Red Hat Enterprise Linux 6

クラスターの管理

High Availability アドオンの設定と管理

Last Updated: 2017-10-12

Red Hat Enterprise Linux 6 クラスターの管理

High Availability アドオンの設定と管理

法律上の通知

概要

High Availability アドオンの設定と管理 は、Red Hat Enterprise Linux 6 向けの High Availability アドオンの設定と管理について説明しています。

目次

はじめに

本ガイドは、Red Hat High Availability アドオンコンポーネントのインストール、設定、管理に関する情報を提供します。Red Hat High Availability アドオンコンポーネントにより、コンピューター (*ノード* 又は *メンバー* と呼称) のグループをつなげてクラスターとして機能させることができます。本ガイドでは、**クラスター** 又は **クラスター群** は、Red Hat High Availability アドオンを実行しているコンピューターの集合を意味します。

本ガイドは、Red Hat Enterprise Linux について高度な運用知識があり、クラスター、ストレージ、サーバーコンピューティングの概念を理解している方を対象としています。

Red Hat Enterprise Linux 6 の詳細については、以下の資料を参照してください。

- 『Red Hat Enterprise Linux インストールガイド』 — Red Hat Enterprise Linux 6 のインストールに関する情報を提供しています。
- 『Red Hat Enterprise Linux 導入ガイド』 — Red Hat Enterprise Linux 6 の導入、設定、管理に関する情報を提供しています。

Red Hat Enterprise Linux 6 の High Availability アドオンと関連製品についての詳しい情報は、以下の資料を参照してください。

- 『High Availability アドオンの概要』 — Red Hat High Availability アドオンについて大まかに説明しています。
- 『論理ボリュームマネージャの管理』 — クラスター化環境における Logical Volume Manager (LVM) の実行に関する情報など LVM について説明しています。
- 『Global File System 2: 設定と管理』 — Resilient Storage アドオンに含まれている Red Hat GFS2 (Red Hat Global File System 2) のインストール、設定、メンテナンスに関する情報を提供しています。
- 『DM マルチパス機能』 — Red Hat Enterprise Linux 6 の Device-Mapper Multipath (デバイスマッパーマルチパス) 機能の使用に関する情報を提供しています。
- 『ロードバランサーの管理』 — Load Balancer アドオンを使用したハイパフォーマンスシステムとサービスの設定に関する情報を提供します。これは、リアルサーバー群のセット全域で IP 負荷分散用に Linux Virtual Servers (LVS) を提供する統合ソフトウェアコンポーネントのセットです。
- 『リリースノート』 — Red Hat 製品の最新リリース情報を提供しています。

Red Hat Cluster Suite ドキュメントと Red Hat ドキュメントは、HTML と PDF 形式で、オンラインで https://access.redhat.com/site/documentation/ から入手できます。 また Red Hat Enterprise Linux ドキュメント CD の RPM 形式で取得可能です。

1. フィードバック

本書内で誤字・脱字を発見された場合や、本書改善のためのご意見がございましたら、弊社にご連絡ください。その場合は、製品 **Red Hat Enterprise Linux 6**、コンポーネント**doc-Cluster_Administration** で Bugzilla (http://bugzilla.redhat.com/bugzilla/) 内でご報告ください。

マニュアルの識別子の記入を忘れないで下さい：

```
Cluster_Administration(EN)-6 (2013-11-13T16:26)
```

このマニュアルの識別子を記入して頂くことにより、ご使用のガイドのバージョンを弊社で的確に把握できます。

このドキュメントの改善案をお持ちの場合は、出来るだけ詳しく説明をお願いします。エラーを発見された場合は、そのセクション番号と周囲の文の一部を含んで頂くと弊社で素早く発見することができます。

第1章 RED HAT HIGH AVAILABILITY アドオンの設定と管理の概要

Red Hat High Availability アドオンにより、コンピューター（*ノード* 又は *メンバー* と呼称）の集合を連結してクラスターとして機能させることができます。使用しているクラスタリング設定のニーズに適合するように Red Hat High Availability アドオンを使用できます (例えば、GFS2 ファイルシステム上でのファイルの共有やサービスのフェイルオーバー設定用にクラスターを設定できます)。

注記

High Availability アドオンおよび Red Hat Global File System 2 (GFS2) を使用して Red Hat Enterprise Linux クラスターを導入およびアップグレードするベストプラクティスについての情報は、Red Hat カスタマーポータルに掲載の "Red Hat Enterprise Linux Cluster, High Availability, and GFS Deployment Best Practices" というタイトルの記事を参照してください https://access.redhat.com/site/articles/40051。

本章は、Red Hat Enterprise Linux 6 の初回リリース以降に Red Hat High Availability アドオンに追加された機能と更新内容についてまとめています。次章以降からは、Red Hat High Availability アドオンの設定と管理の概要を説明していきます。

1.1. 新機能と変更された機能

本セクションは、Red Hat Enterprise Linux 6 の初回リリース以降に追加された Red Hat High Availability アドオンの新機能と変更された機能について説明します。

1.1.1. Red Hat Enterprise Linux 6.1 の新機能と変更された機能

Red Hat Enterprise Linux 6.1 では、ドキュメントと機能が以下のように更新/変更されています。

- Red Hat Enterprise Linux 6.1 リリース以降、Red Hat High Availability アドオンは SNMP トラップに対するサポートを提供しています。Red Hat High Availability アドオンを使用して SNMP トラップを設定する方法については、10章*Red Hat High Availability アドオンを使用した SNMP の設定* を参照してください。

- Red Hat Enterprise Linux 6.1 リリース以降、Red Hat High Availability アドオンは **ccs** クラスター設定コマンドに対するサポートを提供しています。**ccs** コマンドの詳細については、5章***ccs*** *コマンドを使用した Red Hat High Availability アドオンの設定* と 6章*Red Hat High Availability アドオンを使用した* ***ccs*** *の管理* を参照してください。

- Conga を使用した Red Hat High Availability アドオンソフトウェアの設定と管理については、Conga 画面と機能サポートが更新されました。

- Red Hat Enterprise Linux 6.1 リリース以降、**ricci** を使用する場合は、いずれかのノードから更新済みのクラスター設定を初めて伝播する時にパスワードが必要になります。**ricci** に関する情報は、「**ricci** の注意事項」 を参照してください。

- サービスの *Restart-Disable* 障害ポリシーを指定できるようになりました。これは、サービスが失敗した場合にシステムがサービスを再起動を試行することを意味します。ただし、サービスの再起動も失敗した場合は、サービスはクラスター内の別のホストに移動される代わりに無効になります。この機能については「クラスターへのクラスターサービスの追加」 と 付録B *HA リソースパラメーター* で説明しています。

- 独立したサブツリーを非クリティカルと設定できるようになりました。これは、リソースが失

敗した場合にはそのリソースのみが無効になることを意味します。本機能の詳細は、「クラスターへのクラスターサービスの追加」 と 「障害からの回復と独立したサブツリー」 を参照してください。

- 本ガイドには、新しい章として 9章 *クラスター内の問題の診断と修正* が追加されました。

さらに、ドキュメント全体に渡り若干の訂正と明確化を図りました。

1.1.2. Red Hat Enterprise Linux 6.2 の新機能と変更された機能

Red Hat Enterprise Linux 6.2 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。

- Red Hat Enterprise Linux は、アクティブ/アクティブ構成で Clustered Samba を実行するサポートを提供するようになりました。Clustered Samba 設定の詳細については、11章*Clustered Samba の設定* を参照してください。

- **luci** をホストしているシステム上で認証できる全ユーザーは **luci** にログインできます。ただし、Red Hat Enterprise Linux 6.2 以降、管理者 (root ユーザー又は管理者パーミッションを持つユーザー) がユーザーに対してパーミッションを設定するまでは、**luci** を実行しているシステムの root ユーザーのみが **luci** コンポーネントにアクセスできます。ユーザーに **luci** のパーミッションを設定する詳細については、「luci へのアクセス制御」 を参照してください。

- クラスターのノードは、UDP ユニキャストトランスポートのメカニズムを使用して、ノード間の通信が行われます。UDP ユニキャスト設定に関する詳細情報は 「UDP ユニキャストトラフィック」 を参照してください。

- **/etc/sysconfig/luci** ファイルを使用することで、**luci** の動作を設定できるようになりました。例えば、**luci** が機能している IP アドレスのみを特別に設定できます。**luci** が機能している IP アドレスのみを設定する詳細については、表2.2「**luci** を実行するコンピューター上で有効な IP ポート」 を参照してください。**/etc/sysconfig/luci** ファイルの一般的な詳細は、「**/etc/sysconfig/luci** を使用した **luci** の設定」 を参照してください。

- **ccs** コマンドに、利用可能なフェンスデバイスの一覧を表示する **--lsfenceopts** オプションが追加されました。また、**--lsfenceopts** *fence_type* オプションは利用可能なフェンスタイプを表示します。これらのオプションの詳細については、「フェンスデバイスとフェンスデバイスオプションの一覧表示」 を参照してください。

- **ccs** コマンドに、クラスターに現在利用可能なクラスターサービスの一覧を表示する **--lsserviceopts** オプションが追加されました。また、**--lsserviceopts** *service_type* オプションは、特定のタイプのサービスに指定できるオプションの一覧を表示します。これらのオプションの詳細については、「利用可能なクラスターサービスおよびリソースの一覧表示」 を参照してください。

- Red Hat Enterprise Linux 6.2 リリースでは、VMware (SOAP インターフェース) フェンスエージェントに対するサポートを提供しています。フェンスデバイスのパラメーターについての詳細は、付録A *フェンスデバイスパラメーター* を参照してください。

- Red Hat Enterprise Linux 6.2 リリースでは、RHEV 3.0 以降の RHEV-M REST API フェンスエージェントに対するサポートを提供しています。フェンスデバイスのパラメーターについての詳細は、付録A *フェンスデバイスパラメーター* を参照してください。

- Red Hat Enterprise Linux 6.2 リリース以降、**ccs** コマンドを使用してクラスター内で仮想マシンを設定する場合、**--addvm** オプション (**addservice** オプションではない) を使用できます。これにより、クラスター設定ファイルの **rm** 設定ノードで直接 **vm** リソースを定義できます。**ccs** コマンドを使用して仮想マシンリソースを設定する詳細については、「仮想マシンのリソース」 を参照してください。

