



Red Hat Enterprise Linux

7

High Availability Add-On リファレンス ス

Red Hat Enterprise Linux 7 向け High Availability Add-On のリファレンス
ドキュメント

Steven Levine

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

Red Hat Enterprise Linux 7 向け High Availability Add-On のリファレンス ドキュメント

Steven Levine
Red Hat Customer Content Services
slevine@redhat.com

法律上の通知

Copyright © 2016 Red Hat, Inc. and others.

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

概要

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

第1章 Red Hat High Availability Add-On の設定と管理のリファレンス概要	4
1.1. 新機能と変更点	4
1.2. Pacemaker 設定ツールのインストール	5
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	6
1.4. クラスターと Pacemaker の設定ファイル	6
1.5. クラスター設定の注意事項	7
第2章 pcsd Web UI	8
2.1. pcsd Web UI の設定	8
2.2. pcsd Web UI を用いたクラスターの作成	9
2.3. クラスターコンポーネントの設定	11
第3章 pcs コマンドラインインターフェース	14
3.1. pcs コマンド	14
3.2. pcs の使用に関するヘルプ表示	14
3.3. raw クラスター設定の表示	15
3.4. 設定の変更をファイルに保存	15
3.5. 状態の表示	15
3.6. 全クラスター設定の表示	16
3.7. 現在の pcs バージョンの表示	16
3.8. クラスター設定のバックアップおよび復元	16
第4章 クラスターの作成と管理	17
4.1. クラスターの作成	17
4.2. クラスターのタイムアウト値の設定	18
4.3. 冗長リングプロトコル (RRP) の設定	18
4.4. クラスターノードの管理	19
4.5. ユーザーのパーミッション設定	21
4.6. クラスター設定の削除	23
4.7. クラスターの状態表示	23
第5章 フェンス機能: STONITH の設定	24
5.1. STONITH (フェンス) エージェント	24
5.2. フェンスデバイスの一般的なプロパティ	24
5.3. デバイス固有のフェンスオプションの表示	25
5.4. フェンスデバイスの作成	25
5.5. アンフェンスによるストレージベースのフェンスデバイスの設定	26
5.6. フェンスデバイスの表示	26
5.7. フェンスデバイスの修正と削除	26
5.8. フェンスデバイスが接続されているノードの管理	27
5.9. その他のフェンス設定オプション	27
5.10. フェンスレベルの設定	30
5.11. 冗長電源のフェンシング設定	31
第6章 クラスターリソースの設定	32
6.1. リソースの作成	32
6.2. リソースのプロパティ	33
6.3. リソース固有のパラメーター	33
6.4. リソースのメタオプション	33
6.5. リソースグループ	36
6.6. リソースの動作	37
6.7. 設定されているリソースの表示	39
6.8. リソースパラメーターの変更	40

6.9. 複数のモニタリング動作	40
6.10. クラスターリソースの有効化と無効化	41
6.11. クラスターリソースのクリーンアップ	41
第7章 リソースの制約	42
7.1. 場所の制約	42
7.2. 順序の制約	45
7.3. リソースのコロケーション	47
7.4. 制約の表示	49
第8章 クラスターリソースの管理	50
8.1. リソースを手作業で移動する	50
8.2. 障害発生によるリソースの移動	51
8.3. 接続状態変更によるリソースの移動	52
8.4. クラスターリソースの有効化、無効化、および禁止	53
8.5. モニター操作の無効化	54
8.6. 管理リソース	54
第9章 高度なリソースタイプ	55
9.1. リソースのクローン	55
9.2. 多状態のリソース: 複数モードのリソース	57
9.3. pacemaker_remote サービス	59
第10章 クラスタークォーラム	66
10.1. クォーラムオプションの設定	66
10.2. クォーラム管理コマンド (Red Hat Enterprise Linux 7.3 以降)	66
10.3. クォーラムオプションの変更 (Red Hat Enterprise Linux 7.3 以降)	67
10.4. クォーラムアンブロック (quorum unblock) コマンド	68
10.5. クォーラムデバイス (テクノロジーレビュー)	68
第11章 Pacemaker ルール	75
11.1. ノード属性の式	75
11.2. 時刻と日付ベースの式	76
11.3. 日付の詳細	76
11.4. 期間	77
11.5. pcs を用いたルールの設定	77
11.6. 時刻ベースの式のサンプル	77
11.7. ルールを使用したリソースの場所の確定	77
第12章 Pacemaker クラスターのプロパティ	79
12.1. クラスタープロパティとオプションの要約	79
12.2. クラスターのプロパティの設定と削除	81
12.3. クラスタープロパティ設定のクエリー	81
第13章 クラスターイベントのスキ립トのトリガー	83
13.1. Pacemaker アラートエージェント (Red Hat Enterprise Linux 7.3 以降)	83
13.2. モニタリングのリソースを使ったイベント通知	89
第14章 Pacemaker を用いたマルチサイトクラスターの設定 (テクニカルレビュー)	92
付録A Red Hat Enterprise Linux 6 および Red Hat Enterprise Linux 7 でのクラスターの作成	95
A.1. クラスター作成 - rgmanager と Pacemaker	95
A.2. Red Hat Enterprise Linux 6.5 および Red Hat Enterprise Linux 7 で Pacemaker を用いたクラスターの作成	99
付録B 改訂履歴	101

索引	101
----------	-----

第1章 Red Hat High Availability Add-On の設定と管理のリファレンス概要

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** 設定インターフェースまたは **pcsd** GUI インターフェースを使用します。

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** コマンドがすべてのリソースのリソース状態と **failcount** をリセットす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るようになりました。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** を設定する方法をより明確に説明するため、[「モニタ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。
- ※ 本ドキュメントの [「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「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。

- ※ 本バージョンでは、[「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 \(Red Hat Enterprise Linux 7.3 以降\)」](#)に記載されているように、**pcs quorum update** コマンドを使用して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 \(テクノロジープレビュー\)」](#)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 \(テクニカルプレビュー\)](#)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。

1.2. 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 パッケージのインストール後、必ずソフトウェア更新に関する設定で自動インストールが行われないよう設定してください。実行中のクラスターでインストールが行われると予期しない動作の原因となる場合があります。

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

Red Hat High Availability Add-On では、受信トラフィックに対して以下のポートを有効にする必要があります。

- ※ TCP: ポート 2224、3121、21064
- ※ UDP: ポート 5405
- ※ DLM (clvm/GFS2 で DLM ロックマネージャーを使用する場合): ポート 21064

以下のコマンドを実行すると、**firewalld** デーモンを使用してこれらのポートを有効にすることができます。

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

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** と **cib.xml** です。これらのファイルは直接編集せずに、必ず **pcs** または **pcsd** インターフェースを使用して編集してください。

corosync.conf ファイルは、Pacemaker が構築されるクラスターマネージャーである **corosync** によって使用されるクラスターパラメーターを提供します。

cib.xml はクラスターの構成と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) によって使用されます。CIB の内容はクラスター全体で自動的に同期されます。

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。

- ※ Red Hat は完全なクラスターノードが 17 個以上あるクラスターデプロイメントをサポートしません。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ースで 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ー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。

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構成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。[図2.1 「クラスターページの管理」](#)に記載されている **Manage Clusters** ページが表示されます。

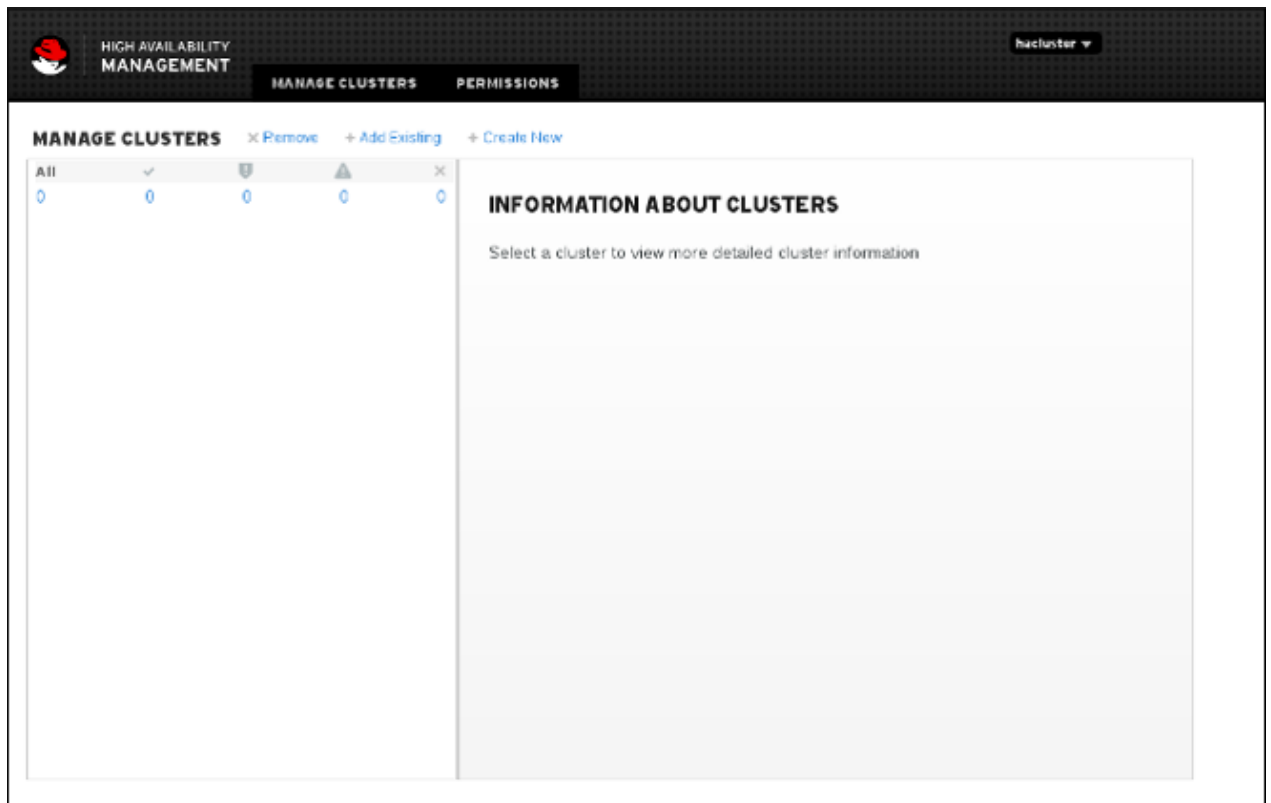


図2.1 クラスターページの管理

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構成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。

クラスターの作成または追加後、クラスター名が **Manage Cluster** ページに表示されます。クラスターを選択すると、そのクラスターの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ースを指定すると、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。

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:

UDPU (default) ▼

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

図2.2 クラスターページの作成

2.2.2. クラスターパーミッションの設定

Manage Clusters ページで **Permissions** をクリックし、個別ユーザーまたはグループのパーミッションを設定します。ユーザー **hacluster** は常にフルパーミッションを持つことに注意してください。



注記

この画面で設定するパーミッションは、Web UI でクラスターを管理するためのもので、[「ACL の設定」](#) に説明のあるアクセス制御リスト (ACL) を使用して設定されるパーミッションとは異なります。

この画面では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)

デフォルトでは、**haclient** グループのメンバーはクラスターに完全アクセスできます。

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」](#) に記載されている **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)。
- ※ **ACLs** ([「ACL の設定」](#) を参照)。
- ※ **Manage Clusters** ([「クラスタープロパティ」](#) を参照)。



図2.3 クラスターコンポーネントのメニュー

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 **Nodes** オプションを選択すると、ノードで実行されているリソースやリソースの場所設定などの、現在設定されているノードと現在選択されたノードの状態が表示されます。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。ノードをスタンバイモードにすることもできます。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。

[「フェンスデバイス」](#)に説明があるように、**Configure Fencing** を選択すると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変更、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、**Add Resource** 画面を表示します。**Type** ドロップダウン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。リソースに指定できる任意の引数を表示するには、**Optional Arguments** をクリックします。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とき、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、その引数の詳細な説明が表示されます。

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章高度なリソースタイプ](#) を参照してください。

最低でも 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** 画面が表示されたらグループ名を入力し、**Create Group** をクリックします。すると、**Resources** 画面に戻り、リソースのグループ名が表示されます。リソースグループの作成後、追加のリソースを作成または編集するときにグループ名をリソースパラメーターとして示すことができます。

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、**Add Fence Device** 画面を表示します。**Type** ドロップダウンメニューからフェンスデバイス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。フェンスデバイスに指定できる任意の引数を表示するには、**Optional Arguments** をクリックします。新しいフェンスデバイスのパラメーターを入力したら **Create Fence Instance** 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 **ACLS** 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ます。アクセス制御リスト (ACL) を使用すると、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。XPath クエリーまたは特定要素の ID のいずれかに適用される各ロールのパーミッション (Read (読み取り)/write (書き込み/deny (拒否)) の数は無制限です。

2.3.5. クラスタープロパティ

クラスター管理ページの上部にあるメニューから **Cluster Properties** 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、これらのプロパティの値をデフォルト値から他の値に変更できます。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。

