddr</b> または <b>Filesystem</b> )
provider	OCF 仕様により、複数のベンダーで同じリソースエージェントを指定できます。Red Hat が提供するエージェントのほとんどは、プロバイダーに <b>heartbeat</b> を使用しています。

[表6.2「リソースプロパティを表示させるコマンド」](#) は、利用可能なリソースプロパティを表示するコマンドの概要を示しています。

表6.2 リソースプロパティを表示させるコマンド

pcs 表示コマンド	出力
<b>pcs resource list</b>	利用できる全リソースの一覧を表示
<b>pcs resource standards</b>	利用できるリソースエージェントの標準を表示
<b>pcs resource providers</b>	利用できるリソースエージェントのプロバイダーを表示
<b>pcs resource list <i>string</i></b>	利用できるリソースを指定文字列でフィルターした一覧を表示。仕様名、プロバイダー名、タイプ名などでフィルターを指定して、リソースを表示できます。

## 6.3. リソース固有のパラメーター

リソースごとに以下のコマンドを使用すると、そのリソースに設定できるパラメーターが表示されます。

```
# pcs resource describe standard:provider:type|type
```

たとえば、以下のコマンドはタイプ **LVM** のリソースに設定できるパラメーターを表示します。



```
# pcs resource describe LVM
Resource options for: LVM
volgrpname (required): The name of volume group.
exclusive: If set, the volume group will be activated exclusively.
partial_activation: If set, the volume group will be activated even
only partial of the physical volumes available. It helps to set to
true, when you are using mirroring logical volumes.
```

## 6.4. リソースのメタオプション

リソースには、リソース固有のパラメーターの他に、リソースオプションを設定できます。このような追加オプションは、クラスターがリソースの動作を決定する際に使用されます。表6.3「リソースのメタオプション」では、このようなオプションを説明します。

表6.3 リソースのメタオプション

フィールド	デフォルト	説明
<b>priority</b>	<b>0</b>	すべてのリソースをアクティブにできない場合に、クラスターは優先度の低いリソースを停止して、優先度の高いリソースを実行し続けます。
<b>target-role</b>	<b>Started</b>	<p>クラスターが維持するリソースのステータスです。使用できる値は以下のようになります。</p> <ul style="list-style-type: none"> <li>* <b>Stopped</b> - リソースの強制停止</li> <li>* <b>Started</b> - リソースの起動を許可 (多状態リソースの場合マスターには昇格されません)</li> <li>* <b>Master</b> - リソースの起動を許可し、必要に応じて昇格</li> </ul>
<b>is-managed</b>	<b>true</b>	クラスターによるリソースの起動と停止が可能かどうか。使用できる値は <b>true</b> または <b>false</b> です。
<b>resource-stickiness</b>	<b>0</b>	リソースを同じ場所に残すための優先度の値です。



フィールド	デフォルト	説明
<b>requires</b>	Calculated	<p>リソースを起動できる条件を示します。</p> <p>以下の条件を除き、<b>fencing</b> がデフォルトに設定されます。以下の値が使用できます。</p> <p>* <b>nothing</b> - クラスターは常にリソースを開始できます。</p> <p>* <b>quorum</b> - クラスターは、設定されているノードの過半数がアクティブな場合に限りこのリソースを起動できます。<b>stonith-enabled</b> が <b>false</b> に設定されている場合、またはリソースの <b>standard</b> が <b>stonith</b> の場合は、このオプションがデフォルト値となります。</p> <p>* <b>fencing</b> - 設定されているノードの過半数がアクティブで <b>且つ</b> 障害が発生しているノードや不明なノードの電源がすべてオフになっている場合にのみ、クラスターによるこのリソースの起動を許可</p> <p>* <b>unfencing</b> - 設定されているノードの過半数がアクティブで <b>且つ</b> 障害が発生しているノードや不明なノードの電源がすべてオフになっている場合にのみ <b>アンフェンス</b> が行われたノードに <b>限定</b> してこのリソースの起動を許可されます。<b>provides=unfencing stonith</b> メタオプションがフェンスデバイスに設定されている場合のデフォルト値です。</p>
<b>migration-threshold</b>	<b>INFINITY</b>	<p>指定したリソースが任意のノードで失敗した回数です。この回数を超えると、そのノードには、このリソースのホストとして不適格とするマークが付けられます。値を0にするとこの機能は無効になり、ノードに不適格マークが付けられることはありません。<b>INFINITY</b> (デフォルト) に設定すると、クラスターは、これを非常に大きい有限数として扱います。このオプションは、失敗した操作に <b>on-fail=restart</b> (デフォルト) が設定されていて、かつ失敗した起動操作のクラスタープロパティー <b>start-failure-is-fatal</b> が <b>false</b> に設定されている場合に限り有効です。<b>migration-threshold</b> オプションの設定の詳細は、「<a href="#">障害発生によるリソースの移動</a>」を参照してください。<b>start-failure-is-fatal</b> オプションの詳細は、<a href="#">表12.1「クラスターのプロパティー」</a>を参照してください。</p>
<b>failure-timeout</b>	<b>0</b> (無効)	<p><b>migration-threshold</b> オプションと併用されます。障害が発生していないかのように動作し、障害が発生したノードにリソースを戻せるようになるまで待機する秒数を示します。タイムベースのアクションと同様、これが、<b>cluster-recheck-interval</b> クラスターパラメーターの値よりも頻繁に確認される保証はありません。<b>failure-timeout</b> オプションの詳細は、「<a href="#">障害発生によるリソースの移動</a>」を参照してください。</p>

フィールド	デフォルト	説明
<b>multiple-active</b>	<b>stop_start</b>	<p>リソースが複数のノードでアクティブであることが検出された場合に、クラスターが実行する動作です。使用できる値は以下のようになります。</p> <ul style="list-style-type: none"> <li>* <b>block</b> - リソースに <code>unmanaged</code> のマークを付けます。</li> <li>* <b>stop_only</b> - 実行中のインスタンスをすべて停止し、以降の動作は行いません。</li> <li>* <b>stop_start</b> - 実行中のインスタンスをすべて停止してから、リソースを1カ所でのみ起動します。</li> </ul>

リソースオプションのデフォルト値を変更する場合は、次のコマンドを使用します。

```
pcs resource defaults options
```

たとえば、次のコマンドは、**resource-stickiness** のデフォルト値を 100 にリセットします。

```
# pcs resource defaults resource-stickiness=100
```

**pcs resource defaults** の *options* パラメーターを省略すると現在設定しているリソースオプションのデフォルト値の一覧を表示します。次の例では **resource-stickiness** のデフォルト値を 100 にリセットした後のコマンド出力を示しています。

```
# pcs resource defaults
resource-stickiness:100
```

リソースのメタオプションにおけるデフォルト値のリセットの有無に関わらず、リソースを作成する際に、特定リソースのリソースオプションをデフォルト以外の値に設定できます。以下は、リソースのメタオプションの値を指定する際に使用する **pcs resource create** コマンドの形式です。

```
pcs resource create resource_id standard:provider:type|type [resource options] [meta
meta_options...]
```

たとえば、以下のコマンドでは **resource-stickiness** の値を 50 に設定したリソースを作成します。

```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.120 cidr_netmask=24 meta
resource-stickiness=50
```

また、次のコマンドを使用すると既存のリソース、グループ、クローン作成したリソース、マスターリソースなどのリソースメタオプションの値を作成することもできます。

```
pcs resource meta resource_id | group_id | clone_id | master_id meta_options
```

以下の例では、**dummy\_resource** という名前の既存リソースがあります。このコマンドは、**failure-timeout** メタオプションの値を 20 秒に設定します。これにより 20 秒でリソースが同じノード上で再起動を試行できるようになります。

```
# pcs resource meta dummy_resource failure-timeout=20s
```

上記のコマンドを実行した後、**failure-timeout=20s** が設定されているか確認するためリソースの値を表示させることができます。

```
# pcs resource show dummy_resource
Resource: dummy_resource (class=ocf provider=heartbeat type=Dummy)
Meta Attrs: failure-timeout=20s
Operations: start interval=0s timeout=20 (dummy_resource-start-timeout-20)
            stop interval=0s timeout=20 (dummy_resource-stop-timeout-20)
            monitor interval=10 timeout=20 (dummy_resource-monitor-interval-10)
```

リソースの clone メタオプションについては「[リソースのクローン](#)」を参照してください。リソースの master メタオプションについては「[多状態のリソース: 複数モードのリソース](#)」を参照してください。

## 6.5. リソースグループ

クラスターの最も一般的な設定要素の1つがリソースセットです。リソースセットはまとめて配置し、順番に起動し、その逆順で停止する必要があります。この設定を簡単にするため、Pacemaker はグループの概念をサポートします。

以下のコマンドを使用してリソースグループを作成し、グループに追加するリソースを指定します。グループが存在しない場合は、このコマンドによりグループが作成されます。グループが存在する場合は、このコマンドにより別のリソースがグループに追加されます。リソースは、このコマンドで指定された順序で起動し、その逆順で停止します。

```
pcs resource group add group_name resource_id [resource_id] ... [resource_id]
[--before resource_id | --after resource_id]
```

このコマンドの **--before** オプションおよび **--after** オプションを使用して、追加するリソースの位置を、そのグループにすでに含まれるリソースを基準にして指定できます。

以下のコマンドを使用して、リソースを作成するときに、既存のグループに新しいリソースを追加することもできます。作成するリソースは *group\_name* というグループに追加されます。

```
pcs resource create resource_id standard:provider:type/type [resource_options] [op operation_action
operation_options] --group group_name
```

以下のコマンドを使用して、グループからリソースを削除します。グループにリソースがない場合、このコマンドはグループ自体を削除します。

```
pcs resource group remove group_name resource_id...
```

以下のコマンドは、現在設定されているリソースグループを一覧表示します。

```
pcs resource group list
```

以下の例では、既存リソースの **IPaddr** と **Email** が含まれるリソースグループ **shortcut** が作成されます。

```
# pcs resource group add shortcut IPaddr Email
```

グループに含まれるリソースの数に制限はありません。グループの基本的なプロパティは以下のとおりです。

- リソースは指定された順序で起動されます (この例では、最初に **IPaddr** が起動された後、**Email** が起動されます)。
- リソースは、指定した順序と逆の順序で停止します。(email first: **IPaddr**)

グループ内に実行できないリソースがあると、そのリソースの後に指定されたリソースは実行できません。

- **IPaddr** を実行できない場合は、**Email** も実行できません。
- **Email** を実行できなくても、**IPaddr** には影響ありません。

グループが大きくなると、リソースグループ作成の設定作業を軽減することが重要になります。

### 6.5.1. グループオプション

リソースグループは、リソースグループに含まれるリソースから **priority** オプション、**target-role** オプション、および **is-managed** オプションを継承します。リソースオプションの詳細は、[表6.3「リソースのメタオプション」](#)を参照してください。

### 6.5.2. グループの Stickiness (粘着性)

粘着性は、リソースを現在の場所に留ませる優先度の度合いを示し、グループで加算されます。グループのアクティブなリソースが持つ stickiness 値の合計が、グループの合計になります。そのため、**resource-stickiness** のデフォルト値が100で、グループに7つのメンバーがあり、そのメンバーの5つがアクティブな場合は、グループ全体でスコアが500の現在の場所が優先されます。

## 6.6. リソースの動作

リソースの健全性を維持するために、リソースの定義に監視操作を追加することができます。モニターリングの動作を指定しないと、**pcs** コマンドはデフォルトでモニターリングの動作を作成します。リソースエージェントでデフォルトの監視間隔が提供されない場合は、pcs コマンドにより 60 秒間隔の監視動作が作成されます。

[表6.4「動作のプロパティ」](#)に、リソースの監視動作のプロパティを示します。

表6.4 動作のプロパティ

フィールド	説明
<b>id</b>	動作の一意の名前。システムは、操作を設定する際に、これを割り当てます。
<b>name</b>	実行する動作。一般的な値は、 <b>monitor</b> 、 <b>start</b> 、 <b>stop</b> です。

フィールド	説明
<b>interval</b>	<p>値をゼロ以外に設定すると、この周波数で繰り返される反復操作 (秒単位) が作成されます。ゼロ以外の値は、アクション <b>名</b> が <b>monitor</b> に設定されている場合に限り有効になります。監視の反復アクションは、リソースの起動が完了するとすぐに実行し、その後の監視アクションの開始は、前の監視アクションが完了した時点でスケジュールされます。たとえば、<b>interval=20s</b> の監視アクションを 01:00:00 に実行すると、次の監視アクションは 01:00:20 ではなく、最初の監視アクションが完了してから 20 秒後に発生します。</p> <p>この値を、デフォルト値であるゼロに設定すると、このパラメーターで、クラスターが作成した操作に使用する値を指定できます。たとえば、<b>interval</b> をゼロに設定し、操作の <b>name</b> を <b>start</b> に設定し、<b>timeout</b> 値を 40 に設定すると、Pacemaker がこのリソースを開始する場合に 40 秒のタイムアウトを使用します。<b>monitor</b> 操作の間隔をゼロに設定した場合は、システムの起動時に Pacemaker が行うプローブの <b>timeout/on-fail/enabled</b> を設定して、デフォルトが望ましくない場合に、すべてのリソースの現在のステータスを取得できます。</p>
<b>timeout</b>	<p>このパラメーターで設定された時間内に操作が完了しないと、操作を中止し、失敗したと見なします。デフォルトの値は、<b>pcs resource op defaults</b> コマンドで設定した場合は、<b>timeout</b> の値です。設定しない場合は、20 秒になります。システムで操作 (<b>start</b>、<b>stop</b>、<b>monitor</b> など) の実行に許可されている時間よりも長い時間が必要なリソースがシステムに含まれている場合は、原因を調査して、実行時間が長いと予想される場合は、この値を増やすことができます。</p> <p><b>timeout</b> 値はいかなる遅延でもありません。また、タイムアウト期間が完了する前に操作が戻ると、クラスターはタイムアウト期間が終わる前に待機を終了します。</p>
<b>on-fail</b>	<p>この動作が失敗した場合に実行する動作。設定できる値は、</p> <ul style="list-style-type: none"> <li>* <b>ignore</b> - リソースが失敗していなかったように振る舞う</li> <li>* <b>block</b> - リソースでこれ以上、一切の操作を行わない</li> <li>* <b>stop</b> - リソースを停止して別の場所で起動しない</li> <li>* <b>restart</b> - リソースを停止して、(おそらく別の場所で) 再起動する</li> <li>* <b>fence</b> - 失敗したリソースがあるノードを STONITH する</li> <li>* <b>standby</b> - 失敗したリソースがあるノード上の <b>すべて</b> のリソースを移行する</li> </ul> <p><b>migrate</b>: 可能であれば、リソースを別のノードに移行します。これは、<b>migration-threshold</b> リソースのメタオプションを 1 に設定するのと同じです。</p> <p>STONITH が有効な場合、<b>stop</b> 動作のデフォルトは <b>fence</b> になります。そうでない場合は <b>block</b> となります。その他のすべての操作は、デフォルトで <b>restart</b> です。</p>
<b>enabled</b>	<p><b>false</b> の場合、操作は存在しないものとして処理されます。使用できる値は <b>true</b> または <b>false</b> です。</p>

### 6.6.1. リソース操作の設定

次のコマンドでリソースを作成すると、監視操作を設定できます。

```
pcs resource create resource_id standard:provider:type/type [resource_options] [op operation_action operation_options [operation_type operation_options]...]
```

たとえば、次のコマンドは、監視操作付きの **IPAddr2** リソースを作成します。新しいリソースには **VirtualIP** という名前が付けられ、**eth2** で IP アドレス 192.168.0.99、ネットマスク 24 になります。監視操作は、30 秒ごとに実施されます。

```
# pcs resource create VirtualIP ocf:heartbeat:IPAddr2 ip=192.168.0.99 cidr_netmask=24 nic=eth2 op
monitor interval=30s
```

また、次のコマンドで既存のリソースに監視操作を追加することもできます。

```
pcs resource op add resource_id operation_action [operation_properties]
```

設定されているリソース操作を削除する場合は、次のコマンドを使用します。

```
pcs resource op remove resource_id operation_name operation_properties
```



#### 注記

操作プロパティを正しく指定して、既存の操作を適切に削除する必要があります。

監視オプションの値を変更する場合は、リソースを更新します。たとえば、以下のコマンドで **VirtualIP** を作成できます。

```
# pcs resource create VirtualIP ocf:heartbeat:IPAddr2 ip=192.168.0.99 cidr_netmask=24 nic=eth2
```

デフォルトでは、次の操作が作成されます。

```
Operations: start interval=0s timeout=20s (VirtualIP-start-timeout-20s)
            stop interval=0s timeout=20s (VirtualIP-stop-timeout-20s)
            monitor interval=10s timeout=20s (VirtualIP-monitor-interval-10s)
```

stop の timeout 操作を変更するには、以下のコマンドを実行します。

```
# pcs resource update VirtualIP op stop interval=0s timeout=40s

# pcs resource show VirtualIP
Resource: VirtualIP (class=ocf provider=heartbeat type=IPAddr2)
Attributes: ip=192.168.0.99 cidr_netmask=24 nic=eth2
Operations: start interval=0s timeout=20s (VirtualIP-start-timeout-20s)
            monitor interval=10s timeout=20s (VirtualIP-monitor-interval-10s)
            stop interval=0s timeout=40s (VirtualIP-name-stop-interval-0s-timeout-40s)
```



#### 注記

**pcs resource update** コマンドでリソースの操作を更新すると、具体的に呼び出されていないオプションは、デフォルト値にリセットされます。

### 6.6.2. グローバルリソース操作のデフォルトの設定

次のコマンドを使用して、監視操作のグローバルデフォルト値を設定できます。

```
pcs resource op defaults [options]
```

たとえば、次のコマンドは、すべての監視操作に対して、**timeout** 値のグローバルデフォルトを 240 秒に設定します。

```
# pcs resource op defaults timeout=240s
```

モニタリング操作の現在設定されているデフォルト値を表示するには、**pcs resource op defaults** コマンドをオプションを指定せずに実行します。

たとえば、以下のコマンドを実行すると、クラスター監視操作のデフォルト値を表示します。この例では、**timeout** 値が 240 秒に設定されています。

```
# pcs resource op defaults
timeout: 240s
```

クラスターリソース定義でオプションが指定されていない場合に限り、クラスターリソースがグローバルデフォルトを使用することに注意してください。デフォルトでは、リソースエージェントがすべての操作の **timeout** オプションを定義します。グローバル操作のタイムアウト値を有効にするには、**timeout** オプションを明示的に指定せずにクラスターリソースを作成するか、次のコマンドのように、クラスターリソースを更新して **timeout** オプションを削除する必要があります。

```
# pcs resource update VirtualIP op monitor interval=10s
```

たとえば、すべての監視操作に、グローバルデフォルトの **timeout** 値 240 秒を設定し、クラスターリソース **VirtualIP** を更新して、**monitor** 操作のタイムアウト値を削除すると、リソース **VirtualIP** には、**start**、**stop**、および **monitor** の操作のタイムアウト値が、それぞれ 20 秒、40 秒、240 秒に設定されます。タイムアウト操作のグローバルなデフォルト値は、ここでは **monitor** にのみ適用されます。ここでは、前のコマンドにより、デフォルトの **timeout** オプションが削除されています。

```
# pcs resource show VirtualIP
Resource: VirtualIP (class=ocf provider=heartbeat type=IPAddr2)
Attributes: ip=192.168.0.99 cidr_netmask=24 nic=eth2
Operations: start interval=0s timeout=20s (VirtualIP-start-timeout-20s)
            monitor interval=10s (VirtualIP-monitor-interval-10s)
            stop interval=0s timeout=40s (VirtualIP-name-stop-interval-0s-timeout-40s)
```

## 6.7. 設定されているリソースの表示

設定されているリソースの一覧を表示する場合は、次のコマンドを使用します。

```
pcs resource show
```

例えば、**VirtualIP** という名前のリソースと **WebSite** という名前のリソースでシステムを設定していた場合、**pcs resource show** コマンドを実行すると次のような出力が得られます。

```
# pcs resource show
VirtualIP (ocf::heartbeat:IPAddr2): Started
WebSite (ocf::heartbeat:apache): Started
```

リソースに設定されているパラメーターを表示する場合は、次のコマンドを使用します。

```
pcs resource show resource_id
```

たとえば、次のコマンドは、現在設定されているリソース **VirtualIP** のパラメーターを表示します。

```
# pcs resource show VirtualIP
Resource: VirtualIP (type=IPAddr2 class=ocf provider=heartbeat)
Attributes: ip=192.168.0.120 cidr_netmask=24
Operations: monitor interval=30s
```

## 6.8. リソースパラメーターの変更

設定されているリソースのパラメーターを変更する場合は、次のコマンドを使用します。

```
pcs resource update resource_id [resource_options]
```

以下のコマンドシーケンスでは、**VirtualIP** リソースに設定したパラメーターの初期値、**ip** パラメーターの値を変更するコマンド、変更されたパラメーター値を示しています。

```
# pcs resource show VirtualIP
Resource: VirtualIP (type=IPAddr2 class=ocf provider=heartbeat)
Attributes: ip=192.168.0.120 cidr_netmask=24
Operations: monitor interval=30s
# pcs resource update VirtualIP ip=192.169.0.120
# pcs resource show VirtualIP
Resource: VirtualIP (type=IPAddr2 class=ocf provider=heartbeat)
Attributes: ip=192.169.0.120 cidr_netmask=24
Operations: monitor interval=30s
```

## 6.9. 複数のモニターリング動作

リソースエージェントが対応する範囲で、1つのリソースに複数の監視操作を設定できます。これにより、1分ごとに表面的なヘルスチェックを行い、徐々に頻度を上げてより正確なチェックを行うこともできます。



### 注記

複数の監視操作を設定する場合は、2種類の操作が同じ間隔で実行されないように注意してください。

リソースに異なるレベルで徹底的なヘルスチェックに対応する追加モニターリング動作を設定するには **OCF\_CHECK\_LEVEL=*n*** オプションを追加します。

たとえば、以下のように **IPAddr2** リソースを設定すると、デフォルトでは 10 秒間隔でタイムアウト値が 20 秒の監視操作が作成されます。

```
# pcs resource create VirtualIP ocf:heartbeat:IPAddr2 ip=192.168.0.99 cidr_netmask=24 nic=eth2
```

仮想 IP が、深さ 10 の様々なチェックに対応する場合は、次のコマンドを実行すると、Pacemaker は、通常の 10 秒間隔の仮想 IP チェックに加えて、60 秒ごとに高度な監視チェックを実行します。なお、上述のとおり、追加の監視操作は 10 秒間隔にしないようにしてください。

```
# pcs resource op add VirtualIP monitor interval=60s OCF_CHECK_LEVEL=10
```



## 6.10. クラスターリソースの有効化と無効化

次のコマンドは **resource\_id** で指定されているリソースを有効にします。

```
pcs resource enable resource_id
```

次のコマンドは **resource\_id** で指定されているリソースを有効にします。

```
pcs resource disable resource_id
```

## 6.11. クラスターリソースのクリーンアップ

リソースに障害が発生すると、クラスターの状態を表示するときに障害メッセージが表示されます。このリソースを解決する場合、**pcs resource cleanup** コマンドで障害状態を消去できます。このコマンドはリソースの状態と **failcount** をリセットし、リソースの動作履歴を消去して現在の状態を再検出するようにクラスターに指示します。

以下のコマンドは、*resource\_id* によって指定されたリソースをクリーンアップします。

```
pcs resource cleanup resource_id
```

*resource\_id* を指定しないと、このコマンドはすべてのリソースのリソース状態と **failcount** をリセットします。

Red Hat Enterprise Linux 7.5 では、**pcs resource cleanup** コマンドは、失敗したアクションとして表示されるリソースのみを検証します。全ノードの全リソースを調査するには、次のコマンドを入力します。

```
pcs resource refresh
```

デフォルトでは、**pcs resource refresh** コマンドは、リソースの状態がわかっているノードだけを調べます。ステータスが分からないすべてのリソースを検証するには、以下のコマンドを実行します。

```
pcs resource refresh --full
```

## 第7章 リソースの制約

リソースの制約を設定して、クラスター内のそのリソースの動作を指定できます。以下の制約のカテゴリーを設定できます。

- **場所** の制約 - この制約は、リソースを実行するノードを指定します。場所の制約については「[場所の制約](#)」で説明しています。
- **order** 制約 - 順序の制約はリソースが実行される順序を決めます。順序の制約については「[順序の制約](#)」で説明しています。
- **コロケーション** の制約 - この制約は、他のリソースとの対比でリソースの配置先を決定します。コロケーションの制約については「[リソースのコロケーション](#)」で説明しています。

複数リソースをまとめて配置して、順番に起動するまたは逆順で停止する一連の制約を簡単に設定する方法として、Pacemaker ではリソースグループという概念に対応しています。リソースグループの詳細は「[リソースグループ](#)」を参照してください。

### 7.1. 場所の制約

場所の制約は、リソースを実行するノードを指定します。場所の制約を設定することで、たとえば特定のノードで優先してリソースを実行する、または特定のノードではリソースを実行しないことを決定できます。

場所の制約に加え、リソースが実行されるノードは、そのリソースの **resource-stickiness** 値に影響されます。これは、リソースが現在実行しているノードに留まることをどの程度優先するかを決定します。**resource-stickiness** 値の設定に関する詳細は、「[現在のノードを優先させるリソースの設定](#)」を参照してください。

#### 7.1.1. 基本的な場所の制約

基本的な場所の制約を設定して、リソースの実行を特定のノードで優先するか、または回避するかを指定できます。オプションの **score** 値を使用して、制約の相対的な優先度を指定できます。

以下のコマンドは、リソースの実行を、指定した1つまたは複数のノードで優先するように、場所の制約を作成します。1回のコマンドで、特定のリソースの制約を複数のノードに対して作成できます。

```
pcs constraint location rsc prefers node[=score] [node[=score]] ...
```

次のコマンドは、リソースが指定ノードを回避する場所の制約を作成します。

```
pcs constraint location rsc avoids node[=score] [node[=score]] ...
```

表7.1「[簡単な場所の制約オプション](#)」では、最も簡単な形式で場所の制約を設定する場合のオプションの意味をまとめています。

表7.1 簡単な場所の制約オプション

フィールド	説明
<b>rsc</b>	リソース名
<b>node</b>	ノード名

フィールド	説明
<b>score</b>	<p>リソースを特定ノードで優先的に実行するか、または実行を回避するかを示す正の整数値。<b>INFINITY</b> は、リソースの場所制約のデフォルト <b>score</b> 値です。</p> <p><b>pcs constraint location rsc prefers</b> コマンドで <b>score</b> の値を <b>INFINITY</b> にすると、そのノードが利用可能な場合は、リソースがそのノードで優先的に実行します。ただし、そのノードが利用できない場合に、別のノードでそのリソースを実行しないようにする訳ではありません。</p> <p><b>pcs constraint location rsc avoids</b> コマンドで <b>score</b> に <b>INFINITY</b> を指定すると、その他のノードが利用できない場合でも、そのリソースはそのノードでは実行されないことを示します。これは、<b>-INFINITY</b> のスコアで <b>pcs constraint location add</b> コマンドを設定するのと同じです。</p>

以下のコマンドは、**Webserver** リソースが、**node1** ノードで優先的に実行するように指定する場所の制約を作成します。

```
# pcs constraint location Webserver prefers node1
```

Red Hat Enterprise Linux 7.4 の **pcs** では、コマンドラインの場所の制約に正規表現に対応しています。この制約は、リソース名に一致する正規表現に基づいて、複数のリソースに適用されます。これにより、1つのコマンドラインで複数の場所の制約を設定できます。

次のコマンドは、**dummy0** から **dummy9** までのリソースの実行が **node1** に優先されるように指定する場所の制約を作成します。

```
# pcs constraint location 'regex%dummy[0-9]' prefers node1
```

Pacemaker は、  
[http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1\\_chap09.html#tag\\_09\\_04](http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap09.html#tag_09_04) で説明しているように、POSIX 拡張正規表現を使用するため、以下のコマンドを実行しても同じ制約を指定できます。

```
# pcs constraint location 'regex%dummy[[:digit:]]' prefers node1
```

### 7.1.2. 高度な場所の制約

ノードに場所の制約を設定する際に、**pcs constraint location** コマンドの **resource-discovery** オプションを指定して、指定したリソースに対して、Pacemaker がこのノードでリソース検出を実行するかどうかの優先度を指定できます。物理的にリソースが稼働可能なノードのサブセットでリソース検出を制限すると、ノードが大量に存在する場合にパフォーマンスを大幅に改善できます。**pacemaker\_remote** を使用して、ノード数を100単位で拡大する場合は、このオプションの使用を検討してください。

以下のコマンドは、**pcs constraint location** コマンドで **resource-discovery** オプションを指定する場合の形式を示しています。*id* は制約 id であることに注意してください。*rsc*、*node*、*score* の意味は表 7.1「簡単な場所の制約オプション」で説明しています。このコマンドでは、基本的な場所の制約に対応します。*score* を正の値にすると、リソースが特定のノードで優先的に実行するように設定されます。*score* を負の値にすると、リソースがノードを回避するように設定されます。基本的な場所の制約と同様に、制約にリソースの正規表現を使用することもできます。

```
pcs constraint location add id rsc node score [resource-discovery=option]
```

表7.2「リソース検出の値」では、**resource-discovery** オプションに指定できる値を説明しています。

表7.2 リソース検出の値

値	説明
<b>always</b>	このノードに指定したリソースで、リソース検出を常に実行します。これは、リソースの場所の制約の <b>resource-discovery</b> のデフォルト値です。
<b>never</b>	このノードで指定したリソースでのリソース検出は実行しません。
<b>exclusive</b>	このノード (および <b>exclusive</b> と同様のマークが付いている他のノード) に指定したリソースに限定してリソース検出を行います。複数のノードで同じリソースの <b>exclusive</b> 検出を使用する複数の場所制約により、 <b>resource-discovery</b> が排他的であるノードのサブセットが作成されます。1つまたは複数のノードで、リソースが <b>exclusive</b> 検出用にマーク付けされている場合、そのリソースは、ノードのサブセット内にのみ配置できます。

**resource-discovery** オプションを **never** または **exclusive** に設定すると、クラスターが認識しなくても、該当する場所でリソースがアクティブになる可能性があります。この場合は、(**systemd** または管理者による起動などの) クラスターが制御できる範囲外でサービスが起動すると、複数の場所でリソースがアクティブになる可能性があります。また、クラスターの一部がダウンしたりスプリットブレインが発生しているときに **resource-discovery** プロパティが変更した場合や、ノードでリソースがアクティブなときにそのリソースやノードに対して **resource-discovery** プロパティが変更した場合にも、複数の場所でリソースがアクティブになる可能性があります。そのため、ノードの数が9以上で、特定の場所しかリソースを実行できない場合 (必要なソフトウェアが他の場所にインストールされていない場合など) に限り、このオプションは適しています。

### 7.1.3. ルールを使用したリソースの場所の確定

より複雑な場所の制約は、Pacemaker ルールを使用してリソースの場所を判断することができます。Pacemaker ルールや設定できるプロパティの概要は、[11章 Pacemaker ルール](#) を参照してください。

以下のコマンドを使用して、ルールを使用する Pacemaker 制約を使用します。**score** を省略すると、デフォルトの INFINITY に設定されます。**resource-discovery** を省略すると、デフォルトの **always** に設定されます。**resource-discovery** オプションの詳細は、「[高度な場所の制約](#)」を参照してください。基本的な場所の制約と同様に、制約にリソースの正規表現を使用することもできます。

ルールを使用して場所の制約を設定する場合は、**score** を正または負の値にすることができます。正の値は prefers を示し、負の値は avoids を示します。

```
pcs constraint location rsc rule [resource-discovery=option] [role=master|slave] [score=score | score-attribute=attribute] expression
```

*expression* オプションは、*duration\_options* および *date\_spec\_options* のいずれかに設定できます。使用できる値は、[表11.5「日付詳細のプロパティ」](#) で説明されているように、hours、monthdays、weekdays、yeardays、months、weeks、years、weekyears、moon になります。