- 本ガイドに、新しい付録として 付録D *クラスターサービスリソースチェックとフェイルオーバーのタイムアウト* が加わりました。この付録は、**rgmanager** がクラスターリソースのステータスをモニターする方法、ステータスチェックの間隔を修正する方法について説明しています。また、操作のタイムアウトによりサービスが失敗することを示した **__enforce_timeouts** サービスパラメーターについても説明しています。

- 本ガイドには、「クラスターコンポーネントに対する iptables ファイアウォールの設定」 の項が追加されています。この項では、様々なクラスターコンポーネントに対して **iptables** ファイアウォールを使用してマルチキャストトラフィックを可能にするために使用できるフィルタリングについて説明しています。

さらに、ドキュメント全体に渡り若干の訂正と明確化を図りました。

1.1.3. Red Hat Enterprise Linux 6.3 の新機能と変更された機能

Red Hat Enterprise Linux 6.3 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。

- Red Hat Enterprise Linux 6.3 リリースでは、**condor** リソースエージェントに対するサポートを提供しています。HA リソースパラメーターの詳細については、付録B *HA リソースパラメーター* を参照してください。

- 本ガイドに、付録F *High Availability LVM (HA-LVM)* の付録が新たに加わりました。

- 本ガイド全体に渡り、クラスターの再起動が必要になる設定変さらについて明確化を図りました。これらの変更は、「設定の変更が反映されない」 にまとめています。

- 本ガイドには、**luci** に 15 分間操作が行われないとログアウトするアイドルタイムアウトがあることが記載されています。**luci** の起動についての詳細は、「**luci** の起動」 を参照してください。

- **fence_ipmilan** フェンスデバイスは、権限レベルのパラメーターをサポートします。フェンスデバイスのパラメーターについての詳細は、付録A *フェンスデバイスパラメーター* を参照してください。

- 本ガイドに、「クラスター化環境での仮想マシンの設定」 のセクションが新たに加わりました。

- 本ガイドに、「luci 設定のバックアップと復元」 のセクションが新たに加わりました。

- 本ガイドに、「クラスターデーモンがクラッシュする」 のセクションが新たに加わりました。

- 本ガイドでは、デバッグオプションの設定について「ロギング」、「デバッグオプションの設定」、「DLM (分散ロックマネージャー) のデバッグログは有効にする」 で説明しています。

- Red Hat Enterprise Linux 6.3 以降、root ユーザー又は **luci** 管理者のパーミッションを付与されたユーザーは、**luci** インターフェースを使用してユーザーをシステムに追加できるようになりました。詳細は 「luci へのアクセス制御」 を参照してください。

- Red Hat Enterprise Linux 6.3 リリース以降、**ccs** コマンドによる設定の検証は、**-h** オプションを使って指定するノードの **/usr/share/cluster/cluster.rng** にあるクラススキーマに従って行います。これまでは、**ccs** コマンドは **ccs** コマンド自体でパッケージ化されたローカルシステムの **/usr/share/ccs/cluster.rng** のクラススキーマを常に使用していました。設定の検証については、「設定の妥当性検証」 を参照してください。

- 付録A *フェンスデバイスパラメーター* のフェンスデバイスパラメーターについて説明した表、付録B *HA リソースパラメーター* の HA リソースパラメーターについて説明した表には、**cluster.conf** ファイルに表示されているとおりパラメーター名も記載されています。

さらに、ドキュメント全体に渡り若干の訂正と明確化を図りました。

1.1.4. Red Hat Enterprise Linux 6.4 の新機能および変更された機能

Red Hat Enterprise Linux 6.4 では、ドキュメントと機能が以下のように更新/変更されています。

- Red Hat Enterprise Linux 6.4 リリースでは、Eaton Network Power Controller (SNMP Interface) フェンスエージェント、HP BladeSystem フェンスエージェント、IBM iPDU フェンスエージェントに対応しています。フェンスデバイスのパラメーターに関する情報は、付録A *フェンスデバイスパラメーター* を参照してください。
- 付録B *HA リソースパラメーター* では、NFS サーバーのリソースエージェントに関する説明が追加されています。
- Red Hat Enterprise Linux 6.4 では、root ユーザーまたは **luci** 管理者権限を持つユーザーは、**luci** インターフェースを使用してユーザーをシステムから削除できるようになりました。これについては、「luci へのアクセス制御」 に記載されています。
- 付録B *HA リソースパラメーター* では、ファイルシステムおよび GFS2 HA リソースの新規パラメーター **nfsrestart** について説明しています。
- 本ガイドには、「以前の設定を上書きするコマンド」 のセクションが追加されました。
- 「IP ポートの有効化」 には、**igmp** 向けに **iptables** ファイアウォールをフィルタリングする情報が含まれています。
- IPMI LAN フェンスエージェントは、付録A *フェンスデバイスパラメーター* に記載されているように、IPMI デバイスの権限レベルを設定するパラメーターに対応するようになりました。
- クラスター内のノード間通信ができるように、イーサネットのボンディングモード 1 以外に、0 と 2 へのサポートが追加されています。本ガイドのトラブルシューティングに関するアドバイスで、対応のボンディングモードのみを使用するように提案されていましたが、現在はこの旨を記載しています。
- VLAN タグのネットワークデバイスは、クラスターのハートビート通信に対応するようになりました。ハートビート通信に対応していない旨のトラブルシューティングアドバイスは、本書から削除されています。
- Red Hat High Availability アドオンは、冗長リングプロトコルの設定に対応するようになりました。この機能の使用および **cluster.conf** 設定ファイルの設定に関する一般情報は、「冗長リングプロトコルの設定」 を参照してください。**luci** で冗長リングプロトコルを設定する方法は、「冗長リングプロトコルの設定」 を参照してください。**ccs** コマンドで、冗長リングプロトコルを設定する方法は、「冗長リングプロトコルの設定」 を参照してください。

さらに、ドキュメント全体に渡り若干の訂正と明確化を図りました。

1.1.5. Red Hat Enterprise Linux 6.5 の新機能および変更された機能

Red Hat Enterprise Linux 6.5 では、ドキュメントと機能が以下のように更新、変更されています。

- 本ガイドに 「nfsexport および nfsserver リソースの設定」 セクションが新たに加わりました。

- 付録A *フェンスデバイスパラメーター* のフェンスデバイスパラメーターについての表が更新され、**luci** インターフェイスのマイナーな更新が反映されています。

さらに、ドキュメント全体にわたって若干の訂正と明確化が図られています。

1.2. 設定の基本

クラスターをセットアップするには、ノード群を特定のクラスターハードウェアに接続して、それらのノードをクラスター環境に設定します。Red Hat High Availability アドオンの設定と管理は、以下の基本的なステップで構成されています:

1. ハードウェアの設定。「ハードウェアの設定」 を参照してください。
2. Red Hat High Availability アドオンソフトウェアのインストール。「Red Hat High Availability アドオンソフトウェアのインストール」 を参照してください。
3. Red Hat High Availability アドオンソフトウェアの設定。「Red Hat High Availability アドオンソフトウェアの設定」 を参照してください。

1.3. ハードウェアの設定

ハードウェアを設定するには、Red Hat High Availability アドオンを実行するために必要となる他のハードウェアにクラスターノードを接続します。ハードウェアの数量とタイプはクラスターの目的と可用性要件により異なります。通常は、エンタープライズレベルのクラスターは以下のタイプのハードウェアを必要とします (図1.1「Red Hat High Availability アドオンハードウェアの概要」 を参照)。ハードウェア及び他のクラスター設定に関する注意事項は、*2章Red Hat High Availability アドオンを設定する前の作業* を参照するか、Red Hat 認定担当者に確認してください。

- クラスターノード — Red Hat Enterprise Linux 6 ソフトウェアを実行可能なコンピューター。RAM 1 GB 以上
- 公共ネットワーク用のネットワークスイッチ — クライアントがクラスターにアクセスするために必要です。
- プライベートネットワーク用のネットワークスイッチ — クラスターノード同士や、ネットワークパワースイッチとファイバーチェンネルスイッチなど、他のクラスターハードウェアとの通信に必要となります。
- フェンスデバイス — フェンスデバイスが必要になります。エンタープライズレベルのクラスターでのフェンシングには、ネットワークパワースイッチが推奨されます。対応するフェンスデバイスについての詳細は、*付録A フェンスデバイスパラメーター* を参照してください。
- ストレージ — 一部のタイプのストレージがクラスターに必要です。図1.1「Red Hat High Availability アドオンハードウェアの概要」 では共有ストレージが表示されていますが、特定の用途には共有ストレージが不要の場合もあります。