第3章 pcs コマンドラインインターフェース

pcs コマンドラインインターフェースは、インターフェースを**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** コマンドの詳細については[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** コマンドの詳細については[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 高度なリソースタイプ](#)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** コマンドについては[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** コマンドについては[7章 リソースの制約](#)を参照してください。

➤ property

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** コマンドについては[「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。次の例は出力の一部です。

```
# 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 クラスター設定の表示

クラスター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 クラスター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** を使うと raw クラスター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。

クラスターが設定済みでアクティブな CIB が存在する場合は、次のコマンドを使用すると raw xml ファイルを保存できます。

```
pcs cluster cib filename
```

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

```
pcs cluster cib testfile
```

次のコマンドでは **testfile1** 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ー構成には追加されません。

```
# pcs -f testfile1 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 filename
```

3.5. 状態の表示

次のコマンドでクラスターおよびクラスターリソースの状態を表示します。

```
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

現在の全クラスター設定を表示させる場合は次のコマンドを使います。

```
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、標準出力が使用されます。**--local** オプションを指定すると、現在のノードのファイルのみが復元されます。

```
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. クラスターを構成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。ユーザー **hacluster** のパスワードも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。
- ※ **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** の両方で構成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構成、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
--token timeout	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)。
--join timeout	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)。
--consensus timeout	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)。
--miss_count_const count	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。
--fail_rcv_const failures	新しい設定の構成前に、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

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ー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 **udpu** トランSPORT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 A には **nodeA-0** と **nodeA-1** の 2 つのインターフェースがあります。ノード B には **nodeB-0** と **nodeB-1** の 2 つのインターフェースがあります。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 トランSPORT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** オプションを使用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

以下の手順にしたがって既存のクラスターに新しいノードを追加します。この例では、既存のクラスターノードは **clusternode-01.example.com**、**clusternode-02.example.com**、および **clusternode-03.example.com** 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。

```
[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に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ユーザー **hacluster** 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. 新しいノードにて、ユーザー **hacluster** をクラスターのすべてのノードに対して認証します。

```
[root@newnode ~]# pcs cluster auth
Username: hacluster
```



```
Password:
clusternode-01.example.com: Authorized
clusternode-02.example.com: Authorized
clusternode-03.example.com: Authorized
newnode.example.com: Already authorized
```

2. 新しいノードでクラスターサービスを開始および有効化します。

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

3. 新しいクラスターノードのフェンスデバイスを設定するようにしてください。フェンスデバイスの設定については [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、クラスターにある他のノードすべてで **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、クラスターを永久的に破壊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

次のコマンドでは指定した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。このノード上で現在実行中のリソースはすべて別のノードに移行されます。**--all** を使用すると全ノードがスタンバイモードになります。

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、設定をテストして実際にはノードのシャットダウンを行わずにリカバリーのシミュレーションを行う場合にも使用できます。

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

次のコマンドは指定したノードのスタンバイモードを外します。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** を使用すると全ノードのスタンバイモードを外します。

```
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を阻止する制約が追加されることになります。制約を取り除く場合は **pcs cluster unstandby** コマンドを実行します。このコマンドに必ずしもリソースを指定ノードに戻すわけではありません。最初にどのようにリソースを設定したかにより、その時点で実行できるノードに移動されます。リソースの制約については [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

デフォルトでは、root ユーザーと **haclient** グループのメンバーであるユーザーは、クラスター設定へ完全に読み書きアクセスできます。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。

以下の 2 つのステップにしたがって、ローカルユーザーのパーミッションを設定します。

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与えています。

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を持つ **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 システム内に **rouser** というユーザーを作成し、**read-only** ロールを割り当てます。

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

5. 現在の ACL を表示させます。

```
# 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 を表示させます。

```
# 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は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
```

第5章 フェンス機能: STONITH の設定

STONITH は「Shoot The Other Node In The Head」の略語です。問題のあるノードや同時アクセスによるデータ破損を防止します。

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。このようなことが発生した場合は、STONITH で全ノードを強制的にオフラインにしてからサービスを別の場所で開始すると安全です。

5.1. STONITH (フェンス) エージェント

次のコマンドは利用できる全 STONITH エージェントを表示します。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

注記

フェンスデバイスやフェンスリソースを無効にする場合にも通常のリソースと同様、**target-role** を設定することができます。

注記

特定のノードがフェンスデバイスを使用しないようにするには、フェンスリソースの場所の制約を設定します。

フェンスデバイスに設定できる汎用プロパティについては [表5.1「フェンスデバイスの一般的なプロパティ」](#) に記載されています。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
-------	-----	-------	----

フィールド	タイプ	デフォルト	説明
priority	整数	0	stonith リソースの優先度です。高い優先度のデバイスから順番に試行されます。
pcmk_host_map	文字列		ホスト名に対応していないデバイスのポート番号とホスト名をマッピングします。例えば、 node1:1;node2:2,3 なら node1 にはポート 1 を使用し node2 にはポート 2 と 3 を使用するようクラスターに指示します。
pcmk_host_list	文字列		このデバイスで制御するマシンの一覧です (オプション、 pcmk_host_check=static-list を除く)。
pcmk_host_check	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: dynamic-list (デバイスに問い合わせ)、 static-list (pcmk_host_list 属性をチェック)、なし (すべてのデバイスで全マシンのフェンスが可能とみなされる)

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
```

5.4. フェンスデバイスの作成

5.4. stonith 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
```

各ノードの異なるポートを使って複数のノードに 1 つのフェンスデバイスを使用する場合はノードごと別にデバイスを作成する必要はありません。**pcmk_host_map** オプションを使用すると、ノードとポートのマッピングを定義できます。たとえば、次のコマンドは **west-apc** という APC 電源スイッチを使用し、ノード **west-13** に 15 番ポートを使用する **myapc-west-13** という名前のフェンスデバイスを 1 つ作成します。

```
# pcs stonith create myapc-west-13 fence_apc pcmk_host_list="west-13" ipaddr="west-apc"
login="apc" passwd="apc" port="15"
```

以下の例は、15 番ポートを使用するフェンスノード **west-13**、17 番ポートを使用するフェンスノード **west-14**、18 番ポートを使用するフェンスノード **west-15**、および 19 番ポートを使用するフェンスノード **west-16** に **west-apc** という APC 電源スイッチを使用します。

```
# pcs stonith create myapc fence_apc pcmk_host_list="west-13,west-14,west-15,west-16"
pcmk_host_map="west-13:15;west-14:17;west-15:18;west-16:19" ipaddr="west-apc" login="apc"
passwd="apc"
```

5.5. アンフェンスによるストレージベースのフェンスデバイスの設定

SAN やストレージフェンスデバイス (つまり電源ベース以外のフェンスエージェントを使用するフェンスデバイス) を作成する場合、**stonith** デバイスの作成時にメタオプション **provides=unfencing** を設定する必要があります。これにより、フェンスされたノードが再起動前にアンフェンスされ、クラスターサービスがそのノードで開始されます。

電源ベースのフェンスデバイスを設定する場合はデバイス自体がノードの起動 (およびクラスターへの再ジョイン試行) に電力を供給するため **provides=unfencing** メタオプションを設定する必要はありません。この場合の起動とはアンフェンスが発生してから起動するという意味です。

次のコマンドでは **fence_scsi** フェンスエージェントを使用する **my-scsi-shooter** という名前の stonith デバイスを設定、デバイスのアンフェンスを有効にしています。

```
pcs stonith create my-scsi-shooter fence_scsi devices=/dev/sda meta provides=unfencing
```

5.6. 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]
```

5.7. 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
```

5.8. フェンスデバイスが接続されているノードの管理

手作業でのノードの排他処理は次のコマンドで行うことができます。**--off** を使用すると stonith に対して **of** API コールが使用されノードは再起動ではなく電源オフになります。

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

次のコマンドで指定したノードの電源が現在オフになっているのかどうかを確認することができます。



注記

指定したノードがクラスターソフトウェアや通常クラスターで制御しているサービスを実行中だった場合はデータ破損やクラスター障害が発生するので注意してください。

```
pcs stonith confirm node
```

5.9. 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オプションを [表5.2「フェンスデバイスの高度なプロパティ」](#) にまとめています。その他のオプションは高度な設定を行う場合に限られます。

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	ポートの代わりに提供する代替のパラメーターです。デバイスによっては、標準のポートパラメーターをサポートしなかったり、追加のパラメーターを提供することがあります。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。クラスターが追加のパラメーターを提供しないようにするには、 none を値として使用します。

フィールド	タイプ	デフォルト	説明
pcmk_reboot_action	文字列	reboot	reboot の代わりに実行する代替のコマンドです。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、このパラメーターを使って再起動の動作を実行するデバイス固有の代替コマンドを指定します。
pcmk_reboot_timeout	時間	60s	stonith-timeout の代わりに再起動の動作に対して使用する代替タイムアウトです。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。再起動の動作に対してデバイス固有の代替タイムアウトを指定する場合に使用します。
pcmk_reboot_retries	整数	2	タイムアウト期間内で reboot コマンドを再試行させる最大回数です。複数接続に対応していないデバイスがあります。別のタスクでビジー状態になるとデバイスが動作に失敗する場合があるため、Pacemaker は残り時間内で動作を自動的に再試行させます。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。
pcmk_off_action	文字列	オフ	off の代わりに実行する代替コマンドです。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、このパラメーターを使ってオフの動作を実行するデバイス固有の代替コマンドを指定します。
pcmk_off_timeout	時間	60s	stonith-timeout の代わりにオフの動作に対して使用する代替タイムアウトです。オフ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オフするデバイスもあります。オフの動作に対してデバイス固有の代替タイムアウトを指定する場合に使用します。
pcmk_off_retries	整数	2	タイムアウト期間内で off コマンドを再試行させる最大回数です。複数接続に対応していないデバイスがあります。別のタスクでビジー状態になるとデバイスが動作に失敗する場合があるため、Pacemaker は残り時間内で動作を自動的に再試行させます。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。
pcmk_list_action	文字列	list	list の代わりに実行する代替のコマンドです。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、このパラメーターを使って list の動作を実行するデバイス固有の代替コマンドを指定します。

フィールド	タイプ	デフォルト	説明
pcmk_list_timeout	時間	60s	stonith-timeout 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。list の完了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 list するデバイスもあります。list の動作に対してデバイス固有の代替タイムアウトを指定する場合に使用します。
pcmk_list_retries	整数	2	タイムアウト期間内で list コマンドを再試行させる最大回数です。複数接続に対応していないデバイスがあります。別のタスクでビジー状態になるとデバイスが動作に失敗する場合があるため、Pacemaker は残り時間内で動作を自動的に再試行させます。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。
pcmk_monitor_action	文字列	monitor	monitor の代わりに実行する代替のコマンドです。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、このパラメーターを使ってモニタリングの動作を実行するデバイス固有の代替コマンドを指定します。
pcmk_monitor_timeout	時間	60s	stonith-timeout の代わりにモニターの動作に対して使用する代替タイムアウトです。モニター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オフするデバイスもあります。モニターの動作に対してデバイス固有の代替タイムアウトを指定する場合に使用します。
pcmk_monitor_retries	整数	2	タイムアウト期間内で monitor コマンドを再試行させる最大回数です。複数接続に対応していないデバイスがあります。別のタスクでビジー状態になるとデバイスが動作に失敗する場合があるため、Pacemaker は残り時間内で動作を自動的に再試行させます。Pacemaker によるモニター動作の再試行回数を変更する場合に使用します。
pcmk_status_action	文字列	status	status の代わりに実行する代替のコマンドです。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、このパラメーターを使って status の動作を実行するデバイス固有の代替コマンドを指定します。
pcmk_status_timeout	時間	60s	stonith-timeout の代わりに status の動作に対して使用する代替タイムアウトです。status 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オフするデバイスもあります。status の動作に対してデバイス固有の代替タイムアウトを指定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
<code>pcmk_status_retries</code>	整数	2	タイムアウト期間内で <code>status</code> コマンドを再試行させる最大回数です。複数接続に対応していないデバイスがあります。別のタスクでビジー状態になるとデバイスが動作に失敗する場合があるため、Pacemaker は残り時間内で動作を自動的に再試行させます。Pacemaker による <code>status</code> 動作の再試行回数を変更する場合に使用します。

5.10. 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のノードのフェンシングをサポートします。トポロジを実装するには、通常どおりに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 1 つ以上のフェンスレベルを定義します。

- ※ 各レベルは 1 から昇順で試行されていきます。
- ※ 任意のフェンスデバイスに障害が発生すると、現行レベルの排他処理は終了します。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排他処理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ません。
- ※ 任意のレベルで成功するまたはすべてのレベルが試行される (失敗する) と動作は終了します。

ノードにフェンスレベルを追加する場合に次のコマンドを使用します。複数デバイスの指定は `stonith ID` を使ってコンマで区切ります。指定されたデバイスが指定され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、**my_ilo** という ilo フェンスデバイスと **my_apc** という apc フェンスデバイスの 2 つがノード **rh7-2** に対して設定されています。これらのコマンドはフェンスレベルを設定します。デバイス **my_ilo** に障害が発生し、ノードをフェンスできない場合、Pacemaker はデバイス **my_apc** の使用を試みます。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
```