- **defined|not\_defined attribute**
- **attribute lt|gt|lte|gte|eq|ne [string|integer|version] value**

- `date gt|lt date`
- `date in-range date to date`
- `date in-range date to duration duration_options ...`
- `date-spec date_spec_options`
- `expression and|or expression`
- `(expression)`

以下の場所の制約は、現在が 2018 年の任意の時点である場合に true の式を設定します。

```
# pcs constraint location Webserver rule score=INFINITY date-spec years=2018
```

以下のコマンドは、月曜日から金曜日までの 9 am から 5 pm までが true となる式を設定します。hours の値 16 には、時間 (hour) の値が一致する 16:59:59 までが含まれます。

```
# pcs constraint location Webserver rule score=INFINITY date-spec hours="9-16" weekdays="1-5"
```

以下のコマンドは、13 日の金曜日が満月になると true になる式を設定します。

```
# pcs constraint location Webserver rule date-spec weekdays=5 monthdays=13 moon=4
```

#### 7.1.4. 場所の制約ストラテジー

「[基本的な場所の制約](#)」、「[高度な場所の制約](#)」、「[ルールを使用したリソースの場所の確定](#)」で説明している場所の制約のいずれかを使用することで、リソースを実行できるノードを指定するための一般的なストラテジーを設定できます。

- オプトインクラスター - デフォルトでは、すべてのリソースを、どのノードでも実行できません。そして、特定のリソースに対してノードを選択的に許可できるようにクラスターを設定します。オプトインクラスターの設定方法は「[オプトインクラスターの設定](#)」で説明しています。
- オプトアウトクラスター - デフォルトでは、すべてのリソースをどのノードでも実行できるクラスターを設定してから、リソースを特定のノードで実行しないように、場所の制約を作成します。オプトアウトクラスターの設定方法は「[オプトアウトクラスターの設定](#)」で説明しています。これは、デフォルトの Pacemaker ストラテジーです。

クラスターでオプトインまたはオプトアウトのどちらを選択するかは、独自に優先する設定やクラスターの設定により異なります。ほとんどのリソースをほとんどのノードで実行できるようにする場合は、オプトアウトを使用した方が設定しやすくなる可能性があります。ほとんどのリソースを、一部のノードでのみ実行する場合は、オプトインを使用した方が設定しやすくなる可能性があります。

##### 7.1.4.1. オプトインクラスターの設定

オプトインクラスターを作成する場合は、クラスタープロパティ **symmetric-cluster** を **false** に設定し、デフォルトでは、いずれのノードでもリソースの実行を許可しないようにします。

```
# pcs property set symmetric-cluster=false
```

個々のリソースでノードを有効にします。以下のコマンドは、場所の制約を設定し、**Webserver** リ

ソースでは **example-1** ノードが優先され、**Database** リソースでは **example-2** ノードが優先されるようにし、いずれのリソースも優先ノードに障害が発生した場合は **example-3** ノードにフェールオーバーできるようにします。オプトインクラスターに場所の制約を設定する場合は、スコアをゼロに設定すると、リソースに対してノードの優先や回避を指定せずに、リソースをノードで実行できます。

```
# pcs constraint location Webserver prefers example-1=200
# pcs constraint location Webserver prefers example-3=0
# pcs constraint location Database prefers example-2=200
# pcs constraint location Database prefers example-3=0
```

#### 7.1.4.2. オプトアウトクラスターの設定

オプトアウトクラスターを作成するには、クラスタープロパティ **symmetric-cluster** を **true** に設定し、デフォルトで、すべてのノードでリソースの実行を許可します。

```
# pcs property set symmetric-cluster=true
```

以下のコマンドを実行すると、「[オプトインクラスターの設定](#)」の例と同じ設定になります。全ノードのスコアは暗黙で 0 になるため、優先ノードに障害が発生した場合はいずれのリソースも **example-3** ノードにフェールオーバーできます。

```
# pcs constraint location Webserver prefers example-1=200
# pcs constraint location Webserver avoids example-2=INFINITY
# pcs constraint location Database avoids example-1=INFINITY
# pcs constraint location Database prefers example-2=200
```

INFINITY は、スコアのデフォルト値であるため、上記コマンドでは、スコアに INFINITY を指定する必要はないことに注意してください。

#### 7.1.5. 現在のノードを優先させるリソースの設定

リソースには、「[リソースのメタオプション](#)」で説明されているように、リソースの作成時にメタ属性として設定できる **resource-stickiness** 値があります。**resource-stickiness** 値は、現在実行しているノード上にリソースが残す量を決定します。Pacemaker は、他の設定 (場所の制約の score 値など) とともに **resource-stickiness** 値を考慮して、リソースを別のノードに移動するか、そのまま残すかを決定します。

デフォルトでは、**resource-stickiness** の値が 0 の状態でリソースが作成されます。**resource-stickiness** が 0 に設定され、場所の制約がない Pacemaker のデフォルト動作では、クラスターノード間で均等に分散されるようにリソースを移動します。この設定では、正常なリソースの移動頻度が想定よりも増える可能性があります。この動作を防ぐには、デフォルトの **resource-stickiness** の値を 1 に設定します。このデフォルトはクラスター内のすべてのリソースに適用されます。この小さい値は、作成する他の制約で簡単に上書きできますが、Pacemaker がクラスター全体で正常なリソースを不必要に移動しないようにするには十分です。

以下のコマンドは、デフォルトの **resource-stickiness** 値を 1 に設定します。

```
# pcs resource defaults resource-stickiness=1
```

**resource-stickiness** の値が設定されている場合は、リソースが新たに追加されたノードに移動しません。この時点でリソースバランスが必要な場合は、**resource-stickiness** の値を一時的に 0 に設定できます。



場所の制約スコアが resource-stickiness 値よりも高い場合、クラスターは正常なリソースを、場所の制約がポイントするノードに依然として移動する可能性があります。

Pacemaker がリソースの配置先を決定する方法の詳細は、「[使用と配置ストラテジー](#)」を参照してください。

## 7.2. 順序の制約

順序の制約でリソースを実行する順序を指定します。

次のコマンドを使って順序の制約を設定します。

```
pcs constraint order [action] resource_id then [action] resource_id [options]
```

表7.3「[順序の制約のプロパティ](#)」では順序の制約を設定する場合のプロパティとオプションについて簡単に示します。

表7.3 順序の制約のプロパティ

フィールド	説明
resource_id	動作を行うリソースの名前。
action	<p>リソースで実行する動作。action プロパティでは、以下の値が使用できます。</p> <ul style="list-style-type: none"> <li>* <b>start</b> - リソースを起動する</li> <li>* <b>stop</b> - リソースを停止する</li> <li>* <b>promote</b> - スレーブリソースからマスターリソースにリソースの昇格を行う</li> <li>* <b>demote</b> - マスターリソースからスレーブリソースにリソースの降格を行う</li> </ul> <p>動作を指定しない場合のデフォルトの動作は <b>start</b> です。マスターリソースとスレーブリソースについての詳細は「<a href="#">多状態のリソース: 複数モードのリソース</a>」を参照してください。</p>
<b>kind</b> オプション	<p>制約の実施方法。kind オプションでは、以下の値が使用できます。</p> <ul style="list-style-type: none"> <li>* <b>Optional</b> - 両方のリソースが指定の動作を実行する場合のみ適用されます。オプションの順序は「<a href="#">勧告的な順序付け</a>」を参照してください。</li> <li>* <b>Mandatory</b> - 常に実施 (デフォルト値) します。1 番目に指定したリソースが停止している場合や起動できない場合は、2 番目に指定したリソースが停止します。必須の順序の詳細は、「<a href="#">強制的な順序付け</a>」を参照してください。</li> <li>* <b>Serialize</b> - リソースセットに対して 2 種類の動作、停止と起動が同時に発生しないようにする</li> </ul>
<b>symmetrical</b> オプション	<p>デフォルトの true に設定されていると逆順でリソースの停止を行ないます。デフォルト値: <b>true</b></p>

### 7.2.1. 強制的な順序付け

mandatory 制約では 1 番目に指定しているリソースが実行されない限り 2 番目に指定しているリソースは実行できません。これが **kind** オプションのデフォルトです。デフォルト値のままにしておくと 1 番

目に指定しているリソースの状態が変化した場合、2 番目に指定したリソースが必ず反応するようになります。

- 1 番目に指定している実行中のリソースを停止すると、2 番目に指定しているリソースも停止されます (実行している場合)。
- 1 番目に指定している実行中のリソースが実行しておらず起動できない場合には、指定したリソースは (実行していれば) 停止します。
- 2 番目に指定しているリソースの実行中に 1 番目に指定しているリソースが再起動されると、2 番目に指定しているリソースが停止され再起動されます。

ただし、クラスターは、それぞれの状態変化に対応することに注意してください。2 番目のリソースで停止操作を開始する前に 1 番目のリソースが再起動し、起動状態にあると、2 番目のリソースを再起動する必要がありません。

### 7.2.2. 勧告的な順序付け

順序の制約に **kind=Optional** オプションが指定された場合、制約は任意として考慮され、両方のリソースが指定のアクションを実行する場合のみ適用されます。1 番目に指定しているリソースの状態を変更しても、2 番目に指定しているリソースには影響しません。

次のコマンドは、**VirtualIP** と **dummy\_resource** という名前のリソースに、勧告的な順序の制約を設定します。

```
# pcs constraint order VirtualIP then dummy_resource kind=Optional
```

### 7.2.3. 順序付けされたリソースセット

一般的に、管理者は、複数のリソースの連鎖を作成する際に順序を設定します (例: リソース A が開始してからリソース B を開始し、その後にリソース C を開始)。複数のリソースを作成して同じ場所に配置し (コロケーションを指定)、起動の順序を設定する必要がある場合は、「[リソースグループ](#)」に従って、このようなリソースが含まれるリソースグループを設定できます。ただし、特定の順序で起動する必要があるリソースをリソースグループとして設定することが適切ではない場合があります。

- リソースを順番に起動するように設定する必要があるものの、リソースは必ずしも同じ場所に配置しない場合
- リソース C の前にリソース A または B のいずれかが起動する必要があるものの、A と B の間には関係が設定されていない場合
- リソース C およびリソース D の前にリソース A およびリソース B の両方が起動している必要があるものの、A と B、または C と D の間には関係が設定されていない場合

このような状況では、**pcs constraint order set** コマンドを使用して、リソースの 1 つまたは複数のセットに対して順序の制約を作成できます。

**pcs constraint order set** コマンドを使用すると、以下のオプションをリソースのセットに設定できます。

- **sequential** - リソースセットに順序を付ける必要があるかどうかを指定します。 **true** または **false** に設定できます。

**sequential** を **false** に設定すると、セットのメンバーに順序を設定せず、順序の制約にあるセット間で順序付けできます。そのため、このオプションは、制約に複数のセットが登録されている場合に限り有効です。それ以外の場合は、制約を設定しても効果がありません。



- **require-all** - 続行する前にセットの全リソースがアクティブである必要があるかどうかを指定します。**true** または **false** に設定できます。**require-all** を **false** に設定すると、次のセットに進む前に、セットの1つのリソースのみを開始する必要があります。**require-all** を **false** に設定しても、**sequential** が **false** に設定されている順序なしセットと併用しない限り、効果はありません。
- **action**: 表7.3「順序の制約のプロパティ」で説明しているように、**start**、**promote**、**demote**、または **stop** に設定できます。

**pcs constraint order set** コマンドの **setoptions** パラメーターの後に、リソースのセットに対する以下の制約オプションを設定できます。

- **id** - 定義する制約の名前を指定します。
- **score** - 制約の優先度を示します。このオプションの詳細は 表7.4「コロケーション制約のプロパティ」を参照してください。

```
pcs constraint order set resource1 resource2 [resourceN]... [options] [set resourceX resourceY ...
[options]] [setoptions [constraint_options]]
```

**D1**、**D2**、**D3** という3つのリソースがある場合は、次のコマンドを実行すると、この3つのリソースを、順序を指定したリソースセットとして設定します。

```
# pcs constraint order set D1 D2 D3
```

#### 7.2.4. 順序の制約からリソースを削除

次のコマンドを使用すると、すべての順序の制約からリソースが削除されます。

```
pcs constraint order remove resource1 [resourceN]...
```

### 7.3. リソースのコロケーション

任意のリソースの場所を別のリソースの場所に依存するよう定義するのがコロケーションの制約です。

2つのリソース間にコロケーション制約を作成すると、リソースがノードに割り当てられる割り当てる順序に重要な影響を及ぼす点に注意してください。リソース B の場所を把握していない場合は、リソース B に相対的となるようにリソース A を配置することができません。このため、コロケーションの制約を作成する場合は、リソース A をリソース B に対してコロケーションを設定するのか、もしくはリソース B をリソース A に対してコロケーションを設定するのかを考慮する必要があります。

また、コロケーションの制約を作成する際に注意しておきたいもう1つの点として、リソース A をリソース B に対してコロケーションすると仮定した場合、クラスターがリソース B に選択するノードを決定する際、リソース A の優先度も考慮に入れます。

次のコマンドはコロケーションの制約を作成します。

```
pcs constraint colocation add [master|slave] source_resource with [master|slave] target_resource
[score] [options]
```

マスターリソースとスレーブリソースについての詳細は「多状態のリソース: 複数モードのリソース」を参照してください。

表7.4「コロケーション制約のプロパティ」は、コロケーション制約を設定するのに使用するプロパティおよびオプションをまとめています。

表7.4 コロケーション制約のプロパティ

フィールド	説明
source_resource	コロケーションソース。制約の条件を満たさない場合、クラスターではこのリソースの実行を許可しないことを決定する可能性があります。
target_resource	コロケーションターゲット。クラスターは、このリソースの配置先を決定してから、ソースリソースの配置先を決定します。
score	正の値を指定するとリソースが同じノードで実行します。負の値を指定すると同じノードで実行しなくなります。デフォルト値である <b>+INFINITY</b> を指定すると、 <i>source_resource</i> は必ず <i>target_resource</i> と同じノードで実行します。- <b>INFINITY</b> は、 <i>source_resource</i> を <i>target_resource</i> と同じノードで実行してはならないことを示しています。

### 7.3.1. 強制的な配置

制約スコアが **+INFINITY** または **-INFINITY** の場合は常に強制的な配置が発生します。制約条件が満たされないと *source\_resource* の実行が許可されません。 **score=INFINITY** の場合、*target\_resource* がアクティブではないケースが含まれます。

**myresource1** を、常に **myresource2** と同じマシンで実行する必要がある場合は、次のような制約を追加します。

```
# pcs constraint colocation add myresource1 with myresource2 score=INFINITY
```

**INFINITY** を使用しているため、(何らかの理由で) **myresource2** がクラスターのいずれのノードでも実行できない場合は、**myresource1** の実行が許可されません。

または、逆の設定、つまり **myresource1** が **myresource2** と同じマシンでは実行されないようにクラスターを設定することもできます。この場合は **score=-INFINITY** を使用します。

```
# pcs constraint colocation add myresource1 with myresource2 score=-INFINITY
```

ここでも、**-INFINITY** を指定することで、制約は結合しています。このため、実行できる場所として残っているノードで **myresource2** がすでに実行されている場合は、いずれのノードでも **myresource1** を実行できなくなります。

### 7.3.2. 勧告的な配置

必ず実行する(または必ず実行しない)状況を強制的な配置とするならば、勧告的な配置は、ある状況下で優先される状況を指しています。制約のスコアが **-INFINITY** より大きく、**INFINITY** より小さい場合、クラスターは希望の設定に対応しようとしますが、クラスターリソースの一部を停止するという選択肢がある場合には、その設定が無視される場合があります。勧告的なコロケーション制約と設定の他の要素を組み合わせると、強制的であるように動作させることができます。

### 7.3.3. 複数リソースのコロケート

コロケーションと順序が適用されるリソースのセットを作成する必要がある場合は、「[リソースグループ](#)」に従って、このようなリソースを含むリソースグループを設定できます。ただし、コロケーション

を設定する必要があるリソースをリソースグループとして設定することが適切ではない場合があります。

- リソースのセットにコロケーションを設定する必要があるものの、リソースが必ずしも順番に起動する必要がない場合
- リソース A または B のいずれかに対してコロケーションを設定する必要があるリソース C が起動したものの、A と B の間に関係が設定されていない場合。
- リソース C およびリソース D を、リソース A およびリソース B の両方に対してコロケーションを設定する必要があるものの、A と B の間、または C と D の間に関係が設定されていない場合

このような状況では、**pcs constraint colocation set** コマンドを使用して、リソースの1つまたは複数のセットでコロケーションの制約を作成できます。

**pcs constraint colocation set** コマンドを使用すると、以下のオプションをリソースのセットに設定できます。

- **sequential** - セットのメンバーで相互のコロケーションが必要であるかどうかを指定します。**true** または **false** に設定できます。

**sequential** を **false** に設定すると、このセットのメンバーがアクティブであるかどうかに関係なく、このセットのメンバーを、制約の中で、このセットの後にリストされている他のセットに対してコロケーションを設定できます。そのため、このオプションは制約でこのセットの後に他のセットが指定されている場合に限り有効です。他のセットが指定されていない場合は、制約の効果がありません。

- **role** - **Stopped**、**Started**、**Master**、または **Slave** に設定できます。マルチステートリソースの詳細は「[多状態のリソース: 複数モードのリソース](#)」を参照してください。

**pcs constraint colocation set** コマンドの **setoptions** パラメーターの後に、リソースのセットに対する以下の制約オプションを設定できます。

- **kind**: 制約の実行方法を示します。このオプションの詳細は [表7.3「順序の制約のプロパティ」](#) を参照してください。
- **symmetrical**: リソースを停止する順序を示します。デフォルトの **true** に設定されていると逆順でリソースの停止を行ないます。デフォルト値: **true**
- **id** - 定義する制約の名前を指定します。

セットのメンバーをリストすると、各メンバーは、自身の前のメンバーに対してコロケーションが設定されます。たとえば、set A B は B が A の場所に配置されることを意味します。しかし、複数のセットをリストする場合、各セットはその後のメンバーと同じ場所に配置されます。たとえば、set C D sequential=false set A B は、C と D のセットが、A と B のセットと同じ場所に配置されることを意味します (ただし、C と D には関係がなく、B は A にはコロケーションが設定されています)。

以下のコマンドは、リソースのセットにコロケーションの制約を作成します。

```
pcs constraint colocation set resource1 resource2 [resourceN]... [options] [set resourceX resourceY ... [options]] [setoptions [constraint_options]]
```

#### 7.3.4. コロケーション制約の削除

コロケーション制約を削除する場合はコマンドに `source_resource` を付けて使用します。

```
pcs constraint colocation remove source_resource target_resource
```

## 7.4. 制約の表示

設定した制約を表示させるコマンドがいくつかあります。

以下のコマンドは、現在の場所、順序、ロケーションの制約をすべて表示します。

```
pcs constraint list|show
```

以下のコマンドは、現在の場所の制約を一覧表示します。

- **resources** を指定すると、リソース別に場所の制約が表示されます。これはデフォルトの動作です。
- **nodes** を指定すると、ノード別に場所の制約が表示されます。
- 特定のリソースまたはノードを指定すると、そのリソースまたはノードの情報のみが表示されます。

```
pcs constraint location [show resources|nodes [specific nodes|resources]] [--full]
```

以下のコマンドは、現在の順序の制約をすべて表示します。--full オプションを指定すると、制約の内部 ID が表示されます。

```
pcs constraint order show [--full]
```

次のコマンドは、現在のコロケーション制約を一覧表示します。--full オプションを指定すると、制約の内部 ID が表示されます。

```
pcs constraint colocation show [--full]
```

以下のコマンドは、特定リソースを参照する制約を一覧表示します。

```
pcs constraint ref resource ...
```

## 第8章 クラスターリソースの管理

本章ではクラスターのリソース管理に使用する各種コマンドについて説明します。次のような手順が含まれます。

- 「リソースを手作業で移動する」
- 「障害発生によるリソースの移動」
- 「クラスターリソースの有効化、無効化、および禁止」
- 「モニター操作の無効化」

### 8.1. リソースを手作業で移動する

クラスターの設定を無視して、強制的にリソースを現在の場所から移行させることができます。次のような2つの状況が考えられます。

- ノードがメンテナンスで、そのノードで実行中の全リソースを別のノードに移行する必要がある
- 個別に指定したリソースを移行する必要がある

ノードで実行中の全リソースを別のノードに移行する場合は、そのノードをスタンバイモードにします。クラスターノードをスタンバイモードにする方法については「[スタンバイモード](#)」を参照してください。

個別に指定したリソースは、以下のいずれかの方法で移行できます。

- 「[現在のノードからリソースを移動](#)」の説明に従って、**pcs resource move** コマンドを使用して現在稼働しているノードからリソースを移動します。
- **pcs resource relocate run** コマンドを使用して、現在のクラスター状態、制約、リソースの場所、およびその他の設定によって決定される優先ノードへリソースを移動します。このコマンドの詳細は、「[リソースの優先ノードへの移動](#)」を参照してください。

#### 8.1.1. 現在のノードからリソースを移動

現在稼働しているノードからリソースを移動するには以下のコマンドを使用し、リソースの *resource\_id* を定義どおりに指定します。移行するリソースを実行する移行先のノードを指定する場合は、**destination\_node** を使用します。

```
pcs resource move resource_id [destination_node] [--master] [lifetime=lifetime]
```

#### 注記

**pcs resource move** コマンドを実行すると、現在実行しているノードでそれが実行されないように、制約がリソースに追加されます。**pcs resource clear** または **pcs constraint delete** コマンドを実行すると制約を削除できます。このコマンドを実行しても、リソースが必ずしも元のノードに戻る訳ではありません。この時点でリソースが実行できる場所は、リソースの最初の設定方法によって異なります。

**pcs resource move** コマンドの **--master** パラメーターを指定すると、制約の範囲がマスターロールに限定され、*resource\_id* の代わりに *master\_id* を指定する必要があります。

任意で **pcs resource move** コマンドの **lifetime** パラメーターを設定すると、制限が維持される期間を指定できます。**lifetime** パラメーターの単位は、ISO 8601 に定義されている形式に従って指定します。ISO 8601 では、Y (年)、M (月)、W (週)、D (日)、H (時)、M (分)、S (秒) のように、単位を大文字で指定する必要があります。

分単位の M と、月単位の M を区別するには、分単位の値の前に PT を指定する必要があります。たとえば、**lifetime** パラメーターが 5M の場合は 5 カ月の間隔を示し、**lifetime** パラメーターが PT5M の場合は 5 分の間隔を示します。

**lifetime** パラメーターは、**cluster-recheck-interval** クラスタプロパティに定義した間隔で確認されます。デフォルト値は 15 分です。このパラメーターをより頻繁に確認する必要がある場合は、次のコマンドを実行して、この値をリセットできます。

```
pcs property set cluster-recheck-interval=value
```

任意で、**pcs resource move** コマンドに **--wait[=n]** パラメーターを設定し、移行先のノードでリソースが起動するまでの待機時間 (秒単位) を指定できます。待機時間がこの値を超えると、リソースが起動した場合に 0 が返され、リソースが起動しなかった場合は 1 が返されます。n の値を指定しないと、デフォルトのリソースのタイムアウト値が使用されます。

以下のコマンドは、**resource1** リソースを **example-node2** ノードへ移動し、1 時間 30 分は元のノードへ戻らないようにします。

```
pcs resource move resource1 example-node2 lifetime=PT1H30M
```

以下のコマンドは、**resource1** リソースを **example-node2** ノードへ移行し、30 分は元のノードへ戻らないようにします。

```
pcs resource move resource1 example-node2 lifetime=PT30M
```

リソース制約の詳細は [7章 リソースの制約](#) を参照してください。

### 8.1.2. リソースの優先ノードへの移動

フェイルオーバーや管理者の手作業によるノードの移行により、リソースが移行した後、フェイルオーバーの原因となった状況が改善されたとしても、そのリソースが必ずしも元のノードに戻るとは限りません。リソースを優先ノードへ移行するには、以下のコマンドを実行します。優先ノードは、現在のクラスタ状態、制約、リソースの場所、およびその他の設定により決定され、時間とともに変更する可能性があります。

```
pcs resource relocate run [resource1] [resource2] ...
```

リソースを指定しないと、すべてのリソースが優先ノードに移行します。

このコマンドは、リソースのスティッキネスを無視し、各リソースの優先ノードを算出します。優先ノードの算出後、リソースを優先ノードに移行する場所の制約を作成します。リソースが移行すると、制約が自動的に削除されます。**pcs resource relocate run** コマンドによって作成された制約をすべて削除するには、**pcs resource relocate clear** コマンドを入力します。リソースの現在の状態と、resource stickiness を無視した最適なノードを表示するには、**pcs resource relocate show** コマンドを実行します。

## 8.2. 障害発生によるリソースの移動



リソースの作成時に、リソースに **migration-threshold** オプションを設定し、定義した回数だけ障害が発生した場合にリソースが新しいノードに移動されるように設定できます。このしきい値に一度到達すると、このノードでは、以下が行われるまで、障害が発生したリソースを実行できなくなります。

- 管理者は **pcs resource failcount** コマンドを使用して手作業でリソースの **failcount** をリセットします。
- リソースの **failure-timeout** 値に達する

デフォルトで、**migration-threshold** の値が **INFINITY** に設定されています。**INFINITY** は、内部的に非常に大きな有限数として定義されます。0 にすると、**migration-threshold** 機能が無効になります。



#### 注記

リソースの **migration-threshold** を設定するのと、リソースの状態を維持しながら別の場所に移動させるようにリソースの移動を設定するのは同じではありません。

次の例では、**dummy\_resource** リソースに、移行しきい値 10 を追加します。この場合は、障害が 10 回発生すると、そのリソースが新しいノードに移動します。

```
# pcs resource meta dummy_resource migration-threshold=10
```

次のコマンドを使用すると、クラスター全体にデフォルトの移行しきい値を追加できます。

```
# pcs resource defaults migration-threshold=10
```

リソースの現在の障害回数とリミットを確認するには **pcs resource failcount** コマンドを使用します。

移行しきい値の概念には、リソース起動の失敗とリソース停止の失敗の 2 つの例外があります。クラスタープロパティ **start-failure-is-fatal** が **true** に設定された場合 (デフォルト) は、起動の失敗により **failcount** が **INFINITY** に設定され、リソースが常に即座に移動するようになります。**start-failure-is-fatal** オプションの詳細は、表 12.1「[クラスターのプロパティ](#)」を参照してください。

停止時の失敗は、起動時とは若干異なり、極めて重大となります。リソースの停止に失敗し STONITH が有効になっていると、リソースを別のノードで起動できるように、クラスターによるノードのフェンスが行われます。STONITH を有効にしていない場合はクラスターに続行する手段がないため、別のノードでのリソース起動は試行されません。ただし、障害のタイムアウト後に再度停止が試行されます。

### 8.3. 接続状態変更によるリソースの移動

以下の 2 つのステップに従って、外部の接続が失われた場合にリソースが移動するようにクラスターを設定します。

1. **ping** リソースをクラスターに追加します。**ping** リソースは、同じ名前のシステムユーティリティを使用して、マシン (DNS ホスト名または IPv4/IPv6 アドレスによって指定される一覧) にアクセス可能であるかをテストし、その結果を使用して **pingd** と呼ばれるノード属性を維持します。
2. 接続が失われたときに別のノードにリソースを移動させる、リソース場所制約を設定します。

表 6.1「[リソースのプロパティ](#)」には、**ping** リソースに設定できるプロパティを示します。

表 8.1 ping リソースのプロパティ

フィールド	説明
<b>dampen</b>	今後の変更が発生するまでに待機する (弱める) 時間。これにより、クラスターノードが、わずかに異なる時間に接続が失われたことに気が付いたときに、クラスターでリソースがバウンスするのを防ぎます。
<b>multiplier</b>	接続された ping ノードの数は、ノードの数にこの値を掛けて、スコアを取得します。複数の ping ノードが設定された場合に便利です。
<b>host_list</b>	現在の接続状態を判断するために接続するマシン。使用できる値には、解決可能な DNS ホスト名、IPv4 アドレス、および IPv6 アドレスが含まれます。ホストリストのエントリーはスペースで区切られます。

次のコマンド例は、**gateway.example.com** への接続を検証する **ping** リソースを作成します。実際には、ネットワークゲートウェイやルーターへの接続を検証します。リソースがすべてのクラスターノードで実行されるように、**ping** リソースをクローンとして設定します。

```
# pcs resource create ping ocf:pacemaker:ping dampen=5s multiplier=1000
host_list=gateway.example.com clone
```

以下の例は、既存のリソース **Webserver** に場所制約ルールを設定します。これにより、**Webserver** リソースが現在実行しているホストが **gateway.example.com** へ ping できない場合に、**Webserver** リソースを **gateway.example.com** へ ping できるホストに移動します。

```
# pcs constraint location Webserver rule score=-INFINITY pingd lt 1 or not_defined pingd
```

## 8.4. クラスターリソースの有効化、無効化、および禁止

「[リソースを手作業で移動する](#)」に説明されている **pcs resource move** および **pcs resource relocate** コマンドの他にも、クラスターリソースの動作を制御するために使用できるコマンドは複数あります。

実行中のリソースを手動で停止し、クラスターが再起動しないようにする場合は、以下のコマンドを使用します。その他の設定 (制約、オプション、失敗など) によっては、リソースが起動した状態のままになる可能性があります。**--wait** オプションを指定すると、**pcs** はリソースが停止するまで *n* 秒間待機します。その後、リソースが停止した場合は 0 を返し、リソースが停止しなかった場合は 1 を返します。*n* を指定しないと、デフォルトの 60 分に設定されます。

```
pcs resource disable resource_id [--wait[=n]]
```

クラスターがリソースを起動できるようにするには、次のコマンドを使用します。他の設定によっては、リソースが停止したままになることがあります。**--wait** オプションを指定すると、**pcs** はリソースが開始するまで最長で *n* 秒間待機します。その後、リソースが開始した場合には 0、リソースが開始しなかった場合には 1 を返します。*n* を指定しないと、デフォルトの 60 分に設定されます。

```
pcs resource enable resource_id [--wait[=n]]
```

指定したノードでリソースが実行されないようにする場合は、次のコマンドを使用します。ノードを指定しないと、現在実行中のノードでリソースが実行されないようになります。

```
pcs resource ban resource_id [node] [--master] [lifetime=lifetime] [--wait[=n]]
```



**pcs resource ban** コマンドを実行すると、場所制約である **-INFINITY** がリソースに追加され、リソースが指定のノードで実行されないようにします。**pcs resource clear** または **pcs constraint delete** コマンドを実行すると制約を削除できます。このコマンドを実行しても、リソースが必ずしも指定のノードに戻る訳ではありません。その時点でリソースが実行できる場所は、リソースを最初に設定した方法によって異なります。リソース制約の詳細は [7章 リソースの制約](#) を参照してください。

**pcs resource ban** コマンドの **--master** パラメーターを指定すると、制約の範囲がマスターロールに限定され、*resource\_id* の代わりに *master\_id* を指定する必要があります。

任意で **pcs resource ban** コマンドに **lifetime** パラメーターを設定し、制約が持続する期間を指定できます。**lifetime** パラメーターの単位や、**lifetime** パラメーターがチェックされる間隔を指定する方法については、「[リソースを手作業で移動する](#)」を参照してください。

任意で、**pcs resource ban** コマンドに **--wait[=*n*]** パラメーターを設定し、移行先のノードでリソースが起動するまでの待機時間 (秒単位) できます。待機時間がこの値を超えると、リソースが起動した場合に 0 が返され、リソースが起動しなかった場合は 1 が返されます。*n* の値を指定しないと、デフォルトのリソースのタイムアウト値が使用されます。

指定したリソースを現在のノードで強制起動する場合は **pcs resource** コマンドの **debug-start** パラメーターを使用します。これは、主にデバッグリソースに使用されます。クラスターでのリソースの起動は (ほぼ) 毎回 Pacemaker で行われるため、直接 **pcs** コマンドを使用した起動は行われません。リソースが起動しない原因は、大抵、リソースが誤って設定されているか (システムログでデバッグします)、リソースが起動しないように制約が設定されているか、リソースが無効になっているかのいずれかになります。この場合は、次のコマンドを使用してリソースの設定をテストできます。ただし、通常は、クラスター内でリソースを起動するのに使用しないでください。