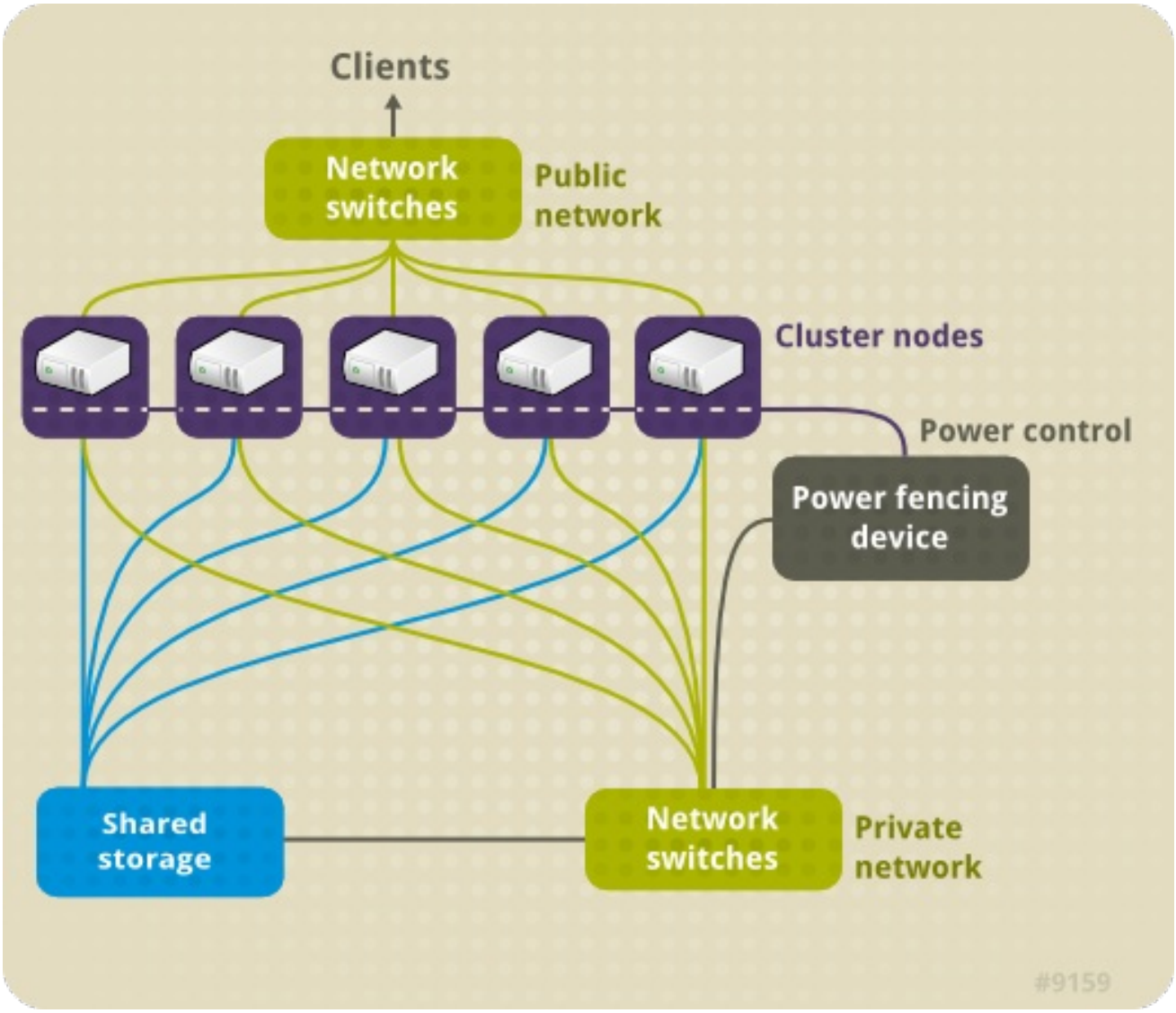


図1.1 Red Hat High Availability アドオンハードウェアの概要

1.4. RED HAT HIGH AVAILABILITY アドオンソフトウェアのインストール

Red Hat High Availability アドオンソフトウェアをインストールするには、このソフトウェアのエンタイトルメントが必要です。**luci** 設定の GUI を使用している場合、この GUI からクラスターソフトウェアをインストールしてください。その他のツールを使用してクラスターを設定している場合は、Red Hat Enterprise Linux ソフトウェアをインストールする場合と同じようにセキュリティを確保してソフトウェアをインストールしてください。

以下の **yum install** コマンドを使用して、Red Hat High Availability アドオンソフトウェアのパッケージをインストールします。

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

rgmanager のみをインストールしても、HA クラスターを作成するために必要な依存性をすべて、HighAvailability チャンネルから引っ張ってきます。**lvm2-cluster** と **gfs2-utils** パッケージは、ResilientStorage チャンネルの一部ですが、お使いのサイトでは必要がない可能性もあります。

Red Hat High Availability アドオンソフトウェアのアップグレード

クラスターを実稼働から外すことなく、任意の Red Hat Enterprise Linux のメジャーリリース上でクラ

スターソフトウェアをアップグレードすることが可能です。これを行うには、一度に 1 つのホスト上のクラスターソフトウェアを無効にして、ソフトウェアをアップグレードした後、ホスト上のクラスターソフトウェアを再起動する必要があります。

1. 任意の単一クラスターノード上の全クラスターサービスをシャットダウンします。ノード上でクラスターソフトウェアを停止する手順は、「クラスターソフトウェアの停止」 を参照してください。**rgmanager** を停止する前に、クラスターで管理中のサービスと仮想マシンを手動でホストから再配置することが望ましいでしょう。

2. **yum update** コマンドを実行してインストールしたパッケージを更新します。

3. クラスターノードを再起動するか、手動でクラスターサービスを再開始します。ノード上でクラスターソフトウェアを起動する手順については 「クラスターソフトウェアの起動」 を参照してください。

1.5. RED HAT HIGH AVAILABILITY アドオンソフトウェアの設定

Red Hat High Availability アドオンソフトウェアを設定するには、クラスターコンポーネント間の関係を指定する設定ツールを使用します。以下のクラスター設定ツールは Red Hat High Availability アドオンと使用できます：

- **Conga** — Red Hat High Availability アドオンをインストール、設定、管理するための総合的なユーザーインターフェースです。**Conga** を使用した High Availability アドオンの設定と管理の詳細については、*3章Conga を使用した Red Hat High Availability アドオンの設定* と *4章Conga を使用した Red Hat High Availability アドオンの管理* を参照してください。

- **ccs** コマンド — このコマンドは Red Hat High Availability アドオンの設定と管理を行います。**ccs** コマンドを使用した High Availability アドオンの設定と管理の詳細については、*5章ccs コマンドを使用した Red Hat High Availability アドオンの設定* と *6章Red Hat High Availability アドオンを使用した ccs の管理* を参照してください。

- コマンドラインツール — Red Hat High Availability アドオンの設定と管理を行うためのコマンドラインツール群です。コマンドラインツールを使用したクラスターの設定と管理の詳細については、*7章Red Hat High Availability の手動での設定* と *8章コマンドラインツールを使用した Red Hat High Availability アドオンの管理* を参照してください。推奨されるコマンドラインツールは *付録E コマンドラインツールの要約* にまとめています。

注記

system-config-cluster は Red Hat Enterprise Linux 6 では利用できません。

第2章 RED HAT HIGH AVAILABILITY アドオンを設定する前の作業

本章は、Red Hat High Availability アドオンのインストールと設定を行う前に実行すべきタスクと注意事項について説明しています。以下のセクションから構成されます。

重要

Red Hat High Availability アドオンの導入がご使用のニーズに適合していて、サポート可能であることを確認してください。導入する前に、Red Hat 認定担当者に連絡して、設定を検証してください。さらに、設定のバーンイン期間を設けて障害モードのテストを実施してください。

- 「設定時の一般的な注意事項」
- 「互換性のあるハードウェア」
- 「IP ポートの有効化」
- 「`/etc/sysconfig/luci` を使用した **luci** の設定」
- 「統合されたフェンスデバイスと使用するための ACPI 設定」
- 「HA サービス設定の注意事項」
- 「設定の妥当性検証」
- 「**NetworkManager** の注意事項」
- 「Quorum Disk 使用に関する注意事項」
- 「Red Hat High Availability アドオンと SELinux」
- 「マルチキャストアドレス」
- 「UDP ユニキャストトラフィック」
- 「`ricci` の注意事項」
- 「クラスター化環境での仮想マシンの設定」

2.1. 設定時の一般的な注意事項

Red Hat High Availability アドオンはご使用のニーズに適合するように様々な方法で設定できます。以下の一般的な注意事項を考慮に入れ、計画、設定、実装を行ってください。

サポートされるクラスターノードの数

High Availability アドオンでサポートされるクラスターノードの最大数は 16 です。

単一のサイトクラスター

今回は単一サイトクラスターのみ完全にサポートされています。複数の物理的な場所に渡るクラスターは正式にはサポートされていません。複数のサイトクラスターについての詳細と説明に関しては、Red Hat セールス又はサポート担当者にご相談ください。

GFS2

GFS2 ファイルシステムは、スタンドアロンシステム、あるいはクラスター設定の一部としても実装できますが、Red Hat は単一ノードファイルシステムとしての GFS2 の使用をサポートしていません。Red Hat は、ハイパフォーマンスの単一ノードファイルシステムを複数サポートしています。これらは、一般的にクラスターファイルシステムより低いオーバーヘッドを持つ単一ノード用に最適化されています。単一ノードのみがファイルシステムをマウントする必要がある状況では、Red Hat は GFS2 よりも、こうしたファイルシステムの使用を推奨します。Red Hat は、既存のお客様に対して単一ノードの GFS2 ファイルシステムを引き続きサポートします。

GFS2 ファイルシステムをクラスターファイルシステムとして設定する場合は、クラスター内の全てのノードが共有ファイルシステムにアクセスできるようにしてください。一部のノードがファイルシステムへのアクセスを持ち、他のノードは持っていない非対称クラスター設定はサポートされません。このような設定では、全てのノードが実際に GFS2 ファイルシステム自体をマウントする必要がありません。

単一障害点のないハードウェア設定

クラスターは、デュアルコアコントローラー RAID アレイ、複数のボンディングされたネットワークチャンネル、クラスターメンバーとストレージ間のマルチパス、冗長 UPS (un-interruptible power supply) システムを含むことができます。これで、1 つの障害によりアプリケーションのダウンタイムやデータの損失が発生しないようにします。

別の方法として、単一障害点のないクラスターよりも低い可用性を提供する低コストのクラスターをセットアップすることも可能です。例えば、シングルコントローラー RAID アレイとシングルイーサネットチャンネルのみを持つクラスターのセットアップが可能です。

ホスト RAID コントローラー、クラスターサポートのないソフトウェア RAID、マルチイニシエーターパラレル SCSI 設定など一部の低コスト代替手段は、互換性がないか、共有クラスターストレージとして使用するには適切ではありません。

データ整合性の保証

データ整合性を確保するために、一度に 1 つのノードだけがクラスターサービスを実行してクラスターサービスのデータにアクセスできます。クラスターハードウェア設定内のパワースイッチを使用することで、フェイルオーバープロセス中に 1 つのノードがそのノードの HA サービスを再開する前に別のノードをパワーサイクルできるようにします。これにより 2 つのノードが同時に同じデータにアクセスしてデータが破損するのを防止します。*フェンスデバイス* (遠隔操作でクラスターノード群の電力供給、シャットダウン、再起動を行うハードウェア又はソフトウェアソリューション) を使用して、全ての障害状況下でデータの整合性を保証します。

イーサネットチャンネルボンディング

クラスタークォーラムとノードの健全性は、イーサネットを介してクラスターノード間のメッセージ通信で判定されます。さらに、クラスターノード群は他のクリティカルな各種クラスター機能 (例えば、フェンシング) にイーサネットを使用します。マルチイーサネットインターフェースは、イーサネットチャンネルボンディングと同一体として動作するよう設定され、クラスターノード群と他のクラスターハードウェア間の標準的なスイッチドイーサネット接続における単一障害点のリスクを軽減します。

Red Hat Enterprise Linux 6.4 ではボンディングモード 0、1、2 がサポートされています。

IPv4 と IPv6

High Availability アドオンは IPv4 と IPv6 の両方のインターネットプロトコルをサポートします。High Availability アドオンでの IPv6 のサポートは Red Hat Enterprise Linux 6 では初めてです。