以下のコマンドは指定され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。指定されたノードやデバイスがない場合、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]
```

次のコマンドを使用すると指定ノードや `stonith id` のフェンスレベルが消去されます。ノードや `stonith id` を指定しないと全フェンスレベルが消去されます。

```
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。空白は入れないでください。例を示します。

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

次のコマンドで定義したフェンスデバイスとノードが実際に存在しているか確認します。

```
pcs stonith level verify
```

5.11. 冗長電源のフェンシング設定

冗長電源のフェンシングを設定する場合、ホストを再起動するときに両方の電源をオフにしてからどちらかの電源をオンにするよう、クラスターによる確認が必要になります。

ノードの電源が完全にオフにならなかった場合、ノードがリソースを解放しないことがあります。この場合、ノードがこのようなリソースに同時にアクセスし、リソースを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
```

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|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]] [--disabled]
```

--group オプションを指定すると、リソースはその名前のリソースグループへ追加されます。指定のグループが存在しない場合、グループが作成され、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すでに存在するリソースを基準として、追加されたリソースの相対的な位置を指定します。

--disabled オプションを指定すると、リソースが自動的に起動されません。

以下のコマンドを実行すると **VirtualIP** という名前のリソースが作成され、標準 **ocf**、**heartbeat** プロバイダー、および **IPAddr2** タイプが指定されます。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。標準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使用します。

```
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については [「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については [「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については [「リソースの動作」](#) を参照してください。

- ※ **--clone** を指定するとクローンリソースが作成されます。**--master** を指定するとマスター/スレーブ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章高度なリソースタイプ](#)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」](#)に詳細を示します。

表6.1 リソースのプロパティ

フィールド	説明
resource_id	Your name for the resource
standard	The standard the script conforms to. Allowed values: ocf , service , upstart , systemd , lsb , stonith
type	The name of the Resource Agent you wish to use, for example IPaddr or Filesystem
provider	The OCF spec allows multiple vendors to supply the same resource agent. Most of the agents shipped by Red Hat use heartbeat as the provider.

利用可能なリソースプロパティを表示するコマンドは [表6.2「リソースプロパティを表示させるコマンド」](#) を参照してください。

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
pcs resource list	利用できる全リソースの一覧を表示
pcs resource standards	利用できるリソースエージェントの標準を表示
pcs resource providers	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
pcs resource list string	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、標準名、プロバイダー名、タイプ名などでフィルターした一覧を表示させることが可能

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

リソース固有のパラメーターの他にも追加のリソースオプションを設定することができます。オプションはリソースの動作を決定するためクラスターによって使用されます。[表6.3「リソースのメタオプション」](#)に詳細を示します。

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
priority	0	If not all resources can be active, the cluster will stop lower priority resources in order to keep higher priority ones active.
target-role	Started	維持するリソースの状態、使用できる値: * Stopped - リソースの強制停止 * Started - 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) * Master - Allow the resource to be started and, if appropriate, promoted
is-managed	true	Is the cluster allowed to start and stop the resource? Allowed values: true , false
resource-stickiness	0	Value to indicate how much the resource prefers to stay where it is.
requires	Calculated	リソースの起動を許可する条件 以下の条件の場合を除き fencing にデフォルト設定、可能な値: * nothing - クラスターによるリソースの起動を常に許可 * quorum - 設定されているノードの過半数がアクティブな場合にのみクラスターはこのリソースを起動可能、 stonith-enabled が false に設定されている場合またはリソースの standard が stonith の場合はこのオプションがデフォルト値となる * fencing - 設定されているノードの過半数がアクティブで 且つ 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 * unfencing - The cluster can only start this resource if a majority of the configured nodes are active and any failed or unknown nodes have been powered off and only on nodes that have been unfenced . This is the default value if the provides=unfencing stonith meta option has been set for a fencing device. For information on the provides=unfencing stonith meta option, see 「アンフェンスによるストレージベースのフェンスデバイスの設定」 .

フィールド	デフォルト	説明
migration-threshold	INFINITY (無効)	How many failures may occur for this resource on a node, before this node is marked ineligible to host this resource. For information on configuring the migration-threshold option, refer to 「障害発生によるリソースの移動」 .
failure-timeout	0 (無効)	Used in conjunction with the migration-threshold option, indicates how many seconds to wait before acting as if the failure had not occurred, and potentially allowing the resource back to the node on which it failed. For information on configuring the failure-timeout option, refer to 「障害発生によるリソースの移動」 .
multiple-active	stop_start	リソースが複数のノードで実行していることが検出された場合にクラスターに行わせる動作、使用できる値: * block - リソースに unmanaged のマークを付ける * stop_only - 実行しているインスタンスをすべて停止する (これ以上の動作は行わない) * stop_start - 実行中のインスタンスをすべて停止してから一ヶ所でのみ起動する

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
```

例えば、次のコマンドでは **resource-stickiness** のデフォルト値を 100 にリセットしています。

```
# 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のデフォルト値をリセットしていたかいないかによって、リソース作成の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することができます。リソースのメタオプションの値を指定する時に使用する **pcs resource create** コマンドの形式を以下に示します。

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

例えば、次のコマンドでは **resource-stickiness** の値を 50 に設定したリソースを作成しています。

```
# 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
```

次の例では、既存の **dummy_resource** というリソースに **failure-timeout** メタオプションの値を 20 秒に設定しているコマンドの例を示します。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

一緒に配置され、順番に起動され、逆順で停止される必要のあるリソースのセットは、クラスターの最も一般的な要素の 1 つです。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、グループに含まれるリソースを指定します。グループが存在しない場合は、このコマンドによってグループが作成されます。グループが存在する場合は、このコマンドによって追加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され、その逆順で停止されます。

```
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** および **--after** オプションを使用すると、リソースグループにすでに存在するリソースを基準として、追加されたリソースの相対的な位置を指定できます。

また、以下のコマンドを使用するとリソースの作成時に新しいリソースを既存グループへ追加できます。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すべて表示します。

```
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 **shortcut** というリソースグループが作成されます。

```
# 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次のとおりです。

- ※ 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)。
- ※ リソースは指定された順序の逆順で停止されます (**Email** が停止された後、**IPAddr** が停止されます)。

グループのリソースの 1 つを実行できない場合、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** には影響しません。

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**、**target-role**、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)

Stickiness はリソースが現在の場所に留まりたい度合いを示し、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 値の合計は、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、グループ全体のスコアは 500 になります。

6.6. リソースの動作

リソースに健全性を維持させるためリソースの定義にモニタリングの動作を追加することができます。モニタ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リングの動作を作成します。モニタリングの間隔はリソースエージェントで確定されます。リソースエージェントでデフォルトのモニタリング間隔が提供されない場合は **pcs** コマンドにより 60 秒間隔のモニタリング動作が作成されます。

[表6.4「動作のプロパティ」](#)にリソースのモニタリング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
id	Unique name for the action. The system assigns this when you configure an operation.
name	The action to perform. Common values: monitor , start , stop
interval	How frequently (in seconds) to perform the operation. Default value: 0 , meaning never.

フィールド	説明
timeout	How long to wait before declaring the action has failed. If you find that your system includes a resource that takes a long time to start or stop or perform a non-recurring monitor action at startup, and requires more time than the system allows before declaring that the start action has failed, you can increase this value from the default of 20 or the value of timeout in "op defaults".
on-fail	<p>この動作が失敗した場合に実行する動作、使用できる値:</p> <ul style="list-style-type: none"> * ignore - リソースが失敗していなかったように振る舞う * block - リソースでこれ以上、一切の動作を行わない * stop - リソースを停止して別の場所で起動しない * restart - リソースを停止してから再起動する (おそらく別の場所) * fence - 失敗したリソースがあるノードを STONITH する * standby - 失敗したリソースがあるノード上のすべてのリソースを移動させる <p>The default for the stop operation is fence when STONITH is enabled and block otherwise. All other operations default to restart.</p>
enabled	If false , the operation is treated as if it does not exist. Allowed values: true , false

次のコマンドでリソースを作成するとモニタリングの動作を設定することができます。

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

例えば、次のコマンドはモニタリング動作付きの **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。モニタリング動作は 30 秒毎に実施されます。

```
# 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モニタリング動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の動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。

モニタリングオプションの値を変更には、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動作を作成します。

```
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 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)
```

モニタリング動作にグローバルのデフォルト値を設定する場合は次のコマンドを使用します。

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

たとえば、以下のコマンドはすべてのモニタリング操作に対して **timeout** 値のグローバルデフォルトを 24 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、以下のコマンドはクラスターのモニタリング操作のデフォルト値を表示します。この例では **timeout** 値は 240 秒に設定されています。

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

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全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
```

設定されたすべてのリソースのリストおよびこれらのリソースに対して設定されたパラメータを表示するには、以下の例のように **pcs resource show** コマンドの **--full** オプションを使用します。

```
# pcs resource show --full
```

```
Resource: VirtualIP (type=IPAddr2 class=ocf provider=heartbeat)
Attributes: ip=192.168.0.120 cidr_netmask=24
Operations: monitor interval=30s
Resource: WebSite (type=apache class=ocf provider=heartbeat)
Attributes: statusurl=http://localhost/server-status configfile=/etc/httpd/conf/httpd.conf
Operations: monitor interval=1min
```

設定されているリソースの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
```

例えば、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
```

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]
```

設定されているリソース **VirtualIP** のパラメーターの値を示すコマンドと初期値を表示している出力、**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表示している出力を以下に示します。

```
# 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
# 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. 複数のモニタリング動作

リソースエージェントが対応する範囲で一つのリソースに複数のモニタリング動作を設定することができます。これにより毎分、表面的なヘルスチェックを行ったり、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

複数のモニタリング動作を設定する場合は 2 種類の動作が同じ間隔で実行されないよう注意してください。

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。

例えば、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 秒のモニタリング動作が作成されます。

```
# pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.0.99 cidr_netmask=24 nic=eth2
```

仮想 IP で深さ 10 の異なるチェックに対応する場合は、次のコマンドを発行すると Packemaker で 10 秒間隔の通常仮想 IP チェックの他に 60 秒間隔の高度なモニタリングチェックが実行されるようになります。(説明されているように 10 秒間隔の追加モニタリング動作は設定しないでください。)

```
# 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無効にします。

```
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。

第7章 リソースの制約

リソースの制約を設定することでクラスター内のそのリソースの動作を決めることができます。設定できる制約は以下のカテゴリーになります。

- ※ **location** 制約 — 場所の制約はリソースを実行できるノードを決めます。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。
- ※ **colocation** 制約 — コロケーションの制約は他のリソースと相対的となるリソースの配置先を決めます。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。

複数リソース一式を一緒に配置、それらを順番に起動させ、また逆順で停止させるため複数の制約を設定する場合、その簡易な方法として Pacemaker ではリソースグループという概念に対応しています。リソースグループについては「[リソースグループ](#)」を参照してください。

7.1. 場所の制約

場所の制約ではリソースを実行させるノードを指定します。場所の制約を設定することで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のリソースの実行を避けるなどの指定を行うことができます。

[表7.1「場所の制約オプション」](#)では場所の制約を設定する場合のオプションについて簡単に示します。

表7.1 場所の制約オプション

フィールド	説明
rsc	リソース名
node	ノード名
score	優先度を示す値、任意のノードでのリソースの実行を優先または避ける INFINITY の値は「すべき」から「しなければならない」に変化、 INFINITY はリソースの場所制約のデフォルト score 値

フィールド	説明
resource-discovery	<p>指定のリソースに対して Pacemaker がこのノードでリソース検出を実行する優先度を示す値。リソースが物理的に稼働可能なノードのサブセットへの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することができます。pacemaker_remote を使用してノード数を100 単位のノード数に拡大する場合にこのオプションを考慮するとよいでしょう。指定可能な値には以下が含まれます。</p> <p>always: このノードの指定されたリソースに対して常にリソース検出を行います。</p> <p>never: このノードの指定されたリソースに対してリソース検出をまったく行いません。</p> <p>exclusive: このノード (および exclusive と同様にマーク付けされた他のノード) で指定されたリソースのみに対してリソースの検出を行います。異なるノードにまたがった同じリソースの exclusive 検出を使用する複数の場所制約によって、resource-discovery が排他的なノードのサブセットが作成されます。1 つ以上のノードでリソースが exclusive 検出に対してマーク付けされた場合、リソースはノードのサブセット内のみに置くことが可能です。</p> <p>このオプションを never または exclusive に設定すると、クラスターが認識しなくてもリソースがこれらの場所でアクティブになる可能性があります。この場合、クラスターが制御できる範囲外でサービスが開始されると (systemd または管理者による開始など)、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たときに resource-discovery プロパティが変更された場合や、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 プロパティが変更された場合にも、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、9 個以上のノードがあり、リソースを特定の場所のみで実行できる場合 (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、適切なオプションになります。</p> <p>always はリソースの場所制約におけるデフォルトの resource-discovery 値です。</p>