**debug-start** コマンドの形式を以下に示します。

```
pcs resource debug-start resource_id
```

## 8.5. モニター操作の無効化

定期的な監視を停止する最も簡単な方法は、監視を削除することです。ただし、一時的に無効にしたい場合もあります。このような場合は、**pcs resource update** コマンドで **enabled="false"** を操作の定義に追加します。モニターリング操作を再度有効にするには、**enabled="true"** を操作の定義に設定します。

**pcs resource update** コマンドでリソースの操作を更新すると、具体的に呼び出されていないオプションは、デフォルト値にリセットされます。たとえば、モニターリング操作のタイムアウト値をカスタムの 600 に設定してある場合に以下のコマンドを使用すると、タイムアウト値がデフォルト値の 20 (または、**pcs resource ops default** コマンドでデフォルト値を設定された任意の値) にリセットされます。

```
# pcs resource update resourceXZY op monitor enabled=false
# pcs resource update resourceXZY op monitor enabled=true
```

このオプションの元の値 600 を維持するために、監視操作に戻す場合は、以下の例のように、その値を指定する必要があります。

```
# pcs resource update resourceXZY op monitor timeout=600 enabled=true
```

## 8.6. 管理リソース

リソースを **unmanaged** モードに設定できます。このモードでは、リソースは設定に維持されますが Pacemaker はリソースを管理しません。

以下のコマンドは、指定のリソースを **非管理** モードに設定します。

```
pcs resource unmanage resource1 [resource2] ...
```

以下のコマンドは、リソースをデフォルトの **管理** モードに設定します。

```
pcs resource manage resource1 [resource2] ...
```

**pcs resource manage** または **pcs resource unmanage** コマンドを使用してリソースグループの名前を指定できます。このコマンドは、グループのすべてのリソースに対して実行されるため、1つのコマンドでグループ内の全リソースすべて **管理** または **非管理** モードに設定し、グループに含まれるリソースを個別に管理できます。

## 第9章 高度な設定

本章では Pacemaker で対応している高度なリソースタイプと高度な設定機能を説明します。

### 9.1. リソースのクローン

リソースが複数のノードでアクティブになるよう、リソースをクローンすることができます。たとえば、ノードの分散のために、クローンとなるリソースを使用して、クラスター全体に分散させる IP リソースのインスタンスを複数設定できます。リソースエージェントが対応していれば、任意のリソースのクローンを作成できます。クローンは、1つのリソースまたは1つのリソースグループで設定されます。



#### 注記

同時に複数のノードでアクティブにできるリソースのみがクローンに適しています。たとえば、共有メモリーデバイスから **ext4** などの非クラスター化ファイルシステムをマウントする **Filesystem** リソースのクローンは作成しないでください。**ext4** パーティションはクラスターを認識しないため、同時に複数のノードから繰り返し行われる読み取りや書き込みの操作には適していません。

#### 9.1.1. クローンリソースの作成と削除

リソースの作成と、そのリソースのクローン作成を同時に行う場合は、次のコマンドを使用します。

```
pcs resource create resource_id standard:provider:type|type [resource options] \
clone [meta clone_options]
```

クローンの名前は **resource\_id-clone** になります。

1つのコマンドで、リソースグループの作成と、リソースグループのクローン作成の両方を行うことはできません。

作成済みリソースまたはリソースグループのクローンを作成する場合は、次のコマンドを実行します。

```
pcs resource clone resource_id| group_name [clone_options]...
```

クローンの名前は **resource\_id-clone** または **group\_name-clone** になります。



#### 注記

リソース設定の変更が必要なのは、1つのノードのみです。



#### 注記

制約を設定する場合は、グループ名またはクローン名を必ず使用します。

リソースのクローンを作成すると、そのクローンの名前は、リソース名に **-clone** を付けた名前になります。次のコマンドは、タイプが **apache** のリソース **webfarm** と、そのクローンとなるリソース **webfarm-clone** を作成します。

```
# pcs resource create webfarm apache clone
```



## 注記

あるリソースまたはリソースグループのクローンを、別のクローンの後にくるように作成する場合は、多くの場合 **interleave=true** オプションを設定する必要があります。これにより、依存されているクローンが同じノードで停止または開始した時に、依存しているクローンのコピーを停止または開始できるようになります。このオプションを設定しない場合は、次のようになります。クローンリソース B がクローンリソース A に依存していると、ノードがクラスターから離れてから戻ってきってから、そのノードでリソース A が起動すると、リソース B の全コピーが、全ノードで再起動します。これは、依存しているクローンリソースに **interleave** オプションが設定されていない場合は、そのリソースの全インスタンスが、そのリソースが依存しているリソースの実行インスタンスに依存するためです。

リソースまたはリソースグループのクローンを削除する場合は、次のコマンドを使用します。リソースやリソースグループ自体は削除されません。

```
pcs resource unclone resource_id | group_name
```

リソースオプションの詳細は、「[リソースの作成](#)」を参照してください。

表9.1「[リソースのクローンオプション](#)」には、クローンのリソースに指定できるオプションを示します。

表9.1 リソースのクローンオプション

フィールド	説明
<b>priority, target-role, is-managed</b>	表6.3「 <a href="#">リソースのメタオプション</a> 」に従って、クローンされたリソースから継承されるオプション。
<b>clone-max</b>	起動するリソースのコピーの数。デフォルトは、クラスター内のノード数です。
<b>clone-node-max</b>	1つのノードで起動できるリソースのコピー数。デフォルト値は <b>1</b> です。
<b>notify</b>	クローンのコピーを停止したり起動する時に、前もって、およびアクションが成功した時に、他のコピーに通知します。使用できる値は <b>false</b> および <b>true</b> です。デフォルト値は <b>false</b> です。
<b>globally-unique</b>	<p>クローンの各コピーで異なる機能を実行させるかどうか。使用できる値は <b>false</b> および <b>true</b> です。</p> <p>このオプションの値を <b>false</b> にすると、リソースが実行しているすべてのノードで同じ動作を行うため、1台のマシンごとに実行できるクローンのコピーは1つです。</p> <p>このオプションの値を <b>true</b> にすると、任意のマシンで実行中のクローンのコピーが、別のインスタンスが別のノードまたは同じノードで実行されているかに関係なく、そのインスタンスと同等ではありません。<b>clone-node-max</b> の値が1より大きい場合にはデフォルト値が <b>true</b> になり、小さい場合は <b>false</b> がデフォルト値になります。</p>
<b>ordered</b>	コピーを、(並列ではなく)連続して開始する必要があります。使用できる値は <b>false</b> および <b>true</b> です。デフォルト値は <b>false</b> です。

フィールド	説明
<b>interleave</b>	2 番目のクローンのすべてのインスタンスが開始または停止するまで待機せず、2 番目のクローンの同じノードでコピーを開始または停止した直後に最初クローンのコピーが開始または停止できるよう、順序制約の動作を変更します。使用できる値は <b>false</b> および <b>true</b> です。デフォルト値は <b>false</b> です。
<b>clone-min</b>	このフィールドに値を指定した場合は、 <b>interleave</b> オプションが <b>true</b> に設定されていても、元のクローンの後に順序付けされたクローンは、元のクローンに指定された数だけインスタンスが実行するまで、起動できません。

### 9.1.2. 制約のクローン作成

ほとんどの場合、アクティブなクラスターノードに対するクローンのコピーは1つです。ただし、リソースクローンの **clone-max** には、そのクラスター内のノード合計より小さい数を設定できます。この場合は、リソースの場所の制約を使用して、クラスターが優先的にコピーを割り当てるノードを指定できます。クローンの ID を使用する点以外、制約は通常のリソースの場合と全く変わらずクローンに記述されます。

次のコマンドは、クラスターがリソースのクローン **webfarm-clone** を **node1** に優先的に割り当てる場所の制約を作成します。

```
# pcs constraint location webfarm-clone prefers node1
```

順序制約の動作はクローンでは若干異なります。以下の例では、**interleave** クローンオプションをデフォルトの **false** のままにしているため、起動する必要がある **webfarm-clone** のすべてのインスタンスが起動するまで、**webfarm-stats** のインスタンスは起動しません。**webfarm-clone** のコピーを1つも起動できない場合にのみ、**webfarm-stats** がアクティブになりません。さらに、**webfarm-stats** が停止するまで待機してから、**webfarm-clone** が停止します。

```
# pcs constraint order start webfarm-clone then webfarm-stats
```

通常のリソース (またはリソースグループ) とクローンのコロケーションは、リソースを、クローンのアクティブコピーを持つ任意のマシンで実行できることを意味します。どのコピーが実行しているマシンでそのリソースを実行させるかはクローンが実行している場所とそのリソース自体の場所の優先度に応じて選択されます。

クローン間のコロケーションも可能です。この場合、クローンに対して許可できる場所は、そのクローンが実行中のノード (または実行するノード) に限定されます。割り当ては通常通り行われます。

以下のコマンドは、コロケーション制約を作成し、**webfarm-stats** リソースが **webfarm-clone** のアクティブなコピーと同じノードで実行するようにします。

```
# pcs constraint colocation add webfarm-stats with webfarm-clone
```

### 9.1.3. 粘着性のクローン作成

安定性のある割り当てパターンにするためクローンはデフォルトで若干の粘着性を備えています。**resource-stickiness** の値を指定しないと、クローンが使用する値は1となります。値を小さくすることで他のリソースのスコア計算への阻害を最小限に抑えながら、Pacemaker によるクラスター内の不要なコピーの移動を阻止することができます。

## 9.2. 多状態のリソース: 複数モードのリソース

多状態リソースはクローンリソースの特殊化です。昇格可能なクローンリソースにより、インスタンスの操作モードは、**Master** および **Slave** のいずれかにできます。モードの名前には特別な意味はありませんが、インスタンスの起動時に、**Slave** 状態で起動する必要があるという制限があります。

以下のコマンドを実行すると、リソースをマスター/スレーブクローンとして作成できます。

```
pcs resource create resource_id standard:provider:type|type [resource options] master
[master_options]
```

マスター/スレーブクローンの名前は **resource\_id-master** になります。



### 注記

Red Hat Enterprise Linux のリリース 7.3 以前では、マスター/スレーブのクローンの作成は以下の形式で行ってください。

```
pcs resource create resource_id standard:provider:type|type [resource options] --
master [meta master_options]
```

また、作成済みのリソースまたはリソースグループからマスター/スレーブリソースを作成することもできます。このコマンドを使用する場合はマスター/スレーブクローンの名前を指定することができます。名前を指定しない場合は **resource\_id-master** または **group\_name-master** になります。

```
pcs resource master master/slave_name resource_id/group_name [master_options]
```

リソースオプションの詳細は、「[リソースの作成](#)」を参照してください。

表9.2「多状態リソースのプロパティ」には、多状態リソースに指定できるオプションが記載されています。

表9.2 多状態リソースのプロパティ

フィールド	説明
<b>id</b>	多状態リソースに付ける名前
<b>priority</b> 、 <b>target-role</b> 、 <b>is-managed</b>	表6.3「リソースのメタオプション」を参照してください。
<b>clone-max</b> 、 <b>clone-node-max</b> 、 <b>notify</b> 、 <b>globally-unique</b> 、 <b>ordered</b> 、 <b>interleave</b>	表9.1「リソースのクローンオプション」を参照してください。
<b>master-max</b>	<b>master</b> ステータスにプロモートできるリソースのコピー数。デフォルト値は1です。
<b>master-node-max</b>	<b>master</b> ステータスにプロモートできるリソースのコピー数。デフォルト値は1です。

### 9.2.1. 多状態リソースの監視

マスターリソースのみに監視操作を追加するには、リソースに別の監視操作を追加します。同じリソース上では各監視操作の間隔が異なるようにする必要があります。

以下の例は、**ms\_resource** のマスターリソースに 11 秒間隔の監視操作を設定します。この監視操作は、10 秒間隔で行われるデフォルトの監視操作とは別に追加されます。

```
# pcs resource op add ms_resource interval=11s role=Master
```

### 9.2.2. 多状態制約

ほとんどの場合、多状態リソースにはアクティブなクラスターノードごとに1つのコピーがあります。そうではない場合は、リソースの場所制約を使用して、クラスターが優先的にコピーを割り当てるノードを指定できます。これらの制約は、通常のリソースと同様に記述されます。

リソースの場所制約については「[場所の制約](#)」を参照してください。

リソースをマスターにするかスレーブにするかを指定するコロケーション制約を作成することができます。次のコマンドは、リソースのコロケーション制約を作成します。

```
pcs constraint colocation add [master|slave] source_resource with [master|slave] target_resource [score] [options]
```

コロケーション制約の詳細は「[リソースのコロケーション](#)」を参照してください。

多状態リソースが含まれる順序制約を設定する場合、リソースに指定できるアクションの1つがリソースのスレーブからマスターへの昇格を指定する **promote** です。さらに、**demote** を指定するとリソースをマスターからスレーブに降格できます。

順序制約を設定するコマンドは次のようになります。

```
pcs constraint order [action] resource_id then [action] resource_id [options]
```

リソースの順序制約については「[順序の制約](#)」を参照してください。

### 9.2.3. 多状態の粘着性 (Stickiness)

安定性のある割り当てパターンを実現するため、多状態リソースはデフォルトで若干の粘着性を備えています。**resource-stickiness** の値を指定しないと、1 が値として使用されます。値を小さくすることで他のリソースのスコア計算への阻害を最小限に抑えながら、Pacemaker によるクラスター内の不要なコピーの移動を阻止することができます。

## 9.3. リソースとしての仮想ドメインの設定

**pcs resource create** コマンドを使用し、**VirtualDomain** をリソースタイプとして指定して、**libvirt** 仮想化フレームワークにより管理される仮想ドメインを設定できます。

仮想ドメインをリソースとして設定する場合は、以下の点を考慮してください。

- 仮想ドメインは、クラスターリソースとして設定する前に停止する必要があります。
- 仮想ドメインをクラスターリソースにすると、クラスターツールを使用しない限り、起動、停止、または移行を行うことができません。



- クラスターリソースとして設定した仮想ドメインを、ホストの起動時に起動するように設定することはできません。
- すべてのノードは、それぞれの管理される仮想ドメインの必要な設定ファイルおよびストレージデバイスにアクセスできる必要があります。

クラスターが仮想ドメイン内のサービスを管理できるようにする場合は、仮想ドメインをゲストノードとして設定できます。ゲストノードの設定は、「[pacemaker\\_remote サービス](#)」を参照してください。

仮想ドメインの設定は、[仮想化の導入および管理ガイド](#)を参照してください。

表9.3「[仮想ドメインリソースのリソースオプション](#)」では、**VirtualDomain** リソースに設定できるリソースオプションを説明します。

表9.3 仮想ドメインリソースのリソースオプション

フィールド	デフォルト	説明
<b>config</b>		(必須) この仮想ドメインの <b>libvirt</b> 設定ファイルへの絶対パス。
<b>hypervisor</b>	システムに依存	接続先のハイパーバイザーの URI。 <b>virsh --quiet uri</b> コマンドを実行してシステムのデフォルト URI を判別できます。
<b>force_stop</b>	<b>0</b>	停止時にドメインを常に強制的にシャットダウン (破棄) します。デフォルト動作では、正常なシャットダウンの試行に失敗した後でのみ、強制シャットダウンを実行します。仮想ドメイン (または仮想化バックエンド) が正常なシャットダウンに対応していない場合に限り、これを <b>true</b> に設定する必要があります。
<b>migration_transport</b>	システムに依存	移行中にリモートハイパーバイザーに接続するのに使用されるトランスポート。このパラメーターを省略すると、リソースは <b>libvirt</b> のデフォルトトランスポートを使用して、リモートハイパーバイザーに接続します。
<b>migration_network_suffix</b>		専用の移行ネットワークを使用します。移行 URI は、このパラメーターの値をノード名の末尾に追加することにより設定されます。ノード名が完全修飾ドメイン名 (FQDN) の場合は、FQDN の最初のピリオド (.) の直前に接尾辞を挿入します。この設定されたホスト名がローカルで解決可能であり、関連する IP アドレスが優先ネットワークを介して到達可能であることを確認してください。
<b>monitor_scripts</b>		仮想ドメイン内でサービスの監視を追加で実行する場合は、監視するスクリプトの一覧とともに、このパラメーターを追加します。 <b>注記:</b> 監視スクリプトを使用する場合、 <b>start</b> および <b>migrate_from</b> の操作は、すべての監視スクリプトが正常に完了した場合にのみ完了します。この遅延に対応できるように、この操作のタイムアウトを必ず設定してください。
<b>autoset_utilization_cpu</b>	<b>true</b>	<b>true</b> に設定すると、監視の実行時に、エージェントが <b>virsh</b> から <b>domainU</b> の <b>vCPU</b> 数を検出し、これをリソースの CPU 使用率に組み込みます。



フィールド	デフォルト	説明
<b>autoset_utilization_hv_memory</b>	<b>true</b>	true に設定すると、監視の実行時に、エージェントが <b>virsh</b> から <b>Max memor</b> 数を検出し、これをリソースの <b>hv_memory</b> の使用率に組み込みます。
<b>migrateport</b>	ランダムハイポート	このポートは、 <b>qemu</b> 移行 URI で使用されます。これを設定しないと、ポートにはランダムハイポートが使用されます。
<b>snapshot</b>		仮想マシンイメージが保存されるスナップショットディレクトリーへのパス。このパラメーターが設定されている場合、仮想マシンの RAM 状態は、停止時にスナップショットディレクトリーのファイルに保存されます。起動時にドメインのステータスファイルが存在すると、ドメインは、最後に停止する直前の状態に復元されます。このオプションは、 <b>force_stop</b> オプションと互換性はありません。

**VirtualDomain** リソースオプションに加えて、**allow-migrate** メタデータオプションを設定して、リソースの別のノードへのライブ移行を許可できます。このオプションを **true** に設定すると、状態を失うことなくリソースを移行できます。このオプションがデフォルトの状態である **false** に設定されていると、仮想ドメインは、ノード間で移行される際に、最初のノードでシャットダウンしてから、2 番目のノードで再起動します。

以下の手順を使用して **VirtualDomain** リソースを作成します。

1. 仮想マシンを管理するために **VirtualDomain** リソースエージェントを作成する場合、Pacemaker は仮想マシンの xml 設定ファイルをディスクのファイルにダンプすることを必要とします。たとえば、**guest1** という名前の仮想マシンを作成し、ホストにあるファイルに xml をダンプします。好きなファイル名を使用できますが、この例では **/etc/pacemaker/guest1.xml** を使用します。

```
# virsh dumpxml guest1 > /etc/pacemaker/guest1.xml
```

2. ゲストノードが実行している場合はシャットダウンします。Pacemaker は、クラスターで設定されているノードを起動します。
3. **VirtualDoman** リソースを **pcs resource create** コマンドを使用して設定します。たとえば、以下のコマンドは、**VM** という名前の **VirtualDomain** リソースを設定します。**allow-migrate** オプションは **true** に設定されるため、**pcs resource move VM nodeX** コマンドはライブ移行として実行されます。

```
# pcs resource create VM VirtualDomain config=../vm.xml \
migration_transport=ssh meta allow-migrate=true
```

## 9.4. PACEMAKER\_REMOTE サービス

**pacemaker\_remote** サービスを使用すると、**corosync** を実行していないノードをクラスターに統合し、そのリソースが実際のクラスターノードであるかのように、クラスターがリソースを管理できます。

**pacemaker\_remote** サービスが提供する機能には以下が含まれます。

- **pacemaker\_remote** サービスは、RHEL 7.7 の Red Hat サポート制限である 32 ノードを超えた拡張を可能にします。
- **pacemaker\_remote** サービスを使用すると、仮想環境をクラスターリソースとして管理でき、さらに仮想環境内の個別のサービスをクラスターリソースとして管理できます。

**pacemaker\_remote** サービスは、以下の用語を使用して記述されます。

- **クラスターノード** - 高可用性サービスを実行しているノード (**pacemaker** および **corosync**)。
- **リモートノード** - **pacemaker\_remote** を実行して、**corosync** クラスターメンバーシップを必要としないクラスターにリモートで統合するノード。リモートノードは、**ocf:pacemaker:remote** リソースエージェントを使用するクラスターリソースとして設定されます。
- **ゲストノード** - **pacemaker\_remote** サービスを実行する仮想ゲストノード。仮想ゲストリソースはクラスターにより管理されます。クラスターにより起動し、リモートノードとしてクラスターに統合されます。
- **pacemaker\_remote**: Pacemaker クラスター環境のリモートノードおよびゲストノード (KVM および LXC) 内でリモートアプリケーション管理を実行できるサービスデーモン。このサービスは、**corosync** を実行していないノード上でリソースをリモートで管理できる Pacemaker のローカルリソース管理デーモン (LRMD) の強化バージョンです。
- **LXC** - **libvirt-lxc** Linux コンテナドライバで定義される Linux Container

**pacemaker\_remote** サービスを実行している Pacemaker クラスターには次のような特徴があります。

- リモートノードおよびゲストノードは、**pacemaker\_remote** サービスを実行します (仮想マシン側で必要な設定はほとんどありません)。
- クラスターノードで実行しているクラスタースタック (**pacemaker** および **corosync**) はリモートノードで **pacemaker\_remote** サービスに接続するため、クラスターに統合できます。
- クラスターノードで実行しているクラスタースタック (**pacemaker** および **corosync**) はゲストノードを開始し、ゲストノードで **pacemaker\_remote** サービスに即座に接続するため、クラスターに統合できます。

クラスターノードと、クラスターノードが管理するリモートおよびゲストノードの主な違いは、リモートおよびゲストノードはクラスタースタックを実行しないことです。そのため、リモートおよびゲストノードには以下の制限があります。

- クォーラムでは実行されない
- フェンスデバイスのアクションを実行しません
- クラスターの指定コントローラー (DC) として機能できない
- **pcs** コマンドのすべてを実行できません

その一方で、リモートノードおよびゲストノードは、クラスタースタックに関連するスケーラビリティの制限に拘束されません。

このような制限事項以外に、リモートノードとゲストノードは、リソース管理に関してクラスターノードと同様に動作し、リモートノードとゲストノード自体をフェンスすることができます。クラスターは、各リモートノードおよびゲストノードのリソースを完全に管理し、監視できます。このようなノー

ドに制約を作成したり、ノードをスタンバイ状態にできます。または、**pcs** コマンドを使用して、クラスターノードでその他の動作を実行することもできます。リモートノードおよびゲストノードは、クラスターノードと同様にクラスターステータスの出力に表示されます。

### 9.4.1. ホストとゲストの認証

クラスターノードと `pacemaker_remote` の間の接続には、TLS (Transport Layer Security) が使用され、PSK (Pre-Shared Key) の暗号化と TCP 上の認証 (デフォルトで 3121 ポートを使用) でセキュア化されます。そのため、クラスターノードと、**pacemaker\_remote** を実行しているノードは、同じ秘密鍵を共有する必要があります。デフォルトでは、クラスターノードとリモートノードの両方でこのキーを `/etc/pacemaker/authkey` に格納する必要があります。

Red Hat Enterprise Linux 7.4 以降では、**pcs cluster node add-guest** コマンドでゲストノードの **authkey** を、**pcs cluster node add-remote** コマンドでリモートノードの **authkey** をセットアップできます。

### 9.4.2. ゲストノードリソースのオプション

ゲストノードとして動作するよう仮想マシンまたは LXC リソースを設定する場合、仮想マシンを管理する **VirtualDomain** リソースを作成します。**VirtualDomain** リソースに設定できるオプションの説明は、[表9.3「仮想ドメインリソースのリソースオプション」](#) を参照してください。

**VirtualDomain** リソースオプションのほかに、メタデータオプションはリソースをゲストノードとして定義し、接続パラメーターを定義します。Red Hat Enterprise Linux 7.4 では、**pcs cluster node add-guest** コマンドを使用してこれらのリソースオプションを設定してください。7.4 以前のリリースでは、リソースを作成する際に、これらのオプションを設定できます。[表9.4「KVM/LXC リソースをリモートノードとして設定するメタデータオプション」](#) では、このようなメタデータオプションについて説明します。

表9.4 KVM/LXC リソースをリモートノードとして設定するメタデータオプション

フィールド	デフォルト	説明
<b>remote-node</b>	<none>	このリソースが定義するゲストノードの名前。リソースをゲストノードとして有効にし、ゲストノードの識別に使用される一意名を定義します。 <b>警告:</b> この値を、リソースやノードの ID と重複させることはできません。
<b>remote-port</b>	3121	<b>pacemaker_remote</b> へのゲスト接続に使用するカスタムのポートを設定します。
<b>remote-addr</b>	ホスト名として使用される <b>remote-node</b> 値	リモートノードの名前がゲストのホスト名ではない場合に接続する IP アドレスまたはホスト名
<b>remote-connect-timeout</b>	60s	保留中のゲスト接続がタイムアウトするまでの時間

### 9.4.3. リモートノードリソースのオプション

リモートノードは、**ocf:pacemaker:remote** がリソースエージェントとして指定された状態で、クラスターリソースとして定義されます。Red Hat Enterprise Linux 7.4 では、**pcs cluster node add-remote** コマンドを使用して、このリソースを作成してください。7.4 以前のリリースでは、**pcs resource**

**create** コマンドを使用してこのリソースを作成できます。表9.5「リモートノードのリソースオプション」は、**remote** リソースに設定できるリソースオプションを説明します。

表9.5 リモートノードのリソースオプション

フィールド	デフォルト	説明
<b>reconnect_interval</b>	0	リモートノードへのアクティブな接続が切断された後、リモートノードへの再接続を試みる前に待機する時間(秒単位)。この待機期間は繰り返し発生します。待機期間の後に再接続に失敗した場合、待機期間の後に、新しい再接続が試行されます。このオプションが使用されると、Pacemaker は待機期間の後に無限にリモートノードへ接続を試みます。
<b>server</b>		接続するサーバーの場所。IP アドレスまたはホスト名を指定できます。
<b>port</b>		接続する TCP ポート。

#### 9.4.4. ポートのデフォルトの場所の変更

Pacemaker または **pacemaker\_remote** のいずれかのポートのデフォルトの場所を変更する必要がある場合は、これらのデーモンのどちらにも影響を与える **PCMK\_remote\_port** 環境変数を設定できます。この環境変数は、以下のように **/etc/sysconfig/pacemaker** に配置して有効にできます。

```
##==##==# Pacemaker Remote
...
#
# Specify a custom port for Pacemaker Remote connections
PCMK_remote_port=3121
```

特定のゲストノードまたはリモートノードによって使用されるデフォルトのポートを変更する場合は、**PCMK\_remote\_port** 変数をそのノードの **/etc/sysconfig/pacemaker** ファイルに設定する必要があります。また、ゲストノードまたはリモートノードの接続を作成するクラスターリソースを同じポート番号で設定する必要もあります(ゲストノードの場合は **remote-port** メタデータオプション、リモートノードの場合は **port** オプションを使用します)。

#### 9.4.5. 設定の概要: KVM ゲストノード

本セクションでは、**libvirt** と KVM 仮想ゲストを使用して、Pacemaker で仮想マシンを起動し、そのマシンをゲストノードとして統合する方法の概要を説明します。

1. 「リソースとしての仮想ドメインの設定」で説明したように、**VirtualDomain** リソースを設定します。
2. Red Hat Enterprise Linux 7.3 以前を使用しているシステムでは、以下の手順に従って、すべてのクラスターノードと仮想マシン上で、パス **/etc/pacemaker/authkey** で同じ暗号化キーを配置します。これにより、リモート通信や認証を保護することができます。
  - a. 以下の一連のコマンドをすべてのノードで実行し、安全なパーミッションの **authkey** ディレクトリを作成します。

```
# mkdir -p --mode=0750 /etc/pacemaker
# chgrp haclient /etc/pacemaker
```

- b. 以下のコマンドは、暗号化キーを作成する方法の1つを示しています。キーは1度だけ作成し、すべてのノードにコピーする必要があります。

```
# dd if=/dev/urandom of=/etc/pacemaker/authkey bs=4096 count=1
```

3. Red Hat Enterprise Linux 7.4 では、すべての仮想マシンで以下のコマンドを実行し、**pacemaker\_remote** パッケージをインストールし、**pcsd** サービスを起動し、これを起動時に実行できるようにし、ファイアウォールを介して TCP のポート 3121 を許可します。

```
# yum install pacemaker-remote resource-agents pcs
# systemctl start pcsd.service
# systemctl enable pcsd.service
# firewall-cmd --add-port 3121/tcp --permanent
# firewall-cmd --add-port 2224/tcp --permanent
# firewall-cmd --reload
```

Red Hat Enterprise Linux 7.3 以前では、すべての仮想マシンで以下のコマンドを実行し、**pacemaker\_remote** パッケージをインストールして、**pacemaker\_remote** サービスを起動し、これを起動時に実行できるようにして、ファイアウォールを介して TCP のポート 3121 を許可します。

```
# yum install pacemaker-remote resource-agents pcs
# systemctl start pacemaker_remote.service
# systemctl enable pacemaker_remote.service
# firewall-cmd --add-port 3121/tcp --permanent
# firewall-cmd --add-port 2224/tcp --permanent
# firewall-cmd --reload
```

4. 各仮想マシンに、すべてのノードが認識できる静的ネットワークアドレスと一意なホスト名を割り当てます。ゲスト仮想マシンに静的 IP アドレスを設定する方法については、『仮想化の導入および管理ガイド』を参照してください。
5. Red Hat Enterprise Linux 7.4 以降では、以下のコマンドを使用して、既存の **VirtualDomain** リソースをゲストノードに変換します。このコマンドは、追加するゲストノードではなく、クラスターノードで実行する必要があります。リソースを変換する以外にも、このコマンドは **/etc/pacemaker/authkey** をゲストノードにコピーし、ゲストノードで **pacemaker\_remote** デーモンを起動し、これを有効にします。

```
pcs cluster node add-guest hostname resource_id [options]
```

Red Hat Linux 7.3 以前では、以下のコマンドを使用して既存の **VirtualDomain** リソースをゲストノードに変換します。このコマンドは、追加するゲストノードではなく、クラスターノードで実行する必要があります。

```
pcs cluster remote-node add hostname resource_id [options]
```

6. **VirtualDomain** リソースの作成後は、クラスターの他のノードと同じように、ゲストノードを扱うことができます。たとえば、クラスターノードから実行される次のコマンドのように、リソースを作成し、リソースにリソース制約を設定してゲストノードで実行できます。Red Hat

Enterprise Linux 7.3 時点では、ゲストノードをグループで含むことができ、ストレージデバイス、ファイルシステム、および VM をグループ化できます。

```
# pcs resource create webserver apache configfile=/etc/httpd/conf/httpd.conf op monitor
interval=30s
# pcs constraint location webserver prefers guest1
```

#### 9.4.6. 設定概要: リモートノード (Red Hat Enterprise Linux 7.4)

本セクションでは、Red Hat Enterprise 7.4 において Pacemaker リモートノードを設定し、そのノードを既存の Pacemaker クラスター環境に統合する手順の概要を説明します。

1. リモートノードを設定するノードで、ローカルファイアウォールを介してクラスター関連のサービスを許可します。

```
# firewall-cmd --permanent --add-service=high-availability
success
# firewall-cmd --reload
success
```



#### 注記

**iptables** を直接使用する場合や、**firewalld** 以外のファイアウォールソリューションを使用する場合は、単に TCP のポート 2224 および 3121 を開きます。

2. リモートノードで **pacemaker\_remote** デーモンをインストールします。

```
# yum install -y pacemaker-remote resource-agents pcs
```

3. リモートノードで、**pcsd** を開始し、有効にします。

```
# systemctl start pcsd.service
# systemctl enable pcsd.service
```

4. リモートノードとして追加するノードに **pcs** を認証していない場合は、認証します。

```
# pcs cluster auth remote1
```

5. 以下のコマンドを使用して、リモートノードリソースをクラスターに追加します。このコマンドは、関連するすべての設定ファイルを新規ノードに追加し、ノードを起動し、これをシステムの起動時に **pacemaker\_remote** を開始するように設定することもできます。このコマンドは、追加するリモートノードではなく、クラスターノードで実行する必要があります。

```
# pcs cluster node add-remote remote1
```