2.2. 互換性のあるハードウェア

Red Hat High Availability アドオンソフトウェアを設定する前に、該当するクラスターが適切なハードウェア (例えば、サポートされているフェンスデバイス、ストレージデバイス、ファイバーチャンネルスイッチ) を使用していることを確認してください。最新のハードウェア互換性の情報については https://hardware.redhat.com/ にある Red Hat Hardware Catalog を参照してください。

2.3. IP ポートの有効化

Red Hat High Availability アドオンを導入する前に、クラスターノード群及び **luci** (**Conga** ユーザーインターフェースサーバー) を実行するコンピューター群上で特定の IP ポートを有効にする必要があります。以下のセクションで、有効にすべき IP ポートを説明しています。

- 「クラスターノードでの IP ポートの有効化」
- 「**luci** の IP ポートの有効化」

以下のセクションでは、Red Hat High Availability アドオンで必要な IP ポートを有効にする際の **`iptables`** ルールを提供しています。

- 「クラスターコンポーネントに対する iptables ファイアウォールの設定」

2.3.1. クラスターノードでの IP ポートの有効化

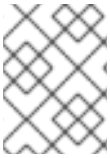
クラスター内のノード間で通信できるように、特定の Red Hat High Availability アドオンコンポーネントに割り当てられた IP ポートを有効化する必要があります。表2.1「Red Hat High Availability アドオンノードで有効にする IP ポート」 では、IP ポート番号、該当するプロトコル、ポート番号が割り当てられたコンポーネントについて記載しています。各クラスターノードで、表2.1「Red Hat High Availability アドオンノードで有効にする IP ポート」 に従い IP ポートを有効化してください。**`system-config-firewall`** を使用して IP ポートを有効化できます。

表2.1 Red Hat High Availability アドオンノードで有効にする IP ポート

IP ポート番号	プロトコル	コンポーネント
5404, 5405	UDP	**`corosync/cman`** (クラスターマネージャー)
11111	TCP	**`ricci`** (クラスターの更新情報を伝播)
21064	TCP	**`dlm`** (分散ロックマネージャー)
16851	TCP	**`modclusterd`**

2.3.2. luci の IP ポートの有効化

クライアントコンピューター群が **luci** (**Conga** ユーザーインターフェースサーバー) を実行する 1 台のコンピューターと通信できるようにするには、**luci** に割り当てられている IP ポートを有効にする必要があります。表2.2「**luci** を実行するコンピューター上で有効な IP ポート」 に従って、**luci** を実行する各コンピューターで IP ポートを有効にします。



注記

クラスターノードが **luci** を実行している場合は、ポート 11111 は既に有効になっているはずです。

表2.2 luci を実行するコンピューター上で有効な IP ポート

IP ポート番号	プロトコル	コンポーネント
8084	TCP	**luci** (**Conga** ユーザーインターフェースサーバー)

/etc/sysconfig/luci ファイルを使用して設定を有効した Red Hat Enterprise Linux 6.1 リリース以降、**luci** が機能する IP アドレスのみを特別に設定できるようになりました。使用しているサーバーインフラストラクチャーに複数のネットワークが組み込まれており、内部ネットワークからのみ **luci** にアクセスしたい場合は、この機能を使用できます。このためには、ファイル内の **host** を指定する行をコメント解除して編集します。例えば、ファイルの **host** 設定を 10.10.10.10 に変更するには、**host** の行を次のように変更します。

```
host = 10.10.10.10
```

/etc/sysconfig/luci ファイルの詳細については、「**/etc/sysconfig/luci** を使用した **luci** の設定」 を参照してください。

2.3.3. クラスターコンポーネントに対する iptables ファイアウォールの設定

以下では、Red Hat Enterprise Linux 6 (High Availability アドオンあり) で必要とされる IP ポートを有効化するための iptable ルールのサンプルを記載しています。これらの例では、サブネットとして 192.168.1.0/24 を使用しますが、これらのルールを使用する場合は適切なサブネットで 192.168.1.0/24 を置き換えてください。

cman (Cluster Manager) では以下のフィルタリングを使用します。

```
$ iptables -I INPUT -m state --state NEW -m multiport -p udp -s
192.168.1.0/24 -d 192.168.1.0/24 --dports 5404,5405 -j ACCEPT
$ iptables -I INPUT -m addrtype --dst-type MULTICAST -m state --state NEW
-m multiport -p udp -s 192.168.1.0/24 --dports 5404,5405 -j ACCEPT
```

dlm (Distributed Lock Manager) の場合:

```
$ iptables -I INPUT -m state --state NEW -p tcp -s 192.168.1.0/24 -d
192.168.1.0/24 --dport 21064 -j ACCEPT
```

ricci (Conga リモートエージェントの一部) の場合:

```
$ iptables -I INPUT -m state --state NEW -p tcp -s 192.168.1.0/24 -d
192.168.1.0/24 --dport 11111 -j ACCEPT
```

modclusterd (Conga リモートエージェントの一部) の場合:

```
$ iptables -I INPUT -m state --state NEW -p tcp -s 192.168.1.0/24 -d
192.168.1.0/24 --dport 16851 -j ACCEPT
```

luci (Conga ユーザーインターフェースサーバー) の場合:

```
$ iptables -I INPUT -m state --state NEW -p tcp -s 192.168.1.0/24 -d
192.168.1.0/24 --dport 8084 -j ACCEPT
```

igmp (Internet Group Management Protocol) の場合:

```
$ iptables -I INPUT -p igmp -j ACCEPT
```

これらのコマンドを実行後、以下のコマンドを実行して現在の設定を保存して、再起動しても変更が維持されるようにします。

```
$ service iptables save ; service iptables restart
```

2.4. /ETC/SYSCONFIG/LUCI を使用した LUCI の設定

Red Hat Enterprise Linux 6.1 リリース以降、**/etc/sysconfig/luci** ファイルを使用して **luci** の動作のいくつかを設定できるようになりました。このファイルで変更できるパラメーターには、サーバー設定だけでなく init スクリプトにより使用される動作環境の補助的な設定も含まれます。また、このファイルを編集して、アプリケーション設定のパラメーターを修正することもできます。ファイル内には、編集によって変更可能な設定パラメーターを説明した手順が記載されています。

所定の形式を維持するために、ファイルの編集時には **/etc/sysconfig/luci** ファイルの設定以外の行は変更しないことをお勧めします。また、このファイル、特に **INITSCRIPT** セクションに必要な構文には従うようにしてください。ここでは、等号記号の前後に空白を入れることはできません。また、空白を含む文字列を囲む引用符を使用する必要があります。

以下の例は、**/etc/sysconfig/luci** ファイルを編集することにより **luci** が機能するポートを変更する方法を示しています。

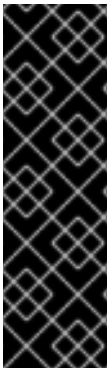
1. **/etc/sysconfig/luci** ファイル内の次の行をコメント解除します:

   ```
   #port = 4443
   ```

2. 4443 を希望するポート番号に置き換えます。これは 1024 (特権ポートではない) かそれ以上でなければなりません。例えば、次のようにファイルの行を編集して、**luci** が機能するポートを 8084 に設定できます。

   ```
   port = 8084
   ```

3. **luci** サービスを再起動して、変更を反映させます。



重要

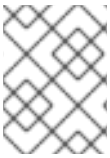
/etc/sysconfig/luci ファイル内の設定パラメーターを修正してデフォルト値を再定義する場合は、デフォルト値の代わりに新しい値を使用することをお勧めします。例えば、**luci** が機能するポートを修正する場合、「**luci** の IP ポートの有効化」 で説明のとおり **luci** の IP ポートを有効にする時に修正した値を指定するようにしてください。

修正されたポートとホストパラメーターは、「**luci** の起動」 の説明のとおり **luci** サービスの起動時に表示される URL に自動的に反映されます。**luci** にアクセスするにはこの URL を使用することをお勧めします。

/etc/sysconfig/luci ファイルで設定できるパラメーターの完全な情報は、ファイル内のドキュメントを参照してください。

2.5. 統合されたフェンスデバイスと使用するための ACPI 設定

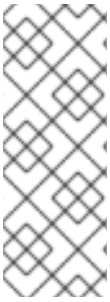
クラスターが統合化されたフェンスデバイスを使用する場合、迅速で完全なフェンシングを確保するために ACPI (Advanced Configuration and Power Interface) を設定する必要があります。



注記

Red Hat High Availability アドオンでサポートされている統合化されたフェンスデバイスの最新情報は、 http://www.redhat.com/cluster_suite/hardware/ を参照してください。

クラスターノードが統合化されたフェンスデバイスによりフェンスされるよう設定されている場合は、そのノードの ACPI Soft-Off を無効にします。ACPI Soft-Off を無効にすると、クリーンシャットダウン (例えば **shutdown -h now**) を試みるのではなく、統合されたフェンスデバイスが迅速かつ完全にノードをオフにできるようにします。これに対して、ACPI Soft-Off が有効な場合、統合されたフェンスデバイスはノードをオフにするのに 4 秒以上かかります (以下の注記参照)。さらには、ACPI Soft-Off が有効で、シャットダウン中にノードがパニックやフリーズを起こすと、統合されたフェンスデバイスがノードをオフにできない場合があります。これらの状況下では、フェンシングは遅延されるか、失敗します。結果的に、ノードが統合されたフェンスデバイスでフェンスされ ACPI Soft-Off が有効であると、クラスターの復帰は遅くなるか、復帰に管理者の介入が必要になります。



注記

ノードをフェンスするのに必要な時間は、使用される統合化されたフェンスデバイスにより異なります。一部の統合化されたフェンスデバイスは、電源ボタンを押し続けることに相当する動作を行います。そのため、フェンスデバイスは 4 ～ 5 秒でノードをオフにします。別の統合化されたフェンスデバイスは、電源ボタンを一瞬だけ押すことに相当する動作を行い、ノードのオフはオペレーティングシステムに依存します。そのため、フェンスデバイスは 4 ～ 5 秒よりもさらに長い時間幅でノードをオフにします。

ACPI Soft-Off を無効にするには、**chkconfig** 管理を使用して、フェンスされた時にノードが直ちにオフになることを確認します。ACPI Soft-Off を無効にするための推奨される方法は **chkconfig** 管理を使用することです。ただし、このメソッドがご使用のクラスターに十分でない場合は、以下の代替メソッドのいずれかを使用して ACPI Soft-Off を無効にできます：

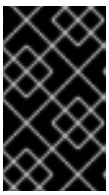
- BIOS 設定を "instant-off" 又は遅延なくノードをオフにする同様の設定に変更