次のコマンドはリソースが指定ノードで優先して実行される場所の制約を作成します。

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

次のコマンドはリソースが指定ノードを避けて実行される場所の制約を作成します。

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

リソースの実行を許可するノード指定には上記以外にも 2 種類の方法があります。

- ※ オプトインクラスター — クラスターを設定し、デフォルトではいずれのノードでもリソース実行を許可せず、特定のリソース用に選択的に許可ノードを有効にします。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 — クラスターを設定し、デフォルトでは全ノードでリソース実行を許可してから、特定ノードでの実行を許可しない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。

オプトインまたはオプトアウトクラスターの設定を選択するべきかどうかは、個人的な好みとクラスターの構成によります。大多数のリソースを大多数のノードで実行できる場合、オプトアウトの設定は簡単になる傾向にあります。ほとんどのリソースをノードの小さなサブセット上でのみ実行できる場合はオプトインの設定のほうが簡単になる傾向にあります。

7.1.1. 「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** に設定してデフォルトではリソースの実行をいずれのノードでも許可しないようにします。

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

リソースごとにノードを有効にします。次のコマンドは場所の制約を設定するため、**Webserver** リソースは **example-1** ノードでの実行を優先させ、**Database** リソースは **example-2** ノードでの実行を優先させるようになります。また、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することができます。

```
# 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
```

7.1.2. 「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場合はクラスタープロパティ **symmetric-cluster** を **true** に設定してデフォルトではリソースの実行をすべてのノードに許可します。

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

次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 score 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することができます。

```
# 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
```

上記コマンドでは score に INFINITY を指定する必要はありません。INFINITY が score のデフォルト値になります。

7.2. 順序の制約

順序の制約はリソースの実行順序を指定します。順序の制約を設定することでリソースの起動と停止の順序を指定することができます。

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7.2「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7.2 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>リソースで行う動作、<i>action</i> プロパティで利用できる値:</p> <ul style="list-style-type: none"> * start - リソースを起動する * stop - リソースを停止する * promote - 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 * demote - 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 <p>動作を指定しないとデフォルトの動作 start が設定されます。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 「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。</p>
kind オプション	<p>制約の実施方法、kind オプションで利用できる値:</p> <ul style="list-style-type: none"> * Optional - 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。順序は 「勧告的な順序付け」 を参照してください。 * Mandatory - 常に実施 (デフォルト値)、1 番目に指定したリソースが停止しているまたは起動できない場合、2 番目に指定されているリソースを停止しなければなりません (「強制的な順序付け」 を参照) * Serialize - 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
symmetrical オプション	If true, which is the default, stop the resources in the reverse order. Default value: true

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 (実行していれば)。
- ※ 1 番目に指定しているリソースが実行されていない状態でまた起動できない場合には 2 番目に指定しているリソースが停止されます (実行していれば)。
- ※ 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。

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。最初に指定するリソースによる状態の変更は 2 番目に指定するリソースに影響を与えません。

次のコマンドは **VirtualIP** というリソースと **dummy_resource** というリソースに勧告的な順序付け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、管理者は順序付けされたリソースのチェーンを作成します (例: リソース C の前にリソース B が開始し、リソース B の前にリソース A が開始)。設定により、順番にコロケートおよび開始されるリソースのセットを作成する必要がある場合、[「リソースグループ」](#)の説明にしたがってこれらのリソースが含まれるリソースグループを設定できます。しかし、リソースが順番に開始されるよう設定を行う必要があり、それらのリソースはコロケートされる必要がない場合、**pcs constraint order set** コマンドを使用してリソースのセットに順序付け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: true** または **false** に設定でき、リソースのセットが相対的に順序付けされる必要があるかどうかを示します。

sequential を **false** に設定すると、順序付けの制約で他のセットに対して相対的に順序付けすることができますが、セットのメンバーは相対的に順序付けされません。そのため、このオプションは制約に複数のセットがリストされている場合のみ有効です。それ以外の場合、制約の効果はありません。
- ✧ **require-all: true** または **false** を設定でき、続行する前にセットのすべてのリソースがアクティブである必要があるかどうかを示します。**require-all** を **false** に設定した場合、次のセットに継続する前にセットの 1 つのリソースのみが開始される必要があります。**require-all** を **false** に設定した場合、**sequential** が **false** に設定された順序付けされていないセットと併用しないと効果がありません。
- ✧ **action:** [表7.2「順序の制約のプロパティ」](#)の説明どおり、**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7.3「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と仮定した場合、次のコマンドはこの 3 つのリソースを順序付けされたひとつの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を削除します。

```
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の制約を作成する場合は重要な副作用がある点に注意してください。ノードにリソースを割り当てる順序に影響します。つまり、リソース B の場所がわからないとリソース B に相対的となるようリソース A を配置することはできません。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 に対してコロケートするのか、リソース B をリソース A に対してコロケートするのかが重要となります。

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事項がもう一つあります。リソース A をリソース B に対してコロケートすると仮定した場合、クラスターはリソース B に選択するノードを決定する際、リソース A の優先傾向も考慮に入れます。

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、スレーブリソースの詳細は「[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。

[表7.3「コロケーション制約のプロパティ」](#)にコロケーション制約設定用のプロパティおよびオプションを示します。

表7.3 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、制約の条件を満たせない場合はこのリソースの実行を全く許可しないと判断されることがあります。
target_resource	コロケーションターゲット、このリソースの実行先が最初に決定されてからソースリソースの実行先が決定されます。
score	正の値を指定するとリソースは同じノードで実行され、負の値を指定するとリソースは同じノードで実行されません。デフォルト値である +INFINITY は source_resource が target_resource と同じノードで実行されなければならないことを示しています。- INFINITY は source_resource が target_resource と同じノードで実行され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 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には (理由の如何に関わらず) myresource1 の実行は許可されません。

また、逆の設定、つまり myresource1 が myresource2 と同じマシンでは実行できないよう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
```

-INFINITY を指定することで制約が結合しています。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には **myresource1** はいずれのノードでも実行できなくなります。

7.3.2. 勧告的な配置

必ず実行する場合や必ず実行しない場合が「強制的な配置」であれば「勧告的な配置」はある状況下で優先される場合を言います。制約のスコアが **-INFINITY** より大きく **INFINITY** より小さい場合、クラスターはユーザーの希望を優先しようとしませんが、クラスターリソースを一部停止することを希望する場合は無視します。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

設定により、順番にコロケートおよび開始されるリソースのセットを作成する必要がある場合、[「リソースグループ」](#)の説明にしたがってこれらのリソースが含まれるリソースグループを設定できます。しかし、リソースのセットをコロケートする必要がある、順番に起動する必要はない場合、**pcs constraint colocation set** コマンドを使用してリソースのセットに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: true** または **false** に設定でき、セットのメンバーがお互いをコロケートする必要があるかどうかを示します。

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後にリストされている他のセットとコロケートすることができます。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リストされている場合のみ有効です。それ以外の場合、制約の効果はありません。

- ✱ **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7.2「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。デフォルト値は**true**で、リソースを逆順で停止します。
- ✱ **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場合、各メンバーはその前のメンバーとコロケートされます。たとえば、「set A B」は B が A とコロケート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コロケートされます。たとえば、「set C D sequential=false set A B」は C と D のセット (C と D との関係がない) が A と B のセット (B は A とコロケートされる) とコロケートされることを意味します。

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。

次のコマンドは現在の場所、順序、ロケーションの制約を表示します。

```
pcs constraint list|show
```

次のコマンドは現在の場所の制約を表示します。

- ※ **resources** を指定すると場所制約がリソースごとに表示されます。デフォルトの動作です。
- ※ **nodes** を指定すると場所制約がノードごとに表示されます。
- ※ 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]
```

次のコマンドは現在の全順序制約を表示します。 **--full** オプションを指定すると制約の内部 ID を表示します。

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

次のコマンドは現在の全コロケーション制約を表示します。 **--full** オプションを指定すると制約の内部 ID を表示します。

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

次のコマンドは特定リソースを参照する制約を表示します。

```
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動させることができます。次のような2タイプの状況が考えられます。

- ✦ ノードのメンテナンスのためそのノードで実行中の全リソースを別のノードに移動する必要がある
- ✦ 個別に指定されたリソースを移動する必要がある場合

ノードで実行中の全リソースを別のノードに移動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。

個別に指定されたリソースは以下の方法のいずれかで移動することができます。

- ✦ [「現在のノードからリソースを移動」](#) の説明にしたがって、**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、ノードの *resource_id* を定義どおりに指定します。移動するリソースを実行する移動先のノードを指定する場合は **destination_node** を指定します。

```
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、現在稼働しているノードで実行されないよう、リソースに制約が追加されます。**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。これら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わけではありません。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によって異なります。

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 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。ISO 8601 に定義された形式に従って **lifetime** パラメーターの単位を指定できます。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、5M の **lifetime** パラメーターは 5 カ月の間隔を示し、PT5M の **lifetime** パラメーターは 5 分の間隔を示します。

lifetime パラメーターは **cluster-recheck-interval** クラスタープロパティによって定義された間隔でチェックされます。デフォルト値は 15 分です。このパラメーターを頻繁にチェックする必要がある場合、以下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 ban** コマンドに **--wait[=n]** パラメーターを設定し、移動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) を指定することができます。待機時間がこの値を超えると、リソースが起動した場合は 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タイムアウト値が使用されます。

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

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

リソースの制約の詳細は [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動によって、リソースが移動された後、フェイルオーバーの原因となった状況が改善されてもそのリソースは必ずしも元のノードに戻るわけではありません。リソースを優先ノードへ移動するには、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、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動されます。

このコマンドは resource stickiness を無視して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、リソースが優先ノードへ移動する原因となる場所の制約を作成します。リソースが移動した後、制約は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、リソースに **migration-threshold** オプションをセットし、セットした回数の障害が発生するとリソースが新しいノードに移動されるよう設定することができます。このしきい値に一旦達してしまうと、このノードは障害が発生したリソースを実行できなくなります。解除には以下が必要になります。

- ※ 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

デフォルトではしきい値は定義されません。



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リソースの移行設定とは異なります。

次の例では **dummy_resource** というリソースに移行しきい値 10 を追加しています。この場合、障害が 10 回発生するとそのリソースは新しいノードに移動されます。

```
# 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することができます。

```
# 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)、起動の失敗によって **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。

停止時の失敗は起動時とは若干異なり重大です。リソースの停止に失敗し STONITH が有効になっている場合、リソースを別のノードで起動できるようクラスターによるノードの排他処理が行われます。STONITH を有効にしていない場合に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、障害タイムアウト後に停止が再度試行されます。

8.3. 接続状態変更によるリソースの移動

以下の 2 つのステップにしたがって、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) 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ための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」](#) では **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
-------	----

フィールド	説明
dampen	The time to wait (dampening) for further changes to occur. This prevents a resource from bouncing around the cluster when cluster nodes notice the loss of connectivity at slightly different times.
multiplier	The number of connected ping nodes gets multiplied by this value to get a score. Useful when there are multiple ping nodes configured.
host_list	The machines to contact in order to determine the current connectivity status. Allowed values include resolvable DNS host names, IPv4 and IPv6 addresses. The entries in the host list are space separated.

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、**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、**Webserver** という既存のリソースの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されているホストが **www.example.com** へ ping できない場合に、**Webserver** リソースを **www.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作業で停止し、クラスターが再起動しないようにするには、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 30 秒間 (または指定した秒間) 待機します。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。

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

クラスターによるリソースの起動を許可する場合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起動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** はリソースの停止を最長で 30 秒間 (または「*n*」を使って秒数を指定する) 待ってから、リソースが停止した場合には 0、リソースが停止しなかった場合には 1 を返します。

```
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。これら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わけではありません。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** コマンドの **lifetime** パラメーターを設定すると、制約を維持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、移動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) を指定することができます。待機時間がこの値を超えると、リソースが起動した場合は 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。クラスターの推奨は無視され強制起動しているリソースからの出力を表示します。このコマンドは主にリソースをデバッグする際に使用されます。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** コマンドを使った起動は行われません。リソースが起動しない場合、大抵はリソースが誤って設定されている、リソースが起動しないよう制約が設定されている、リソースが無効になっているのいずれかが原因です。このような場合に、このコマンドを使ってリソースの設定をリストアップすることができます。ただし、クラスター内でのリソースの通常起動には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

モニタリングが再実行されないようにする最も簡単な方法はモニタリングを削除することです。しかし、場合によっては一時的に無効にしたいときがあります。このような場合は **enabled="false"** を操作の定義に追加します。モニタ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。

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 **unmanaged** 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 **managed** 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。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のリソースすべてを **managed** または **unmanaged** モードに設定し、リソースを個別に管理できます。

第9章 高度なリソースタイプ

本章では Pacemaker で対応している高度なリソースタイプについて説明しています。

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、クローンしたリソースを使用して IP リソースの複数のインスタンスを設定し、クラスター全体で分散することができます。リソースエージェントがサポートするリソースはすべてクローンできます。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構成されます。