6. **remote** リソースをクラスターに追加した後、リモートノードを、クラスター内の他のノードを処理するのと同じように処理できます。たとえば、以下のコマンドをクラスターノードから実行すると、リソースを作成し、そのリソースにリソース制約を配置して、リモートノードで実行できます。

```
# pcs resource create webserver apache configfile=/etc/httpd/conf/httpd.conf op monitor
interval=30s
# pcs constraint location webserver prefers remote1
```



### 警告

リソースグループ、コロケーション制約、または順序制約でノード接続リソースを利用しないでください。

7. リモートノードのフェンスリソースを設定します。リモートノードは、クラスターノードと同じ方法でフェンスされます。クラスターノードと同様に、リモートノードで使用するフェンスリソースを設定します。リモートノードはフェンスアクションを開始できないことに注意してください。クラスターノードのみが、実際に別のノードに対してフェンシング操作を実行できます。

#### 9.4.7. 設定概要: リモートノード (Red Hat Enterprise Linux 7.3 以前)

本セクションでは、Red Hat Enterprise 7.3 以前のシステムにおいて Pacemaker リモートノードを設定し、そのノードを既存の Pacemaker クラスター環境に統合する手順の概要を説明します。

1. リモートノードを設定するノードで、ローカルファイアウォールを介してクラスター関連のサービスを許可します。

```
# firewall-cmd --permanent --add-service=high-availability
success
# firewall-cmd --reload
success
```



### 注記

**iptables** を直接使用する場合や、**firewalld** 以外のファイアウォールソリューションを使用する場合は、単に TCP のポート 2224 および 3121 を開きます。

2. リモートノードで **pacemaker\_remote** デーモンをインストールします。

```
# yum install -y pacemaker-remote resource-agents pcs
```

3. 適切に通信するには、すべてのノード (クラスターノードとリモートノードの両方) に同じ認証キーがインストールされている必要があります。既存ノードにすでにキーがある場合、そのキーを使用してリモートノードにコピーします。それ以外の場合はリモートノードで新しいキーを作成します。

リモートノードで以下のコマンドを実行し、セキュアなパーミッションを持つ認証キーのディレクトリーを作成します。

```
# mkdir -p --mode=0750 /etc/pacemaker
# chgrp haclient /etc/pacemaker
```

以下のコマンドは、リモートノードで暗号化キーを作成する方法の1つを示しています。

```
# dd if=/dev/urandom of=/etc/pacemaker/authkey bs=4096 count=1
```

4. リモートノードで **pacemaker\_remote** デーモンを開始し、有効にします。

```
# systemctl enable pacemaker_remote.service
# systemctl start pacemaker_remote.service
```

5. クラスターノードにて、リモートノードの認証キーと同じパスを持つ共有された認証キーの保存場所を作成し、そのディレクトリーにキーをコピーします。この例では、キーが作成されたリモートノードからキーがコピーされます。

```
# mkdir -p --mode=0750 /etc/pacemaker
# chgrp haclient /etc/pacemaker
# scp remote1:/etc/pacemaker/authkey /etc/pacemaker/authkey
```

6. クラスターノードから以下のコマンドを実行し、**remote** リソースを作成します。この例では、リモートノードは **remote1** になります。

```
# pcs resource create remote1 ocf:pacemaker:remote
```

7. **remote** リソースの作成後、リモートノードをクラスターの他のノードと同じように扱うことができます。たとえば、以下のコマンドをクラスターノードから実行すると、リソースを作成し、そのリソースにリソース制約を配置して、リモートノードで実行できます。

```
# pcs resource create webserver apache configfile=/etc/httpd/conf/httpd.conf op monitor
interval=30s
# pcs constraint location webserver prefers remote1
```



#### 警告

リソースグループ、コロケーション制約、または順序制約でノード接続リソースを利用しないでください。

8. リモートノードのフェンスリソースを設定します。リモートノードは、クラスターノードと同じ方法でフェンスされます。クラスターノードと同様に、リモートノードで使用するフェンスリソースを設定します。リモートノードはフェンスアクションを開始できないことに注意してください。クラスターノードのみが、実際に別のノードに対してフェンシング操作を実行できます。

### 9.4.8. システムアップグレードおよび **pacemaker\_remote**

Red Hat Enterprise Linux 7.3 より、アクティブな Pacemaker リモートノードで **pacemaker\_remote** サービスが停止した場合にノードの停止前にクラスターがノードをリソースから正常に移行するようになります。これにより、クラスターからノードを削除せずに、ソフトウェアのアップグレードやその



他の定期的なメンテナンスを実行できるようになりました。ただし、**pacemaker\_remote** がシャットダウンすると、クラスターは即座に再接続を試みます。リソースのモニタータイムアウトが発生する前に **pacemaker\_remote** が再起動されないと、クラスターは監視操作が失敗したと判断します。

アクティブな Pacemaker リモートノードで、**pacemaker\_remote** サービスが停止したときに監視が失敗しないようにするには、以下の手順に従って、**pacemaker\_remote** を停止する可能性があるシステム管理を実行する前に、ノードをクラスターから削除します。

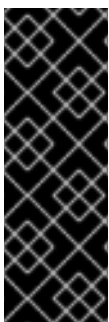


### 警告

Red Hat Enterprise Linux 7.2 以前のリリースでは、現在クラスターに統合されているノード上で **pacemaker\_remote** が停止するとクラスターはそのノードをフェンスします。**yum update** のプロセスの一部として自動的に停止した場合、システムが不安定な状態になることがあります (特に **pacemaker\_remote** と同時にカーネルもアップグレードされる場合)。Red Hat Enterprise Linux 7.2 以前のリリースでは、以下の手順にしたがって、**pacemaker\_remote** を停止する可能性があるシステム管理を実行する前にノードをクラスターから除去する必要があります。

1. ノードからすべてのサービスを除去する **pcs resource disable resourcename** コマンドを使用して、ノードの接続リソースを停止します。ゲストノードでは VM も停止されるため、VM をクラスターの外部で起動して (**virsh** などを使用) メンテナンスを実行する必要があります。
2. 必要なメンテナンスを実行します。
3. ノードをクラスターに戻す準備ができたなら、**pcs resource enable** でリソース再度有効にします。

## 9.5. DOCKER コンテナの PACEMAKER サポート (テクノロジープレビュー)



### 重要

Docker コンテナの Pacemaker サポートは、テクノロジープレビュー目的でのみ実現されています。テクノロジープレビューの意味については、[テクノロジープレビュー機能のサポート範囲](#) を参照してください。

テクノロジープレビューであるこの機能には1つの例外があります。Red Hat Enterprise Linux 7.4 以降、Red Hat は、Red Hat Openstack Platform (RHOSP) デプロイメントで Pacemaker バンドルの使用を完全にサポートします。

Pacemaker は、必要なインフラストラクチャーで Docker コンテナを起動するための特殊構文 (*bundle*) に対応しています。Pacemaker バンドルを作成したら、バンドルがカプセル化する Pacemaker リソースを作成できます。

- 「[Pacemaker バンドルリソースの設定](#)」では、Pacemaker バンドルを作成するコマンドについて説明し、各バンドルパラメーターに定義できるパラメーターについてまとめた表が記載されています。

- 「[バンドルでの Pacemaker リソースの設定](#)」では、Pacemaker バンドルに含まれているリソースの設定について説明しています。
- 「[Pacemaker バンドルの制限](#)」では、Pacemaker バンドルの制限について説明しています。
- 「[Pacemaker バンドル設定の例](#)」は、Pacemaker バンドルの設定例を説明しています。

### 9.5.1. Pacemaker バンドルリソースの設定

Docker コンテナの Pacemaker バンドルを作成するコマンド構文は以下の通りです。このコマンドを使用すると、その他のリソースをカプセル化しないバンドルが作成されます。バンドルでクラスターリソースを作成する方法については、「[バンドルでの Pacemaker リソースの設定](#)」を参照してください。

```
pcs resource bundle create bundle_id container docker [container_options] [network
network_options] [port-map port_options]... [storage-map storage_options]... [meta meta_options] [--
disabled] [--wait[=n]]
```

必要な *bundle\_id* パラメーターは、バンドルに対する一意の名前にする必要があります。--disabled オプションを指定すると、バンドルは自動的に起動されません。--wait オプションを指定すると、Pacemaker は、バンドルが起動し、成功時に 0 を、エラー時に 1 を返すまで最大 *n* 秒待機します。*n* を指定しないと、デフォルトの 60 分に設定されます。

以下のセクションでは、Pacemaker バンドルの各要素に設定できるパラメーターを説明します。

#### 9.5.1.1. Docker パラメーター

表9.6「[Docker コンテナのパラメーター](#)」では、バンドルに設定できる **docker** コンテナオプションを説明します。



#### 注記

Pacemaker で **docker** バンドルを設定する前に、Docker をインストールし、バンドルの実行が許可されている各ノード上に完全に設定した Docker イメージを展開します。

表9.6 Docker コンテナのパラメーター

フィールド	デフォルト	説明
<b>image</b>		Docker イメージタグ (必須)
<b>replicas</b>	正であれば <b>promoted-max</b> の値。それ以外は 1。	起動するコンテナインスタンスの数を指定する正の整数
<b>replicas-per-host</b>	1	単一ノードで起動できるコンテナインスタンスの数を指定する正の整数
<b>promoted-max</b>	0	正であれば、非負の整数は、マスターロールにおいてサービスを実行できる多くのレプリカとともに、コンテナ化されたサービスが複数のサービスとして扱われる必要があることを示しています。

フィールド	デフォルト	説明
<b>network</b>		これが指定されると、Docker コンテナのネットワーク設定として <b>docker run</b> コマンドに渡されます。
<b>run-command</b>	<b>/usr/sbin/pacemaker _remoted</b> バンドルが リソースを含む場合、そ れ以外はなし	このコマンドは、起動する際にコンテナ内で実行されます ("PID 1")。バンドルにリソースが含まれる場合、このコマンドは <b>pacemaker _remoted</b> デーモンを起動する必要があります (ただし、その他のタスクも実行するスクリプトでも問題ありません)。
<b>options</b>		<b>docker run</b> コマンドに渡す、その他のコマンドラインオプション

#### 9.5.1.2. (バンドルネットワークパラメーター

表9.7「バンドルリソースネットワークパラメーター」では、バンドルに設定できる **network** オプションを説明します。

表9.7 バンドルリソースネットワークパラメーター

フィールド	デフォルト	説明
<b>add-host</b>	TRUE	TRUE で <b>ip-range-start</b> が使用される場合は、Pacemaker により自動的に、コンテナ内の <b>/etc/hosts</b> ファイルに、各レプリカ名と割り当てられている IP のエントリーが指定されるようにします。
<b>ip-range-start</b>		これが指定されると、Pacemaker は、各コンテナインスタンスに対して暗黙的な <b>ocf:heartbeat:IPAddr2</b> リソースを作成します。これは、この IP アドレスで始まり、Docker 要素の <b>replicas</b> として指定されている限りのシーケンシャルアドレスとして使用します。これらのアドレスは、ホストのネットワークから使用して、コンテナ内のサービスに到達できます。ただし、コンテナ自身の中では表示されません。現在サポートされているアドレスは、IPv4 のみです。
<b>host-netmask</b>	32	<b>ip-range-start</b> が指定されると、IP アドレスがこの CIDR ネットマスクで (多数のビットとして) 作成されます。
<b>host-interface</b>		<b>ip-range-start</b> が指定されていると、IP アドレスがこのホストインターフェイスで作成されます (デフォルトでは、IP アドレスから決まります)。

フィールド	デフォルト	説明
<b>control-port</b>	3121	<p>バンドルに Pacemaker リソースが含まれる場合、クラスターは、コンテナ内の Pacemaker Remote との通信に、この整数の TCP ポートを使用します。コンテナがデフォルトポートでリッスンできない場合は、このポートを変更すると便利です。これは、コンテナが <b>ip-range-start</b> (この場合 <b>replicas-per-host</b> は1である必要があります) ではなく、ホストのネットワークを使用しようとしているか、バンドルがデフォルトポートですでにリッスンしている Pacemaker Remote ノードを実行できる場合に起こります。ホストやコンテナで設定されている <b>PCMK_remote_port</b> 環境変数は、バンドル接続に対しては無視されます。</p> <p>Pacemaker バンドル設定が <b>control-port</b> パラメーターを使用するときに、バンドルに独自の IP アドレスがある場合には、corosync を実行しているすべてのクラスターノード上およびそのクラスターノードから、その IP アドレスでポートを開く必要があります。あるいは、バンドルが <b>network="host"</b> コンテナパラメーターを設定している場合には、すべてのクラスターノードから、それぞれのクラスターノードの IP アドレスでポートを開く必要があります。</p>



## 注記

レプリカは、バンドル ID とダッシュそしてゼロから始まる整数カウンターで名前が付けられます。**httpd-bundle** というバンドル名により **replicas=2** が設定されれば、そのコンテナの名前は **httpd-bundle-0** と **httpd-bundle-1** になります。

ネットワークパラメーターに加え、バンドルに **port-map** パラメーターを任意で指定できます。[表 9.8「バンドルリソースポートマップパラメーター」](#) では **port-map** パラメーターを説明します。

表9.8 バンドルリソースポートマップパラメーター

フィールド	デフォルト	説明
<b>id</b>		ポートマッピングの一意の名前 (必須)
<b>port</b>		これが指定されると、ホストネットワーク上 ( <b>ip-range-start</b> が指定されている場合は、コンテナの、割り当てられた IP アドレス上) のこの TCP ポート番号に対する接続がコンテナネットワークに転送されます。 <b>port</b> または <b>range</b> のいずれ1つがポートマッピングで指定される必要があります。
<b>internal-port</b>	<b>port</b> の値	<b>port</b> と <b>internal-port</b> が指定されている場合は、ホストネットワーク上の <b>port</b> に対する接続が、コンテナネットワーク上のこのポートに転送されます。

フィールド	デフォルト	説明
<b>range</b>		<b>range</b> が指定されると、ホストネットワーク上 ( <b>ip-range-start</b> が指定されている場合は、コンテナの、割り当てられた IP アドレス上) のこれらの TCP ポート番号 ( <i>first_port-last_port</i> ) に対する接続が、コンテナネットワークの同じポートに転送されます。 <b>port</b> または <b>range</b> のいずれ1つがポートマッピングで指定される必要があります。



### 注記

バンドルにソースが含まれる場合、Pacemaker は自動的に **control-port** をマッピングします。そのため、ポートマッピングでは、そのポートを指定する必要はありません。

#### 9.5.1.3. バンドルストレージパラメーター

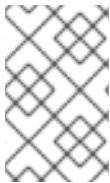
必要に応じて、バンドルに **storage-map** パラメーターを設定することができます。表9.9「バンドルリソースストレージマッピングパラメーター」では、これらのパラメーターを説明します。

表9.9 バンドルリソースストレージマッピングパラメーター

フィールド	デフォルト	説明
<b>id</b>		ストレージマッピングの一意の名前 (必須)
<b>source-dir</b>		コンテナにマッピングされるホストファイルシステム上の絶対パス。 <b>storage-map</b> パラメーターを設定する際には、 <b>source-dir</b> と <b>source-dir-root</b> のいずれかを指定する必要があります。
<b>source-dir-root</b>		各コンテナインスタンスのホスト上で異なるサブディレクトリーを使用した、コンテナにマッピングされるホストのファイルシステム上のパスの開始。このサブディレクトリーには、バンドル名と同じ名前が付けられ、ダッシュと 0 から始まる整数カウンターも加えられます。 <b>storage-map</b> パラメーターを設定するときには、 <b>source-dir</b> と <b>source-dir-root</b> のいずれかを完全に指定する必要があります。
<b>target-dir</b>		ホストストレージがマッピングされるコンテナ内のパス名 (必須)
<b>options</b>		ストレージをマッピングする際に使用するファイルシステムマウントオプション

ホスト上のサブディレクトリーが **source-dir-root** パラメーターで名前が付けられる仕組みの例として、**source-dir-root=/path/to/my/directory**、**target-dir=/srv/appdata**、バンドルに **replicas=2** で **mybundle** という名前が指定されている場合、クラスターは、**mybundle-0** と **mybundle-1** というホスト名で2つのコンテナインスタンスを作成します。また、コンテナ **/path/to/my/directory/mybundle-0** と **/path/to/my/directory/mybundle-1** を実行しているホストで2

つのディレクトリを作成します。各コンテナには、いずれかのディレクトリが与えられ、そのコンテナ内で実行しているアプリケーションには、`/srv/appdata` というディレクトリが表示されます。



#### 注記

Pacemaker は、ソースディレクトリが既にホストに存在しない場合の動作を定義しません。ただし、コンテナテクノロジーまたはそのリソースエージェントがソースディレクトリを作成します。



#### 注記

バンドルに Pacemaker リソースが含まれる場合、Pacemaker は自動的に **source-dir=/etc/pacemaker/authkey** **target-dir=/etc/pacemaker/authkey** と **source-dir-root=/var/log/pacemaker/bundle** **target-dir=/var/log** と同等のものをコンテナにマッピングします。そのため、**storage-map** パラメーターを設定する際には、これらのパスを指定する必要はありません。



#### 重要

**PCMK\_authkey\_location** 環境変数は、クラスターのノード上の `/etc/pacemaker/authkey` のデフォルト以外に設定することはできません。

### 9.5.2. バンドルでの Pacemaker リソースの設定

バンドルは必要に応じて、1つの Pacemaker クラスターリソースを含めることができます。バンドルに含まれていないリソースと同様に、クラスターリソースには、操作、インスタンス属性、メタデータ属性を定義することができます。バンドルにリソースが含まれる場合、コンテナイメージは Pacemaker Remote デーモンを含む必要があります。また、**ip-range-start** または **control-port** をバンドルで設定する必要があります。Pacemaker は、接続に対して暗黙的な **ocf:pacemaker:remote** リソースを作成し、コンテナ内で Pacemaker Remote を起動して、Pacemaker Remote でリソースを監視・管理します。バンドルに2つ以上のコンテナインスタンス(レプリカ)がある場合、Pacemaker リソースは暗黙的なクローンとして機能します。バンドルが **promoted-max** オプションを、0を超えるように設定した場合、これは多状態クローンになります。

コマンドの **bundle** パラメーター、リソースを含めるバンドル ID を指定して、**pcs resource create** コマンドで Pacemaker バンドルでリソースを作成します。リソースを含む Pacemaker バンドルの作成例は、「[Pacemaker バンドル設定の例](#)」を参照してください。



#### 重要

リソースを含むバンドルのコンテナにはアクセス可能なネットワーク環境が必要です。それにより、クラスターノードの Pacemaker はコンテナ内の Pacemaker Remote にコンタクトできます。たとえば、**docker** オプション **--net=none** はリソースと合わせて使用しないでください。オプションの **ip-range-start** は、リソースとは使うべきではありません。**docker** オプション **--net=host** が使用されている場合(コンテナがホストのネットワーク領域を共有している)場合、バンドルごとに一意の **control-port** パラメーターを指定する必要があります。ファイアウォールでは、**control-port** にアクセスできるように設定を行う必要があります。

#### 9.5.2.1. ノード属性とバンドルリソース



バンドルにクラスターリソースが含まれる場合、リソースエージェントはマスタースコアなどのノード属性を設定する可能性があります。ただし、コンテナでは、ノードが属性を取得するべきかどうかははっきりしません。

コンテナがホストされているノードに関係なく、コンテナが同じ共有ストレージを使用している場合は、バンドルノード自体でマスタースコアを使用することが適切です。一方、コンテナが、基礎となるホストからエクスポートされたストレージを使用する場合は、基礎となるホストでマスタースコアを使用することがより適切です。これは特定の状況に依存するため、**container-attribute-target** リソースメタデータ属性では、ユーザーは使用するアプローチを指定することができます。**host** に設定されると、ユーザー定義型のノード属性は、基礎となるホストでチェックされます。その他の場合は、ローカルノード(この場合はバンドルノード)が使用されます。この動作は、ユーザー定義属性にのみ適用されます。クラスターは、**#uname** などのクラスター定義型の属性に対してローカルノードを常にチェックします。

**container-attribute-target** が **host** に設定されると、クラスターは、追加の環境変数をリソースエージェントに渡します。これにより、ノード属性を正しく設定できるようになります。

#### 9.5.2.2. メタデータ属性とバンドルリソース

バンドルで設定されているメタデータ属性は、バンドルに含まれるリソースや、バンドルに対して Pacemaker によって作成されたリソースによって継承されます。これには、**priority**、**target-role**、**is-managed** などのオプションが含まれます。

#### 9.5.3. Pacemaker バンドルの制限

Pacemaker バンドルは以下の制限で動作します。

- バンドルはグループに含まれていないことや、**pcs** コマンドで明示的にクローン化されていないことがあります。これには、バンドルが含むリソースや、バンドルに対して Pacemaker によって明示的に作成されたリソースが含まれます。ただし、バンドルの **replicas** の値が1未満で設定されると、バンドルはクローンであるかのように動作することに注意してください。
- バンドルが管理されていない場合や、クラスターがメンテナンスモードの際に Pacemaker を再起動すると、バンドルが不具合を起こすことがあります。
- バンドルには、インスタンス属性、使用率属性、または操作がありません。しかし、バンドルに含まれるリソースには、これらがあります。
- リソースを含むバンドルは、バンドルが、一意な異なる **control-port** を使用する場合にのみ、Pacemaker Remote ノードで実行できます。

#### 9.5.4. Pacemaker バンドル設定の例

以下の例では、**httpd-bundle** というバンドル ID で Pacemaker **bundle** を作成します。これには、**httpd** というリソース ID を持つ **ocf:heartbeat:apache** が含まれます。

この手順には、以下の前提設定が必要です。

- Docker が、クラスターの各ノードでインストールされ有効化されている。
- **pcmktest:http** という既存の Docker イメージが存在する。
- コンテナイメージに、Pacemaker Remote デーモンが含まれている。
- コンテナイメージに、設定済みの Apache Web サーバーが含まれている。

- クラスターの各ノードに、**/var/local/containers/httpd-bundle-0**、**/var/local/containers/httpd-bundle-1**、**/var/local/containers/httpd-bundle-2** というディレクトリーがあり、Web サーバーの root に **index.html** ファイルが含まれている。稼働中は、単一の共有ドキュメント root が対象となるが、この例では、この設定により、各ホスト上の **index.html** ファイルを別のものにすることが可能。そのため、Web サーバーに接続して、サービスされる **index.html** ファイルを確認することが可能。

この手順により、Pacemaker バンドルに以下のパラメーターが設定されます。

- バンドル ID は **httpd-bundle** です。
- 以前設定した Docker コンテナイメージは **pcmktest:http** です。
- この例は、3 コンテナインスタンスを起動します。
- この例では、コマンドラインオプション **--log-driver=journald** を **docker run** コマンドに渡します。このパラメーターは必要ではありませんが、その他のオプションを **docker** コマンドに渡す方法を示すために追加しています。**--log-driver=journald** の値は、コンテナ内のシステムログが基礎となるホストの **systemd** ジャーナルにログインされることを意味します。
- Pacemaker は、3 つの連続した暗黙的な **ocf:heartbeat:IPaddr2** リソースを作成します。これは、各コンテナに対して 1 つ作成され、IP アドレス 192.168.122.131 で始まります。
- IP アドレスは、ホストインターフェイス **eth0** で作成されます。
- IP アドレスは、CIDR ネットマスクが 24 で作成されます。
- この例では、**http-port** というポートマップ ID を作成します。コンテナに割り当てられている IP アドレスのポート 80 に対する接続がコンテナネットワークに転送されます。
- この例では、**httpd-root** というストレージマップ ID を作成します。このストレージマッピングについて以下で説明します。
  - **source-dir-root** の値は **/var/local/containers** です。これは各コンテナインスタンスに対して、ホスト上の異なるサブディレクトリーを使用し、コンテナにマッピングされるホストのファイルシステム上のパスの開始を指定します。
  - **target-dir** の値は **/var/www/html** で、ホストストレージがマッピングされるコンテナ内のパス名を指定します。
  - このファイルシステム **rw** マウントオプションは、ストレージのマッピングの際に使用されます。
  - この例のコンテナはリソースを含むため、Pacemaker は自動的に、コンテナに **source-dir=/etc/pacemaker/authkey** と同等のものをマッピングします。このコンテナでは、ストレージマッピングにそのパスを指定する必要はありません。

この例では、既存のクラスター設定が、**temp-cib.xml** という名前の一時ファイルに配置され、**temp-cib.xml.deltasrc** にコピーされます。クラスター設定に対する設定のすべては、**tmp-cib.xml** ファイルに対して行われます。updates が完了すると、この手順では **pcs cluster cib-push** コマンドの **diff-against** を使用して、設定ファイルの更新のみがアクティブな設定ファイルにプッシュされるようにします。

```
# pcs cluster cib tmp-cib.xml
# cp tmp-cib.xml tmp-cib.xml.deltasrc
# pcs -f tmp.cib.xml resource bundle create httpd-bundle \
container docker image=pcmktest:http replicas=3 \
```



```
options=--log-driver=journald \
network ip-range-start=192.168.122.131 host-interface=eth0 \
host-netmask=24 port-map id=httpd-port port=80 \
storage-map id=httpd-root source-dir-root=/var/local/containers \
target-dir=/var/www/html options=rw \
# pcs -f tmp-cib.xml resource create httpd ocf:heartbeat:apache \
statusurl=http://localhost/server-status bundle httpd-bundle
# pcs cluster cib-push tmp-cib.xml diff-against=tmp-cib.xml.deltasrc
```

## 9.6. 使用と配置ストラテジー

Pacemaker は、すべてのノードのリソース割り当てスコアに従って、リソースを配置する場所を決定します。このリソースは、リソースのスコアが最も高いノードに割り当てられます。この割り当てスコアは、リソースの制約、**resource-stickiness** の設定、各ノードにおけるリソースの過去の障害履歴、各ノードの使用率などの要因の組み合わせから導出されます。

すべてのノードでリソース割り当てスコアが等しい場合、デフォルトの配置ストラテジーにより、Pacemaker は、負荷を分散するために、割り当てられたリソースの数が最も少ないノードを選択します。各ノードのリソースの数が等しい場合は、CIB の最初の対象ノードがリソースを実行するのに選択されます。

ただし、多くの場合、リソースが使用するノードの容量 (メモリーや I/O など) の割合は、状況によって大きく異なります。そのため、ノードに割り当てられているリソースの数のみを考慮して、いつでも思想的な負荷分散が行われるとは限りません。また、リソースの合計要件が、指定の容量を超えるように配置されている場合は、リソースが完全に起動できなくなるか、パフォーマンスが低下した状態で起動することがあります。このような要因を考慮するために、Pacemaker では次の要素を設定できます。

- 特定のノードの容量
- 特定のリソースが必要な容量
- リソースの配置の全体的なストラテジー

以下のセクションでは、これらのコンポーネントの設定方法を説明します。

### 9.6.1. 使用率属性

ノードの容量、またはリソースが必要な容量を設定するには、ノードとリソースに *utilization attributes* を使用します。これを行うには、リソースの使用率変数を設定し、その変数に値を割り当ててリソースに必要なものを示してから、同じ使用率変数をノードに設定し、その変数に値を割り当ててそのノードが提供するものを示します。

設定に応じて使用率属性に名前を付け、設定に必要な数だけ名前と値のペアを定義できます。使用率属性の値は整数である必要があります。

Red Hat Enterprise Linux 7.3 では、**pcs** コマンドを使用して使用率属性を設定できます。

以下の例では、2つのノードに対して CPU キャパシティの使用率属性を設定して、属性 **cpu** を命名します。また、RAM 容量の使用率属性も設定して、属性 **memory** を命名します。この例では、以下のように設定されています。

- ノード 1 は、CPU 容量 2 と、RAM 容量 2048 を指定するように定義されています。
- ノード 2 は、CPU 容量 4 と、RAM 容量 2048 を指定するように定義されています。

```
# pcs node utilization node1 cpu=2 memory=2048
# pcs node utilization node2 cpu=4 memory=2048
```

次の例では、3つの異なるリソースに必要な、同じ使用率属性を指定します。この例の場合:

- **dummy-small** リソースには、CPU 容量 1 と、RAM 容量 1024 が必要です。
- **dummy-medium** リソースには、CPU 容量 2 と、RAM 容量 2048 が必要です。
- **dummy-large** リソースには、CPU 容量 1 と、RAM 容量 3072 が必要です。

```
# pcs resource utilization dummy-small cpu=1 memory=1024
# pcs resource utilization dummy-medium cpu=2 memory=2048
# pcs resource utilization dummy-large cpu=3 memory=3072
```

使用率属性で定義されているリソースの要件を満たすのに十分な空き容量があれば、ノードはリソースに適格と見なされます。

### 9.6.2. 配置ストラテジー

ノードが提供する容量と、リソースが必要とする容量を設定したら、**placement-strategy** クラスタープロパティを設定する必要があります。これを設定しないと、容量を設定しても効果がありません。クラスターのプロパティの詳細は [12章 Pacemaker クラスターのプロパティ](#) を参照してください。

**placement-strategy** クラスタープロパティには、4つの値を使用できます。

- **default** - 使用率の値は全く考慮されません。リソースは、割り当てスコアに従って割り当てられます。スコアが同じであれば、リソースはノード間で均等に分散されます。
- **utilization** - 使用率の値は、ノードが適格と見なされるかどうか (つまり、リソースの要件を満たすのに十分な空き容量があるかどうか) を決定する場合にのみ考慮されます。負荷分散は、ノードに割り当てられたリソースの数に基づいて行われます。
- **balanced** - 使用率の値は、ノードがリソースを提供する資格があるかどうかを判断する際、および負荷分散する際に考慮されるため、リソースのパフォーマンスを最適化する方法でリソースを分散しようとします。
- **minimal** - 使用率の値は、ノードがリソースを提供する資格があるかどうかを決定する場合に限り考慮されます。負荷分散は、できるだけ少ないノードにリソースを集中させて、残りのノードで電力を節約できるようにします。

以下の例のコマンドでは、**placement-strategy** の値を **balanced** に設定します。このコマンドを実行すると、Pacemaker は、複雑な一連のコロケーション制約を使用せずに、リソースからの負荷がクラスター全体に均等に分散されるようにします。

```
# pcs property set placement-strategy=balanced
```

### 9.6.3. リソースの割り当て

以下のサブセクションでは、Pacemaker がリソースを割り当てる仕組みをまとめています。

#### 9.6.3.1. ノードの優先順位

Pacemaker は、以下のストラテジーに従ってリソースを割り当てる際に優先されるノードを決定します。

- 重みが最も高いノードが最初に消費されます。ノードの重みは、ノードの状態を表すためにクラスターによって維持されるスコアです。
- ノードの重みが、複数のノードで同じ場合は、以下のようになります。
  - **placement-strategy** クラスタープロパティが **default** または **utilization** の場合は、以下のようになります。
    - 割り当てられているリソースの数が最も少ないノードが最初に使用されます。
    - 割り当てられているリソースの数が等しい場合は、CIB に登録されている最初の対象ノードが最初に使用されます。
  - **placement-strategy** クラスタープロパティが **balanced** である場合は、以下のようになります。
    - 空き容量が最も多いノードが最初に使用されます。
    - ノードの空き容量が等しい場合は、割り当てられているリソースの数が最も少ないノードが最初に使用されます。
    - ノードの空き容量が等しく、割り当てられているリソースの数が等しい場合は、CIB に最初に登録されている対象ノードが最初に使用されます。
  - **placement-strategy** クラスタープロパティが **minimal** の場合は、CIB に登録されている最初の対象ノードが最初に使用されます。

### 9.6.3.2. ノードの容量

Pacemaker は、以下のストラテジーに従って、どのノードに最も空き容量があるかを判断します。

- 使用率属性が1種類だけ定義されている場合、空き容量は単純な数値比較となります。
- 定義されている使用属性の種類が複数になる場合は、ほとんどの属性タイプで数値が最も高いノードの空き容量が、最も大きくなります。以下に例を示します。
  - NodeA の空き CPU が多く、NodeB の空きメモリーが多い場合は、互いの空き容量は等しくなります。
  - NodeA の空き CPU が多く、NodeB の空きメモリーとストレージが多い場合、NodeB の方が空き容量が多くなります。