  注記

  BIOS で ACPI Soft-Off を無効にすることは、一部のコンピューターでは不可能な場合があります。

- **/boot/grub/grub.conf** ファイルのカーネルブートコマンドの行に **acpi=off** を追記

  重要

  このメソッドは ACPI を完全に無効にします。ACPI が完全に無効の場合、一部のコンピューターは正しく起動しません。このメソッドは、他のメソッドがご使用のクラスターに効果的でない場合に ***のみ*** 使用してください。

以下のセクションは、ACPI Soft-Off を無効にするための推奨されるメソッドと代替メソッドの手順を記載しています。

- 「**chkconfig** 管理を使用した ACPI Soft-Off の無効化」 — 推奨されるメソッド
- 「BIOS を使用した ACPI Soft-Off の無効化」 — 1 番目の代替メソッド
- 「**grub.conf** ファイル内での ACPI の完全無効化」 — 2 番目の代替メソッド

2.5.1. chkconfig 管理を使用した ACPI Soft-Off の無効化

chkconfig 管理を使用して ACPI Soft-Off を無効にするには、**chkconfig** 管理から ACPI デーモン (**acpid**) を削除するか、**acpid** をオフにします。

注記

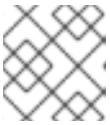
これが ACPI Soft-Off を無効にする推奨されるメソッドです。

以下のようにして、各クラスターノードで **chkconfig** 管理を使用して ACPI Soft-Off を無効にします。

1. 以下のどちらかのコマンドを実行します。
   - **chkconfig --del acpid** — このコマンドは **chkconfig** 管理から **acpid** を削除します。

     — または —

   - **chkconfig --level 345 acpid off** — このコマンドは **acpid** をオフにします。
2. ノードを再起動します。
3. クラスターが設定され稼働している間に、フェンスされた時点でノードが直ちにオフになることを確認します。

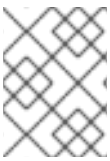


注記

ノードは **fence_node** コマンド、又は **Conga** でフェンスできます。

2.5.2. BIOS を使用した ACPI Soft-Off の無効化

ACPI Soft-Off を無効にするための推奨されるメソッドは、**chkconfig** 管理 (「**chkconfig** 管理を使用した ACPI Soft-Off の無効化」) を使用することです。しかし、このメソッドがご使用のクラスターに効果的でない場合は、本セクションの手順に従ってください。



注記

BIOS で ACPI Soft-Off を無効にすることは、一部のコンピューターでは不可能な場合があります。

以下のようにして、各クラスターノードの BIOS を設定することで ACPI Soft-Off を無効にできます。

1. ノードを再起動して、**BIOS CMOS Setup Utility** プログラムを起動します。

2. **Power** メニュー（又は同等の電源管理メニュー）に移動します。

3. **Power** メニューで **Soft-Off by PWR-BTTN** 機能 (又は同等の機能) を **Instant-Off** (又は、遅延なく電源ボタンを使ってノードをオフにできるのと同等の設定) にセットします。例2.1「**BIOS CMOS Setup Utility**: **Soft-Off by PWR-BTTN** を **Instant-Off** に設定」 は、**ACPI Function** が **Enabled** に、**Soft-Off by PWR-BTTN** が **Instant-Off** にセットされている **Power** メニューを示しています。

注記

ACPI Function、 **Soft-Off by PWR-BTTN**、**Instant-Off** と同等の機能は、コンピューターによって異なります。しかし、この手順の目的は、遅延なしに電源ボタンを介してコンピューターの電源オフができるように BIOS を設定することです。

4. **BIOS CMOS Setup Utility** プログラムを終了して、BIOS 設定を保存します。

5. クラスターが設定され稼働している間に、フェンスされた時点でノードが直ちにオフになることを確認します。

注記

ノードは **fence_node** コマンド、又は **Conga** でフェンスできます。

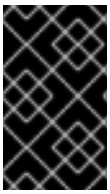
例2.1 BIOS CMOS Setup Utility: Soft-Off by PWR-BTTN を Instant-Off に設定

```
+---------------------------------------------|-------------------+
|    ACPI Function             [Enabled]      |    Item Help      |
|    ACPI Suspend Type         [S1(POS)]      |-------------------|
|  x Run VGABIOS if S3 Resume   Auto          |   Menu Level   *  |
|    Suspend Mode              [Disabled]     |                   |
|    HDD Power Down            [Disabled]     |                   |
|    Soft-Off by PWR-BTTN      [Instant-Off   |                   |
|    CPU THRM-Throttling       [50.0%]        |                   |
|    Wake-Up by PCI card       [Enabled]      |                   |
|    Power On by Ring          [Enabled]      |                   |
|    Wake Up On LAN            [Enabled]      |                   |
|  x USB KB Wake-Up From S3     Disabled      |                   |
|    Resume by Alarm           [Disabled]     |                   |
|  x  Date(of Month) Alarm       0            |                   |
|  x  Time(hh:mm:ss) Alarm       0 :  0 :     |                   |
|    POWER ON Function         [BUTTON ONLY   |                   |
|  x KB Power ON Password       Enter         |                   |
|  x Hot Key Power ON           Ctrl-F1       |                   |
|                                             |                   |
|                                             |                   |
+---------------------------------------------|-------------------+
```

この例では、**ACPI Function** が **Enabled** に、**Soft-Off by PWR-BTTN** が **Instant-Off** に設定されています。

2.5.3. grub.conf ファイル内での ACPI の完全無効化

ACPI Soft-Off を無効にするための推奨されるメソッドは、**chkconfig** 管理 (「**chkconfig** 管理を使用した ACPI Soft-Off の無効化」) を使用することです。使用中のクラスターにこの推奨されるメソッドが効果的でない場合は、BIOS 電源管理 (「BIOS を使用した ACPI Soft-Off の無効化」) を使用して ACPI Soft-Off を無効にできます。どのメソッドも効果的でない場合は、**grub.conf** ファイルのカーネルブートコマンドラインに **acpi=off** を追記すると ACPI を完全に無効にできます。



重要

このメソッドは ACPI を完全に無効にします。ACPI が完全に無効の場合、一部のコンピューターは正しく起動しません。このメソッドは、他のメソッドがご使用のクラスターに効果的でない場合に **のみ** 使用してください。

以下のようにして、各クラスターの **grub.conf** ファイルを編集することにより ACPI を完全に無効にできます。

1. テキストエディターで **/boot/grub/grub.conf** を開きます。
2. **/boot/grub/grub.conf** 内のカーネルコマンドラインに **acpi=off** を追記します (例 2.2「**acpi=off** が追記されたカーネルブートコマンドライン」 を参照) 。
3. ノードを再起動します。
4. クラスターが設定され稼働している間に、フェンスされた時点でノードが直ちにオフになることを確認します。