注記

クローンに適しているのは同時に複数のノードで実行することができるリソースのみです。たとえば、共有ストレージデバイスから **ext4** などのクラスター化していないファイルシステムをマウントする **Filesystem** リソースなど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。

リソースグループの作成とそのリソースグループのクローン作成は一つのコマンドではできません。

作成済みリソースまたはリソースグループのクローンは次のコマンドで作成できます。

```
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一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、その名前はリソース名に **-clone** を付けた名前が付けられます。次のコマンドではタイプが **apache** の **webfarm** というリソースとそのクローンとして **webfarm-clone** というリソースを作成します。

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

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については「[リソースの作成](#)」を参照してください。

クローンのリソースに指定できるオプションを[表9.1「クローンのリソース用オプション」](#)に示します。

表9.1 クローンのリソース用オプション

フィールド	説明
priority, target-role, is-managed	表6.3「リソースのメタオプション」 の説明どおり、クローンされたリソースから継承されるオプション。
clone-max	How many copies of the resource to start. Defaults to the number of nodes in the cluster.
clone-node-max	How many copies of the resource can be started on a single node; the default value is 1 .
notify	When stopping or starting a copy of the clone, tell all the other copies beforehand and when the action was successful. Allowed values: false , true . The default value is false .
globally-unique	各クローンのコピーに異なる機能を行わせるかどうか、使用できる値: false 、 true このオプションの値が false の場合はリソースが実行しているいずれのノードであってもすべて同じ動作を行うため、1 台のマシンごと実行できるクローンのコピーは 1 つ このオプションの値が true の場合は任意のマシンで実行中のクローンのコピーは別のマシンまたは同じマシンで実行している他のコピーとは同じにならない、 clone-node-max の値が「1」より大きい場合にはデフォルト値は true になり、これ以外の場合は false がデフォルト値になる
ordered	Should the copies be started in series (instead of in parallel). Allowed values: false , true . The default value is false .
interleave	Changes the behavior of ordering constraints (between clones/masters) so that copies of the first clone can start or stop as soon as the copy on the same node of the second clone has started or stopped (rather than waiting until every instance of the second clone has started or stopped). Allowed values: false , true . The default value is false .

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してクローンのコピーはひとつです。ただし、**clone max** にはクラスター内のノード合計数より小さい数をリソースのクローン数の値として設定することができます。この場合、リソースの場所制約を付けたコピーを優先的に割り当てるノードを示すことができます。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。

次のコマンドでは場所の制約を作成し、リソースのクローン **webfarm-clone** が **node1** に優先的に割り当てられるようにしています。

```
# 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、開始される必要がある **webfarm-clone** のコピーがすべて開始されるまで待機した後に **webfarm-stats** が開始されます。**webfarm-clone** のコピーを1つも開始できない場合、**webfarm-stats** はアクティブになりません。さらに、**webfarm-stats** が停止されるまで待機してから **webfarm-clone** が停止されます。

```
# 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) をクローンとコロケートすると、そのリソースはクローンの複数コピーが実行されている複数マシンいずれでも実行できるということになります。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。

クローン同士で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されるようにします。

```
# 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** に値を与えないとクローンは 1 の値を使用します。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 によるクラスターノード内での不必要なコピーの移動を適切に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。多状態リソースにより、インスタンスは **Master** および **Slave** の 2 つの操作モードのいずれかになることができます。モードの名前に特別な意味はありませんが、インスタンスの起動時に **Slave** 状態にならないとなければならない制限があります。

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 [meta 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。

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については「[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
id	多状態リソースに付ける名前
priority、target-role、is-managed	表6.3「リソースのメタオプション」 を参照
clone-max、clone-node-max、notify、globally-unique、ordered、interleave	表9.1「クローンのリソース用オプション」 を参照
master-max	How many copies of the resource can be promoted to master status; default 1.
master-node-max	How many copies of the resource can be promoted to master status on a single node; default 1.

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、11 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、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ています。

```
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を以下に示します。

```
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)

To achieve a stable allocation pattern, multistate resources are slightly sticky by default. If no value for **resource-stickiness** is provided, the multistate resource will use a value of 1. Being a small value, it causes minimal disturbance to the score calculations of other resources but is enough to prevent Pacemaker from needlessly moving copies around the cluster.

9.3. `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、それらのノードのリソースはクラスターによって実際のクラスターノードと同様に管理されます。

`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 `corosync` の 16 ノード制限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。

- ※ **cluster node**: 高可用性サービスを実行しているノード (`pacemaker` および `corosync`)。
- ※ **remote node**: `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ために `pacemaker_remote` を実行しているノード。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。
- ※ **guest node**: `pacemaker_remote` サービスを実行している仮想ゲストノード。ゲストノードは、`ocf:pacemaker:VirtualDomain` などのリソースエージェントの `remote-node` メタデータオプションを使用して設定されます。仮想ゲストリソースはクラスターによって管理されます。仮想ゲストリソースはクラスターによって開始され、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`) はリモートノード上で `pacemaker_remote` サービスに接続するため、クラスターに統合することができます。
- ※ 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、ゲストノード上で `pacemaker_remote` サービスに即座に接続するため、クラスターに統合することができます。

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。

- ※ クォーラムで実行されません
- ※ 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) にはなれません
- ※ `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。

前述の制限以外では、リモー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、リモートおよびゲストノード自体をフェンシングできます。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および監視できます。これらのノードに制約を作成したり、これらのノードをスタンバイ状態にすることができます。また、**pcs** コマンドを使用してクラスターノードでその他のアクションを実行することもできます。リモート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状態の出力に表示されます。

9.3.1. ホストとゲストの認証

クラスターノードと `pacemaker_remote` 間の接続は TLS (Transport Layer Security) が使用され、PSK (Pre-Shared Key) の暗号化と TCP 上の認証 (デフォルトでポート 3121 を使用) でセキュア化されます。そのため、**pacemaker_remote** を実行しているクラスターノードおよびノードは同じプライベートキー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。

9.3.2. 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を表示するには、次のコマンドを実行します。

```
# pcs resource describe VirtualDomain
```

VirtualDomain リソースオプションの他にも、メタデータオプションを設定してリソースをゲストノードとして有効にし、接続パラメーターを定義することができます。メタデータオプションの説明は [表 9.3「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」](#) を参照してください。

表9.3 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
remote-node	<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。警告: この値はリソースやノードの ID と重複してはなりません。
remote-port	3121	pacemaker_remote へのゲスト接続に使用するカスタムのポートを設定
remote-addr	ホスト名として使用される remote-node 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
remote-connect-timeout	60s	保留中のゲスト接続がタイムアウトするまでの時間

9.3.3. リモートノードリソースのオプション

リモートノードをクラスターリソースとして設定するには、**pcs resource create** コマンドを使用し、**ocf:pacemaker:remote** をリソースタイプとして指定します。**remote** リソースに設定できるリソースオプションの説明は [表9.4「リモートノードのリソースオプション」](#) を参照してください。

表9.4 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
-------	-------	----

フィールド	デフォルト	説明
reconnect_interval	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)。待機期間は繰り返されます。待機期間の後に再接続に失敗した場合、さらなる待機期間の後に再接続を試みます。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。
server		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。
port		接続する TCP ポート。

9.3.4. デフォルトの **pacemaker_remote** オプションの変更

デフォルトのポートまたは Pacemaker や **pacemaker_remote** いずれかの **authkey** の場所を変更する必要がある場合は、両方のデーモンに反映させることができる環境変数を設定することができます。以下のように **/etc/sysconfig/pacemaker** ファイルに配置するとこの環境変数を有効にすることができます。

```
#==#==# Pacemaker Remote
# Use a custom directory for finding the authkey.
PCMK_authkey_location=/etc/pacemaker/authkey
#
# Specify a custom port for Pacemaker Remote connections
PCMK_remote_port=3121
```

特定ノード (クラスターノード、ゲストノード、またはリモートノード) でのデフォルトのキーの場所を変更すると、そのノードの **PCMK_authkey_location** を設定できることに注意してください (またその場所にキーを格納できます)。すべてのノードで場所を同じにする必要はありませんが、同じにした方が管理が楽になります。

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)。

9.3.5. 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. 仮想化ソフトウェアをインストールし、クラスターノード上で **libvirt** サービスを有効にした後、すべてのクラスターノードと仮想マシンの **/etc/pacemaker/authkey** に同じ暗号化キーを保存します。これにより、通信と認証がセキュア化されます。

すべてのノードで以下のコマンドセットを実行し、セキュアなパーミッションを持つ **authkey** 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、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
```

- 各仮想マシンに **pacemaker_remote** パッケージをインストールし、**pacemaker_remote** サービスを開始します。**pacemaker_remote** が起動時に実行されるよう設定し、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
# systemctl start pacemaker_remote.service
# systemctl enable pacemaker_remote.service
# firewall-cmd --add-port 3121/tcp --permanent
# firewall-cmd --reload
```

- 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各仮想マシンに割り当てます。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。
- 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
```

- ゲストノードが実行されている場合はシャットダウンします。そのノードがクラスターで設定されると Pacemaker によって開始されます。
- VirtualDomain** リソースを作成し、**remote-note** リソースメタオプションを設定して仮想マシンはリソースの実行が可能なゲストノードであることを示します。

以下の例では、このリソースはクラスターに統合可能で、ホスト名が **guest1** のゲストノードであることを **remote-node=guest1** メタ属性が Pacemaker に伝えます。クラスターは開始後、ホスト名 **guest1** で仮想マシンの **pacemaker_remote** サービスに通信しようとしています。

クラスターノードから以下のコマンドを入力します。

```
# pcs resource create vm-guest1 VirtualDomain hypervisor="qemu:///system"
config="/virtual_machines/vm-guest1.xml" meta remote-node=guest1
```

- VirtualDomain** リソースの作成後、ゲス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、作成したリソースに対してリソース制約が作成され、ゲストノードで実行されるようになります。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
```

9.3.6. 設定の概要: リモートノード

本セクションでは、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または **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、さまざまなクラスタリングコンポーネントによって使用される TCP ポート 2224、3121、21064、および UDP ポート 5405 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、作成したリソースに対してリソース制約が作成され、リモートノードで実行されるようになります。

```
# 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様にフェンスされます。クラスターノードと同様に、リモートノードと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してはならないことに注意してください。他のノードに対してフェンス操作を実行できるのはクラスターノードのみです。

9.3.7. 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した。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。**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れば、**pcs resource enable** でリソース再度有効にします。

9.3.8. VM リソースのゲストノードへの変換

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。リソースが最初からゲストノードとして作成された場合、このコマンドを実行する必要はありません。

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

以下のコマンドを使用して、指定のノードでゲストノードとして設定されたリソースを無効にします。

```
pcs cluster remote-node remove hostname
```

第10章 クラスタークォーラム

Red Hat Enterprise Linux High Availability Add-On クラスターは **votequorum** サービスとフェンシングを併用してスプリットブレインが発生しないようにします。クラスターの各システムには投票数が割り当てられ、過半数の票がある場合のみクラスターの操作を続行できます。サービスはすべてのノードにロードするか、すべてのノードにロードしない必要があります。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、クォーラム設定の特別な機能を設定できます。これらのオプションは [表10.1「クォーラムオプション」](#) に記載されています。

表10.1 クォーラムオプション

オプション	説明
--auto_tie_breaker	<p>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、auto_tie_breaker_node に設定された nodeid (設定されていない場合は最小の nodeid) と通信する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>auto_tie_breaker オプションを指定すると、クラスターは均等の分割では操作を継続できるため、偶数個のノードを持つクラスターで使用されます。複数で不均等の分割など、より複雑な障害では「クォーラムデバイス (テクノロジープレビュー)」に説明されているクォーラムデバイスの使用が推奨されます。</p>
--wait_for_all	<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クラスターは初回だけクォーラムに達します。</p> <p>wait_for_all オプションは、主にクォーラムデバイスの lms (last man standing) アルゴリズムを使用する、2 ノードまたは偶数のノードで構成されるクラスターによって使用されます。</p> <p>wait_for_all 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、auto_tie_breaker が無効になっている場合に自動的に有効になります。wait_for_all 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>
--last_man_standing	<p>有効にすると、特定の状況でクラスターは動的に expected_votes とクォーラムを再計算します。このオプションを有効にするには wait_for_all を有効にする必要があります。</p>
--last_man_standing_window	<p>クラスターのノードが失われた後、expected_votes 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)。</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]
```

以下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、クラスターはクォーラムがなくても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作業で変更したことにより実行されているクラスターが 50% 未満となる場合、クラスターの他のノードを別々に起動でき、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、**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。

以下のコマンドは、ライブクラスターの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のみに影響し、設定ファイルは変更されません。リロードが発生した場合、**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。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を無効にし、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 (テクノロジープレビュー)



重要

クォーラムデバイスの機能はテクノロジープレビューとしてのみ提供されます。「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。

Red Hat Enterprise Linux 7.3 より、クラスター用のサードパーティーのアービトレーションデバイスとして機能する別のクォーラムデバイスを設定することが可能となりました。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、偶数のノードで構成されるクラスターに推奨され、2 ノード構成のクラスターには強く推奨されます。