### 9.6.3.3. リソースの割り当て設定

Pacemaker は、以下のストラテジーに従って、最初に割り当てられるリソースを決定します。

- 優先度の最も高いリソースが最初に割り当てられます。リソースの優先度の設定は、[表6.3「リソースのメタオプション」](#) を参照してください。
- リソースの優先度が等しい場合は、リソースのシャッフルを防ぐために、実行中のノードで最も高いスコアを持つリソースが最初に割り当てられます。
- リソースが実行しているノードのリソーススコアが等しい場合や、リソースが実行していない場合は、優先ノードで最もスコアが高いリソースが最初に割り当てられます。この場合、優先

ノードのリソーススコアが等しい場合は、CIB に最初に登録されている実行可能なリソースが最初に割り当てられます。

#### 9.6.4. リソース配置ストラテジーガイドライン

リソースに対する Pacemaker の配置ストラテジーの動作効率を最適化するためにも、システムを設定する際には以下を考慮してください。

- 物理容量が十分にあることを確認します。

ノードの物理容量が、通常の状態ではほぼ最大に使用されているとすると、フェイルオーバーの際に問題が発生する可能性があります。使用率機能がなくても、タイムアウトや二次障害が発生する可能性があります。

- ノードに設定する容量にバッファをいくつか構築します。

物理的に存在するよりもわずかに多くのノードリソースが通知されます。ここでは、Pacemaker リソースが、設定した CPU、メモリーなどを常に 100% 使用しないことが想定されます。このような状況は、オーバーコミットと呼ばれることもあります。

- リソースの優先度を指定します。

クラスターがサービスを犠牲にする場合、犠牲にするサービスが一番重要でないことが理想的です。最も重要なリソースが最初にスケジュールされるように、リソースの優先度が適切に設定されていることを確認してください。リソースの優先度の設定は、[表6.3「リソースのメタオプション」](#)を参照してください。

#### 9.6.5. NodeUtilization リソースエージェント (Red Hat Enterprise Linux 7.4 以降)

Red Hat Enterprise Linux 7.4 は **NodeUtilization** リソースエージェントに対応しています。NodeUtilization エージェントは、使用可能な CPU、ホストメモリーの可用性、およびハイパーバイザーのメモリーの可用性に関するシステムパラメーターを検出し、このようなパラメーターを CIB に追加します。エージェントをクローンリソースとして実行して、各ノードにこのようなパラメーターを自動的に入力することができます。

**NodeUtilization** リソースエージェントと、このエージェントのリソースオプションの説明は、**pcs resource describe NodeUtilization** コマンドを実行してください。

### 9.7. PACEMAKER で管理されていないリソースの依存関係の起動順の設定 (RED HAT ENTERPRISE LINUX 7.4 以降)

クラスターは、クラスターが管理していない依存関係を持つリソースを含めることができます。この場合は、Pacemaker を起動する前にその依存関係を起動し、Pacemaker が停止した後に停止する必要があります。

Red Hat Enterprise Linux 7.4 では、この状況を明らかにするには、**systemd resource-agents-deps** ターゲットで、スタートアップ順を設定します。このターゲットに対して **systemd** ドロップインユニットを作成すると、Pacemaker はこのターゲットに対して相対的な順序を適切に設定できます。

たとえば、クラスターに、そのクラスターが管理しない外部サービス **foo** に依存するリソースが含まれる場合は、以下を含むドロップインユニット **/etc/systemd/system/resource-agents-deps.target.d/foo.conf** を作成できます。

```
[Unit]
Requires=foo.service
After=foo.service
```

ドロップインユニットを作成した後に、**systemctl daemon-reload** コマンドを実行します。

この方法で指定するクラスターの依存関係はサービス以外のものとなります。例えば、**/srv** にファイルをマウントする依存関係も考えられます。この場合、**systemd** ドキュメンテーションに従って、**systemd** ファイル **srv.mount** を作成して、ここで説明しているように **.conf** ファイルの **srv.mount foo.service** の代わりにドロップインユニットを作成します。

## 9.8. SNMP での PACEMAKER クラスターを照会 (RED HAT ENTERPRISE LINUX 7.5 以降)

Red Hat Enterprise Linux 7.5 では、**pcs\_snmp\_agent** デーモンを使用して、SNMP でデータについて Pacemaker クラスターを照会できます。**pcs\_snmp\_agent** デーモンは、**agentx** プロトコルでマスターエージェント (**snmpd**) に接続する SNMP エージェントです。**pcs\_snmp\_agent** エージェントは、マスターエージェントにデータを提供するのみなので、スタンドアローンとして動作しません。

以下の手順では、システムが Pacemaker クラスターで SNMP を使うための基本的な設定を行います。この手順は、SNMP を使用してクラスターのデータを取得する、クラスターの各ノードで行います。

1. クラスターの各ノードに **pcs-snmp** パッケージをインストールします。これにより、**snmp** を使うことのできる **net-snmp** パッケージもインストールされています。

```
# yum install pcs-snmp
```

2. **/etc/snmp/snmpd.conf** 設定ファイルに以下の行を追加して、**master agentx** として **snmpd** を設定します。

```
master agentx
```

3. **/etc/snmp/snmpd.conf** 設定ファイルに以下の行を追加して、同じ SNMP 設定で **pcs\_snmp\_agent** を有効にします。

```
view systemview included .1.3.6.1.4.1.32723.100
```

4. **pcs\_snmp\_agent** サービスを開始します。

```
# systemctl start pcs_snmp_agent.service
# systemctl enable pcs_snmp_agent.service
```

5. 設定を確認するには、**pcs status** を使用してクラスターステータスを表示してから、SNMP からのデータの取得を試行し、出力と一致しているかどうかを見てください。SNMP を使用してデータを取得する際には、プリミティブなリソースのみが与えられることに注意してください。

以下の例では、アクションに 1 回失敗した状態で実行しているクラスター上での **pcs status** コマンドの出力を示しています。

```
# pcs status
Cluster name: rhel75-cluster
```

Stack: corosync  
 Current DC: rhel75-node2 (version 1.1.18-5.el7-1a4ef7d180) - partition with quorum  
 Last updated: Wed Nov 15 16:07:44 2017  
 Last change: Wed Nov 15 16:06:40 2017 by hacluster via cibadmin on rhel75-node1

2 nodes configured  
 14 resources configured (1 DISABLED)

Online: [ rhel75-node1 rhel75-node2 ]

Full list of resources:

```
fencing    (stonith:fence_xvm): Started rhel75-node1
dummy5 (ocf::pacemaker:Dummy): Stopped (disabled)
dummy6 (ocf::pacemaker:Dummy): Stopped
dummy7 (ocf::pacemaker:Dummy): Started rhel75-node2
dummy8 (ocf::pacemaker:Dummy): Started rhel75-node1
dummy9 (ocf::pacemaker:Dummy): Started rhel75-node2
Resource Group: group1
  dummy1    (ocf::pacemaker:Dummy): Started rhel75-node1
  dummy10   (ocf::pacemaker:Dummy): Started rhel75-node1
Clone Set: group2-clone [group2]
  Started: [ rhel75-node1 rhel75-node2 ]
Clone Set: dummy4-clone [dummy4]
  Started: [ rhel75-node1 rhel75-node2 ]
```

Failed Actions:

```
* dummy6_start_0 on rhel75-node1 'unknown error' (1): call=87, status=complete,
exitreason=",
  last-rc-change='Wed Nov 15 16:05:55 2017', queued=0ms, exec=20ms
```

```
# snmpwalk -v 2c -c public localhost PACEMAKER-PCS-V1-MIB::pcmkPcsV1Cluster
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterName.0 = STRING: "rhel75-cluster"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterQuorate.0 = INTEGER: 1
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterNodesNum.0 = INTEGER: 2
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterNodesNames.0 = STRING: "rhel75-node1"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterNodesNames.1 = STRING: "rhel75-node2"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterCorosyncNodesOnlineNum.0 = INTEGER: 2
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterCorosyncNodesOnlineNames.0 = STRING:
"rhel75-node1"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterCorosyncNodesOnlineNames.1 = STRING:
"rhel75-node2"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterCorosyncNodesOfflineNum.0 = INTEGER: 0
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterPcmkNodesOnlineNum.0 = INTEGER: 2
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterPcmkNodesOnlineNames.0 = STRING:
"rhel75-node1"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterPcmkNodesOnlineNames.1 = STRING:
"rhel75-node2"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterPcmkNodesStandbyNum.0 = INTEGER: 0
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterPcmkNodesOfflineNum.0 = INTEGER: 0
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesNum.0 = INTEGER: 11
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.0 = STRING: "fencing"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.1 = STRING: "dummy5"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.2 = STRING: "dummy6"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.3 = STRING: "dummy7"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.4 = STRING: "dummy8"
```

```

PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.5 = STRING: "dummy9"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.6 = STRING: "dummy1"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.7 = STRING: "dummy10"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.8 = STRING: "dummy2"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.9 = STRING: "dummy3"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterAllResourcesIds.10 = STRING: "dummy4"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesNum.0 = INTEGER: 9
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.0 = STRING:
"fencing"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.1 = STRING:
"dummy7"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.2 = STRING:
"dummy8"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.3 = STRING:
"dummy9"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.4 = STRING:
"dummy1"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.5 = STRING:
"dummy10"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.6 = STRING:
"dummy2"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.7 = STRING:
"dummy3"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterRunningResourcesIds.8 = STRING:
"dummy4"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterStoppedResroucesNum.0 = INTEGER: 1
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterStoppedResroucesIds.0 = STRING:
"dummy5"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterFailedResourcesNum.0 = INTEGER: 1
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterFailedResourcesIds.0 = STRING: "dummy6"
PACEMAKER-PCS-V1-MIB::pcmkPcsV1ClusterFailedResourcesIds.0 = No more variables
left in this MIB View (It is past the end of the MIB tree)

```

## 9.9. クリーンノードのシャットダウンで停止するようにリソースを設定 (RED HAT ENTERPRISE LINUX 7.8 以降)

クラスターノードがシャットダウンしたときの Pacemaker のデフォルトの応答は、シャットダウンが正常なシャットダウンであっても、そのノードで実行中のすべてのリソースを停止し、別の場所でリソースを復元することです。Red Hat Enterprise Linux 7.8 以降では、ノードが正常にシャットダウンすると、ノードに接続されているリソースがノードにロックされ、シャットダウンしたノードがクラスターに再度参加するときに再び起動するまで、他の場所で起動できないように、Pacemaker を設定できます。これにより、ノードのリソースをクラスター内の他のノードにフェールオーバーせずに、サービスの停止が許容できるメンテナンスウィンドウ中にノードの電源を切ることができます。

### 9.9.1. クリーンノードシャットダウンで停止するようにリソースを設定するためのクラスタープロパティ

ノードの正常なシャットダウンでリソースがフェイルオーバーしないようにする機能は、以下のクラスタープロパティで実装されます。

#### shutdown-lock

このクラスタープロパティをデフォルト値の **false** に設定すると、クラスターは、ノードで適切にシャットダウンしているノードでアクティブなリソースを復旧します。このプロパティを **true** に設定すると、適切にシャットダウンしているノードでアクティブなリソースは、クラスターに再参

加した後にノードで再起動するまで別の場所で起動できなくなります。

**shutdown-lock** プロパティはクラスターノードまたはリモートノードのいずれかで機能しますが、ゲストノードは機能しません。

**shutdown-lock** を **true** に設定すると、ノードがダウンした場合にクラスターリソースのロックを削除し、以下のコマンドを実行してノードで手動更新を実行してリソースを別の場所で起動できるようになります。

```
pcs resource refresh resource --node node
```

リソースのロックが解除されると、クラスターはリソースを別の場所に移動できるようになることに注意してください。リソースのスティッキネスの値または場所の設定を使用することにより、これが発生する可能性を抑制できます。

### 注記

手動更新は、最初に次のコマンドを実行するとリモートノードで機能します。

1. リモートノードで **systemctl stop pacemaker\_remote** コマンドを実行してノードを停止します。
2. **pcs resource disable remote-connection-resource** コマンドを実行します。

その後、リモートノードで手動更新を実行できます。

### shutdown-lock-limit

このクラスタープロパティをデフォルト値の 0 以外の値に設定すると、シャットダウンを開始してから指定した時間内にノードが再参加しない場合に、他のノードの復旧にリソースが利用可能になります。ただし、この間隔は、**cluster-recheck-interval** クラスタープロパティの値よりも頻繁に確認されないことに注意してください。

### 注記

**shutdown-lock-limit** プロパティは、以下のコマンドを最初に実行した場合に限りリモートノードで動作します。

1. リモートノードで **systemctl stop pacemaker\_remote** コマンドを実行してノードを停止します。
2. **pcs resource disable remote-connection-resource** コマンドを実行します。

このコマンドの実行後、**shutdown-lock-limit** で指定した時間が経過すると、リモートノード上で実行しているリソースが他のノードの復旧に利用できます。

## 9.9.2. shutdown-lock クラスタープロパティの設定

以下の例では、サンプルのクラスターで **shutdown-lock** クラスタープロパティを **true** に設定し、ノードをシャットダウンして再起動したときの影響を示しています。この例のクラスターは、**z1.example.com**、**z2.example.com**、および **z3.example.com** の 3 つのノードで設定されます。



1. **shutdown-lock** プロパティを **true** に設定し、その値を確認します。この例では、**shutdown-lock-limit** プロパティはデフォルト値 0 を維持します。

```
[root@z3.example.com ~]# pcs property set shutdown-lock=true
[root@z3.example.com ~]# pcs property list --all | grep shutdown-lock
shutdown-lock: true
shutdown-lock-limit: 0
```

2. クラスターのステータスを確認します。この例では、3 番目と 5 番目のリソースが **z1.example.com** で実行しています。

```
[root@z3.example.com ~]# pcs status
...
Full List of Resources:
...
* first (ocf::pacemaker:Dummy): Started z3.example.com
* second (ocf::pacemaker:Dummy): Started z2.example.com
* third (ocf::pacemaker:Dummy): Started z1.example.com
* fourth (ocf::pacemaker:Dummy): Started z2.example.com
* fifth (ocf::pacemaker:Dummy): Started z1.example.com
...
```

3. **z1.example.com** をシャットダウンします。これにより、そのノードで実行中のリソースを停止します。

```
[root@z3.example.com ~] # pcs cluster stop z1.example.com
Stopping Cluster (pacemaker)...
Stopping Cluster (corosync)...
```

**pcs status** コマンドを実行すると、ノードの **z1.example.com** がオフラインであることを示し、**z1.example.com** で実行していたリソースは、ノードの停止時に **LOCKED** になります。

```
[root@z3.example.com ~]# pcs status
...

Node List:
* Online: [ z2.example.com z3.example.com ]
* OFFLINE: [ z1.example.com ]

Full List of Resources:
...
* first (ocf::pacemaker:Dummy): Started z3.example.com
* second (ocf::pacemaker:Dummy): Started z2.example.com
* third (ocf::pacemaker:Dummy): Stopped z1.example.com (LOCKED)
* fourth (ocf::pacemaker:Dummy): Started z3.example.com
* fifth (ocf::pacemaker:Dummy): Stopped z1.example.com (LOCKED)
...
```

4. クラスターサービスを **z1.example.com** で再度起動し、クラスターに再参加できるようにします。ロックされたリソースは、そのノードで開始する必要がありますが、いったん起動すると、必ずしも同じノードに留まるわけではありません。

```
[root@z3.example.com ~]# pcs cluster start z1.example.com
Starting Cluster...
```

この例では、3 番目と 5 番目のリソースが z1.example.com ノードで復元されます。

```
[root@z3.example.com ~]# pcs status
...

Node List:
* Online: [ z1.example.com z2.example.com z3.example.com ]

Full List of Resources:
..
* first (ocf::pacemaker:Dummy): Started z3.example.com
* second (ocf::pacemaker:Dummy): Started z2.example.com
* third (ocf::pacemaker:Dummy): Started z1.example.com
* fourth (ocf::pacemaker:Dummy): Started z3.example.com
* fifth (ocf::pacemaker:Dummy): Started z1.example.com
...
```

## 第10章 クラスタークォーラム

Red Hat High Availability Add-On クラスターは、スプリットブレインの状況を回避するために、**votequorum** サービスをフェンシングと併用します。クラスターの各システムには多くの投票数が割り当てられ、過半数の票を取得しているものだけがクラスターの操作を継続できます。サービスは、すべてのノードに読み込むか、いずれのノードにも読み込まないようにする必要があります。サービスをクラスターノードのサブセットに読み込むと、結果が予想できなくなります。サービスがクラスターノードのサブセットにロードされると、結果が予想不可能になります。**votequorum** サービスの設定および操作の詳細は、**votequorum(5)** の man ページを参照してください。

### 10.1. クォーラムオプションの設定

**pcs cluster setup** コマンドを使用してクラスターを作成する場合は、クォーラム設定の特殊な機能を使用できます。表10.1「クォーラムオプション」では、このようなオプションをまとめています。

表10.1 クォーラムオプション

オプション	説明
<b>--auto_tie_breaker</b>	<p>これを有効にすると、クラスターは、決定論的に最大 50% のノードが同時に失敗しても存続されます。クラスターパーティションや、<b>auto_tie_breaker_node</b> に設定された <b>nodeid</b> (設定されていない場合は最小の <b>nodeid</b>) と通信したままのノードのセットは、クォーラムに達した状態を維持します。その他のノードはクォーラムに達しません。</p> <p><b>auto_tie_breaker</b> オプションを指定すると、均等の分割でクラスターが動作を継続できるようになるため、主に偶数個のノードがあるクラスターで使用されます。複数で不均等の分割などの、より複雑な障害は、「クォーラムデバイス」で説明されているようにクォーラムデバイスを使用することが推奨されます。<b>auto_tie_breaker</b> オプションは、クォーラムデバイスと互換性がありません。</p>
<b>--wait_for_all</b>	<p>有効にすると、最低1回、同時にすべてのノードが現れた後に、初回だけ、クラスターがクォーラムに達します。</p> <p><b>wait_for_all</b> オプションは、主にクォーラムデバイス <b>lms</b> (last man standing) アルゴリズムを使用する 2 ノードクラスター、および偶数のノードで設定されるクラスターに使用されます。</p> <p><b>wait_for_all</b> オプションは、クラスターに 2 つのノードがあり、クォーラムデバイスを使用せず、<b>auto_tie_breaker</b> が無効になっている場合に自動的に有効になります。<b>wait_for_all</b> を明示的に 0 に設定すると、このオプションをオーバーライドできます。</p>
<b>--last_man_standing</b>	<p>有効にすると、クラスターは特定の状況で <b>expected_votes</b> とクォーラムを動的に再計算します。このオプションを有効にする場合は、<b>wait_for_all</b> を有効にする必要があります。<b>last_man_standing</b> オプションには、クォーラムデバイスとの互換性がありません。</p>
<b>--last_man_standing_window</b>	<p>クラスターのノードが失われた後の、<b>expected_votes</b> およびクォーラムを再計算するまでの待ち時間 (ミリ秒単位) です。</p>

これらのオプションの設定および使用に関する詳細は、**votequorum(5)** の man ページを参照してください。

## 10.2. クォーラム管理コマンド (RED HAT ENTERPRISE LINUX 7.3 以降)

クラスターの稼働後、以下のクラスタークォーラムコマンドを実行できます。

次のコマンドは、クォーラムの設定を表示します。

```
pcs quorum [config]
```

次のコマンドは、クォーラムのランタイム状態を表示します。

```
pcs quorum status
```

長時間クラスターからノードを除去したためクォーラムの損失が発生した場合、**pcs quorum expected-votes** コマンドを使用するとライブクラスターの **expected\_votes** パラメーターの値を変更できます。これにより、クォーラムがない場合でも、クラスターは操作を継続できます。



### 警告

ライブクラスターで期待される票数 (vote) を変更する場合は、細心の注意を払って行ってください。期待される票数を手動で変更したために、実行しているクラスターが 50% 未満となる場合は、クラスターの他のノードを個別に起動してクラスターサービスを開始できるため、データの破損や予期せぬ結果が発生することがあります。この値を変更する場合は、**wait\_for\_all** パラメーターが有効になっていることを確認してください。

次のコマンドは、ライブクラスターで期待される票数を、指定の値に設定します。これはライブクラスターにのみ影響し、設定ファイルは変更されません。リロードが行われると、**expected\_votes** の値は、設定ファイルの値にリセットされます。

```
pcs quorum expected-votes votes
```

## 10.3. クォーラムオプションの変更 (RED HAT ENTERPRISE LINUX 7.3 以降)

Red Hat Enterprise Linux 7.3 より、**pcs quorum update** コマンドを使用してクラスターの一般的なクォーラムオプションを変更できるようになりました。稼働中のシステムでは、**quorum.two\_node** および **quorum.expected\_votes** オプションを変更できます。その他すべてのクォーラムオプションについては、このコマンドを実行するには、クラスターを停止する必要があります。クォーラムオプションの詳細は **votequorum(5)** の man ページを参照してください。

**pcs quorum update** コマンドの形式は次のとおりです。

```
pcs quorum update [auto_tie_breaker=[0|1]] [last_man_standing=[0|1]] [last_man_standing_window=[time-in-ms]] [wait_for_all=[0|1]]
```

以下の一連のコマンドは、**wait\_for\_all** クォーラムオプションを変更し、このオプションの更新された状態を表示します。クラスターの稼働中はこのコマンドを実行できないことに注意してください。

```
[root@node1:~]# pcs quorum update wait_for_all=1
Checking corosync is not running on nodes...
Error: node1: corosync is running
Error: node2: corosync is running
```

```
[root@node1:~]# pcs cluster stop --all
node2: Stopping Cluster (pacemaker)...
node1: Stopping Cluster (pacemaker)...
node1: Stopping Cluster (corosync)...
node2: Stopping Cluster (corosync)...
```

```
[root@node1:~]# pcs quorum update wait_for_all=1
Checking corosync is not running on nodes...
node2: corosync is not running
node1: corosync is not running
Sending updated corosync.conf to nodes...
node1: Succeeded
node2: Succeeded
```

```
[root@node1:~]# pcs quorum config
Options:
  wait_for_all: 1
```

## 10.4. クォーラムアンブロック (QUORUM UNBLOCK) コマンド

クォーラムに達していない状態でクラスターにリソース管理を続行させたい場合、以下のコマンドを使用してクォーラムの確立時にクラスターがすべてのノードを待たないようにします。



### 注記

このコマンドは細心の注意を払って使用する必要があります。このコマンドを実行する前に、現在クラスターにないノードの電源を切り、共有リソースにアクセスできない状態であることを確認する必要があります。

```
# pcs cluster quorum unblock
```

## 10.5. クォーラムデバイス

Red Hat Enterprise Linux 7.4 は、個別のクォーラムを設定する機能に完全に対応しています。主要な用途は、クォーラムルールによって許容されるノード障害の数よりも多くのノード障害をクラスターが許容するようにすることです。クォーラムデバイスは、偶数のノードを持つクラスターに推奨されます。2 ノードクラスターでクォーラムデバイスを使用すると、スプリットブレインの状況で存続するノードをより適切に判別できます。

クォーラムデバイスを設定する場合は、以下を考慮する必要があります。

- クォーラムデバイスは、クォーラムデバイスを使用するクラスターと同じ場所にある別の物理ネットワークで実行することが推奨されます。理想としては、クォーラムデバイスホストを、メインクラスターとは別のラックに置くか、少なくとも別の PSU に置くようにします。corosync リングと同じネットワークセグメントには置かないようにしてください。
- 複数のクォーラムデバイスをクラスターで同時に使用することはできません。

- 複数のクォーラムデバイスをクラスターで同時に使用することはできません。ただし、複数のクラスターが1つのクォーラムデバイスを同時に使用することはできます。アルゴリズムやクォーラムオプションはクラスターノード自体に保存されるため、同じクォーラムデバイスを使用する各クラスターが、複数のアルゴリズムやクォーラムオプションを使用できます。たとえば、**ffsplit** ((fifty/fifty split) アルゴリズムを使用するクラスターと、**lms** (last man standing) アルゴリズムを使用する別のクラスターが、1つのクォーラムデバイスを使用できます。
- クォーラムデバイスは、既存のクラスターノードで実行しないでください。

### 10.5.1. クォーラムデバイスパッケージのインストール

クラスターにクォーラムデバイスを設定するには、以下のパッケージをインストールする必要があります。

- 既存クラスターのノードで、**corosync-qdevice** をインストールします。

```
[root@node1:~]# yum install corosync-qdevice
[root@node2:~]# yum install corosync-qdevice
```

- クォーラムデバイスホストに、**pcs** および **corosync-qnetd** をインストールします。

```
[root@qdevice:~]# yum install pcs corosync-qnetd
```

- **pcsd** サービスを起動し、システムの起動時に **pcsd** がクォーラムデバイスホストで有効になるようにします。

```
[root@qdevice:~]# systemctl start pcsd.service
[root@qdevice:~]# systemctl enable pcsd.service
```

### 10.5.2. クォーラムデバイスの設定

本セクションでは、Red Hat High Availability クラスターでクォーラムデバイスを設定する手順例を説明します。以下の手順では、クォーラムデバイスを設定し、そのクォーラムデバイスをクラスターに追加します。この例の場合:

- クォーラムデバイスに使用されるノードは **qdevice** です。
- クォーラムデバイスモデルは **net** で、これは現在対応している唯一のクォーラムデバイスモデルです。**net** モデルは、以下のアルゴリズムに対応します。
  - **ffsplit** (fifty-fifty split) -これにより、アクティブなノードの数が最も多いパーティションに1票が提供されます。
  - **lms** (last-man-standing) -ノードが **qnetd** サーバーを確認できるクラスター内に残っている唯一のノードである場合に、1票が返されます。



### 警告

LMS アルゴリズムにより、ノードが1つしか残っていてもクラスターはクォーラムを維持できますが、`number_of_nodes - 1`と同じであるため、クォーラムデバイスの投票力が大きいことを意味します。クォーラムデバイスとの接続が失われると、`number_of_nodes - 1`の票が失われます。つまり、(クォーラムデバイスを無効にすることで) すべてのノードがアクティブなクラスターのみがクォーラムに達したままになります。他のクラスターは、クォーラムに達しなくなります。

これらのアルゴリズムの実装の詳細は、man ページの **corosync-qdevice**(8) を参照してください。

- クラスターノードは **node1** と **node2** です。

以下の手順は、クォーラムデバイスを設定し、そのクォーラムデバイスをクラスターに追加します。

1. クォーラムデバイスをホストするために使用するノードで以下のコマンドを使用し、クォーラムデバイスを設定します。このコマンドは、クォーラムデバイスモデルである **net** を設定および開始し、システムの起動時にデバイスが開始するように設定します。

```
[root@qdevice:~]# pcs qdevice setup model net --enable --start
Quorum device 'net' initialized
quorum device enabled
Starting quorum device...
quorum device started
```

クォーラムデバイスの設定後、そのステータスを確認できます。**corosync-qnetd** デーモンが実行中であり、この時点でクライアントが接続されていないことが分かります。**--full** コマンドオプションを指定すると詳細が出力されます。

```
[root@qdevice:~]# pcs qdevice status net --full
QNetd address:      *:5403
TLS:                Supported (client certificate required)
Connected clients:  0
Connected clusters: 0
Maximum send/receive size: 32768/32768 bytes
```

2. 以下のコマンドを実行して、**firewalld** で **high-availability** サービスを有効にして、**pcsd** デーモンおよび **net** クォーラムデバイスに必要なファイアウォールのポートを有効にします。

```
[root@qdevice:~]# firewall-cmd --permanent --add-service=high-availability
[root@qdevice:~]# firewall-cmd --add-service=high-availability
```

3. 既存クラスターのいずれかのノードにより、クォーラムデバイスをホストしているノードで **hacluster** ユーザーを認証します。

```
[root@node1:~] # pcs cluster auth qdevice
Username: hacluster
Password:
```

qdevice: Authorized

#### 4. クォーラムデバイスをクラスターに追加します。

クォーラムデバイスを追加する前に、後で比較するために、クォーラムデバイスの現在の設定と状況を確認できます。このコマンドの出力は、クラスターがクォーラムデバイスを使用していないことを示しています。

```
[root@node1:~]# pcs quorum config
Options:
```

```
[root@node1:~]# pcs quorum status
Quorum information
-----
Date:           Wed Jun 29 13:15:36 2016
Quorum provider: corosync_votequorum
Nodes:          2
Node ID:         1
Ring ID:         1/8272
Quorate:         Yes
```

```
Votequorum information
-----
Expected votes: 2
Highest expected: 2
Total votes:    2
Quorum:         1
Flags:          2Node Quorate
```

```
Membership information
-----
Nodeid  Votes  Qdevice Name
  1      1      NR node1 (local)
  2      1      NR node2
```

以下のコマンドは、作成しておいたクォーラムデバイスをクラスターに追加します。複数のクォーラムデバイスをクラスターで同時に使用することはできません。ただし、複数のクラスターが1つのクォーラムデバイスを同時に使用することはできます。以下のコマンド例では、**ffsplit** アルゴリズムを使用するようにクォーラムデバイスを設定します。クォーラムデバイスの設定オプションの詳細は、man ページの **corosync-qdevice(8)** を参照してください。

```
[root@node1:~]# pcs quorum device add model net host=qdevice algorithm=ffsplit
Setting up qdevice certificates on nodes...
node2: Succeeded
node1: Succeeded
Enabling corosync-qdevice...
node1: corosync-qdevice enabled
node2: corosync-qdevice enabled
Sending updated corosync.conf to nodes...
node1: Succeeded
node2: Succeeded
Corosync configuration reloaded
```



```
Starting corosync-qdevice...
node1: corosync-qdevice started
node2: corosync-qdevice started
```

5. クォーラムデバイスの設定状態をチェックします。

クラスター側から以下のコマンドを実行すると、設定の変更内容を確認できます。

**pcs quorum config** は設定されたクォーラムデバイスを表示します。

```
[root@node1:~]# pcs quorum config
Options:
Device:
  Model: net
  algorithm: ffsplit
  host: qdevice
```

**pcs quorum status** コマンドは、使用中のクォーラムデバイスとクォーラムのランタイム状態を表示します。

```
[root@node1:~]# pcs quorum status
Quorum information
-----
Date:          Wed Jun 29 13:17:02 2016
Quorum provider: corosync_votequorum
Nodes:         2
Node ID:        1
Ring ID:        1/8272
Quorate:        Yes

Votequorum information
-----
Expected votes: 3
Highest expected: 3
Total votes:    3
Quorum:         2
Flags:          Quorate Qdevice

Membership information
-----


| Nodeid | Votes | Qdevice Name          |
|--------|-------|-----------------------|
| 1      | 1     | A,V,NMW node1 (local) |
| 2      | 1     | A,V,NMW node2         |
| 0      | 1     | Qdevice               |


```

**pcs quorum device status** はクォーラムデバイスのランタイム状態を表示します。

```
[root@node1:~]# pcs quorum device status
Qdevice information
-----
Model:          Net
Node ID:         1
Configured node list:
  0 Node ID = 1
  1 Node ID = 2
```

Membership node list: 1, 2

Qdevice-net information

```
-----
Cluster name:      mycluster
QNetd host:        qdevice:5403
Algorithm:         ffsplit
Tie-breaker:       Node with lowest node ID
State:             Connected
```

クォーラムデバイスから次のコマンドを実行して、**corosync-qnetd** デーモンのステータスを表示できます。

```
[root@qdevice:~]# pcs qdevice status net --full
QNetd address:      *:5403
TLS:                Supported (client certificate required)
Connected clients:  2
Connected clusters: 1
Maximum send/receive size: 32768/32768 bytes
Cluster "mycluster":
  Algorithm:        ffsplit
  Tie-breaker:      Node with lowest node ID
  Node ID 2:
    Client address:  ::ffff:192.168.122.122:50028
    HB interval:     8000ms
    Configured node list: 1, 2
    Ring ID:         1.2050
    Membership node list: 1, 2
    TLS active:      Yes (client certificate verified)
    Vote:            ACK (ACK)
  Node ID 1:
    Client address:  ::ffff:192.168.122.121:48786
    HB interval:     8000ms
    Configured node list: 1, 2
    Ring ID:         1.2050
    Membership node list: 1, 2
    TLS active:      Yes (client certificate verified)
    Vote:            ACK (ACK)
```