注記

ノードは **fence_node** コマンド、又は **Conga** でフェンスできます。

例2.2 acpi=off が追記されたカーネルブートコマンドライン

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this
file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/mapper/vg_doc01-lv_root
#          initrd /initrd-[generic-]version.img
#boot=/dev/hda
default=0
timeout=5
serial --unit=0 --speed=115200
terminal --timeout=5 serial console
title Red Hat Enterprise Linux Server (2.6.32-193.el6.x86_64)
        root (hd0,0)
        kernel /vmlinuz-2.6.32-193.el6.x86_64 ro
root=/dev/mapper/vg_doc01-lv_root console=ttyS0,115200n8 acpi=off
        initrd /initramrs-2.6.32-131.0.15.el6.x86_64.img
```

この例では、**acpi=off** がカーネルブートコマンドライン (「kernel /vmlinuz-2.6.32-193.el6.x86_64.img」で始まる行) に追加されました。

2.6. HA サービス設定の注意事項

HA (高可用性) サービスを設定することにより、ご使用のニーズに合う高可用性クラスターを作成できます。Red Hat High Availability アドオン内の HA サービス管理の核となるコンポーネントである **rgmanager** は、既製のアプリケーション用にコールドフェイルオーバーを実装します。Red Hat High Availability アドオンでは、アプリケーションは他のクラスターリソースにより設定されます。これにより、クラスタークライアントに対する明らかな中断なしに 1 つのクラスターノードから別のノードにフェイルオーバーできる HA サービスを形成できます。HA サービスのフェイルオーバーが発生する可能性があるのは、クラスターノードが失敗した場合、又はクラスターシステム管理者がサービスをあるクラスターノードから別のノードへ移動した場合 (例えば、予定されたクラスターノードの停止) です。

HA サービスを作成するには、クラスター設定ファイル内で設定する必要があります。HA サービスはクラスター *リソース* で構成されています。クラスターリソースとは、クラスター設定ファイル内で作成/管理を行うためのビルディングブロックです (例: IP アドレス、アプリケーション初期化スクリプト、Red Hat GFS2 共有パーティション)。

HA サービスは、データ整合性を維持するため一度に 1 つのノードでのみ実行可能です。フェイルオーバーの優先度はフェイルオーバードメイン内で指定できます。フェイルオーバーの優先度を指定するには、フェイルオーバードメイン内の各ノードに優先度レベルを割り当てます。この優先度レベルがフェイルオーバーの順番 (HA サービスがフェイルオーバーすべき対象のノード) を決定します。フェイルオーバーの優先度を指定しない場合は、HA サービスはフェイルオーバードメイン内の任意のノードにフェイルオーバーできます。また、HA サービスが関連したフェイルオーバードメインのノード上でのみ実行するように制限するかどうかも指定できます (HA サービスが無制限のフェイルオーバードメインに関連付けられている場合、フェイルオーバードメインのメンバーがない時は HA サービスはどのクラスターノードでも開始できます)。

図2.1「ウェブサーバーのクラスターサービスの例」 は、「content-webserver」と呼ばれるウェブサーバーである HA サービスの例を示しています。このサービスは、クラスターノード B で実行中であり、ノード A、B、及び D から成るフェイルオーバードメイン内に存在します。さらに、このフェイルオーバードメインは、ノード A の前にノード D にフェイルオーバーして、フェイルオーバードメイン内のノードのみにフェイルオーバーを制限するようなフェイルオーバー優先度で設定されています。HA サービスは以下のようなクラスターリソースで構成されています。

- IP アドレスリソース — IP アドレス 10.10.10.201
- 「httpd-content」と呼ばれるアプリケーションリソース — ウェブサーバーアプリケーション init スクリプト **/etc/init.d/httpd** (**httpd** を指定)
- ファイルシステムリソース —「gfs2-content-webserver」と呼ばれる Red Hat GFS2

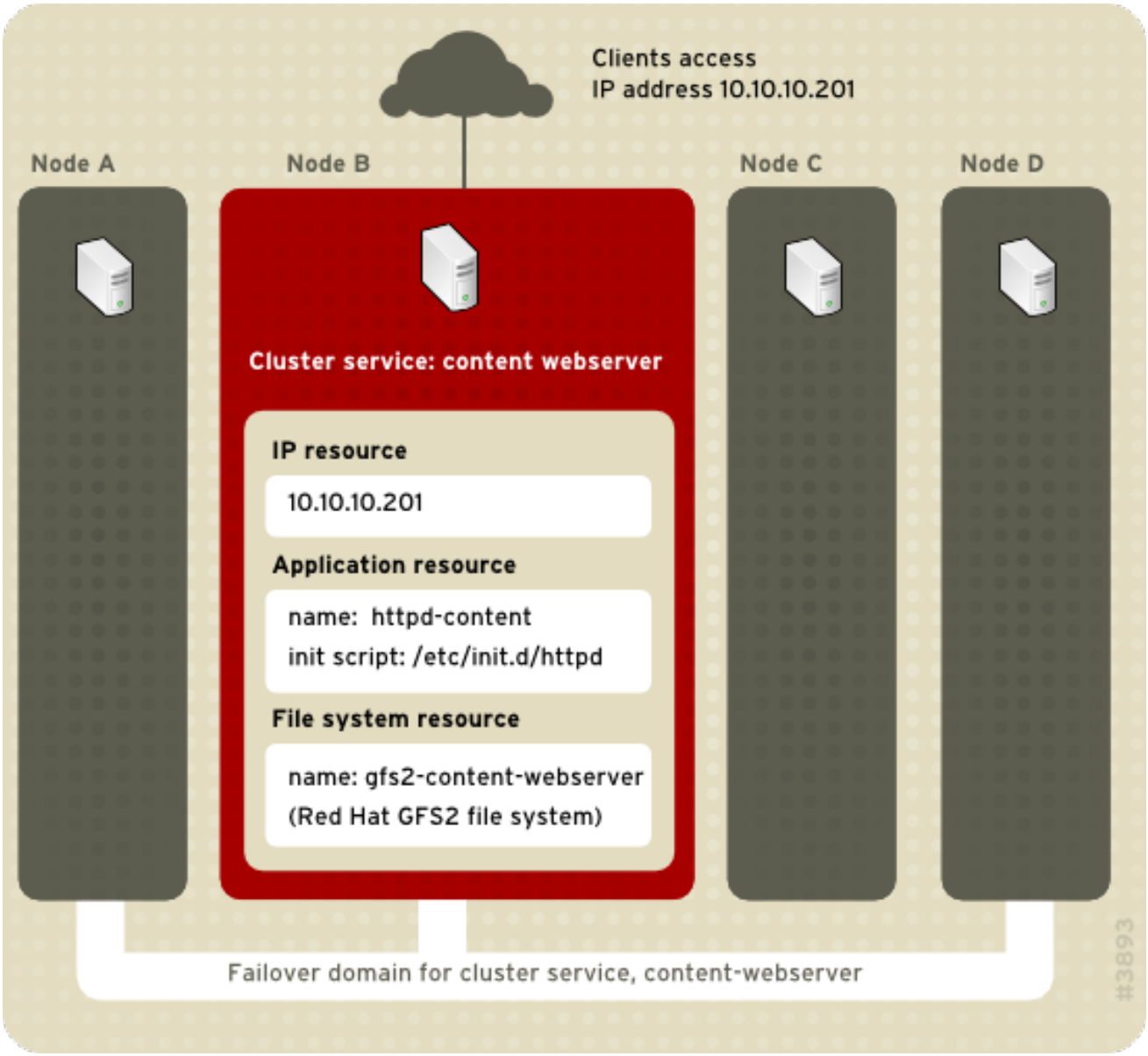


図2.1 ウェブサーバーのクラスターサービスの例

クライアントは、IP アドレス 10.10.10.201 を介して HA サービスにアクセスして、ウェブサーバーアプリケーションである httpd-content との通信を可能にします。httpd-content アプリケーションは gfs2-content-webserver ファイルシステムを使用します。ノード B が失敗した場合、content-webserver HA サービスはノード D にフェイルオーバーすることになります。ノード D が利用できない、又は失敗した場合は、サービスはノード A にフェイルオーバーします。フェイルオーバーは、クラスタークライアントに対してサービスの中断を最小限に抑えながら発生します。例えば、HTTP サービスでは、特定の状態の情報 (セッションデータなど) が消失する場合があります。HA サービスはフェイルオーバーが発生する前のように同じ IP アドレスを介して他のクラスターノードからもアクセス可能になります。

注記

HA サービスとフェイルオーバードメインについての詳細は、『High Availability アドオンの概要』 を参照してください。フェイルオーバードメインの設定についての詳細は、*3章Conga を使用した Red Hat High Availability アドオンの設定* (**Conga** の使用) 又は *7章Red Hat High Availability の手動での設定* (コマンドラインユーティリティを使用) を参照してください。

HA サービスは、クライアントに特化したサービスを提供する一貫したエンティティに設定されたクラスターリソースのグループです。HA サービスは、(各クラスターノードの) クラスター設定ファイル

/etc/cluster/cluster.conf 内にリソースツリーとして表されます。クラスター設定ファイル内では、各リソースツリーは各リソース、その属性、リソースツリー内での他のリソースとの関係 (親、子、兄弟の関係) を指定する XML 表現です。

注記

HA サービスは階層ツリー型に編成されたリソースで構成されているため、サービスは *リソースツリー* 又は *リソースグループ* と呼ばれることもあります。両方の呼称とも **HA サービス** と同義語です。

各リソースツリーの根元には、特別なリソースのタイプである、*サービスリソース* があります。他のリソースのタイプがサービスの残りの部分を構成し、その特性を決定します。HA サービスの設定は、サービスリソースの作成、下位クラスターリソースの作成、サービスの階層制限に従う一貫したエンティティへそれらを組織化することによって行います。

HA サービスを設定する時に検討すべき考慮事項は大きく分けて 2 点あります。

- サービスを作成するために必要なリソースのタイプ
- リソース間での親、子、兄弟の関係

リソースのタイプとリソースの階層は、設定しているサービスのタイプに応じて異なります。

クラスターリソースのタイプは 付録B *HA リソースパラメーター* に一覧表示しています。リソース間の親、子、兄弟の関係は 付録C *HA リソースの動作* で説明しています。

2.7. 設定の妥当性検証

クラスターの設定は、スタートアップ及び設定の再ロード時に **/usr/share/cluster/cluster.rng** のクラスタースキーマに沿って自動的に妥当性が検証されます。また、**ccs_config_validate** コマンドを使用するとクラスター設定の妥当性をいつでも検証できます。**ccs** コマンドを使用した設定の妥当性検証については、「設定の妥当性検証」 を参照してください。

注釈付きのスキーマは **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** (例えば **/usr/share/doc/cman-3.0.12/cluster_conf.html**) を参照してください。

設定の妥当性検証は、以下の基本的なエラーをチェックします。

- XML 妥当性 — 設定ファイルが有効な XML ファイルであることをチェックします。
- 設定のオプション — オプション (XML の要素と属性) が有効であることをチェックします。
- オプションの値 — オプションに有効なデータが含まれていることをチェックします (限定的)。

以下の例は、妥当性検証を示した有効な設定と無効な設定を表示しています。

- 有効な設定 — 例2.3「**cluster.conf** のサンプル設定：有効なファイル」
- 無効な XML — 例2.4「**cluster.conf** のサンプル設定：無効な XML」
- 無効なオプション — 例2.5「**cluster.conf** のサンプル設定：無効なオプション」
- 無効なオプション値 — 例2.6「**cluster.conf** のサンプル設定：無効なオプション値」

例2.3 cluster.conf のサンプル設定：有効なファイル

```
<cluster name="mycluster" config_version="1">
  <logging debug="off"/>
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
   </fencedevices>
   <rm>
   </rm>
</cluster>
```

例2.4 `cluster.conf` のサンプル設定：無効な XML

```
<cluster name="mycluster" config_version="1">
  <logging debug="off"/>
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
   </fencedevices>
   <rm>
   </rm>
<cluster>         <----------------INVALID
```

この例では、設定の最後の行 ("INVALID" として注釈) にスラッシュがありません。**</cluster>** ではなく **<cluster>** となっています。

例2.5 cluster.conf のサンプル設定：無効なオプション

```
<cluster name="mycluster" config_version="1">
  <loging debug="off"/>         <----------------INVALID
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
   </fencedevices>
   <rm>
   </rm>
<cluster>
```

この例では、設定の 2 行目 ("INVALID" として注釈) に無効な XML 要素が含まれています。**logging** ではなく **loging** となっています。

例2.6 cluster.conf のサンプル設定：無効なオプション値

```
<cluster name="mycluster" config_version="1">
  <loging debug="off"/>
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="-1">  <--------
INVALID
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
         </fence>
     </clusternode>
```