クォーラムデバイスの設定時に以下を考慮する必要があります。

- ✧ 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異なる物理ネットワークでクォーラムデバイスを実行することが推奨されます。理想的には、クォーラムデバイスホストはメインクラスターとは別のラックに置くようにし、最低でも別の PSU に置くようにしてください。corosync リングと同じネットワークセグメントには置かないでください。
- ✧ 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が、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は異なるアル

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することができます。

- ※ 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。

- ※ クラスターノードで **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で **corosync-qnetd** をインストールします。

```
[root@qdevice:~]# yum install corosync-qnetd
```

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。以下の手順はクォーラムデバイスを設定し、クラスターに追加します。この例では以下を前提とします。

- ※ 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、現在唯一サポートされているモデルです。
- ※ 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、デバイスが起動時に開始されるよう設定します。

```
[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、その状態をチェックできます。**corosync-qnetd** デーモンが実行され、この時点では接続されているクライアントがないはずです。**--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. 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、クォーラムデバイスの現在の設定と状況をチェックして後で比較することができます。これら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

以下のコマンドは以前作成し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が、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。このコマンド例は、クォーラムデバイスが **lms** (last man standing) アルゴリズムを使用するよう設定します。クォーラムデバイスの設定オプションに関する情報は、**corosync-qdevice(8)** の man ページを参照してください。

```
[root@node1:~]# pcs quorum device add model net host=qdevice algorithm=lms
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
```

3. 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することができます。

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: lms
  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:       LMS
Tie-breaker:     Node with lowest node ID
State:           Connected
```

クォーラムデバイス側から以下のコマンドを実行すると、**corosync-qnetd** デーモンの状態を表示できます。

```
[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:        LMS
  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。これらのコマンドは **corosync-qnetd** サービスのみに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 **ffsplit** (fifty/fifty split) に変更します。

```
[root@node1:~]# pcs quorum device update model algorithm=ffsplit
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の削除後、クォーラムデバイスの状態を確認すると以下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化および停止し、設定ファイルをすべて削除するには、以下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

ルールを使用すると設定をより動的にすることができます。一般的な例としては、就業時間内に **resource-stickiness** の値を設定してリソースが最も優先される場所に戻されないようにし、出社する社員がいない週末に別の値を設定します。

これ以外にも、時間に応じて異なる処理グループにマシンを割り当て(ノード属性を使用)、場所の制約を作成する時にその属性を使用する方法もあります。

各ルールには日付の式など各種の式の他、他のルールも含ませることができます。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。ルールが使用された状況に応じて次に起こる動作は異なります。

表11.1 ルールのプロパティ

フィールド	説明
role	Limits the rule to apply only when the resource is in that role. Allowed values: Started , Slave , and Master . NOTE: A rule with role="Master" cannot determine the initial location of a clone instance. It will only affect which of the active instances will be promoted.
score	The score to apply if the rule evaluates to true . Limited to use in rules that are part of location constraints.
score-attribute	The node attribute to look up and use as a score if the rule evaluates to true . Limited to use in rules that are part of location constraints.
boolean-op	How to combine the result of multiple expression objects. Allowed values: and and or . The default value is and .

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ノード属性の式を使用します。

表11.2 式のプロパティ

フィールド	説明
value	User supplied value for comparison
attribute	The node attribute to test
type	Determines how the value(s) should be tested. Allowed values: string , integer , version

フィールド	説明
operation	<p>実行する比較動作、使用できる値:</p> <ul style="list-style-type: none"> * lt - ノード属性の値が value 未満の場合に True * gt - ノード属性の値が value を越える場合に True * lte - ノード属性の値が value 未満または同等になる場合に True * gte - 属性の値が value を越えるまたは同等になる場合に True * eq - ノード属性の値が value と同等になる場合に True * ne - ノード属性の値が value と同等ではない場合に True * defined - ノードに指定属性がある場合に True * not_defined - True if the node does not have the named attribute

11.2. 時刻と日付ベースの式

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日付の式を使用します。オプションで日付の詳細を含ませることができます。

表11.3 日付の式のプロパティ

フィールド	説明
start	A date/time conforming to the ISO8601 specification.
end	A date/time conforming to the ISO8601 specification.
operation	<p>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。使用できる値は次のとおりです。</p> <ul style="list-style-type: none"> * gt - 現在の日付と時刻が start 以降の場合は True * lt - 現在の日付と時刻が end 以前の場合は True * in-range - 現在の日付と時刻が start 以降で且つ end 以前の場合は True * date-spec - performs a cron-like comparison to the current date/time

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ために使用されます。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無視され、デフォルトの 0 としてみなされません。

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 時までの時間と一致しますが、複数の範囲が含まれる **weekdays="1,2"** や **weekdays="1-2,5-6"** は指定できません。

表11.4 日付詳細のプロパティ

フィールド	説明
id	A unique name for the date
hours	Allowed values: 0-23

フィールド	説明
monthdays	Allowed values: 0-31 (depending on month and year)
weekdays	Allowed values: 1-7 (1=Monday, 7=Sunday)
yeardays	Allowed values: 1-366 (depending on the year)
months	Allowed values: 1-12
weeks	Allowed values: 1-53 (depending on weekyear)
years	Year according the Gregorian calendar
weekyears	May differ from Gregorian years; for example, 2005-001 Ordinal is also 2005-01-01 Gregorian is also 2004-W53-6 Weekly
moon	Allowed values: 0-7 (0 is new, 4 is full moon).

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 を用いたルールの設定

ルールを設定する場合は次のコマンドを使用します。**score** を省略すると INFINITY にデフォルト設定されます。**id** を省略すると *constraint_id* で生成されます。**rule_type** は **expression** または **date_expression** のいずれかにしてください。

```
pcs constraint rule add constraint_id [rule_type] [score=score] [id=rule_id]
expression/date_expression/date_spec options
```

ルールを削除する場合は次のコマンドを使用します。削除す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ることになります。

```
pcs constraint rule remove rule_id
```

11.6. 時刻ベースの式のサンプル

次のコマンド設定の場合は 2005 年以内ならいつでも true となります。

```
# pcs constraint location Webserver rule score=INFINITY date-spec years=2005
```

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 日の金曜日が満月であると true になる式を設定します。

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

11.7. ルールを用いたリソースの場所の確定

ルールを使用してリソースの場所を特定

次のコマンドを使用するとルールを使ってリソースの場所を確定することができます。

```
pcs constraint location resource_id rule [rule_id] [role=master|slave] [score=score expression]
```

expression には以下のいずれかを使用します。

- ✱ **defined|not_defined *attribute***
- ✱ ***attribute* lt|gt|lte|gte|eq|ne *value***
- ✱ **date [*start=start*] [*end=end*] operation=gt|lt|in-range**
- ✱ **date-spec *date_spec_options***

第12章 Pacemaker クラスターのプロパティ

クラスター動作中に起こる可能性がある状況に直面した場合にクラスターのプロパティで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」](#) ではクラスターのプロパティオプションを説明します。
- ※ [「クラスターのプロパティの設定と削除」](#) ではクラスタープロパティの設定方法について説明します。
- ※ [「クラスタープロパティ設定のクエリー」](#) では現在設定されているクラスタープロパティを表示させる方法について説明します。

12.1. クラスタープロパティとオプションの要約

Pacemaker クラスターのプロパティのデフォルト値および設定可能な値などを [表12.1「クラスターのプロパティ」](#) で簡単に示します。

注記

表に記載しているプロパティ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があります。このようなプロパティについては、そのデフォルト値を別の値には変更しないよう推奨しています。

表12.1 クラスターのプロパティ

オプション	デフォルト	説明
batch-limit	30	The number of jobs that the transition engine (TE) is allowed to execute in parallel. The "correct" value will depend on the speed and load of your network and cluster nodes.
migration-limit	-1 (無制限)	The number of migration jobs that the TE is allowed to execute in parallel on a node.
no-quorum-policy	stop	What to do when the cluster does not have quorum. Allowed values: * ignore - 全リソースの管理を続行 * freeze - リソースの管理は続行するが、影響を受けるパーティション外のノードのリソースは復帰させない * 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 * 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排他処理する
symmetric-cluster	true	Indicates whether resources can run on any node by default.

オプション	デフォルト	説明
stonith-enabled	true	Indicates that failed nodes and nodes with resources that cannot be stopped should be fenced. Protecting your data requires that you set this true . true または未設定の場合、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る
stonith-action	reboot	Action to send to STONITH device. Allowed values: reboot , off . The value poweroff is also allowed, but is only used for legacy devices.
cluster-delay	60s	Round trip delay over the network (excluding action execution). The "correct" value will depend on the speed and load of your network and cluster nodes.
stop-orphan-resources	true	Indicates whether deleted resources should be stopped.
stop-orphan-actions	true	Indicates whether deleted actions should be canceled.
start-failure-is-fatal	true	Indicates whether a failure to start a resource on a particular node prevents further start attempts on that node. When set to false , the cluster will decide whether to try starting on the same node again based on the resource's current failure count and migration threshold. For information on setting the migration-threshold option for a resource, see 「障害発生によるリソースの移動」 .
pe-error-series-max	-1 (all)	The number of PE inputs resulting in ERRORS to save. Used when reporting problems.
pe-warn-series-max	-1 (all)	The number of PE inputs resulting in WARNINGS to save. Used when reporting problems.
pe-input-series-max	-1 (all)	The number of "normal" PE inputs to save. Used when reporting problems.
cluster-infrastructure		The messaging stack on which Pacemaker is currently running. Used for informational and diagnostic purposes; not user-configurable.
dc-version		Version of Pacemaker on the cluster's Designated Controller (DC). Used for diagnostic purposes; not user-configurable.
last-lrm-refresh		Last refresh of the Local Resource Manager, given in units of seconds since epoca. Used for diagnostic purposes; not user-configurable.
cluster-recheck-interval	15分	Polling interval for time-based changes to options, resource parameters and constraints. Allowed values: Zero disables polling, positive values are an interval in seconds (unless other SI units are specified, such as 5min).
default-action-timeout	20s	Timeout value for a Pacemaker action. The setting for an operation in a resource itself always takes precedence over the default value set as a cluster option.
maintenance-mode	false	Maintenance Mode tells the cluster to go to a "hands off" mode, and not start or stop any services until told otherwise. When maintenance mode is completed, the cluster does a sanity check of the current state of any services, and then stops or starts any that need it.
shutdown-escalation	20min	The time after which to give up trying to shut down gracefully and just exit. Advanced use only.
stonith-timeout	60s	How long to wait for a STONITH action to complete.

オプション	デフォルト	説明
stop-all-resources	false	Should the cluster stop all resources.
default-resource-stickiness	5000	Indicates how much a resource prefers to stay where it is. It is recommended that you set this value as a resource/operation default rather than as a cluster option.
is-managed-default	true	Indicates whether the cluster is allowed to start and stop a resource. It is recommended that you set this value as a resource/operation default rather than as a cluster option.
enable-acl	false	(Red Hat Enterprise Linux 7.1 and later) Indicates whether the cluster can use access control lists, as set with the pcs acl command.