### 10.5.3. クォーラムデバイスサービスの管理

以下のコマンド例のとおり、PCS はローカルホスト (**corosync-qnetd**) でクォーラムデバイスサービスを管理する機能を提供します。このコマンドは、**corosync-qnetd** サービスにのみ影響することに注意してください。

```
[root@qdevice:~]# pcs qdevice start net
[root@qdevice:~]# pcs qdevice stop net
[root@qdevice:~]# pcs qdevice enable net
[root@qdevice:~]# pcs qdevice disable net
[root@qdevice:~]# pcs qdevice kill net
```

### 10.5.4. クラスターでのクォーラムデバイス設定の管理

以下のセクションは、クラスターでクォーラムデバイス設定を管理するために使用する PCS コマンドについて説明し、「[クォーラムデバイスの設定](#)」のクォーラムデバイス設定を基にした例を使用します。

#### 10.5.4.1. クォーラムデバイス設定の変更

クォーラムデバイスの設定を変更するには **pcs quorum device update** コマンドを使用します。



##### 警告

クォーラムデバイスモデル **net** の **host** オプションを変更するには、**pcs quorum device remove** および **pcs quorum device add** コマンドを使用し、設定プロパティを設定します (変更前のホストと変更後のホストが同じマシンである場合を除く)。

以下のコマンドは、クォーラムデバイスアルゴリズムを **lms** に変更します。

```
[root@node1:~]# pcs quorum device update model algorithm=lms
Sending updated corosync.conf to nodes...
node1: Succeeded
node2: Succeeded
Corosync configuration reloaded
Reloading qdevice configuration on nodes...
node1: corosync-qdevice stopped
node2: corosync-qdevice stopped
node1: corosync-qdevice started
node2: corosync-qdevice started
```

#### 10.5.4.2. クォーラムデバイスの削除

以下のコマンドを使用して、クラスターノードに設定されたクォーラムデバイスを削除します。

```
[root@node1:~]# pcs quorum device remove
Sending updated corosync.conf to nodes...
node1: Succeeded
node2: Succeeded
Corosync configuration reloaded
Disabling corosync-qdevice...
node1: corosync-qdevice disabled
node2: corosync-qdevice disabled
Stopping corosync-qdevice...
node1: corosync-qdevice stopped
node2: corosync-qdevice stopped
Removing qdevice certificates from nodes...
node1: Succeeded
node2: Succeeded
```

クォーラムデバイスを削除すると、クォーラムデバイスの状態を表示するときに、次のエラーメッセージが表示されます。

-

```
[root@node1:~]# pcs quorum device status
Error: Unable to get quorum status: corosync-qdevice-tool: Can't connect to QDevice socket (is
QDevice running?): No such file or directory
```

#### 10.5.4.3. クォーラムデバイスの破棄

クォーラムデバイスホストのクォーラムデバイスを無効にするか停止して、設定ファイルをすべて削除するには、次のコマンドを実行します。

```
[root@qdevice:~]# pcs qdevice destroy net
Stopping quorum device...
quorum device stopped
quorum device disabled
Quorum device 'net' configuration files removed
```

## 第11章 PACEMAKER ルール

ルールは、設定をより動的にするのに使用できます。ルールには、(ノード属性を使用して) 時間ベースで異なる処理グループにマシンを割り当て、場所の制約の作成時にその属性を使用する方法があります。

各ルールには、日付などの様々な式だけでなく、その他のルールも含めることができます。各種の式の結果が **boolean-op** フィールドに応じて処理され、最終的にそのルールが **true** または **false** どちらの評価になるかが確定されます。次の動作は、ルールが使用される状況に応じて異なります。

表11.1 ルールのプロパティ

フィールド	説明
<b>role</b>	リソースが指定のロールにある場合にのみ適用するルールを制限します。使用できる値は <b>Started</b> 、 <b>Slave</b> 、および <b>Master</b> です。注意: <b>role="Master"</b> が指定されたルールは、クローンインスタンスの最初の場所を判断できません。どのアクティブインスタンスが昇格されるかにのみ影響します。
<b>score</b>	ルールが <b>true</b> に評価される場合に適用されるスコア。場所の制約として、ルールでの使用に制限されます。
<b>score-attribute</b>	ルールが <b>true</b> に評価されると検索し、スコアとして使用するノード属性。場所の制約として、ルールでの使用に制限されます。
<b>boolean-op</b>	複数の式オブジェクトからの結果を組み合わせる方法。使用できる値は <b>and</b> および <b>or</b> です。デフォルト値は <b>and</b> です。

### 11.1. ノード属性の式

ノードで定義される属性に応じてリソースを制御する場合に使用されるノード属性の式です。

表11.2 式のプロパティ

フィールド	説明
<b>attribute</b>	テストするノード属性。
<b>type</b>	値をテストする方法を指定します。使用できる値は、 <b>string</b> 、 <b>integer</b> 、 <b>version</b> です。デフォルト値は <b>string</b> です。

フィールド	説明
<b>operation</b>	<p>実行する比較動作。設定できる値は、</p> <ul style="list-style-type: none"> <li>* <b>lt</b> - ノード属性の値が <b>value</b> 未満の場合は True</li> <li>* <b>gt</b> - ノード属性の値が <b>value</b> を超える場合は True</li> <li>* <b>lte</b> - ノード属性の値が <b>value</b> 未満または同等になる場合は True</li> <li>* <b>gte</b> - ノード属性の値が <b>value</b> を超えるか、または同等になる場合は True</li> <li>* <b>eq</b> - ノード属性の値が <b>value</b> と同等になる場合に True</li> <li>* <b>ne</b> - ノード属性の値が <b>value</b> と同等にならない場合は True</li> <li>* <b>defined</b> - ノードに指定属性がある場合は True</li> <li>* <b>not_defined</b> - ノードに指定属性がない場合は True</li> </ul>
<b>value</b>	比較のためにユーザーが提示する値 (必須)

管理者が追加する属性のほかに、[表11.3「組み込みノード属性」](#)で説明されているように、クラスターは、使用可能な各ノードに特殊な組み込みノード属性を定義します。

表11.3 組み込みノード属性

名前	説明
<b>#uname</b>	ノード名
<b>#id</b>	ノード ID
<b>#kind</b>	ノードタイプ。使用できる値は、 <b>cluster</b> 、 <b>remote</b> 、および <b>container</b> です。 <b>kind</b> の値は、 <b>ocf:pacemaker:remote</b> リソースで作成された Pacemaker リモートノードの <b>remote</b> 、および Pacemaker リモートゲストノードおよびバンドルノードの <b>container</b> です。
<b>#is_dc</b>	このノードが指定コントローラー (DC) の場合は <b>true</b> 、それ以外の場合は <b>false</b>
<b>#cluster_name</b>	<b>cluster-name</b> クラスタープロパティの値 (設定されている場合)。
<b>#site_name</b>	<b>site-name</b> ノード属性の値 (設定されている場合)。それ以外は <b>#cluster-name</b> と同じ。
<b>#role</b>	関連する多状態リソースがこのノードで果たすロール。多状態リソースの場所の制約のルール内でのみ有効です。

## 11.2. 時刻と日付ベースの式

日付の式は、現在の日付と時刻に応じてリソースまたはクラスターオプションを制御する場合に使用します。オプションで日付の詳細を含めることができます。

表11.4 日付の式のプロパティ

フィールド	説明
start	ISO 8601 仕様に準じた日付と時刻。
end	ISO 8601 仕様に準じた日付と時刻。
operation	状況に応じて、現在の日付と時刻を start と end のいずれかの日付、または両方の日付と比較します。設定できる値は、  * <b>gt</b> - 現在の日付と時刻が <b>start</b> 以降の場合は True  * <b>lt</b> - 現在の日付と時刻が <b>end</b> 以前の場合は True  * <b>in-range</b> - 現在の日付と時刻が <b>start</b> 以降で且つ <b>end</b> 以前の場合は True  * <b>date-spec</b> - 現在の日付/時刻に対して cron のような比較を実行します。

11.3. 日付の詳細

日付の詳細は、時間に関係する cron のような式を作成するのに使用されます。各フィールドには1つの数字または範囲が含まれます。指定のないフィールドは、デフォルトを 0 に設定するのではなく、無視されます。

たとえば、**monthdays="1"** は各月の最初の日と一致し、**hours="09-17"** は午前 9 時から午後 5 時まで (両時間を含む) の時間と一致します。ただし、**weekdays="1,2"** または **weekdays="1-2,5-6"** には複数の範囲が含まれるため、指定することはできません。

表11.5 日付詳細のプロパティ

フィールド	説明
id	日付の一意の名前
hours	使用できる値 - 0~23
monthdays	使用できる値 - 0~31 (月と年に応じて異なる)
weekdays	使用できる値 - 1~7 (1=月曜日、7=日曜日)
yeardays	使用できる値 - 1~366 (年に応じて異なる)
months	使用できる値 - 1~12
weeks	使用できる値 - 1~53 ( <b>weekyear</b> に応じて異なる)
years	グレゴリオ暦 (新暦) に準じる年
weekyears	グレゴリオ暦の年とは異なる場合がある (例: <b>2005-001 Ordinal</b> は <b>2005-01-01 Gregorian</b> であり <b>2004-W53-6 Weekly</b> でもある)
moon	使用できる値 - 0~7 (0 は新月、4 は満月)

## 11.4. 期間

`operation=in_range` で **end** の値が与えられていない場合は期間を使ってその値を算出します。**date\_spec** オブジェクトと同じフィールドがありますが制限はありません (つまり 19 ヶ月の期間を持たせることが可能)。**date\_specs** 同様、未入力のフィールドは無視されます。

## 11.5. PCS でのルールの設定

**pcs** を使用してルールを設定するには、[「ルールを使用したリソースの場所の確定」](#) の説明に従って、ルールを使用する場所の制約を設定します。

ルールを削除する場合は次のコマンドを使用します。削除しているルールがその制約内で最後のルールになる場合は、その制約も削除されます。

```
pcs constraint rule remove rule_id
```



## 第12章 PACEMAKER クラスターのプロパティー

クラスターのプロパティーは、クラスター動作中に発生する可能性がある状況に直面した場合に、クラスターの動作を制御します。

- [表12.1「クラスターのプロパティー」](#) では、クラスターのプロパティーオプションを説明します。
- 「[クラスターのプロパティーの設定と削除](#)」では、クラスタープロパティーの設定方法を説明します。
- 「[クラスタープロパティー設定のクエリー](#)」では、現在設定されているクラスタープロパティーを一覧表示する方法を説明しています。

### 12.1. クラスタープロパティーとオプションの要約

[表12.1「クラスターのプロパティー」](#) には、Pacemaker クラスターのプロパティーのデフォルト値や、設定可能な値などをまとめています。



#### 注記

この表に記載しているプロパティー以外にも、クラスターソフトウェアで公開されるクラスタープロパティーがあります。このようなプロパティーでは、デフォルト値を別の値には変更しないことが推奨されます。

表12.1 クラスターのプロパティー

オプション	デフォルト	説明
<b>batch-limit</b>	0	クラスターを並列に実行できるリソースアクションの数。正しい値は、ネットワークおよびクラスターノードの速度と負荷によって異なります。
<b>migration-limit</b>	-1 (無制限)	クラスターが、ノードで並行に実行することが許可されている移行ジョブの数。
<b>no-quorum-policy</b>	stop	<p>クラスターにクォーラムがない場合のアクション。設定できる値は、</p> <ul style="list-style-type: none"> <li>* ignore - 全リソースの管理を続行する</li> <li>* freeze - リソース管理は継続するが、影響を受けるパーティションに含まれないノードのリソースは復帰させない</li> <li>* stop - 影響を受けるクラスターパーティション内の全リソースを停止する</li> <li>* suicide - 影響を受けるクラスターパーティション内の全ノードをフェンスする</li> </ul>
<b>symmetric-cluster</b>	true	リソースを、デフォルトで任意のノードで実行できるかどうかを示します。

オプション	デフォルト	説明
<b>stonith-enabled</b>	true	<p>障害が発生したノードと、停止できないリソースが含まれるノードをフェンスする必要があることを示します。データを保護するには、<b>true</b> に設定する必要があります。</p> <p><b>true</b> または未設定の場合は、STONITH リソースが設定されていない限り、クラスターによりリソースの起動が拒否されます。</p>
<b>stonith-action</b>	reboot	STONITH デバイスに送るアクション。使用できる値は <b>reboot</b> 、 <b>off</b> です。 <b>poweroff</b> 値も使用できますが、レガシーデバイスでのみ使用されます。
<b>cluster-delay</b>	60s	(アクションの実行を除く) ネットワーク上のラウンドトリップ遅延です。正しい値は、ネットワークおよびクラスターノードの速度と負荷によって異なります。
<b>stop-orphan-resources</b>	true	削除されたリソースを停止すべきかどうかを示します。
<b>stop-orphan-actions</b>	true	削除されたアクションをキャンセルするかどうかを示します。
<b>start-failure-is-fatal</b>	true	<p>特定のノードでリソースの起動に失敗した場合に、そのノードで開始試行を行わないようにするかを示します。<b>false</b> に設定すると、リソースの現在の失敗数と移行しきい値を基にしてクラスターが同じノードの開始を再試行するかどうかを決定。リソースの <b>migration-threshold</b> オプションの設定は「<a href="#">障害発生によるリソースの移動</a>」を参照してください。</p> <p><b>start-failure-is-fatal</b> を <b>false</b> に設定すると、リソースを起動できない障害があるノードが、すべての依存アクションを遅らせる可能性があるというリスクが発生します。これにより、<b>start-failure-is-fatal</b> のデフォルトは <b>true</b> となっています。<b>start-failure-is-fatal=false</b> を設定するリスクは、移行しきい値を低く設定することで軽減できます。これにより、何度も失敗してもその他のアクションを続行できます。</p>
<b>pe-error-series-max</b>	-1 (すべて)	ERROR となる PE 入力を保存する数。問題を報告する場合に使用されます。
<b>pe-warn-series-max</b>	-1 (すべて)	WARNING となる PE 入力を保存する数。問題を報告する場合に使用されます。
<b>pe-input-series-max</b>	-1 (すべて)	normal となる PE 入力を保存する数。問題を報告する場合に使用されます。
<b>cluster-infrastructure</b>		Pacemaker が現在実行しているメッセージングスタック。情報提供および診断目的に使用されます。ユーザーは設定できません。

オプション	デフォルト	説明
<b>dc-version</b>		クラスターの DC (Designated Controller) で Pacemaker のバージョン。診断目的に使用され、ユーザーは設定できません。
<b>last-lrm-refresh</b>		epoca 以降の秒単位で指定されたローカルリソースマネージャーの最終更新です。診断目的に使用され、ユーザーは設定できません。
<b>cluster-recheck-interval</b>	15 分	時間ベースでオプション、リソースパラメーター、および制約を変更するポーリング間隔。使用できる値は、ポーリングを無効にする 0 (ゼロ) と、秒単位で間隔を示す正の値です (5min など、他の SI 単位が指定されていない場合に限り)。この値は確認が行われる間隔に対する最大時間であることに注意してください。クラスターイベントが、この値で指定された時間よりも早く発生した場合は、確認が行われるタイミングが早くなります。
<b>maintenance-mode</b>	false	メンテナンスモードでは、クラスターが干渉されないモードになり、指示されない限り、サービスを起動したり、停止したりしません。メンテナンスモードが完了すると、クラスターは、サービスの現在の状態のサニティーチェックを実行してから、これを必要とするサービスを停止するか、または開始します。
<b>shutdown-escalation</b>	20min	正常にシャットダウンして終了を試みるのをやめる時間。高度な使用のみ。
<b>stonith-timeout</b>	60s	STONITH アクションが完了するのを待つ時間。
<b>stop-all-resources</b>	false	クラスターがすべてのリソースを停止します。
<b>enable-acl</b>	false	(Red Hat Enterprise Linux 7.1 以降) <b>pcs acl</b> コマンドで設定したように、クラスターがアクセス制御リストを使用できるかどうかを示します。
<b>placement-strategy</b>	<b>default</b>	クラスターノードでリソースの配置を決定する際に、クラスターが使用率属性を考慮にいれるかどうかと、どのように考慮するかを示します。使用率属性および配置ストラテジーの詳細は、「 <a href="#">使用と配置ストラテジー</a> 」を参照してください。
<b>fence-reaction</b>	<b>stop</b>	(Red Hat Enterprise Linux 7.8 以降) 独自のフェンシングの通知を受信した場合は、クラスターノードがどのように反応するかを決定します。クラスターノードは、フェンシングの設定が間違っている場合に独自のフェンシングの通知を受信するか、またはファブリックフェンシングがクラスター通信を遮断しない状態である可能性があります。許可される値は、Pacemaker をすぐに停止し、停止したままにする <b>stop</b> と、ローカルノードを直ちに再起動して失敗した場合に停止する <b>panic</b> です。

## 12.2. クラスターのプロパティの設定と削除

クラスタープロパティの値を設定する場合は次の **pcs** コマンドを使用します。

```
pcs property set property=value
```

たとえば、**symmetric-cluster** の値を **false** に設定する場合は、次のコマンドを使用します。

```
# pcs property set symmetric-cluster=false
```

設定からクラスタープロパティを削除する場合は、次のコマンドを使用します。

```
pcs property unset property
```

代わりに **pcs property set** コマンドの値フィールドを空白にしてもクラスタープロパティを削除することができます。これにより、そのプロパティの値がデフォルト値に戻されます。たとえば、**symmetric-cluster** プロパティを **false** に設定したことがある場合は、設定した値が次のコマンドにより削除され、**symmetric-cluster** の値がデフォルト値の **true** に戻されます。

```
# pcs property set symmetric-cluster=
```

## 12.3. クラスタープロパティ設定のクエリー

ほとんどの場合、各種のクラスターコンポーネントの値を表示するため **pcs** コマンドを使用する際、**pcs list** または **pcs show** を交互に使用することができます。次の例では **pcs list** は複数プロパティのすべての設定の全一覧表示に使用する形式になります。一方、**pcs show** は特定のプロパティの値を表示する場合に使用する形式になります。

クラスターに設定されたプロパティ設定の値を表示する場合は次の **pcs** コマンドを使用します。

```
pcs property list
```

明示的に設定されていないプロパティ設定のデフォルト値など、クラスターのプロパティ設定の値をすべて表示する場合は、次のコマンドを使用します。

```
pcs property list --all
```

特定のクラスタープロパティの現在の値を表示する場合は、次のコマンドを使用します。

```
pcs property show property
```

たとえば、**cluster-infrastructure** プロパティの現在の値を表示する場合は、次のコマンドを実行します。

```
# pcs property show cluster-infrastructure
Cluster Properties:
cluster-infrastructure: cman
```

情報提供の目的で、次のコマンドを使用して、プロパティがデフォルト以外の値に設定されているかどうかに関わらず、プロパティのデフォルト値の一覧を表示できます。

```
pcs property [list|show] --defaults
```

■

## 第13章 クラスタイベントのスキプトのトリガー

Pacemaker クラスタはイベント駆動型のシステムで、イベントはリソースやノードの障害、設定の変更、またはリソースの開始や停止になります。Pacemaker クラスタアラートを設定すると、クラスタイベントの発生時に外部で一部の処理を行うことができます。クラスタアラートを設定するには、以下の2つの方法の1つを使用します。

- Red Hat Enterprise Linux 7.3 より、アラートエージェントを使用して Pacemaker アラートを設定できるようになりました。アラートエージェントは、リソース設定と操作を処理するためにクラスタ呼び出しのリソースエージェントと同様にクラスタが呼び出す外部プログラムです。Pacemaker アラートエージェントの説明は「[Pacemaker アラートエージェント \(Red Hat Enterprise Linux 7.3 以降\)](#)」を参照してください。
- **ocf:pacemaker:ClusterMon** リソースはクラスタの状態を監視でき、クラスタイベントごとにアラートをトリガーできます。このリソースは、標準の間隔で **crm\_mon** コマンドをバックグラウンドで実行します。**ClusterMon** リソースの詳細は「[モニターリングのリソースを使ったイベント通知](#)」を参照してください。

### 13.1. PACEMAKER アラートエージェント (RED HAT ENTERPRISE LINUX 7.3 以降)

クラスタイベントの発生時に Pacemaker アラートエージェントを作成して外部で一部の処理を行うことができます。クラスタは、環境変数を用いてイベントの情報をエージェントに渡します。エージェントは、Eメールメッセージの送信、ログのファイルへの記録、監視システムの更新など、この情報を自由に使用できます。

- Pacemaker は、デフォルトで **/usr/share/pacemaker/alerts** にインストールされるアラートエージェントのサンプルを複数提供します。これらのサンプルスクリプトは、コピーしてそのまま使用したり、目的に合わせて編集するテンプレートとして使用することもできます。対応する全属性は、サンプルエージェントのソースコードを参照してください。サンプルアラートエージェントを使用するアラートの基本的な設定手順の例は、「[サンプルアラートエージェントの使用](#)」を参照してください。
- アラートエージェントの設定および管理に関する一般的な情報は、「[アラートの作成](#)」、「[アラートの表示、編集、および削除](#)」、「[アラートの受信側](#)」、「[アラートメタオプション](#)」、および「[アラート設定コマンドの例](#)」を参照してください。
- Pacemaker アラートの独自のアラートエージェントを作成することができます。アラートエージェントの作成に関する詳細は、「[アラートエージェントの作成](#)」を参照してください。

#### 13.1.1. サンプルアラートエージェントの使用

サンプルアラートエージェントの1つを使用するとき、スクリプトがニーズにあっていないことを確認してください。サンプルエージェントは、特定のクラスタ環境用のカスタムスクリプトを作成するためのテンプレートとして提供されます。Red Hat は、Pacemaker との通信にアラートエージェントスクリプトが使用するインターフェイスをサポートしますが、カスタムエージェント自体にはサポートを提供していないことに注意してください。

サンプルアラートエージェントの1つを使用するには、クラスタの各ノードにエージェントをインストールする必要があります。たとえば、次のコマンドは、**alert\_file.sh.sample** スクリプトを **alert\_file.sh** としてインストールします。

```
# install --mode=0755 /usr/share/pacemaker/alerts/alert_file.sh.sample
/var/lib/pacemaker/alert_file.sh
```

スクリプトをインストールしたら、スクリプトを使用するアラートを作成できます。

以下の例では、インストールした **alert\_file.sh** アラートエージェントを使用してイベントのログをファイルに記録するアラートを設定します。アラートエージェントは、最低限のパーミッションを持つ **hacluster** ユーザーとして実行します。

この例では、イベントの記録に使用するログファイル **pcmk\_alert\_file.log** を作成します。また、アラートエージェントを作成し、その受信先としてログファイルへのパスを追加します。

```
# touch /var/log/pcmk_alert_file.log
# chown hacluster:haclient /var/log/pcmk_alert_file.log
# chmod 600 /var/log/pcmk_alert_file.log
# pcs alert create id=alert_file description="Log events to a file." path=/var/lib/pacemaker/alert_file.sh
# pcs alert recipient add alert_file id=my-alert_logfile value=/var/log/pcmk_alert_file.log
```

以下の例では、**alert\_snmp.sh.sample** スクリプトを **alert\_snmp.sh** としてインストールし、インストールした **alert\_snmp.sh** アラートエージェントを使用してクラスタイベントを NSMP トラップとして送信するアラートを設定します。デフォルトでは、正常な監視呼び出し以外のすべてのイベントを SNMP サーバーに送信します。この例では、タイムスタンプの形式をメタオプションとして設定します。メタオプションの詳細は、「[アラートメタオプション](#)」を参照してください。この例では、アラートの設定後にアラートの受信側が設定され、アラート設定が表示されます。

```
# install --mode=0755 /usr/share/pacemaker/alerts/alert_snmp.sh.sample
/var/lib/pacemaker/alert_snmp.sh
# pcs alert create id=snmp_alert path=/var/lib/pacemaker/alert_snmp.sh meta timestamp-
format="%Y-%m-%d,%H:%M:%S.%01N"
# pcs alert recipient add snmp_alert value=192.168.1.2
# pcs alert
Alerts:
Alert: snmp_alert (path=/var/lib/pacemaker/alert_snmp.sh)
Meta options: timestamp-format=%Y-%m-%d,%H:%M:%S.%01N.
Recipients:
Recipient: snmp_alert-recipient (value=192.168.1.2)
```

以下の例は、**alert\_smtp.sh** エージェントをインストールし、インストールしたアラートエージェントを使用するアラートを設定して、クラスタイベントを E メールメッセージとして送信します。この例では、アラートの設定後に受信側が設定され、アラート設定が表示されます。

```
# install --mode=0755 /usr/share/pacemaker/alerts/alert_smtp.sh.sample
/var/lib/pacemaker/alert_smtp.sh
# pcs alert create id=smtp_alert path=/var/lib/pacemaker/alert_smtp.sh options
email_sender=donotreply@example.com
# pcs alert recipient add smtp_alert value=admin@example.com
# pcs alert
Alerts:
Alert: smtp_alert (path=/var/lib/pacemaker/alert_smtp.sh)
Options: email_sender=donotreply@example.com
Recipients:
Recipient: smtp_alert-recipient (value=admin@example.com)
```

**pcs alert create** および **pcs alert recipient add** コマンドの形式に関する情報は「[アラートの作成](#)」および「[アラートの受信側](#)」を参照してください。

### 13.1.2. アラートの作成



次のコマンドは、クラスターアラートを作成します。設定するオプションは、追加の環境変数として指定するパスで、アラートエージェントスクリプトに渡されるエージェント固有の設定値です。**id** の値を指定しないと、値が生成されます。アラートメタオプションの詳細は、「[アラートメタオプション](#)」を参照してください。

```
pcs alert create path=path [id=alert-id] [description=description] [options [option=value]...] [meta [meta-option=value]...]
```

複数のアラートエージェントを設定できます。クラスターは、各イベントに対して、すべてのアラートエージェントを呼び出します。アラートエージェントはクラスターノードでのみ呼び出されます。アラートエージェントは、Pacemaker リモートノードが関係するイベントに対して呼び出されますが、このようなノードでは呼び出されません。

以下の例は、各イベントで **myscript.sh** を呼び出す簡単なアラートを作成します。

```
# pcs alert create id=my_alert path=/path/to/myscript.sh
```

サンプルアラートエージェントの1つを使用するクラスターアラートの作成方法の例は、「[サンプルアラートエージェントの使用](#)」を参照してください。

### 13.1.3. アラートの表示、編集、および削除

次のコマンドは、設定されたすべてのアラートと、設定されたオプションの値を表示します。

```
pcs alert [config|show]
```

以下のコマンドは、指定した *alert-id* 値を持つ既存のアラートを更新します。

```
pcs alert update alert-id [path=path] [description=description] [options [option=value]...] [meta [meta-option=value]...]
```

以下のコマンドは、指定の *alert-id* 値を持つアラートを削除します。

```
pcs alert remove alert-id
```

代わりに、**pcs alert delete** コマンドを実行できます。これは、**pcs alert remove** コマンドと同じです。**pcs alert delete** コマンドおよび **pcs alert remove** コマンドの両方を使用すると、複数のアラートを削除できるようになります。

### 13.1.4. アラートの受信側

通常、アラートは受信側に送信されます。したがって、各アラートには、1人以上の受信者を追加で設定できます。クラスターは、受信側ごとに別々にエージェントを呼び出します。

受信側は、IP アドレス、メールアドレス、ファイル名、特定のエージェントがサポートするものなど、アラートエージェントが認識できるものを設定します。

次のコマンドは、新しい受信側を指定のアラートに追加します。

```
pcs alert recipient add alert-id value=recipient-value [id=recipient-id] [description=description] [options [option=value]...] [meta [meta-option=value]...]
```

次のコマンドは、既存のアラート受信側を更新します。



```
pcs alert recipient update recipient-id [value=recipient-value] [description=description] [options
[option=value]...] [meta [meta-option=value]...]
```

次のコマンドは、指定のアラート受信側を削除します。

```
pcs alert recipient remove recipient-id
```

代わりに、**pcs alert recipient delete** コマンドを実行できます。これは、**pcs alert recipient remove** コマンドと同じです。**pcs alert recipient remove** コマンドおよび **pcs alert recipient delete** コマンドの両方を使用すると、複数のアラート受信者を削除できます。

次のコマンド例は、受信者 ID が **my-recipient-id** のアラート受信側 **my-alert-recipient** を、アラート **my-alert** に追加します。これにより、クラスターが各イベントの **my-alert** 用に設定したアラートスクリプトを呼び出すように設定され、受信者 **some-address** が環境変数として渡されます。

```
# pcs alert recipient add my-alert value=my-alert-recipient id=my-recipient-id options value=some-address
```

### 13.1.5. アラートメタオプション

リソースエージェントと同様に、メタオプションをアラートエージェントに対して設定すると、Pacemaker の呼び出し方法を調整できます。表13.1「アラートメタオプション」は、アラートメタオプションを示しています。メタオプションは、アラートエージェントごと、または受信側ごとに設定できます。

表13.1 アラートメタオプション

メタ属性	デフォルト	説明
<b>timestamp-format</b>	%H:%M:%S.%06N	イベントのタイムスタンプをエージェントに送信するときにクラスターが使用する形式です。この文字列は <b>date(1)</b> コマンドで使用されます。
<b>timeout</b>	30s	アラートエージェントがこの時間内に完了しないと終了させられます。

以下の例は、**myscript.sh** スクリプトを呼び出すアラートを設定し、2つの受信側をアラートに追加します。最初の受信側の ID は **my-alert-recipient1** で、2つ目の受信側の ID は **my-alert-recipient2** です。スクリプトは各イベントで2回呼び出され、呼び出しのタイムアウト値はそれぞれ15秒です。呼び出しの1つは受信側 **someuser@example.com** に渡され、タイムスタンプの形式は **%D %H:%M** になります。もう1つの呼び出しは受信側 **otheruser@example.com** へ渡され、タイムスタンプの形式は **%c** になります。

```
# pcs alert create id=my-alert path=/path/to/myscript.sh meta timeout=15s
# pcs alert recipient add my-alert value=someuser@example.com id=my-alert-recipient1 meta
timestamp-format="%D %H:%M"
# pcs alert recipient add my-alert value=otheruser@example.com id=my-alert-recipient2 meta
timestamp-format=%c
```

### 13.1.6. アラート設定コマンドの例

以下の例は、基本的なアラート設定コマンドの一部と、アラートの作成、受信側の追加、および設定されたアラートの表示に使用される形式を表しています。アラートエージェント自体はクラスター内の各ノードにインストールする必要がありますが、pcs コマンドの実行は1回だけで済みます。

以下のコマンドは簡単なアラートを作成し、アラートに2つの受信側を追加した後、設定された値を表示します。

- アラート ID の値が指定されていないため、**alert** のアラート ID が作成されます。
- 最初の受信側作成コマンドは、**rec\_value** の受信側を指定します。このコマンドには受信側 ID が指定されていないため、**alert-recipient** の値が受信側 ID として使用されます。
- 2 番目の受信側作成コマンドは、**rec\_value2** の受信側を指定します。このコマンドは、**my-recipient** を受信側 ID として指定します。

```
# pcs alert create path=/my/path
# pcs alert recipient add alert value=rec_value
# pcs alert recipient add alert value=rec_value2 id=my-recipient
# pcs alert config
Alerts:
Alert: alert (path=/my/path)
Recipients:
Recipient: alert-recipient (value=rec_value)
Recipient: my-recipient (value=rec_value2)
```

以下のコマンドは、2 番目のアラートとそのアラートの受信側を追加します。2 番目のアラートのアラート ID は **my-alert** で、受信側の値は **my-other-recipient** です。受信側 ID が指定されていないため、**my-alert-recipient** が受信側 ID として使用されます。

```
# pcs alert create id=my-alert path=/path/to/script description=alert_description options
option1=value1 opt=val meta timeout=50s timestamp-format="%H%B%S"
# pcs alert recipient add my-alert value=my-other-recipient
# pcs alert
Alerts:
Alert: alert (path=/my/path)
Recipients:
Recipient: alert-recipient (value=rec_value)
Recipient: my-recipient (value=rec_value2)
Alert: my-alert (path=/path/to/script)
Description: alert_description
Options: opt=val option1=value1
Meta options: timestamp-format=%H%B%S timeout=50s
Recipients:
Recipient: my-alert-recipient (value=my-other-recipient)
```