```
    </clusternodes>
    <fencedevices>
    </fencedevices>
    <rm>
    </rm>
 <cluster>
```

この例では、設定の 4 行目 ("INVALID" として注釈) に XML 属性に無効な値が含まれています。**node-01.example.com** の **clusternode** の行に **nodeid** があります。この値は正の値 ("1") ではなく負の値 ("-1") となっています。**nodeid** 属性の値は、正の値でなければなりません。

2.8. NETWORKMANAGER の注意事項

NetworkManager の使用は、クラスターノード上ではサポートされていません。クラスターノードに **NetworkManager** をインストールした場合は、削除/無効にしてください。

注記

NetworkManager が実行中あるいは **chkconfig** コマンドを使用して実行するように設定されている場合、**cman** サービスは起動しません。

2.9. QUORUM DISK 使用に関する注意事項

Quorum Disk は、ディスクベースの quorum （定足数）デーモン **qdiskd** で、補助的なヒューリスティックを提供してノードの健全性を判定します。ヒューリスティックを使用することで、ネットワークパーティションが発生した場合に、ノードの運用に重要となる要素を決定できます。例えば、3:1 に分かれた 4 ノードクラスターでは、通常 3 つのノードが 3 対 1 の過半数で自動的に「勝ち」ます。この状況では、1 つのノードがフェンスされます。しかし、**qdiskd** を使用すると、クリティカルなリソース (例: クリティカルなネットワークパス) へのアクセスを基にしてその 1 つのノードが勝てるようにヒューリスティックを設定できます。使用しているクラスターにノードの健全性を判定する追加のメソッドが必要な場合は、そのニーズに合うように **qdiskd** を設定することをお勧めします。

注記

ノードの健全性に特別な要件がない限りは **qdiskd** を設定する必要はありません。特別な要件の例としては、「all-but-one (1 つ以外は全て)」の設定があります。all-but-one 設定では、1 つのノードだけが機能している場合でも定足数を維持するための十分な定足数投票数を提供するように **qdiskd** を設定します。

重要

総括すると、導入のためのヒューリスティック及び他の **qdiskd** パラメーターはサイトの環境と特別な要件によって異なります。ヒューリスティック及び他の **qdiskd** パラメーターの使用を理解するためには、qdisk(5) の man ページを参照してください。**qdiskd** を理解してそれをサイトに使用する上でサポートが必要な場合は、Red Hat 認定サポート担当者にご連絡ください。

qdiskd を使用する必要がある場合は、以下の注意事項を考慮に入れてください。

クラスターノードの投票数

Quorum Disk の使用時は、各クラスターノードは 1 つの投票数を持っている必要があります。

CMAN メンバーシップのタイムアウト値

qdiskd メンバーシップタイムアウト値は、CMAN メンバーシップタイムアウト値 (ノードがメンバーではなく停止していると CMAN が判定するまでのノードが応答しない時間) に基づいて自動的に設定されます。**qdiskd** は、CMAN タイムアウト内で実行できることを確認するための追加の健全性チェックも行います。この値の再設定が必要な場合は、以下の点を考慮する必要があります。

CMAN メンバーシップタイムアウト値は、少なくとも **qdiskd** メンバーシップタイムアウト値の 2 倍にすることをお勧めします。その理由は、quorum（定足数）デーモンは失敗したノードをそれ自身で検出しなければならず、その実行に CMAN よりもかなり長い時間がかかる場合があるためです。他のサイト特有の条件が CMAN と **qdiskd** のメンバーシップタイムアウト値の関係に影響を与える場合があります。CMAN メンバーシップタイムアウト値の調節に関するサポートについては、Red Hat 認定サポート担当者にご連絡ください。

フェンシング

qdiskd の使用時にフェンシングが確実に信頼できるようにするには、パワーフェンシングを使用します。**qdiskd** で設定されていないクラスターには他のタイプのフェンシングが信頼できる場合もありますが、**qdiskd** で設定されたクラスター用には信頼できません。

ノードの最大数

qdiskd で設定されたクラスターは最大で 16 のノードをサポートします。この上限はスケーラビリティによるものです。つまり、ノード数を増加すると、共有 Quorum Disk デバイス上の同期 I/O 競合の量が増加します。

Quorum disk デバイス

quorum disk デバイスは、クラスター内の全ノードによる並行読み取り/書き込みのアクセスを持つ共有ブロックデバイスです。ブロックデバイスの最小限サイズは 10 メガバイトです。**qdiskd** で使用できる共有ブロックデバイスの例としては、マルチポート SCSI RAID アレイ、ファイバーチャンネル RAID SAN、RAID 構成の iSCSI ターゲットがあります。Cluster Quorum Disk Utility の **mkqdisk** を使用して quorum disk デバイスを作成することができます。このユーティリティの使用方法については、mkqdisk(8) の man ページをご覧ください。

注記

JBOD を quorum disk として使用することは推奨されません。JBOD は信頼できるパフォーマンスを提供できないため、ノードがそれに迅速に書き込むことができない可能性があります。ノードが quorum disk デバイスに迅速に書き込むことができないと、ノードは誤ってクラスターから削除されます。

2.10. RED HAT HIGH AVAILABILITY アドオンと SELINUX

Red Hat Enterprise Linux 6 対応の High Availability アドオンでは、SELinux ポリシータイプを **targeted** にセットした状態で **enforcing** の状態の SELinux をサポートします。

注記

仮想マシン環境で High Availability アドオンを使って SELinux を使用する場合、SELinux ブール値 **fenced_can_network_connect** が永続的に **on** に設定されていることを確認してください。この設定により、**fence_xvm** フェンシングエージェントが正常に機能し、システムによる仮想マシンのフェンシングが可能になります。

SELinux の詳細については、Red Hat Enterprise Linux 6 の 『導入ガイド』 を参照してください。

2.11. マルチキャストアドレス

クラスター内のノードは、マルチキャストアドレスを使用して通信します。そのため、各ネットワークスイッチとRed Hat High Availability アドオン内の関連のネットワーク装置は、マルチキャストアドレスを有効化し、IGMP (Internet Group Management Protocol) に対応するように設定する必要があります。各ネットワークスイッチと Red Hat High Availability アドオン内の関連のネットワーク装置がマルチキャストアドレスと IGMP に対応可能であることを確認します。対応している場合は、マルチキャストアドレスと IGMP を有効にします。マルチキャストと IGMP がないと、クラスターに参加できないノードが出てきてしまい、クラスターに問題が発生する可能性があります。「UDP ユニキャストトラフィック」 で記載されているように、このような環境では UDP ユニキャストを使用してください。

注記

ネットワークスイッチ及び関連ネットワーキング機器を設定する手順は、各製品によって異なります。マルチキャストアドレスと IGMP を有効にするためのネットワークスイッチ及び関連ネットワーキング機器の設定については、該当するベンダーのドキュメントやその他の情報を参照してください。

2.12. UDP ユニキャストトラフィック

Red Hat Enterprise Linux 6.2 リリースでは、クラスター内のノードは UDP ユニキャストトランスポートのメカニズムを使用することで通信可能となっています。ただし、クラスターネットワークについては IP マルチキャストの使用を推奨してます。UDP ユニキャストは、IP マルチキャストを使用できない場合に活用する代替手段となります。

cluster.conf 設定ファイルの **cman transport="udpu"** パラメーターを設定することによって、UDP ユニキャストを使用するように Red Hat High-Availability アドオンを設定できます。また、「ネットワークの設定」 の説明のとおり **Conga** ユーザーインターフェースの **Network Configuration** ページからユニキャストを指定することも可能です。

2.13. RICCI の注意事項

Red Hat Enterprise Linux 6 では、**ricci** が **ccsd** に置き換わります。そのため、各クラスターノードで **ricci** が実行している必要があります。この目的は、**ricci** が **cman_tool version -r** コマンド、**ccs** コマンド、又は **luci** ユーザーインターフェースサーバーを介していようと、更新されたクラスター設定を伝播するためです。**ricci** を起動するには、**service ricci start** を使用するか、**chkconfig** を使ってブート時の起動を有効にします。**ricci** の IP ポートを有効にする方法については 「クラスターノードでの IP ポートの有効化」 を参照してください。

Red Hat Enterprise Linux 6.1 リリース以降では、いずれかの特定のノードから更新したクラスター設定を初めて伝播する時に **ricci** を使用するにはパスワードが必要です。使用しているシステムに **ricci** をインストール後、**passwd ricci** コマンドを使って root としてユーザー **ricci** に **ricci** のパスワードを設定します。

2.14. クラスター化環境での仮想マシンの設定

仮想マシンリソースでクラスターを設定する場合、仮想マシンを起動/停止するには **rgmanager** ツールを使用することをお勧めします。**virsh** を使用してマシンを起動すると、複数の場所で仮想マシンが実行することにつながり、仮想マシン内のデータが破損する恐れがあります。

管理者がクラスター環境にあるクラスターツールと非クラスターツールの両方を使用して、仮想マシンを同時に 2 台起動してしまう可能性を減らすため、デフォルトではない場所に仮想マシンの設定ファイルを保存してシステムを構成するようにしてください。デフォルト以外の場所で仮想マシンの設定ファイルを保存すると、**virsh** ではこの設定ファイルを設定しない限り認識されないため、**virsh** を使用することで誤って仮想マシンを起動しにくくなります。

仮想マシンの設定ファイル用のデフォルト以外の場所は、どこでも可能です。NFS 共有又は共有 GSF2 ファイルシステムを使用する利点は、管理者がクラスターメンバー全体に設定ファイルを同期し続ける必要がない点です。ただし、管理者が何らかの方法でクラスター全体でコンテンツを同期し続ける限りは、ローカルディレクトリを使用することも許可されます。

クラスター設定では、仮想マシンはそのリソースの **path** の属性を使用することで、デフォルト以外の場所を参照できます。**path** 属性はディレクトリ又はコロン (:) の記号で区切られたディレクトリのセットであって、特定のファイルへのパスではない点に注意してください。

警告

libvirt-guests サービスは、**rgmanager** を実行している全ノードで無効にすることをお勧めします。仮想マシンが自動起動/再開すると、複数の場所で仮想マシンが実行することにつながり、仮想マシン内のデータが破損する恐れがあります。

仮想マシンのリソースの属性についての詳細は 表B.24「仮想マシン (**vm** リソースResource)」 を参照してください。

第3章 CONGA を使用した RED HAT HIGH AVAILABILITY アドオンの設定

本章では、**Conga** を使用した Red Hat High Availability アドオンソフトウェアの設定方法を説明します。**Conga** を使用して実行しているクラスターを管理する場合は、*4章Conga を使用した Red Hat High Availability アドオンの管理* を参照してください。

注記

Conga は Red Hat High Availability アドオンの管理に使用可能なグラフィカルユーザーインターフェースです。ただし、このインターフェースを効果的に使用するには、その根底にある概念を適切かつ明確に理解しておく必要があります。クラスター設定を学ぶ方法として、ユーザーインターフェース内の利用可能な機能を探すことは推奨されません。コンポーネントに障害が発生した場合に全サービスを実行し続けるほどシステムが堅牢でない場合があるためです。

本章は以下のセクションで構成されます。

- 「設定のタスク」
- 「**luci** の起動」
- 「luci へのアクセス制御」
- 「クラスターの作成」
- 「グローバルクラスタープロパティ」
- 「フェンスデバイスの設定」
- 「クラスターメンバー用のフェンシングの設定」
- 「フェイルオーバードメインの設定」
- 「グローバルクラスターリソースの設定」
- 「クラスターへのクラスターサービスの追加」

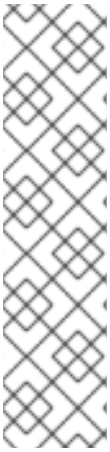
3.1. 設定のタスク

Conga を使用した Red Hat High Availability アドオンソフトウェアの設定は、以下の手順で行います：

1. **Conga** 設定ユーザーインターフェース — **luci** サーバーの設定と実行。「**luci** の起動」 を参照してください。
2. クラスターを作成。「クラスターの作成」 を参照してください。
3. グローバルクラスタープロパティを設定。「グローバルクラスタープロパティ」 を参照してください。
4. フェンスデバイスを設定。「フェンスデバイスの設定」 を参照してください。
5. クラスターメンバー用にフェンシングを設定。「クラスターメンバー用のフェンシングの設定」 を参照してください。

6. フェイルオーバードメインを作成。「フェイルオーバードメインの設定」 を参照してください。
7. リソースを作成。「グローバルクラスターリソースの設定」 を参照してください。
8. クラスターサービスを作成。「クラスターへのクラスターサービスの追加」 を参照してください。

3.2. LUCI の起動



注記

luci を使用してクラスターを設定するには、「**ricci** の注意事項」 で説明のとおり **ricci** がインストールされ、クラスターノード上で稼働している必要があります。そのセクションで記述していますが、**ricci** を使用する場合は 「クラスターの作成」 の説明のとおり、クラスター作成時に各クラスターノード用に **luci** が要求するパスワードが必要です。

luci の起動前に、**luci** が今後通信するいずれかのノード上で **luci** サーバーからポート 11111 への接続を、使用しているクラスターノードの IP ポートが許可することを確認するようにしてください。クラスターノード上で IP ポートを有効にする方法の詳細については 「クラスターノードでの IP ポートの有効化」 を参照してください。

Conga を使用して Red Hat High Availability アドオンを管理するには、以下のようにして **luci** をインストールして実行します：

1. **luci** をホストするコンピューターを選択して、そのコンピューターに **luci** ソフトウェアをインストールします。例えば：