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
```

例えば、**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。例えば、以前に **symmetric-cluster** プロパティを **false** に設定したことがある場合は、次のコマンドにより設定した値が削除され、**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
```

明示的には設定されていなかったプロパティ設定のデフォルト値も含め、クラスターのプロパティ設定のすべての値を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
```

例えば、**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
```

情報としてプロパティの全デフォルト値の一覧を表示させ、その値がデフォルト以外で設定されているかどうかを確認することができます。次のコマンドを使用します。

```
pcs property [list|show] --defaults
```


第13章 クラスターイベントのスキプトのトリガー

Pacemaker クラスター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 クラスターアラートを設定すると、クラスター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。クラスター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。これは推奨される方法で、クラスターアラートをより簡単に設定できます。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。**ClusterMon** リソースの詳細は [「モニタ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)

クラスター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。クラスター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。サポートされる属性は、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記述に関する情報は、[「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、必要性に見合うようスクリプトを確認してください。サンプルエージェントは、特定のクラスター環境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。

サンプルアラートエージェントの1つを使用するには、クラスター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、以下のコマンドは **alert_snmp.sh.sample** スクリプトを **alert_snmp.sh** としてインストールします。

```
# install --mode=0755 /usr/share/pacemaker/alerts/alert_snmp.sh.sample
/var/lib/pacemaker/alert_snmp.sh
```

スクリプトをインストールした後、そのスクリプトを使用するアラートを作成できます。以下の例は、インストールされた **alert_snmp.sh** アラートエージェントを使用するアラートを設定し、クラスターイベントを SNMP トラップとして送信します。デフォルトでは、正常なモニター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はメタオプションとして設定されま

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。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。

```
# 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 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、インストールされ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ーイベントを E メールメッセージとして送信します。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 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」](#) and [「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

以下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に関する情報は、[「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。この場合、クラスターは各イベントで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上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、これらのノード上では呼び出されません。

以下の例は、各イベントで **my-script.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

以下のコマンドは、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
```

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。よって、各アラートに 1 つ以上の受信側を追加設定できます。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、E メールアドレス、ファイル名など、エージェントがサポートし認識できるものになります。

以下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 recipient-value [id=recipient-id] [description=description] [options [option=value]...] [meta [meta-option=value]...]
```

以下のコマンドは、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]...]
```

以下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
```

以下のコマンド例は、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、**my-alert** に設定されたアラートスクリプトを各イベントで呼び出すよう、クラスターが設定されます。

```
# pcs alert recipient add my-alert 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。アラートメタオプションの説明は [表13.1「アラートメタオプション」](#) を参照してください。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
timestamp-format	%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。この文字列は date(1) コマンドと使用されます。
timeout	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、**my-script.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。1つの呼び出しは受信側 **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/my-script.sh meta timeout=15s
# pcs alert recipient add my-alert someuser@example.com id=my-alert-recipient1 meta
timestamp-format=%D %H:%M
# pcs alert recipient add my-alert 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。

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** が値となるアラート 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、受信側 ID の値として **alert-recipient** が使用されます。
- ※ 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 rec_value
# pcs alert recipient add alert 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 meta-option1=2 m=val
# pcs alert recipient add my-alert 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: m=val meta-option1=2
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 m=newval
# pcs alert recipient update my-alert-recipient options option1=new meta metaopt1=newopt
# 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: m=newval meta-option1=2
Recipients:
  Recipient: my-alert-recipient (value=my-other-recipient)
Options: option1=new
Meta options: metaopt1=newopt

```

以下のコマンドは、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
Options: opt=val option1=newvalue1
Meta options: m=newval meta-option1=2
Recipients:
  Recipient: my-alert-recipient (value=my-other-recipient)
Options: option1=new
Meta options: metaopt1=newopt

```

以下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る環境変数は、アラートの種類によって異なります。アラートエージェントに渡される環境変数と、特定のアラートの種類に関連する環境変数の説明は、[表13.2「アラートエージェントに渡される環境変数」](#)を参照してください。

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
CRM_alert_node_sequence	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。Pacemaker によってアラートが発行された順序を参照するために使用することができます。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確実に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。
CRM_alert_timestamp	エージェントの実行前に timestamp-format メタオプションによって指定された形式で作成されたタイムスタンプ。これにより、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って遅延される可能性があります) に関係なく、エージェントは信頼できる精度の高いイベント発生時間を使用できます。
CRM_alert_node	影響を受けるノードの名前
CRM_alert_desc	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 CRM_alert_status と同等の読み取り可能な文字列になります。
CRM_alert_nodeid	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)。
CRM_alert_task	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)。
CRM_alert_rc	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)。
CRM_alert_rsc	影響を受けるリソースの名前 (リソースアラートのみ)。
CRM_alert_interval	リソース操作の間隔 (リソースアラートのみ)
CRM_alert_target_rc	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)。
CRM_alert_status	操作の結果を示すために Pacemaker によって使用される数値コード (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とき、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側がない状態で呼び出されることがあります (受信側が設定されていない場合)。そのため、エージェントはこの状態に対応できなければなりません (このような状況では終了する場合でも)。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た場合、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実行できない場合、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。設定されたアラートと受信側の数や、アラートエージェント内で実行されることによって、負荷が急激に増加する可能性があります。リソースが集中するアクションを直接実行せず、他のインスタンスのキューに置くなど、これを考慮してエージェントを記述する必要があります。

- ※ 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されます。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**、および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ー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。設定エラーから保護する必要があります。また、クラスターへの **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、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** に設定されているリソースがクラスターに含まれる場合、障害時に複数のフェンス通知が送信されます (このパラメーターが設定されたリソースごとに 1 つの通知とそれに加えてさらに 1 つの通知)。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ースは、**ocf:pacemaker:ClusterMon** リソースによって使用される外部スキプトインターフェー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_** が付けられ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。**ClusterMon** リソースは root ユーザーとして実行し、アラートエージェントは **hacluster** ユーザーとして実行することがこの互換性の例外になります。**ClusterMon** によるスキプトの設定に関する情報は、[「モニタ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。フェンスイベントをリッスンできるようにするには、**ClusterMon** リソースの設定時に **--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、**extra_options** パラメーターを用いてクラスター通知の処理を判断します。[表13.3「外部監視プログラムへ渡される環境変数」](#)には、プログラムに渡される環境変数が記載されています。

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
CRM_notify_recipient	リソース定義からの静的な外部受信側。
CRM_notify_node	状態が変更したノード。
CRM_notify_rsc	状態を変更したリソースの名前。
CRM_notify_task	状態が変更する原因となった操作。
CRM_notify_desc	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。
CRM_notify_rc	操作の戻りコード。
CRM_target_rc	操作の予期される戻りコード。

環境変数	説明
CRM_notify_status	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** を指定すると、開始/停止/監視、開始/監視、フェンスリソースの停止など、リソースイベントの他にフェンスイベントの監視が有効になります。

```
# 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 (テクニカルレビュー)



重要

Booth チケットマネージャーはテクノロジーレビューとしてのみ提供されます。「テクノロジーレビュー」の意味については、[テクノロジーレビュー機能のサポート範囲](#)を参照してください。

クラスターが複数のサイトにまたがる場合、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。接続が切断されても、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、離れすぎているため同期を保つのが難しい2つのサイトで高可用性サービスを提供するのは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 7.3 以降には 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追加されました。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が必要になるよう、リソースを設定することができます。これにより、チケットが付与された場合にリソースは1度に1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、異なるサイトで実行されるクラスターで構成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してみなすことができます。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 が Pacemaker のチケット抑制を基にクラスターのリソースを実行するかどうかを判断します。チケットマネージャーを使用する場合、各クラスターは独自のリソースと共有リソースを実行できます。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るとします。リソース G および H は、チケットによって決定されたこれら2つのクラスターのいずれかで実行されます。また、別のチケットによってこれら2つのクラスターのいずれかで実行されることが決まるリソース 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構成の設定方法を示しています。

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** で構成されます。
- ※ 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** で構成されます。
- ※ 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。
- ※ arbitrator (調停役) 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、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の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、これがフェイ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. 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 arbitrator (調停役) 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。

2. Booth 設定のチケットを作成します。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ためにこのチケットを使用します。

このフェイルオーバー設定手順は基本的な手順で、1つのチケットのみを使用します。各チケットが異なる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
```

3. 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
```

4. arbitrator (調停役) ノードから、Booth 設定を arbitrator 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
```

5. 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。arbitrator 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
```

6. arbitrator で Booth を開始および有効化します。



注記

Booth はクラスターで Pacemaker リソースとして実行されるため、クラスターのノードで Booth を手動で開始または有効化してはなりません。

```
[arbitrator-node ~] # pcs booth start
[arbitrator-node ~] # pcs booth enable
```

- 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
```

- 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
```

以下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]
```

- 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定義されたチケット抑制を準備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追加および削除でき、この手順の完了後でも追加と削除が可能です。チケットの追加または削除後、この手順の説明どおりに、他のノード、クラスター、および arbitrator 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 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ースが必要になります。[「クラスター作成 - rgmanager と Pacemaker」](#)ではクラスターコンポーネントごとに設定の違いを説明します。

Red Hat Enterprise Linux 6.5 リリースは **pcs** 設定ツールを使用して、Pacemaker を用いたクラスター設定をサポートします。[「Red Hat Enterprise Linux 6.5 および Red Hat Enterprise Linux 7 で Pacemaker を用いたクラスターの作成」](#)では、Red Hat Enterprise Linux 6.5 の **pcs** サポートと Red Hat Enterprise Linux 7.0 の **pcs** サポートで異なる設定について説明します。

A.1. クラスター作成 - rgmanager と Pacemaker

表A.1 [「gmanager 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A.1 gmanager 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 cluster.conf 、目的に応じて直接編集が可能、または luci か ccs を使用	クラスターおよび Pacemaker 設定ファイルは corosync.conf および cib.xml です。これらのファイルは直接編集しないでください。必ず pcs または pcsd インターフェー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	luci 、 ccs コマンド、手作業で cluster.conf ファイルを編集	pcs または pcsd
インストール	rgmanager のインストール (ricci 、 luci 、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)、必要に応じて lvm2-cluster および gfs2-utils をインストール	pcs 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 lvm2-cluster および gfs2-utils 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"> 1. rgmanager と cman、必要であれば clvmd と gfs2 も起動する 2. ricci を起動、luci インターフェースを使用している場合は luci を起動 3. 必要なサービスに対して chkconfig on を実行し各ランタイムで起動するようにする <p>または、ccs --start 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"> 1. 各ノードで、systemctl start pcsd.service を実行した後に systemctl enable pcsd.service を実行し、起動時に開始するように pcsd を有効にします。 2. クラスターの 1 つのノードで、pcs cluster start --all を実行し、corosync および pacemaker を起動します。
設定ツールへのアクセスの制御	luci の場合、 luci にアクセスできるのは root ユーザーまたは luci パーミッションを持つユーザーになる、すべてのアクセスにノードの ricci パスワードが必要	pcsd GUI では、共通のシステムユーザーであるユーザー hacluster として認証される必要があります。root ユーザーは hacluster のパスワードを設定できます。
クラスター作成	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 luci または ccs で行うか cluster.conf ファイルを直接編集	pcs cluster setup コマンドまたは pcsd 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。 pcs cluster node add コマンドまたは pcsd 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。
クラスター設定を全ノードに伝える	クラスターを luci で設定する場合は設定は自動的に伝わる、 ccs の場合は --sync オプションを使用する、 cman_tool version -r コマンドの使用も可	クラスターおよび Pacemaker 設定ファイルである corosync.conf および cib.xml 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。
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 rgmanager によってサポートされます。</p> <ul style="list-style-type: none"> * 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 * 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 * RRP プロトコルを使用したクラスター設定が可能 	<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"> * クラスターに no-quorum-policy 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 * 設定可能な他のクラスタープロパティーは 表12.1「クラスターのプロパティー」 を参照
ログ機能	グローバルおよびデーモン固有のログ記録設定が可能	ログ記録を手作業で設定する方法については /etc/sysconfig/pacemaker ファイルを参照

設定コンポーネント	rgmanager	Pacemaker
クラスターの検証	luci および ccs 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る、または pcs cluster verify 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 * ccs を使用するか cluster.conf ファイルを編集して two_node=1 および expected_votes=1 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。	pcs は 2 ノードクラスターに必要なオプションを自動的に corosync へ追加します。
クラスターの状態	luci では、クラスターの現在の状態がインターフェースのさまざまなコンポーネントで表示されます。クラスターの状態はリフレッシュできます。 ccs コマンドの --getconf 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。 clustat 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。	pcs status 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 luci または ccs コマンドを使って行うか cluster.conf 設定ファイルを編集して行う	pcs resource create コマンドまたは pcsd 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 を使用したクラスターリソースの設定については 6章 クラスターリソースの設定 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 サービス 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"> * リソース動作の一部はリソースオプションとして設定 * 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 * 順序の制約を使ってリソースの実行順序を指定 <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>詳細は 6章 クラスターリソースの設定 および 7章 リソースの制約 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	luci でクラスター、クラスターの個別ノード、およびクラスターサービスの管理が可能、 ccs ではクラスターの管理が可能、クラスターサービスの管理には clusvadm を使用	pcs cluster standby 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。 pcs resource disable 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	luci でクラスター内の全ノード削除を選択しクラスター全体を削除、クラスター内の各ノードの cluster.conf を削除することも可能	pcs cluster destroy 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、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、リソースのクローンおよびマスターとスレーブリソースについては 9章 高度なリソースタイプ を参照

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またはローカルに作成しノードに追加、クラスター全体に post-fail delay と post-join delay の値を定義することが可能	pcs stonith create または pcsd 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。また、1 つのコマンドですべてのノードにフェンスデバイスを設定するよう pcmk_host_map を定義することもできます。 pcmk_host_map 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。クラスター全体の stonith-timeout の値を定義できます。
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 luci または ccs コマンドを使用するか cluster.conf ファイルを直接編集	フェンスレベルを設定

A.2. Red Hat Enterprise Linux 6.5 および Red Hat Enterprise Linux 7 で Pacemaker を用いたクラスターの作成

Red Hat Enterprise Linux 6.5 リリースは **pcs** 設定ツールを使用して、Pacemaker を用いたクラスター設定をサポートします。Pacemaker を使用する場合、Red Hat Enterprise Linux 6.5 と Red Hat Enterprise Linux 7 とでは、クラスターのインストールおよび作成に若干の違いがあります。ここでは、これらのリリースで異なるコマンドを簡単に説明します。Red Hat Enterprise Linux 7 におけるクラスターのインストールおよび作成の詳細は、[1章 Red Hat High Availability Add-On の設定と管理のリファレンス概要](#) および [4章 クラスターの作成と管理](#) を参照してください。

A.2.1. Red Hat Enterprise Linux 6.5 および Red Hat Enterprise Linux 7 での Pacemaker のインストール