以下のコマンドは、アラート **my-alert** と受信側 **my-alert-recipient** のアラート値を変更します。

```
# pcs alert update my-alert options option1=newvalue1 meta timestamp-format="%H%M%S"
# pcs alert recipient update my-alert-recipient options option1=new meta timeout=60s
# pcs alert
Alerts:
Alert: alert (path=/my/path)
Recipients:
Recipient: alert-recipient (value=rec_value)
Recipient: my-recipient (value=rec_value2)
Alert: my-alert (path=/path/to/script)
```

```
Description: alert_description
Options: opt=val option1=newvalue1
Meta options: timestamp-format=%H%M%S timeout=50s
Recipients:
  Recipient: my-alert-recipient (value=my-other-recipient)
  Options: option1=new
  Meta options: timeout=60s
```

次のコマンドは、受信側 **my-alert-recipient** を **alert** から削除します。

```
# pcs alert recipient remove my-recipient
# pcs alert
Alerts:
  Alert: alert (path=/my/path)
  Recipients:
    Recipient: alert-recipient (value=rec_value)
  Alert: my-alert (path=/path/to/script)
  Description: alert_description
  Meta options: timestamp-format="%M%B%S" timeout=50s
  Meta options: m=newval meta-option1=2
  Recipients:
    Recipient: my-alert-recipient (value=my-other-recipient)
    Options: option1=new
    Meta options: timeout=60s
```

次のコマンドは、設定から **myalert** を削除します。

```
# pcs alert remove my-alert
# pcs alert
Alerts:
  Alert: alert (path=/my/path)
  Recipients:
    Recipient: alert-recipient (value=rec_value)
```

13.1.7. アラートエージェントの作成

Pacemaker アラートには、ノードアラート、フェンスアラート、およびリソースアラートの3種類があります。アラートエージェントに渡された環境変数は、アラートのタイプにより異なる可能性があります。表13.2「アラートエージェントに渡される環境変数」は、アラートエージェントに渡される環境変数を示し、環境変数が特定のアラートタイプに関連付けられるタイミングを指定します。

表13.2 アラートエージェントに渡される環境変数

環境変数	説明
CRM_alert_kind	アラートの種類 (ノード、フェンス、またはリソース)
CRM_alert_version	アラートを送信する Pacemaker のバージョン
CRM_alert_recipient	設定された送信側

環境変数	説明
<b>CRM_alert_node_sequence</b>	アラートがローカルノードで発行されるたびに増加するシーケンス番号。これは、Pacemaker によりアラートが発行された順序を参照するのに使用できます。後で発生したイベントのアラートは、先に発生したイベントのアラートよりもシーケンス番号が大きくなります。この番号は、クラスター全体を対象とする番号ではないことに注意してください。
<b>CRM_alert_timestamp</b>	<b>timestamp-format</b> メタオプションで指定された形式で、エージェントの実行前に作成されたタイムスタンプ。これにより、エージェントは、エージェント自体が呼び出されたタイミング (システムの負荷やその他の状況により遅延する可能性があります) に関係なく、信頼できる高精度のイベント発生時間を使用できます。
<b>CRM_alert_node</b>	影響を受けるノードの名前
<b>CRM_alert_desc</b>	イベントの詳細。ノードアラートの場合は、ノードの現在の状態 (番号または lost) になります。フェンスアラートの場合は、フェンス操作の要求元、ターゲット、フェンス操作のエラーコードなどを含む要求されたフェンス操作の概要になります。リソースアラートの場合は、 <b>CRM_alert_status</b> と同等の読み取り可能な文字列になります。
<b>CRM_alert_nodeid</b>	状態が変更したノードの ID (ノードアラートの場合のみ提供)。
<b>CRM_alert_task</b>	要求されたフェンスまたはリソース操作 (フェンスおよびリソースアラートの場合のみ提供)。
<b>CRM_alert_rc</b>	フェンスまたはリソース操作の数値の戻りコード (フェンスおよびリソースアラートの場合のみ提供)。
<b>CRM_alert_rsc</b>	影響を受けるリソースの名前 (リソースアラートのみ)。
<b>CRM_alert_interval</b>	リソース操作の間隔 (リソースアラートのみ)
<b>CRM_alert_target_rc</b>	操作の予期される数値の戻りコード (リソースアラートのみ)。
<b>CRM_alert_status</b>	Pacemaker が、操作の結果を示すために使用する数値コード (リソースアラートのみ)。

アラートエージェントを記述する際は、以下を考慮する必要があります。

- アラートエージェントは受信者なしで呼び出されることがあります (受信者が設定されていない場合)。したがって、エージェントは、このような状況では終了しませんが、この状態に対応できなければなりません。設定を段階的に変更し、後で受信側を追加することもできます。
- 1つのアラートに複数の受信側が設定されると、アラートエージェントは受信側ごとに1回呼び出されます。エージェントが同時に実行できない場合は、受信側を1つのみ設定する必要があります。エージェントは、受信側をリストとして解釈することができます。
- クラスターイベントの発生時、すべてのアラートは別々のプロセスとして同時に発生します。設定されているアラートと受信者の数、およびアラートエージェント内で行われている内容に応じて、負荷が急激に増加する可能性があります。たとえば、リソースを大量に消費するアク

ションを直接実行するのではなく、別のインスタンスのキューに追加することで、これを考慮に入れるようにエージェントを作成できます。

- アラートエージェントは、最低限のパーミッションを持つ **hacluster** ユーザーで実行します。アラートエージェントに追加のパーミッションが必要な場合は、**sudo** を設定してアラートエージェントが適切な特権を持つ別ユーザーとして必要なコマンドを実行できるようにすることが推奨されます。
- **CRM\_alert\_timestamp** (このコンテンツはユーザー設定の **timestamp-format** によって指定)、**CRM\_alert\_recipient**、すべてのアラートオプションなど、ユーザー設定のパラメータを検証およびサニタイズする場合は十分注意してください。これは、設定エラーから保護するために必要です。また、クラスターノードへの **hacluster** レベルのアクセスがなくても CIB を変更できるユーザーが存在する場合は、セキュリティの問題が発生する可能性もあり、コードを挿入できないようにする必要があります。
- **onfail** パラメーターが **fence** に設定されている操作を持つリソースがクラスターに含まれる場合は、障害発生時に複数のフェンス通知 (このパラメーターが設定されているリソースごとに1つの通知と、追加の通知1つ) が送信されます。STONITH デーモンと **crmd** デーモンの両方が通知を送信します。この場合、送信される通知の数に関係なく、Pacemaker は1つのフェンス操作のみを実際に実行します。



#### 注記

アラートインターフェイスは、**ocf:pacemaker:ClusterMon** リソースで使用する外部スキプトインターフェイスと後方互換性を維持するよう設計されています。この互換性を維持するには、先頭に **CRM\_notify\_** および **CRM\_alert\_** が付けいたアラートエージェントに渡される環境変数を使用できます。互換性の問題の1つは、アラートエージェントが **hacluster** ユーザーで実行している最中に、**ClusterMon** リソースが root ユーザーで外部スキプトを実行したことです。**ClusterMon** によってトリガーされるスキプトの設定については、「[モニターリングのリソースを使ったイベント通知](#)」を参照してください。

## 13.2. モニターリングのリソースを使ったイベント通知

**ocf:pacemaker:ClusterMon** リソースはクラスターの状態を監視でき、クラスターイベントごとにアラートをトリガーできます。このリソースは、標準の間隔で **crm\_mon** コマンドをバックグラウンドで実行します。

デフォルトでは、**crm\_mon** コマンドはリソースイベントのみをリッスンします。イベントのフェンシングのリッスンを有効にするには、**ClusterMon** リソース設定するときに、**--watch-fencing** オプションをコマンドに指定します。**crm\_mon** コマンドはメンバーシップの問題を監視しませんが、フェンスが開始され、そのノードに対して監視が開始されたときにメッセージが出力されます。これはメンバーがクラスターに参加したことを意味します。

**ClusterMon** リソース は外部プログラムを実行して、**extra\_options** パラメーターでクラスター通知を行うものを判別できます。[表13.3「外部監視プログラムへ渡される環境変数」](#) は、発生したクラスターイベントのタイプを記述する、プログラムに渡される環境変数を一覧表示します。

表13.3 外部監視プログラムへ渡される環境変数

環境変数	説明
<b>CRM_notify_recipient</b>	リソース定義からの静的な外部受信側。

環境変数	説明
<b>CRM_notify_node</b>	状態が変更したノード。
<b>CRM_notify_rsc</b>	状態を変更したリソースの名前。
<b>CRM_notify_task</b>	状態が変更する原因となった操作。
<b>CRM_notify_desc</b>	状態が変更する原因となった操作 (該当の操作がある場合) のテキスト出力の関連エラーコード。
<b>CRM_notify_rc</b>	操作の戻りコード。
<b>CRM_target_rc</b>	操作の予期される戻りコード。
<b>CRM_notify_status</b>	操作の状態の数値表現。

以下の例は、外部プログラム **crm\_logger.sh** を実行する **ClusterMon** リソースを設定します。このプログラムは指定されたイベント通知をログに記録します。

以下の手順は、リソースが使用する **crm\_logger.sh** プログラムを作成します。

1. クラスターのノードの1つで、イベント通知をログに記録するプログラムを作成します。

```
# cat <<-END >/usr/local/bin/crm_logger.sh
#!/bin/sh
logger -t "ClusterMon-External" "${CRM_notify_node} ${CRM_notify_rsc} \
${CRM_notify_task} ${CRM_notify_desc} ${CRM_notify_rc} \
${CRM_notify_target_rc} ${CRM_notify_status} ${CRM_notify_recipient}";
exit;
END
```

2. プログラムの所有者とパーミッションを設定します。

```
# chmod 700 /usr/local/bin/crm_logger.sh
# chown root.root /usr/local/bin/crm_logger.sh
```

3. **scp** コマンドを使用して **crm\_logger.sh** プログラムをクラスターの他のノードにコピーし、これらのノードの同じ場所にプログラムを格納し、プログラムに同じ所有者とパーミッションを設定します。

以下の例は、**/usr/local/bin/crm\_logger.sh** プログラムを実行する **ClusterMon-External** という名前の **ClusterMon** リソースを設定します。**ClusterMon** リソースは **html** ファイル (この例の場合は **/var/www/html/cluster\_mon.html**) にクラスターの状態を出力します。**pidfile** は **ClusterMon** がすでに実行されているかどうかを検出します (この例では **/var/run/crm\_mon-external.pid** ファイル)。このリソースはクローンとして作成されるため、クラスターの各ノードで実行されます。**watch-fencing** は、フェンスリソースの **start/stop/monitor**、**start/monitor** など、リソースイベントなどのフェンスイベントの監視を有効にするために指定されます。

```
# pcs resource create ClusterMon-External ClusterMon user=root \
update=10 extra_options="-E /usr/local/bin/crm_logger.sh --watch-fencing" \
```

```
htmlfile=/var/www/html/cluster_mon.html \
pidfile=/var/run/crm_mon-external.pid clone
```



## 注記

以下は、このリソースが実行する **crm\_mon** コマンド (手作業による実行も可能) になります。

```
# /usr/sbin/crm_mon -p /var/run/crm_mon-manual.pid -d -i 5 \
-h /var/www/html/crm_mon-manual.html -E "/usr/local/bin/crm_logger.sh" \
--watch-fencing
```

以下の例は、この例によって生成される監視通知の出力形式になります。

```
Aug 7 11:31:32 rh6node1pcmk ClusterMon-External: rh6node2pcmk.examplerh.com ClusterIP
st_notify_fence Operation st_notify_fence requested by rh6node1pcmk.examplerh.com for peer
rh6node2pcmk.examplerh.com: OK (ref=b206b618-e532-42a5-92eb-44d363ac848e) 0 0 0 #177
Aug 7 11:31:32 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterIP start
OK 0 0 0
Aug 7 11:31:32 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterIP
monitor OK 0 0 0
Aug 7 11:33:59 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com fence_xvms
monitor OK 0 0 0
Aug 7 11:33:59 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterIP
monitor OK 0 0 0
Aug 7 11:33:59 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterMon-
External start OK 0 0 0
Aug 7 11:33:59 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com fence_xvms
start OK 0 0 0
Aug 7 11:33:59 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterIP start
OK 0 0 0
Aug 7 11:33:59 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterMon-
External monitor OK 0 0 0
Aug 7 11:34:00 rh6node1pcmk crmd[2887]: notice: te_rsc_command: Initiating action 8: monitor
ClusterMon-External:1_monitor_0 on rh6node2pcmk.examplerh.com
Aug 7 11:34:00 rh6node1pcmk crmd[2887]: notice: te_rsc_command: Initiating action 16: start
ClusterMon-External:1_start_0 on rh6node2pcmk.examplerh.com
Aug 7 11:34:00 rh6node1pcmk ClusterMon-External: rh6node1pcmk.examplerh.com ClusterIP stop
OK 0 0 0
Aug 7 11:34:00 rh6node1pcmk crmd[2887]: notice: te_rsc_command: Initiating action 15: monitor
ClusterMon-External_monitor_10000 on rh6node2pcmk.examplerh.com
Aug 7 11:34:00 rh6node1pcmk ClusterMon-External: rh6node2pcmk.examplerh.com ClusterMon-
External start OK 0 0 0
Aug 7 11:34:00 rh6node1pcmk ClusterMon-External: rh6node2pcmk.examplerh.com ClusterMon-
External monitor OK 0 0 0
Aug 7 11:34:00 rh6node1pcmk ClusterMon-External: rh6node2pcmk.examplerh.com ClusterIP start
OK 0 0 0
Aug 7 11:34:00 rh6node1pcmk ClusterMon-External: rh6node2pcmk.examplerh.com ClusterIP
monitor OK 0 0 0
```

## 第14章 PACEMAKER を用いたマルチサイトクラスターの設定

クラスターが複数のサイトにまたがる場合は、サイト間のネットワーク接続の問題が原因でスプリットブレインが発生する可能性があります。接続が切断されたときに、別のサイトのノードで障害が発生したのか、またはサイト間の接続に失敗した状態で別サイトのノードが機能しているかどうかをノードが判断する方法がありません。さらに、同時に維持するには離れすぎている2つのサイト間で高可用性サービスを提供することが問題になることがあります。

この問題に対応するため、Red Hat Enterprise Linux 7.4 は Booth クラスターチケットマネージャーを使用して複数のサイトにまたがる高可用性クラスターを設定する機能に完全に対応しています。Booth チケットマネージャーは、特定サイトでクラスターノードに接続するネットワークとは異なる物理ネットワークで実行するための分散サービスです。これにより、Booth フォーメーションという別の非厳密なクラスターがサイトの通常クラスターの上に置かれます。この集約通信層は、個別の Booth チケットに対して合意ベースの決定プロセスを促進します。

Booth チケットは Booth フォーメーションのシングルトンで、時間依存の移動可能な承認の単位を表します。実行には特定のチケットを要求するようにリソースを設定できます。これにより、1つまたは複数のチケットが付与されているサイトで、リソースは一度に1つのサイトでのみ実行されるようになります。

Booth フォーメーションは、複数のサイトで実行しているクラスターで設定され、元のクラスターがすべて独立しているオーバーレイクラスターと考えることができます。チケット付与の有無についてクラスターと通信するのは Booth サービスで、Pacemaker のチケット制約に基づいてクラスターでリソースを実行するかどうかを判断するのは Pacemaker です。これは、チケットマネージャーを使用する場合に、各クラスターが独自のリソースと共有リソースを実行できることを示しています。たとえば、リソース A、B、および C は1つのクラスターでのみ実行され、リソース D、E、および F は別のクラスターでのみ実行されとします。リソース G および H は、チケットによって決定されたこの2つのクラスターのいずれかで実行されます。また、別のチケットにより、この2つのクラスターのいずれかで実行できるリソース J を追加することもできます。

以下の手順は、Booth チケットマネージャーを使用するマルチサイト設定を設定する手順の概要を示しています。

ここで使用するコマンド例は以下を前提とします。

- Cluster 1 は、ノード **cluster1-node1** および **cluster1-node2** で設定されます。
- Cluster 1 に割り当てられたフローティング IP アドレスは 192.168.11.100 です。
- Cluster 2 は、**cluster2-node1** および **cluster2-node2** で設定されます。
- Cluster 2 に割り当てられたフローティング IP アドレスは 192.168.22.100 です。
- 仲裁ノードは **arbitrator-node** で、IP アドレスは 192.168.99.100 です。
- この設定が使用する Booth チケットの名前は **apacheticket** です。

ここで使用するコマンド例は、Apache サービスのクラスターリソースが、各クラスターの **apachegroup** リソースグループの一部として設定されていることを前提としています。各クラスターの Pacemaker インスタンスは独立しているため、このリソースのチケット制約を設定するために、各クラスターでリソースとリソースグループが同じである必要はありませんが、これはフェールオーバーの一般的な事例になります。

クラスターで Apache サービスを設定するクラスター設定の完全な手順は、『High Availability Add-On の管理』の例を参照してください。



設定手順の実行中に **pcs booth config** コマンドを実行すると現在のノードまたはクラスターの Booth 設定を表示できます。また、**pcs booth status** コマンドを実行するとローカルノードの現在の Booth 状態を表示できます。

1. **booth-site** Booth チケットマネージャーパッケージを、両方のクラスターの各ノードにインストールします。

```
[root@cluster1-node1 ~]# yum install -y booth-site
[root@cluster1-node2 ~]# yum install -y booth-site
[root@cluster2-node1 ~]# yum install -y booth-site
[root@cluster2-node2 ~]# yum install -y booth-site
```

2. **pcs** パッケージ、**booth-core** パッケージ、および **booth-arbitrator** パッケージを仲裁ノードにインストールします。

```
[root@arbitrator-node ~]# yum install -y pcs booth-core booth-arbitrator
```

3. 9929/tcp ポートと 9929/udp ポートがすべてのクラスターノードと仲裁ノードで解放されていることを確認します。

たとえば、両方のクラスターのすべてのノードと仲裁ノードで次のコマンドを実行すると、そのノードの 9929/tcp ポートおよび 9929/udp ポートにアクセスできます。

```
# firewall-cmd --add-port=9929/udp
# firewall-cmd --add-port=9929/tcp
# firewall-cmd --add-port=9929/udp --permanent
# firewall-cmd --add-port=9929/tcp --permanent
```

この手順自体により、任意のマシンがノード上の 9929 ポートにアクセスできることに注意してください。お使いのサイトで、ノードが必要なノードに対してのみ開いていることを確認する必要があります。

4. 1つのクラスターの1つのノードで Booth 設定を作成します。各クラスターおよび仲裁ノードに指定するアドレスは IP アドレスでなければなりません。各クラスターにはフローティング IP アドレスを指定します。

```
[cluster1-node1 ~] # pcs booth setup sites 192.168.11.100 192.168.22.100 arbitrators
192.168.99.100
```

このコマンドを実行すると、**/etc/booth/booth.conf** および **/etc/booth/booth.key** 設定ファイルがノード上に作成されます。

5. Booth 設定のチケットを作成します。このチケットは、このチケットがクラスターに付与された場合のみリソースの実行を許可するリソース抑制を定義するのに使用します。

このフェイルオーバー設定手順は基本的な手順で、チケットを1つだけ使用します。各チケットが別の1つ以上のリソースに関連付けられる、より複雑な事例では追加のチケットを作成します。

```
[cluster1-node1 ~] # pcs booth ticket add apacheticket
```

6. 現在のクラスターのすべてのノードに対して Booth 設定を同期します。

```
[cluster1-node1 ~] # pcs booth sync
```

7. 仲裁ノードから、Booth 設定を仲裁者へプルします。この作業をこれまで行ったことがない場合は、最初に、設定をプルするノードに **pcs** を認証する必要があります。

```
[arbitrator-node ~] # pcs cluster auth cluster1-node1
[arbitrator-node ~] # pcs booth pull cluster1-node1
```

8. Booth 設定を別のクラスターにプルし、そのクラスターのすべてのノードを同期します。仲裁ノードでこの作業を行ったことがない場合は、最初に、設定をプルするノードに **pcs** を認証する必要があります。

```
[cluster2-node1 ~] # pcs cluster auth cluster1-node1
[cluster2-node1 ~] # pcs booth pull cluster1-node1
[cluster2-node1 ~] # pcs booth sync
```

9. 仲裁ノードで Booth を開始して有効にします。



#### 注記

Booth はクラスターで Pacemaker リソースとして実行するため、クラスターのノードで Booth を手動で開始したり有効にしたりしないでください。

```
[arbitrator-node ~] # pcs booth start
[arbitrator-node ~] # pcs booth enable
```

10. Booth を設定し、両方のクラスターサイトでクラスターリソースとして実行されるようにします。**booth-ip** および **booth-service** をグループのメンバーとするリソースグループが作成されます。

```
[cluster1-node1 ~] # pcs booth create ip 192.168.11.100
[cluster2-node1 ~] # pcs booth create ip 192.168.22.100
```

11. 各クラスターに定義したリソースグループにチケット制約を追加します。

```
[cluster1-node1 ~] # pcs constraint ticket add apacheticket apachegroup
[cluster2-node1 ~] # pcs constraint ticket add apacheticket apachegroup
```

次のコマンドを実行すると、現在設定されているチケット制約を表示できます。

```
pcs constraint ticket [show]
```

12. この設定用に作成したチケットを最初のクラスターに付与します。

チケットを付与する前にチケット抑制を定義する必要はありません。最初にチケットをクラスターに付与した後、**pcs booth ticket revoke** コマンドで手動でオーバーライドしない限り、Booth はチケットの管理を引き継ぎます。**pcs booth** 管理コマンドの詳細は、**pcs booth** コマンドの PCS ヘルプ画面を参照してください。

```
[cluster1-node1 ~] # pcs booth ticket grant apacheticket
```

チケットは、いつでも (この手順の完了後でも) 追加および削除できます。ただし、チケットを追加または削除した後、この手順の説明どおりに、他のノード、クラスター、および仲裁ノードに対して設定ファイルを同期し、チケットを付与する必要があります。

Booth 設定ファイル、チケット、およびリソースのクリーンアップや削除に使用できるその他の Booth 管理コマンドに関する情報は、**pcs booth** コマンドの PCS ヘルプ画面を参照してください。

## 付録A OCF 戻りコード

この付録では、OCF 戻りコードと、Pacemaker に解釈される仕組みを説明します。

エージェントがコードを返したときに、クラスターがまず行うことは、期待されている結果通りにコードを返しているかどうかを確認します。そして、結果が期待されている値に一致しない場合、その操作が失敗したものとみなされ、復元操作が開始されます。

起動するには、起動した操作の結果の呼び出し元を通知する、定義した戻りコードでリソースエージェントを終了する必要があります。

表A.1「クラスターが行う復旧タイプ」の説明どおり、障害回復には3種類あります。

表A.1 クラスターが行う復旧タイプ

タイプ	説明	クラスターが行った操作
軽度	一時的なエラーが発生しました。	リソースを再起動するか、新しい場所に移します。
重度	現在のノードに固有である可能性のある一時的ではないエラーが発生しました。	リソースを別の場所に移動し、現在のノードで再試行されないようにします。
致命的	すべてのクラスターノードに共有となる一時的でないエラーが発生しました (例: 指定された設定がよくありません)。	リソースを停止し、いかなるクラスターノードでも起動されないようにします。

表A.2「OCF 戻りコード」では、OCF 戻りコードと、不具合のあるコードを受信したときにクラスターが開始する復旧タイプについて説明しています。0 (OCF エイリアス **OCF\_SUCCESS**) を返すイベントアクションは、0 が、期待されている戻り値でない場合に、失敗とみなされます。

表A.2 OCF 戻りコード

戻りコード	OCF ラベル	説明
0	<b>OCF_SUCCESS</b>	<div>操作が無事に完了しました。これは、起動、停止、昇格、降格コマンドに対して想定される戻りコードです。</div> <div>予期しない場合のタイプ: ソフト</div>
1	<b>OCF_ERR_GENERIC</b>	<div>この操作は、一般的なエラーを返しました。</div> <div>タイプ: ソフト</div> <div>リソースマネージャーは、リソースの復元と、新しい場所への移動を試行します。</div>

戻りコード	OCF ラベル	説明
2	<b>OCF_ERR_ARGS</b>	<p>このマシンのリソースの設定が正しくありません。たとえば、ノードで見つからない場所を参照しています。</p> <p>タイプ: 重度</p> <p>リソースマネージャーがリソースを別の場所に移動し、現在のノードで再試行されないようにします。</p>
3	<b>OCF_ERR_UNIMPLEMENTED</b>	<p>要求された操作が実装されていません。</p> <p>タイプ: 重度</p>
4	<b>OCF_ERR_PERM</b>	<p>このリソースエージェントには、このタスクを完了するのに十分な特権がありません。これは、エージェントが特定のファイルを開けない場合や、特定のソケットでリッスンできない場合、ディレクトリへの書き込みを行えない場合が考えられます。</p> <p>タイプ: 重度</p> <p>特に設定されていない限り、リソースマネージャーは、別のノード (パーミッションが存在しない) でリソースを再起動することで、このエラーで失敗したリソースの復旧を試行します。</p>
5	<b>OCF_ERR_INSTALLED</b>	<p>操作が実行されたノードに、必要なコンポーネントが欠如しています。これは、必要なバイナリーが実行不可であるか、重要な設定ファイルが読み込み不可になっていることが原因の場合があります。</p> <p>タイプ: 重度</p> <p>特に設定されていない限り、リソースマネージャーは、別のノード (必要なファイルまたはバイナリーが存在しない) でリソースを再起動することで、このエラーで失敗したリソースの復旧を試行します。</p>

戻りコード	OCF ラベル	説明
6	<b>OCF_ERR_CONFIGURED</b>	<p>ローカルノード上のリソースの設定が正しくありません。</p> <p>タイプ: 致命的</p> <p>このコードがかえされると、Pacemaker は、サービス設定がその他のノードで正しくても、クラスター内のノードでリソースが実行されないようにします。</p>
7	<b>OCF_NOT_RUNNING</b>	<p>このリソースは安全に停止します。これは、リソースが正常にシャットダウンされたか、起動されていないことを意味します。</p> <p>予期しない場合のタイプ: ソフト</p> <p>クラスターは、いかなる操作に対しても、これを返すリソースの停止を試行しません。</p>
8	<b>OCF_RUNNING_MASTER</b>	<p>リソースはマスターモードで実行されています。</p> <p>予期しない場合のタイプ: ソフト</p>
9	<b>OCF_FAILED_MASTER</b>	<p>リソースはマスターモードで実行されていますが、不具合が発生しています。</p> <p>タイプ: ソフト</p> <p>リソースは降格されて停止し、再起動されています (再び昇格されている可能性もあります)。</p>
その他	該当なし	カスタムエラーコード

## 付録B RED HAT ENTERPRISE LINUX 6 および RED HAT ENTERPRISE LINUX 7 でのクラスターの作成

Pacemaker を用いて Red Hat Enterprise Linux 7 で Red Hat High Availability Cluster を設定する場合、**rgmanager** を用いて Red Hat Enterprise Linux 6 のクラスターを設定する場合とは異なる設定ツールと管理インターフェイスが必要になります。「[クラスター作成 - rgmanager と Pacemaker](#)」では、さまざまなクラスターコンポーネント間の設定の違いをまとめています。

Red Hat Enterprise Linux 6.5 以降のリリースは **pcs** 設定ツールを使用した Pacemaker によるクラスター設定に対応しています。「[Red Hat Enterprise Linux 6 および Red Hat Enterprise Linux 7 での Pacemaker のインストール](#)」では、Red Hat Enterprise Linux 6 と Red Hat Enterprise Linux 7 における Pacemaker インストールの相違点をまとめています。

### B.1. クラスター作成 - RGMANAGER と PACEMAKER

表B.1「[rgmanager と Pacemaker を使用した場合のクラスター設定に関する比較](#)」では、Red Hat Enterprise Linux 6 で **rgmanager** を使用した場合と Red Hat Enterprise Linux 7 で Pacemaker を使用した場合のクラスターコンポーネントの設定方法を比較しています。

表B.1 rgmanager と Pacemaker を使用した場合のクラスター設定に関する比較

設定コンポーネント	rgmanager	Pacemaker
クラスター設定ファイル	各ノード上のクラスター設定ファイルは、直接編集が可能な <b>cluster.conf</b> ファイルです。あるいは、 <b>luci</b> か <b>ccs</b> インターフェイスを使用して、クラスター設定を定義します。	クラスターおよび Pacemaker 設定ファイルは <b>corosync.conf</b> および <b>cib.xml</b> です。 <b>cib.xml</b> ファイルは直接編集しないでください。代わりに <b>pcs</b> または <b>pcsd</b> インターフェイスを使用して編集してください。
ネットワーク設定	クラスター設定前に IP アドレスおよび SSH を設定	クラスター設定前に IP アドレスおよび SSH を設定
クラスター設定ツール	<b>luci</b> 、 <b>ccs</b> コマンド、手作業で <b>cluster.conf</b> ファイルを編集	<b>pcs</b> または <b>pcsd</b>
インストール	<b>rgmanager</b> のインストール ( <b>ricci</b> 、 <b>luci</b> 、リソース、フェンスエージェントなどすべての依存パッケージをインストール)。必要な場合は <b>lvm2-cluster</b> および <b>gfs2-utils</b> をインストールします。	<b>pcs</b> と必要なフェンスエージェントをインストールします。必要な場合は <b>lvm2-cluster</b> および <b>gfs2-utils</b> をインストールします。

設定コンポーネント	rgmanager	Pacemaker
クラスターサービスの起動	<p>次の手順でクラスターサービスを起動、有効にする</p> <ol style="list-style-type: none"> <li>1. <b>rgmanager</b> と <b>cman</b>、必要であれば <b>clvmd</b> と <b>gfs2</b> も起動する</li> <li>2. <b>ricci</b> を起動、<b>luci</b> インターフェイスを使用している場合は <b>luci</b> を起動</li> <li>3. 必要なサービスに対して <b>chkconfig on</b> を実行し各ランタイムで起動するようにする</li> </ol> <p>または、<b>ccs --start</b> を実行してクラスターサービスを開始および有効化します。</p>	<p>次の手順でクラスターサービスを起動、有効にする</p> <ol style="list-style-type: none"> <li>1. 各ノードで、<b>systemctl start pcsd.service</b> を実行した後に <b>systemctl enable pcsd.service</b> を実行し、起動時に開始するように <b>pcsd</b> を有効にします。</li> <li>2. クラスターの1つのノードで、<b>pcs cluster start --all</b> を実行し、<b>corosync</b> および <b>pacemaker</b> を起動します。</li> </ol>
設定ツールへのアクセスの制御	<p><b>luci</b> の場合、root ユーザーまたは <b>luci</b> パーミッションを持つユーザーは <b>luci</b> にアクセスできます。すべてのアクセスには、ノードの <b>ricci</b> パスワードが必要です。</p>	<p><b>pcsd</b> GUI では、共通のシステムユーザーであるユーザー <b>hacluster</b> として認証される必要があります。root ユーザーは <b>hacluster</b> のパスワードを設定できます。</p>
クラスター作成	<p>クラスターの命名、クラスターに含ませるノードの定義などは <b>luci</b> または <b>ccs</b> で行うか <b>cluster.conf</b> ファイルを直接編集</p>	<p><b>pcs cluster setup</b> コマンドまたは <b>pcsd</b> Web UI を使用して、クラスターに名前を付け、ノードを含めます。<b>pcs cluster node add</b> コマンドまたは <b>pcsd</b> Web UI を使用すると、ノードを既存のクラスターに追加できます。</p>
クラスター設定を全ノードに伝える	<p>クラスターを <b>luci</b> で設定する場合は設定は自動です。<b>ccs</b> で、<b>--sync</b> オプションを使用します。<b>cman_tool version -r</b> コマンドを使用することもできます。</p>	<p>クラスターおよび Pacemaker 設定ファイルである <b>corosync.conf</b> および <b>cib.xml</b> は、クラスターの設定時やノードまたはリソースの追加時に自動的に伝播されます。</p>
グローバルのクラスタープロパティー	<p>以下の機能は、Red Hat Enterprise Linux 6 の <b>rgmanager</b> によってサポートされます。</p> <ul style="list-style-type: none"> <li>* クラスターネットワーク内での IP マルチキャストに使用するマルチキャストアドレスの選択をシステム側で行うよう設定することが可能</li> <li>* IP マルチキャストが利用できない場合に UDP Unicast トランスポートメカニズムが使用可能</li> <li>* RRP プロトコルを使用したクラスター設定が可能</li> </ul>	<p>Red Hat Enterprise Linux 7 の Pacemaker はクラスターに対して以下の機能をサポートします。</p> <ul style="list-style-type: none"> <li>* クラスターに <b>no-quorum-policy</b> を設定しクラスターが定足数を持たない場合のシステムの動作を指定できる</li> <li>* 設定可能な追加のクラスタープロパティーについては、<a href="#">表12.1「クラスターのプロパティー」</a>を参照してください。</li> </ul>
ロギング	<p>グローバルおよびデーモン固有のログ記録設定が可能</p>	<p>ログ記録を手作業で設定する方法については <b>/etc/sysconfig/pacemaker</b> ファイルを参照</p>