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 6.5 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
[root@rhel6]# yum install pcs
[root@rhel6]# chkconfig corosync off
```

Red Hat Enterprise Linux 7 では、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インストールするだけでなく、**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** サービスを開始および有効にします。また、クラスターのノードの管理アカウントも認証します。

Red Hat Enterprise Linux 7 では、以下のコマンドをクラスターの各ノードで実行します。

```
[root@rhel7]# yum install pcs 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、以下のコマンドをクラスターのノードの 1 つで実行します。

```
[root@rhel7]# pcs cluster auth [node] [...] [-u username] [-p password]
```

A.2.2. Red Hat Enterprise Linux 6.5 および Red Hat Enterprise Linux 7 で Pacemaker を用いたクラスターの作成

Red Hat Enterprise Linux 6.5 で Pacemaker クラスターを作成する場合はクラスターの各ノードでクラスターを作成し、クラスターサービスを開始する必要があります。たとえば、**z1.example.com** と **z2.example.com** のノードで構成される **my_cluster** というクラスターを作成し、これらのノードでクラスターサービスを開始するには、**z1.example.com** と **z2.example.com** の両方で以下のコマンドを実行します。

```
[root@rhel6]# pcs cluster setup --name my_cluster z1.example.com z2.example.com  
[root@rhel6]# pcs cluster start
```

Red Hat Enterprise Linux 7 では、クラスターのノードの 1 つでクラスター作成のコマンドを実行します。以下のコマンドを 1 つのノードのみで実行すると、**z1.example.com** および **z2.example.com** というノードで構成される **my_cluster** という名前のクラスターが作成され、これらのノードでクラスターサービスが開始されます。

```
[root@rhel7]# pcs cluster setup --start --name my_cluster z1.example.com z2.example.com
```

付録B 改訂履歴

改訂 3.1-8.2 翻訳完了	Mon Jul 3 2017	Junko Ito
改訂 3.1-8.1 翻訳ファイルを XML ソースバージョン 3.1-8 と同期	Thu Apr 27 2017	Terry Chuang
改訂 3.1-8 7.3 GA 公開用バージョンの更新。	Mon Apr 17 2017	Steven Levine
改訂 3.1-4.2 翻訳ファイルを XML ソースバージョン 3.1-4 と同期	Wed Mar 1 2017	Terry Chuang
改訂 3.1-4.1 翻訳ファイルを XML ソースバージョン 3.1-4 と同期	Tue Nov 22 2016	Terry Chuang
改訂 3.1-4 7.3 GA 公開用バージョン	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

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](#)

クラスターの状態

- 表示, [クラスターの状態表示](#)

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](#)

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, [リソースを手作業で移動する](#)

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](#)

ルール, [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](#)

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](#)

場所の制約, [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](#)

新機能と変更点, [新機能と変更点](#)

概要

- 新機能と変更点, [新機能と変更点](#)

状態

- 表示, [クラスターの状態表示](#)

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](#)

順序の制約, [順序の制約](#)

順序付け, [順序の制約](#)

- , [クラスターの作成](#)

A

Action

- Property
 - 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](#)

Action Property, [リソースの動作](#)

attribute, [ノード属性の式](#)

- Constraint Expression, [ノード属性の式](#)

Attribute Expression, [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](#)

B

batch-limit, [クラスタープロパティとオプションの要約](#)

- Cluster Option, [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](#)

- Constraint Rule, [Pacemaker ルール](#)

C

Clone

- Option
 - 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](#)

Clone Option, [クローンリソースの作成と削除](#)

clone-max, [クローンリソースの作成と削除](#)

- Clone Option, [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](#)

- Clone Option, [クローンリソースの作成と削除](#)

Cluster

- Option
 - 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](#)
 - default-action-timeout, [クラスタープロパティとオプションの要約](#)
 - default-resource-stickiness, [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](#)
 - is-managed-default, [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](#)
 - 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](#)
- Querying Properties, [クラスタープロパティ設定のクエリー](#)

- Removing Properties, [クラスターのプロパティの設定と削除](#)
- Setting Properties, [クラスターのプロパティの設定と削除](#)

Cluster Option, [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 Option, [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 Option, [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](#)

- Cluster Option, [クラスタープロパティとオプションの要約](#)

Constraint

- Attribute Expression, [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](#)
- Date Specification, [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](#)
- Date/Time Expression, [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](#)
- Duration, [期間](#)
- Rule, [Pacemaker ルール](#)
 - boolean-op, [Pacemaker ルール](#)
 - role, [Pacemaker ルール](#)
 - score, [Pacemaker ルール](#)
 - score-attribute, [Pacemaker ルール](#)

Constraint Expression, [ノード属性の式](#), [時刻と日付ベースの式](#)

Constraint Rule, [Pacemaker ルール](#)

Constraints

- Colocation, [リソースのコロケーション](#)
- Location
 - id, [場所の制約](#)
 - score, [場所の制約](#)
- Order, [順序の制約](#)
 - kind, [順序の制約](#)

D

dampen, [接続状態変更によるリソースの移動](#)

- Ping Resource Option, [接続状態変更によるリソースの移動](#)

Date Specification, [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](#)

Date/Time Expression, [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](#)

dc-version, [クラスタープロパティとオプションの要約](#)

- Cluster Option, [クラスタープロパティとオプションの要約](#)

default-action-timeout, [クラスタープロパティとオプションの要約](#)

- Cluster Option, [クラスタープロパティとオプションの要約](#)

default-resource-stickiness, [クラスタープロパティとオプションの要約](#)

- Cluster Option, [クラスタープロパティとオプションの要約](#)

disabling

- resources, [クラスターリソースの有効化と無効化](#)

Duration, [期間](#)

E

enable-acl, [クラスタープロパティとオプションの要約](#)

- Cluster Option, [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](#)

- Action Property, [リソースの動作](#)

enabling

- resources, [クラスターリソースの有効化と無効化](#)

end, [時刻と日付ベースの式](#)

- Constraint Expression, [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](#)

- Resource Option, [リソースのメタオプション](#)

G

globally-unique, [クローンリソースの作成と削除](#)
- Clone Option, [クローンリソースの作成と削除](#)

Groups, [グループの Stickiness \(粘着性\)](#)

H

host_list, [接続状態変更によるリソースの移動](#)
- Ping Resource Option, [接続状態変更によるリソースの移動](#)

hours, [日付の詳細](#)
- Date Specification, [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](#)
- Action Property, [リソースの動作](#)
- Date Specification, [日付の詳細](#)
- Location Constraints, [場所の制約](#)
- Multi-State Property, [多状態のリソース: 複数モードのリソース](#)
- Resource, [リソースのプロパティ](#)

interleave, [クローンリソースの作成と削除](#)
- Clone Option, [クローンリソースの作成と削除](#)

interval, [リソースの動作](#)
- Action Property, [リソースの動作](#)

is-managed, [リソースのメタオプション](#)
- Resource Option, [リソースのメタオプション](#)

is-managed-default, [クラスタープロパティとオプションの要約](#)
- Cluster Option, [クラスタープロパティとオプションの要約](#)

K

kind, [順序の制約](#)
- Order Constraints, [順序の制約](#)

L

last-lrm-refresh, [クラスタープロパティとオプションの要約](#)
- Cluster Option, [クラスタープロパティとオプションの要約](#)

Location
- Determine by Rules, [ルールを使用したリソースの場所の確定](#)
- score, [場所の制約](#)

M

maintenance-mode, [クラスタープロパティとオプションの要約](#)
- Cluster Option, [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 Option, [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 Option, [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](#)

- Cluster Option, [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](#)

- Resource Option, [リソースのメタオプション](#)

monthdays, [日付の詳細](#)

- Date Specification, [日付の詳細](#)

months, [日付の詳細](#)

- Date Specification, [日付の詳細](#)

moon, [日付の詳細](#)

- Date Specification, [日付の詳細](#)

Moving

- Resources, [リソースを手作業で移動する](#)

Multi-State

- Option
 - 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](#)
- Property
 - id, [多状態のリソース: 複数モードのリソース](#)

Multi-State Option, [多状態のリソース: 複数モードのリソース](#)**Multi-State Property, [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](#)**

- Resource Option, [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 Resource Option, [接続状態変更によるリソースの移動](#)

Multistate, [多状態の粘着性 \(Stickiness\)](#)**N****name, [リソースの動作](#)**

- Action Property, [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](#)

- Cluster Option, [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](#)

- Clone Option, [クローンリソースの作成と削除](#)

O**on-fail, [リソースの動作](#)**

- Action Property, [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](#)

- Constraint Expression, [ノード属性の式, 時刻と日付ベースの式](#)

Option

- 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](#)
- default-action-timeout, [クラスタープロパティとオプションの要約](#)
- default-resource-stickiness, [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](#)
- 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](#)
- is-managed-default, [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](#)
- 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](#)
- 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](#)

Order

- kind, [順序の制約](#)

Order Constraints, [順序の制約](#)

- symmetrical, [順序の制約](#)

ordered, [クローンリソースの作成と削除](#)

- Clone Option, [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](#)

- Cluster Option, [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](#)

- Cluster Option, [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](#)

- Cluster Option, [クラスタープロパティとオプションの要約](#)

Ping Resource

- Option
 - 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 Resource Option, [接続状態変更によるリソースの移動](#)**priority, [リソースのメタオプション](#)**

- Resource Option, [リソースのメタオプション](#)

Property

- 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](#)

provider, [リソースのプロパティ](#)

- Resource, [リソースのプロパティ](#)

Q**Querying**

- Cluster Properties, [クラスタープロパティ設定のクエリー](#)

R**Removing**

- Cluster Properties, [クラスターのプロパティの設定と削除](#)

requires, [リソースのメタオプション](#)**Resource, [リソースのプロパティ](#)**

- Constraint
 - Attribute Expression, [ノード属性の式](#)
 - Date Specification, [日付の詳細](#)
 - Date/Time Expression, [時刻と日付ベースの式](#)
 - Duration, [期間](#)
 - Rule, [Pacemaker ルール](#)
- Constraints
 - Colocation, [リソースのコロケーション](#)
 - Order, [順序の制約](#)

- Location
 - Determine by Rules, [ルールを使用したリソースの場所の確定](#)
- Location Relative to other Resources, [リソースのコロケーション](#)
- Moving, [リソースを手作業で移動する](#)
- Option
 - 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](#)
- Property
 - 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](#)
- Start Order, [順序の制約](#)

Resource Option, [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](#)**
- Groups, [グループの Stickiness \(粘着性\)](#)
 - Multi-State, [多状態の粘着性 \(Stickiness\)](#)
 - Resource Option, [リソースのメタオプション](#)

Resources

- Clones, [リソースのクローン](#)
- Groups, [リソースグループ](#)
- Multistate, [多状態のリソース: 複数モードのリソース](#)

resources

- cleanup, [クラスターリソースのクリーンアップ](#)
- disabling, [クラスターリソースの有効化と無効化](#)
- enabling, [クラスターリソースの有効化と無効化](#)

role, [Pacemaker ルール](#)

- Constraint Rule, [Pacemaker ルール](#)

Rule

- boolean-op, [Pacemaker ルール](#)
- Determine Resource Location, [ルールを使用したリソースの場所の確定](#)
- role, [Pacemaker ルール](#)
- score, [Pacemaker ルール](#)
- score-attribute, [Pacemaker ルール](#)

S

score, [場所の制約](#), [Pacemaker ルール](#)

- Constraint Rule, [Pacemaker ルール](#)
- Location Constraints, [場所の制約](#)

score-attribute, [Pacemaker ルール](#)

- Constraint Rule, [Pacemaker ルール](#)

Setting

- Cluster Properties, [クラスターのプロパティの設定と削除](#)

shutdown-escalation, [クラスタープロパティとオプションの要約](#)

- Cluster Option, [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](#)

- Resource, [リソースのプロパティ](#)

start, [時刻と日付ベースの式](#)

- Constraint Expression, [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](#)

- Cluster Option, [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](#)

- Cluster Option, [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](#)

- Cluster Option, [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](#)

- Cluster Option, [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](#)

- Cluster Option, [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](#)

- Cluster Option, [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](#)

- Cluster Option, [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](#)

- Cluster Option, [クラスタープロパティとオプションの要約](#)

symmetrical, [順序の制約](#)

- Order Constraints, [順序の制約](#)

T

target-role, [リソースのメタオプション](#)

- Resource Option, [リソースのメタオプション](#)

Time Based Expressions, [時刻と日付ベースの式](#)

timeout, [リソースの動作](#)

- Action Property, [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](#)

- Constraint Expression, [ノード属性の式](#)
- Resource, [リソースのプロパティ](#)

V

value, [ノード属性の式](#)

- Constraint Expression, [ノード属性の式](#)

W

weekdays, [日付の詳細](#)

- Date Specification, [日付の詳細](#)

weeks, [日付の詳細](#)

- Date Specification, [日付の詳細](#)

weekyears, [日付の詳細](#)

- Date Specification, [日付の詳細](#)

Y

yeardays, [日付の詳細](#)

- Date Specification, [日付の詳細](#)

years, [日付の詳細](#)

- Date Specification, [日付の詳細](#)