設定コンポーネント	rgmanager	Pacemaker
クラスターの検証	<b>luci</b> および <b>ccs</b> ではクラスタースキーマを使った自動検証、クラスターは起動時に自動的に検証されるクラスターは起動時に自動的に検証されます。	クラスターは起動時に自動的に検証される、または <b>pcs cluster verify</b> を使った検証も可
2 ノードクラスターのクォーラム	2 ノードのクラスターの場合、システムによるクォーラムの決定方法を設定できます。  * クォーラムディスクの設定  * <b>ccs</b> を使用するか <b>cluster.conf</b> ファイルを編集して <b>two_node=1</b> および <b>expected_votes=1</b> を設定し、1つのノードがクォーラムを維持できるようにします。	<b>pcs</b> は 2 ノードクラスターに必要なオプションを自動的に <b>corosync</b> へ追加します。
クラスターの状態	<b>luci</b> では、クラスターの現在の状態がインターフェイスのさまざまなコンポーネントで表示されます。 <b>ccs</b> コマンドの <b>--getconf</b> オプションを使用して、現在の設定ファイルを確認できます。オプションを使用すると、現在の設定ファイルを確認できます。 <b>clustat</b> コマンドを使用するとクラスターの状態を表示できます。	<b>pcs status</b> コマンドを使用して、現在のクラスターの状態を表示できます。
リソース	定義したタイプのリソースの追加およびリソース固有のプロパティの設定は <b>luci</b> または <b>ccs</b> コマンドを使用して行うか <b>cluster.conf</b> 設定ファイルを編集して行う	<b>pcs resource create</b> コマンドまたは <b>pcsd</b> Web UI を使用して、定義されたタイプのリソースを追加し、リソース固有のプロパティを設定します。Pacemaker を使用してクラスターリソースを設定する方法は、 <a href="#">6章 クラスターリソースの設定</a> を参照してください。

設定コンポーネント	rgmanager	Pacemaker
リソースの動作、グループ化、起動と停止の順序	リソースの通信方法の設定にクラスターの <b>サービス</b> を定義	<p>Pacemaker では、同時に配置され、順番に開始および停止される必要があるリソースのセットを定義する簡単な方法としてリソースグループを使用します。さらに、以下の方法でリソースの動作および対話方法を定義できます。</p> <ul style="list-style-type: none"> <li>* リソース動作の一部はリソースオプションとして設定</li> <li>* 場所の制約を使ってリソースを実行させるノードを指定</li> <li>* 順序の制約を使ってリソースの実行順序を指定</li> </ul> <p>コロケーション制約を使って任意のリソースの場所が別のリソースの場所に依存することを指定</p> <p>これらのトピックの詳細については <a href="#">6 章 クラスターリソースの設定</a> および <a href="#">7 章 リソースの制約</a> を参照してください。</p>
リソース管理: リソースの移動、起動、停止	<b>luci</b> を使用すると、クラスター、個別のクラスターノード、およびクラスターサービスを管理できます。 <b>ccs</b> コマンドを使用すると、クラスターを管理できます。 <b>clusvadm</b> を使用してクラスターサービスを管理できます。	<b>pcs cluster standby</b> コマンドを使ってノードを一時的に無効にし、リソースをホストできないようにすることができます。これにより、リソースを移行することになります。 <b>pcs resource disable</b> コマンドでリソースを停止できます。
クラスターの設定を完全に削除	<b>luci</b> では、削除してクラスターを完全に削除するために、クラスター内のすべてのノードを選択できます。クラスターの各ノードから <b>cluster.conf</b> を削除することもできます。	<b>pcs cluster destroy</b> コマンドを使用するとクラスター設定を削除できます。
複数のノードで実行中のリソース、複数モードで複数のノード上で実行中のリソース	該当なし。	Pacemaker ではリソースのクローンを作成し複数のノードで実行させることが可能で、作成したリソースのクローンをマスターリソースとスレーブリソースとして定義し複数のモードで実行させることが可能です。クローン作成したリソースおよびマスター/スレーブリソースに関する情報は、 <a href="#">9 章 高度な設定</a> を参照してください。

設定コンポーネント	rgmanager	Pacemaker
フェンス機能 -- 1 ノードにつき 1 フェンス	フェンスデバイスをグローバルに作成するか、またはローカルに作成し、ノードに追加します。 <b>post-fail delay</b> と <b>post-join delay</b> の値を全体として定義できます。	<b>pcs stonith create</b> または <b>pcsd</b> Web UI を使用して各ノードにフェンスデバイスを作成します。複数のノードをフェンスできるデバイスでは、各ノードで個別に定義せずに 1 度に定義する必要があります。 <b>pcmk_host_map</b> を定義して、1 つのコマンドで全ノードのフェンスデバイスを設定することもできます。 <b>pcmk_host_map</b> の詳細は、表 5.1「フェンスデバイスの一般的なプロパティ」を参照してください。クラスターの <b>stonith-timeout</b> の値を全体として定義できます。
1 ノードごとに複数の (バックアップ) フェンスデバイス	バックアップデバイスの定義は <b>luci</b> または <b>ccs</b> コマンドを使用するか <b>cluster.conf</b> ファイルを直接編集	フェンスレベルを設定

## B.2. RED HAT ENTERPRISE LINUX 6 および RED HAT ENTERPRISE LINUX 7 での PACEMAKER のインストール

Red Hat Enterprise Linux 6.5 以降のリリースは **pcs** 設定ツールを使用した Pacemaker によるクラスター設定に対応しています。Pacemaker を使用する際、Red Hat Enterprise Linux 6 と Red Hat Enterprise Linux 7 におけるクラスターのインストールには、いくつか相違点があります。

以下のコマンドは、Pacemaker が必要とする Red Hat High Availability Add-On ソフトウェアパッケージを Red Hat Enterprise Linux 6 にインストールし、**cman** を使用して **corosync** が開始されるようにします。クラスターの各ノードでこれらのコマンドを実行する必要があります。

```
[root@rhel6]# yum install pacemaker cman pcs
[root@rhel6]# chkconfig corosync off
[root@rhel6]# chkconfig cman off
```

クラスターの各ノードでは、**hacluster** という名前の **pcs** 管理者アカウントのパスワードを設定し、**pcsd** サービスの起動と有効化を行います。

```
[root@rhel6]# passwd hacluster
[root@rhel6]# service pcsd start
[root@rhel6]# chkconfig pcsd on
```

クラスターの 1 つのノードでは、クラスターのノードの管理者アカウントの認証を行います。

```
[root@rhel6]# pcs cluster auth [node] [...] [-u username] [-p password]
```

Red Hat Enterprise Linux 7 では、クラスターの各ノードで以下のコマンドを実行して Pacemaker が必要な Red Hat High Availability Add-On ソフトウェアパッケージをインストールして、**hacluster** という名前の **pcs** 管理アカウントのパスワードを設定し、**pcsd** サービスの起動と有効化を行います。

```
[root@rhel7]# yum install pcs pacemaker fence-agents-all
[root@rhel7]# passwd hacluster
[root@rhel7]# systemctl start pcsd.service
```

```
[root@rhel7]# systemctl enable pcsd.service
```

Red Hat Enterprise Linux 7 では、Red Hat Enterprise Linux 6 と同様に、クラスターのノードで以下のコマンドを実行して、クラスターのノードの管理者アカウントを認証します。

```
[root@rhel7]# pcs cluster auth [node] [...] [-u username] [-p password]
```

Red Hat Enterprise Linux 7 のインストールは、[1章Red Hat High Availability Add-On 設定および管理リファレンスの概要](#)と [4章 クラスターの作成と管理](#) を参照してください。

## 付録C 更新履歴

改訂 8.1-1 7.8 Beta 公開用ドキュメントバージョン	Fri Feb 28 2020	Steven Levine
改訂 7.1-1 7.7 GA 公開用ドキュメントバージョン	Wed Aug 7 2019	Steven Levine
改訂 6.1-1 7.6 GA 公開用ドキュメントバージョン	Thu Oct 4 2018	Steven Levine
改訂 5.1-2 7.5 GA 公開用ドキュメントバージョン	Thu Mar 15 2018	Steven Levine
改訂 5.1-0 7.5 ベータ版公開用ドキュメントバージョン	Thu Dec 14 2017	Steven Levine
改訂 4.1-9 7.4 のバージョンを更新	Tue Oct 17 2017	Steven Levine
改訂 4.1-5 7.4 GA 公開用ドキュメントバージョン	Wed Jul 19 2017	Steven Levine
改訂 4.1-2 7.4 ベータ版公開用ドキュメントの準備	Wed May 10 2017	Steven Levine
改訂 3.1-10 7.3 GA 公開用バージョンの更新。	Tue May 2 2017	Steven Levine
改訂 3.1-4 7.3 GA リリースのバージョン	Mon Oct 17 2016	Steven Levine
改訂 3.1-3 7.3 ベータ公開用ドキュメントの準備	Wed Aug 17 2016	Steven Levine
改訂 2.1-8 7.2 GA 公開用ドキュメントの準備	Mon Nov 9 2015	Steven Levine
改訂 2.1-5 7.2 ベータ公開用ドキュメントの準備	Mon Aug 24 2015	Steven Levine
改訂 1.1-9 7.1 GA リリース向けのバージョン	Mon Feb 23 2015	Steven Levine
改訂 1.1-7 7.1 ベータリリース向けバージョン	Thu Dec 11 2014	Steven Levine
改訂 0.1-41 7.0 GA リリース向けバージョン	Mon Jun 2 2014	Steven Levine
改訂 0.1-2 初回ドラフトの初版	Thu May 16 2013	Steven Levine

## 索引

シンボル

## アクション

### プロパティ

enabled, [リソースの動作](#)

id, [リソースの動作](#)

interval, [リソースの動作](#)

name, [リソースの動作](#)

on-fail, [リソースの動作](#)

timeout, [リソースの動作](#)

## アクションプロパティ, [リソースの動作](#)

### オプション

batch-limit, [クラスタープロパティとオプションの要約](#)

clone-max, [クローンリソースの作成と削除](#)

clone-node-max, [クローンリソースの作成と削除](#)

cluster-delay, [クラスタープロパティとオプションの要約](#)

cluster-infrastructure, [クラスタープロパティとオプションの要約](#)

cluster-recheck-interval, [クラスタープロパティとオプションの要約](#)

dampen, [接続状態変更によるリソースの移動](#)

dc-version, [クラスタープロパティとオプションの要約](#)

enable-acl, [クラスタープロパティとオプションの要約](#)

failure-timeout, [リソースのメタオプション](#)

fence-reaction, [クラスタープロパティとオプションの要約](#)

globally-unique, [クローンリソースの作成と削除](#)

host\_list, [接続状態変更によるリソースの移動](#)

interleave, [クローンリソースの作成と削除](#)

is-managed, [リソースのメタオプション](#)

last-lrm-refresh, [クラスタープロパティとオプションの要約](#)

maintenance-mode, [クラスタープロパティとオプションの要約](#)

master-max, [多状態のリソース: 複数モードのリソース](#)

master-node-max, [多状態のリソース: 複数モードのリソース](#)

migration-limit, [クラスタープロパティとオプションの要約](#)

migration-threshold, [リソースのメタオプション](#)

multiple-active, [リソースのメタオプション](#)

multiplier, [接続状態変更によるリソースの移動](#)

no-quorum-policy, [クラスタープロパティとオプションの要約](#)

notify, [クローンリソースの作成と削除](#)

ordered, [クローンリソースの作成と削除](#)

pe-error-series-max, [クラスタープロパティとオプションの要約](#)

pe-input-series-max, [クラスタープロパティとオプションの要約](#)

pe-warn-series-max, [クラスタープロパティとオプションの要約](#)

placement-strategy, [クラスタープロパティとオプションの要約](#)

priority, [リソースのメタオプション](#)

requires, [リソースのメタオプション](#)

resource-stickiness, [リソースのメタオプション](#)

shutdown-escalation, [クラスタープロパティとオプションの要約](#)

start-failure-is-fatal, [クラスタープロパティとオプションの要約](#)

stonith-action, [クラスタープロパティとオプションの要約](#)

stonith-enabled, [クラスタープロパティとオプションの要約](#)

stonith-timeout, [クラスタープロパティとオプションの要約](#)

stop-all-resources, [クラスタープロパティとオプションの要約](#)

stop-orphan-actions, [クラスタープロパティとオプションの要約](#)

stop-orphan-resources, [クラスタープロパティとオプションの要約](#)

symmetric-cluster, [クラスタープロパティとオプションの要約](#)

target-role, [リソースのメタオプション](#)

オプションのクエリー, [クラスタープロパティ設定のクエリー](#)

クエリー

[クラスターのプロパティ](#), [クラスタープロパティ設定のクエリー](#)

クラスター

オプション

batch-limit, [クラスタープロパティとオプションの要約](#)

cluster-delay, [クラスタープロパティとオプションの要約](#)

cluster-infrastructure, [クラスタープロパティとオプションの要約](#)

cluster-recheck-interval, [クラスタープロパティとオプションの要約](#)

dc-version, [クラスタープロパティとオプションの要約](#)

enable-acl, [クラスタープロパティとオプションの要約](#)

fence-reaction, [クラスタープロパティとオプションの要約](#)

last-lrm-refresh, [クラスタープロパティとオプションの要約](#)

maintenance-mode, [クラスタープロパティとオプションの要約](#)

migration-limit, [クラスタープロパティとオプションの要約](#)

no-quorum-policy, [クラスタープロパティとオプションの要約](#)

pe-error-series-max, [クラスタープロパティとオプションの要約](#)

pe-input-series-max, [クラスタープロパティとオプションの要約](#)

pe-warn-series-max, [クラスタープロパティとオプションの要約](#)

placement-strategy, [クラスタープロパティとオプションの要約](#)

shutdown-escalation, [クラスタープロパティとオプションの要約](#)

start-failure-is-fatal, [クラスタープロパティとオプションの要約](#)

stonith-action, [クラスタープロパティとオプションの要約](#)

stonith-enabled, [クラスタープロパティとオプションの要約](#)

stonith-timeout, [クラスタープロパティとオプションの要約](#)

stop-all-resources, [クラスタープロパティとオプションの要約](#)

stop-orphan-actions, [クラスタープロパティとオプションの要約](#)

stop-orphan-resources, [クラスタープロパティとオプションの要約](#)

symmetric-cluster, [クラスタープロパティとオプションの要約](#)

プロパティのクエリー, [クラスタープロパティ設定のクエリー](#)

プロパティの削除, [クラスターのプロパティの設定と削除](#)

プロパティの設定, [クラスターのプロパティの設定と削除](#)

クラスターのステータス

display, [クラスターの状態表示](#)

クラスターのプロパティ, [クラスターのプロパティの設定と削除](#), [クラスタープロパティ設定のクエリー](#)

クラスターオプション, [クラスタープロパティとオプションの要約](#)

クラスター管理

ACPI の設定, [統合フェンスデバイスで使用する ACPI の設定](#)

クローン, [リソースのクローン](#)

オプション

clone-max, [クローンリソースの作成と削除](#)

clone-node-max, [クローンリソースの作成と削除](#)

globally-unique, [クローンリソースの作成と削除](#)

interleave, [クローンリソースの作成と削除](#)



notify, クローンリソースの作成と削除

ordered, クローンリソースの作成と削除

クローンオプション, クローンリソースの作成と削除

グループ, リソースグループ, グループの Stickiness (粘着性)

グループリソース, リソースグループ

コロケーション, リソースのコロケーション

プロパティ

enabled, リソースの動作

id, リソースのプロパティ, リソースの動作, 多状態のリソース: 複数モードのリソース

interval, リソースの動作

name, リソースの動作

on-fail, リソースの動作

provider, リソースのプロパティ

standard, リソースのプロパティ

timeout, リソースの動作

type, リソースのプロパティ

プロパティの削除, クラスターのプロパティの設定と削除

プロパティの設定, クラスターのプロパティの設定と削除

リソース, リソースのプロパティ, リソースを手作業で移動する

cleanup, クラスターリソースのクリーンアップ

オプション

failure-timeout, リソースのメタオプション

is-managed, リソースのメタオプション

migration-threshold, リソースのメタオプション

multiple-active, リソースのメタオプション

priority, リソースのメタオプション

requires, リソースのメタオプション

resource-stickiness, リソースのメタオプション

target-role, リソースのメタオプション

クローン, リソースのクローン

グループ, リソースグループ

プロパティ

id, [リソースのプロパティ](#)

provider, [リソースのプロパティ](#)

standard, [リソースのプロパティ](#)

type, [リソースのプロパティ](#)

他のリソースに相対的となる場所, [リソースのコロケーション](#)

制約

コロケーション, [リソースのコロケーション](#)

ルール, [Pacemaker ルール](#)

属性式, [ノード属性の式](#)

日付/時刻の式, [時刻と日付ベースの式](#)

日付の詳細, [日付の詳細](#)

期間, [期間](#)

順序, [順序の制約](#)

場所

ルールで確定, [ルールを使用したリソースの場所の確定](#)

多状態, [多状態のリソース: 複数モードのリソース](#)

有効化, [クラスターリソースの有効化と無効化](#)

無効化, [クラスターリソースの有効化と無効化](#)

移動する, [リソースを手作業で移動する](#)

起動順序, [順序の制約](#)

リソースのクローン, [リソースのクローン](#)

リソースの場所の確定, [ルールを使用したリソースの場所の確定](#)

リソースオプション, [リソースのメタオプション](#)

ルール, [Pacemaker ルール](#)

boolean-op, [Pacemaker ルール](#)

score, [Pacemaker ルール](#)

score-attribute, [Pacemaker ルール](#)

リソースの場所の確定, [ルールを使用したリソースの場所の確定](#)

ロール, [Pacemaker ルール](#)

ルールで確定, [ルールを使用したリソースの場所の確定](#)

ロール, [Pacemaker ルール](#)

制約ルール, [Pacemaker ルール](#)

他のリソースに相対的となる場所, [リソースのコロケーション](#)

使用率属性, [使用と配置ストラテジー](#)

制約

コロケーション, [リソースのコロケーション](#)

ルール, [Pacemaker ルール](#)

boolean-op, [Pacemaker ルール](#)

score, [Pacemaker ルール](#)

score-attribute, [Pacemaker ルール](#)

ロール, [Pacemaker ルール](#)

場所

id, [基本的な場所の制約](#)

score, [基本的な場所の制約](#)

属性式, [ノード属性の式](#)

attribute, [ノード属性の式](#)

operation, [ノード属性の式](#)

type, [ノード属性の式](#)

value, [ノード属性の式](#)

日付/時刻の式, [時刻と日付ベースの式](#)

end, [時刻と日付ベースの式](#)

operation, [時刻と日付ベースの式](#)

start, [時刻と日付ベースの式](#)

日付の詳細, [日付の詳細](#)

hours, [日付の詳細](#)

id, [日付の詳細](#)

monthdays, [日付の詳細](#)

months, [日付の詳細](#)

moon, [日付の詳細](#)

weekdays, [日付の詳細](#)

weeks, [日付の詳細](#)

weekyears, [日付の詳細](#)

yeardays, [日付の詳細](#)

years, [日付の詳細](#)

期間, [期間](#)

順序, [順序の制約](#)

種類, [順序の制約](#)

制約ルール, [Pacemaker ルール](#)

制約式, [ノード属性の式](#), [時刻と日付ベースの式](#)

削除中

クラスターのプロパティ, [クラスターのプロパティの設定と削除](#)

場所

score, [基本的な場所の制約](#)

ルールで確定, [ルールを使用したリソースの場所の確定](#)

場所の制約, [基本的な場所の制約](#)

多状態, [多状態のリソース: 複数モードのリソース](#), [多状態の粘着性 \(Stickiness\)](#)

対称, [順序の制約](#)

順序の制約, [順序の制約](#)

属性式, [ノード属性の式](#)

attribute, [ノード属性の式](#)

operation, [ノード属性の式](#)

type, [ノード属性の式](#)

value, [ノード属性の式](#)

新機能と変更点, [新機能と変更点](#)

日付/時刻の式, [時刻と日付ベースの式](#)

end, [時刻と日付ベースの式](#)

operation, [時刻と日付ベースの式](#)

start, [時刻と日付ベースの式](#)

日付の詳細, [日付の詳細](#)

hours, [日付の詳細](#)

id, [日付の詳細](#)

monthdays, [日付の詳細](#)

months, [日付の詳細](#)

moon, [日付の詳細](#)

weekdays, [日付の詳細](#)

weeks, [日付の詳細](#)

weekyears, [日付の詳細](#)

yeardays, [日付の詳細](#)

years, [日付の詳細](#)

時間ベースの式, [時刻と日付ベースの式](#)

有効化

リソース, [クラスターリソースの有効化と無効化](#)

期間, [期間](#)

概要

新機能と変更点, [新機能と変更点](#)

無効化

リソース, [クラスターリソースの有効化と無効化](#)

移動する, [リソースを手作業で移動する](#)

リソース, [リソースを手作業で移動する](#)

種類, [順序の制約](#)

順序の制約, [順序の制約](#)

統合フェンスデバイス

ACPI の設定, [統合フェンスデバイスで使用する ACPI の設定](#)

設定

クラスターのプロパティ, [クラスターのプロパティの設定と削除](#)

起動順序, [順序の制約](#)

配置ストラテジー, [使用と配置ストラテジー](#)

順序

種類, [順序の制約](#)

順序の制約, [順序の制約](#)

対称, [順序の制約](#)

順序付け, [順序の制約](#)

, [クラスターの作成](#)

## A

### ACPI

設定, [統合フェンスデバイスで使用する ACPI の設定](#)

attribute, [ノード属性の式](#)

制約式, [ノード属性の式](#)

## B

batch-limit, [クラスタープロパティとオプションの要約](#)

クラスターオプション, [クラスタープロパティとオプションの要約](#)

boolean-op, [Pacemaker ルール](#)

制約ルール, [Pacemaker ルール](#)

## C

clone-max, [クローンリソースの作成と削除](#)

クローンオプション, [クローンリソースの作成と削除](#)

clone-node-max, [クローンリソースの作成と削除](#)

クローンオプション, [クローンリソースの作成と削除](#)

cluster-delay, [クラスタープロパティとオプションの要約](#)

クラスターオプション, [クラスタープロパティとオプションの要約](#)

cluster-infrastructure, [クラスタープロパティとオプションの要約](#)

クラスターオプション, [クラスタープロパティとオプションの要約](#)

cluster-recheck-interval, [クラスタープロパティとオプションの要約](#)

クラスターオプション, [クラスタープロパティとオプションの要約](#)

## D

dampen, [接続状態変更によるリソースの移動](#)

Ping リソースオプション, [接続状態変更によるリソースの移動](#)

dc-version, [クラスタープロパティとオプションの要約](#)

クラスターオプション, [クラスタープロパティとオプションの要約](#)

## E

enable-acl, [クラスタープロパティとオプションの要約](#)

クラスターオプション, [クラスタープロパティとオプションの要約](#)

enabled, [リソースの動作](#)

アクションプロパティ, [リソースの動作](#)

end, [時刻と日付ベースの式](#)

制約式, [時刻と日付ベースの式](#)

## F

failure-timeout, [リソースのメタオプション](#)

リソースオプション, [リソースのメタオプション](#)

fence-reaction, [クラスタープロパティとオプションの要約](#)

クラスターオプション, [クラスタープロパティとオプションの要約](#)

## G

globally-unique, [クローンリソースの作成と削除](#)

クローンオプション, [クローンリソースの作成と削除](#)

## H

host\_list, [接続状態変更によるリソースの移動](#)

Ping リソースオプション, [接続状態変更によるリソースの移動](#)

hours, [日付の詳細](#)

日付の詳細, [日付の詳細](#)

## I

id, [リソースのプロパティ](#), [リソースの動作](#), [日付の詳細](#)

Multi-State プロパティ, [多状態のリソース: 複数モードのリソース](#)

アクションプロパティ, [リソースの動作](#)

リソース, [リソースのプロパティ](#)

場所の制約, [基本的な場所の制約](#)

日付の詳細, [日付の詳細](#)

interleave, [クローンリソースの作成と削除](#)

クローンオプション, [クローンリソースの作成と削除](#)

interval, [リソースの動作](#)

アクションプロパティ, [リソースの動作](#)

is-managed, [リソースのメタオプション](#)

[リソースオプション](#), [リソースのメタオプション](#)

## L

last-lrm-refresh, [クラスタープロパティとオプションの要約](#)

[クラスターオプション](#), [クラスタープロパティとオプションの要約](#)

## M

maintenance-mode, [クラスタープロパティとオプションの要約](#)

[クラスターオプション](#), [クラスタープロパティとオプションの要約](#)

master-max, [多状態のリソース: 複数モードのリソース](#)

[Multi-State オプション](#), [多状態のリソース: 複数モードのリソース](#)

master-node-max, [多状態のリソース: 複数モードのリソース](#)

[Multi-State オプション](#), [多状態のリソース: 複数モードのリソース](#)

migration-limit, [クラスタープロパティとオプションの要約](#)

[クラスターオプション](#), [クラスタープロパティとオプションの要約](#)

migration-threshold, [リソースのメタオプション](#)

[リソースオプション](#), [リソースのメタオプション](#)

monthdays, [日付の詳細](#)

[日付の詳細](#), [日付の詳細](#)

months, [日付の詳細](#)

[日付の詳細](#), [日付の詳細](#)

moon, [日付の詳細](#)

[日付の詳細](#), [日付の詳細](#)

## Multi-State

### オプション

master-max, [多状態のリソース: 複数モードのリソース](#)

master-node-max, [多状態のリソース: 複数モードのリソース](#)

### プロパティ

id, [多状態のリソース: 複数モードのリソース](#)

[Multi-State オプション](#), [多状態のリソース: 複数モードのリソース](#)



Multi-State プロパティ, [多状態のリソース: 複数モードのリソース](#)

multiple-active, [リソースのメタオプション](#)

[リソースオプション](#), [リソースのメタオプション](#)

multiplier, [接続状態変更によるリソースの移動](#)

[Ping リソースオプション](#), [接続状態変更によるリソースの移動](#)

## N

name, [リソースの動作](#)

[アクションプロパティ](#), [リソースの動作](#)

no-quorum-policy, [クラスタープロパティとオプションの要約](#)

[クラスターオプション](#), [クラスタープロパティとオプションの要約](#)

notify, [クローンリソースの作成と削除](#)

[クローンオプション](#), [クローンリソースの作成と削除](#)

## O

OCF

[戻りコード](#), [OCF 戻りコード](#)

on-fail, [リソースの動作](#)

[アクションプロパティ](#), [リソースの動作](#)

operation, [ノード属性の式](#), [時刻と日付ベースの式](#)

[制約式](#), [ノード属性の式](#), [時刻と日付ベースの式](#)

ordered, [クローンリソースの作成と削除](#)

[クローンオプション](#), [クローンリソースの作成と削除](#)

## P

pe-error-series-max, [クラスタープロパティとオプションの要約](#)

[クラスターオプション](#), [クラスタープロパティとオプションの要約](#)

pe-input-series-max, [クラスタープロパティとオプションの要約](#)

[クラスターオプション](#), [クラスタープロパティとオプションの要約](#)

pe-warn-series-max, [クラスタープロパティとオプションの要約](#)

[クラスターオプション](#), [クラスタープロパティとオプションの要約](#)

Ping リソース

## オプション

dampen, [接続状態変更によるリソースの移動](#)

host\_list, [接続状態変更によるリソースの移動](#)

multiplier, [接続状態変更によるリソースの移動](#)

Ping リソースオプション, [接続状態変更によるリソースの移動](#)

placement-strategy, [クラスタープロパティとオプションの要約](#)

クラスターオプション, [クラスタープロパティとオプションの要約](#)

priority, [リソースのメタオプション](#)

リソースオプション, [リソースのメタオプション](#)

provider, [リソースのプロパティ](#)

リソース, [リソースのプロパティ](#)

## R

requires, [リソースのメタオプション](#)

resource-stickiness, [リソースのメタオプション](#)

Multi-State, [多状態の粘着性 \(Stickiness\)](#)

グループ, [グループの Stickiness \(粘着性\)](#)

リソースオプション, [リソースのメタオプション](#)

## S

score, [基本的な場所の制約](#), [Pacemaker ルール](#)

制約ルール, [Pacemaker ルール](#)

場所の制約, [基本的な場所の制約](#)

score-attribute, [Pacemaker ルール](#)

制約ルール, [Pacemaker ルール](#)

shutdown-escalation, [クラスタープロパティとオプションの要約](#)

クラスターオプション, [クラスタープロパティとオプションの要約](#)

standard, [リソースのプロパティ](#)

リソース, [リソースのプロパティ](#)

start, [時刻と日付ベースの式](#)

制約式, [時刻と日付ベースの式](#)

start-failure-is-fatal, [クラスタープロパティとオプションの要約](#)  
クラスターオプション, [クラスタープロパティとオプションの要約](#)

status  
display, [クラスターの状態表示](#)

stonith-action, [クラスタープロパティとオプションの要約](#)  
クラスターオプション, [クラスタープロパティとオプションの要約](#)

stonith-enabled, [クラスタープロパティとオプションの要約](#)  
クラスターオプション, [クラスタープロパティとオプションの要約](#)

stonith-timeout, [クラスタープロパティとオプションの要約](#)  
クラスターオプション, [クラスタープロパティとオプションの要約](#)

stop-all-resources, [クラスタープロパティとオプションの要約](#)  
クラスターオプション, [クラスタープロパティとオプションの要約](#)

stop-orphan-actions, [クラスタープロパティとオプションの要約](#)  
クラスターオプション, [クラスタープロパティとオプションの要約](#)

stop-orphan-resources, [クラスタープロパティとオプションの要約](#)  
クラスターオプション, [クラスタープロパティとオプションの要約](#)

symmetric-cluster, [クラスタープロパティとオプションの要約](#)  
クラスターオプション, [クラスタープロパティとオプションの要約](#)

## T

target-role, [リソースのメタオプション](#)  
リソースオプション, [リソースのメタオプション](#)

timeout, [リソースの動作](#)  
アクションプロパティ, [リソースの動作](#)

type, [リソースのプロパティ, ノード属性の式](#)  
リソース, [リソースのプロパティ](#)  
制約式, [ノード属性の式](#)

## V

value, [ノード属性の式](#)  
制約式, [ノード属性の式](#)

## W

weekdays, [日付の詳細](#)

[日付の詳細](#), [日付の詳細](#)

weeks, [日付の詳細](#)

[日付の詳細](#), [日付の詳細](#)

weekyears, [日付の詳細](#)

[日付の詳細](#), [日付の詳細](#)

## Y

yeardays, [日付の詳細](#)

[日付の詳細](#), [日付の詳細](#)

years, [日付の詳細](#)

[日付の詳細](#), [日付の詳細](#)