```
# yum install luci
```

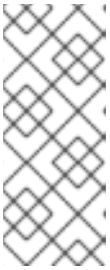
注記

通常は、サーバーケージ又はデータセンター内のコンピューターが **luci** をホストします。ただし、クラスターコンピューターは **luci** をホストできます。

2. **service luci start** を使用して **luci** を開始します。例えば：

```
# service luci start
Starting luci: generating https SSL certificates...  done
                                                           [  OK
]

Please, point your web browser to https://nano-01:8084 to access
luci
```



注記

Red Hat Enterprise Linux release 6.1 以降、**/etc/sysconfig/luci** ファイルを使用してポートやホストパラメーターなど **luci** の動作を設定できるようになりました。詳細は 「**/etc/sysconfig/luci** を使用した **luci** の設定」 を参照してください。修正したポートやホストパラメーターは、**luci** サービスの起動時に表示される URL に自動的に反映されます。

3. Web ブラウザーで、**luci** サーバーの URL を URL アドレスバーに指定して、**Go** (それに該当するもの) をクリックします。**luci** サーバーの URL 構文は、**https://*luci_server_hostname*:*luci_server_port*** で、*luci_server_port* のデフォルト値は **8084** となっています。

   luci への初回アクセス時には、(**luci** サーバーの) 自己署名 SSL 証明書に関するウェブブラウザーのプロンプトが表示されます。ダイアログボックスが確認されると、ウェブブラウザーに **luci** ログインページが表示されます。

4. **luci** をホストしているシステム上で認証できる全ユーザーは **luci** にログインできます。ただし、Red Hat Enterprise Linux 6.2 以降、管理者 (root ユーザー又は管理者パーミッションを持つユーザー) がユーザーにパーミッションを設定するまでは、**luci** を実行しているシステムの root ユーザーのみが **luci** コンポーネントにアクセスできます。ユーザーに **luci** のパーミッションを設定する詳細については、「luci へのアクセス制御」 を参照してください。

   luci にログインすると、図3.1「luci Homebase ページ」 のように **luci Homebase** ページが表示されます。

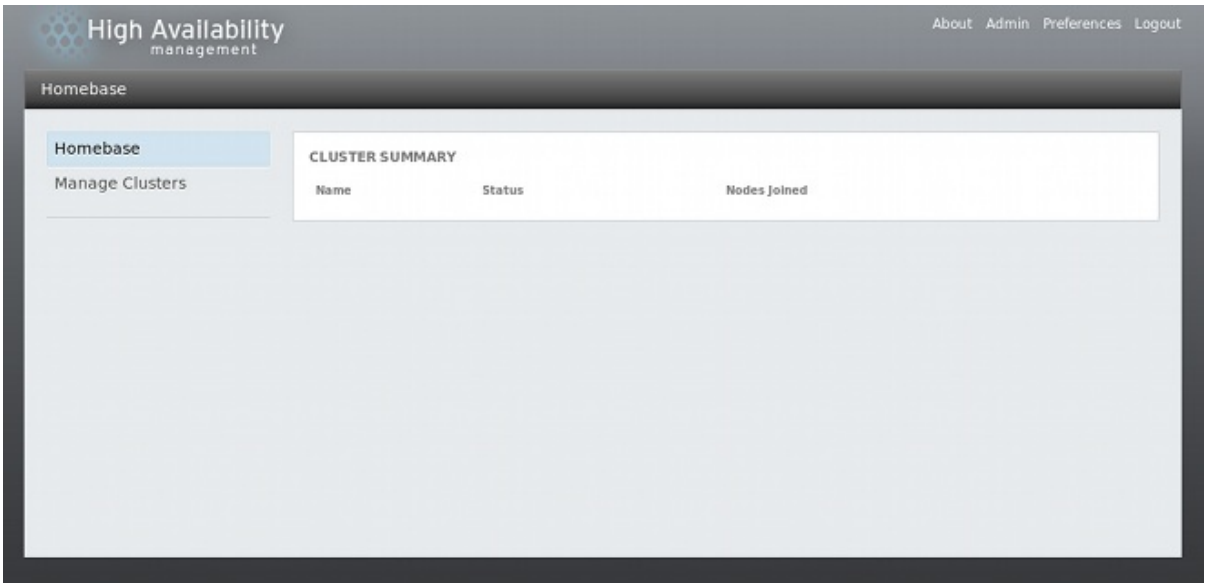


図3.1 luci Homebase ページ

注記

luci には 15 分間操作が行われないとログアウトするアイドルタイムアウトがあります。

3.3. LUCI へのアクセス制御

Red Hat Enterprise Linux 6 の初期リリースから、以下の機能が **Users and Permissions** ページに追加されています。

- Red Hat Enterprise Linux 6.2 では、root ユーザーまたは **luci** アプリケーションを実行しているシステムで **luci** 管理者権限を持つユーザーは、システム上の各ユーザーにパーミッションを設定することで、様々な **luci** コンポーネントへのアクセスを制御することができます。
- Red Hat Enterprise Linux 6.3 では、root ユーザーまたは **luci** 管理者のパーミッションを付与されたユーザーは、**luci** インターフェースにユーザーを追加し、これらのユーザーのユーザーパーミッションを設定できるようになりました。これらユーザーをシステムに追加してパスワードの設定をする必要はありますが、この機能により、このユーザーが **luci** に初めてログインする前にユーザーのパーミッションを設定できるようになりました。
- Red Hat Enterprise Linux 6.4 では、root ユーザーまたは **luci** 管理者権限を持つユーザーは、**luci** インターフェースを使用してユーザーを **luci** インターフェースから削除できるようになりました。これでそのユーザーに設定されたパーミッションがリセットされます。

注記

システム上の **/etc/pam.d/luci** ファイルを編集することで、**luci** による認証動作を修正することができます。Linux-PAM の使用方法に関しては、**pam**(8) man ページを参照してください。

ユーザーの追加、削除、ユーザーのパーミッションを設定するには、**root** または事前に管理者権限を付与されたユーザーでログインして、**luci** 画面の右上隅の**Admin** セクションをクリックしてください。すると、**Users and Permissions** ページが表示され、既存ユーザーが表示されます。

luci インターフェイスにユーザーを追加するには、**Add a User** をクリックして追加するユーザー名を入力します。この追加されたユーザーについてはパスワードを設定する必要はありますが、パーミッションの設定が可能になります。

ユーザーを **luci** インターフェイスから削除して設定したパーミッションをリセットするには、該当のユーザーを選択し、**Delete Selected** をクリックします。

ユーザーのパーミッションを設定または変更するには、**User Permissions** のドロップダウンメニューからユーザーを選択します。ここから、以下のパーミッションを設定することができます。

Luci Administrator

ユーザーに root ユーザーと同じパーミッションを付与します。全クラスター上の完全なパーミッションであり、root 以外の他の全ユーザーのパーミッションを設定/削除することができます。これらのパーミッションは制限できません。

Can Create Clusters

「クラスターの作成」 の説明のとおり、ユーザーは新規クラスターを作成できるようになります。

Can Import Existing Clusters

「luci インターフェースへの既存クラスターの追加」 の説明のとおり、ユーザーは既存のクラスターを **luci** インターフェースに追加できるようになります。

作成又は **luci** にインポートされた各クラスターについては、ユーザーに対して以下のパーミッションを設定できます。

Can View This Cluster

ユーザーは特定のクラスターを閲覧できるようになります。

Can Change the Cluster Configuration

ユーザーは特定のクラスターの設定を修正できるようになります。ただし、クラスターノードの追加および削除は対象外です。

Can Enable, Disable, Relocate, and Migrate Service Groups

「高可用性サービスの管理」 の説明のとおり、ユーザーは高可用性サービスを管理できるようになります。

Can Stop, Start, and Reboot Cluster Nodes

「クラスターノードの管理」 の説明のとおり、ユーザーはクラスターの個々のノードを管理できるようになります。

Can Add and Delete Nodes

「クラスターの作成」 の説明のとおり、ユーザーはクラスターからノードの追加/削除ができるようになります。

Can Remove This Cluster from Luci

「クラスターの起動、停止、再起動、削除」 の説明のとおり、ユーザーは **luci** インターフェースからクラスターを削除できるようになります。

Submit (提出) をクリックすると、パーミッションが反映されます。初期値に戻したい場合は **Reset** をクリックしてください。

3.4. クラスターの作成

luci を使用してクラスターを作成するには、クラスターへの名前の割り当て、クラスターへのクラスターノードの追加、各ノード用の **ricci** パスワードの入力、 クラスターを作成する要求の提出を行います。ノード情報とパスワードが正しければ、**Conga** は (適切なソフトウェアパッケージが現在インストールされていない場合は) 自動的にソフトウェアをクラスターノードにインストールして、クラスターを起動します。クラスターの作成方法は、以下のとおりです。

1. **luci Homebase** ページの左側にあるメニューから **Manage Clusters (クラスターの管理)** をクリックします。図3.2「luci のクラスター管理のページ」 に示してあるように、**Clusters (クラスター)** 画面が表示されます。

図3.2 luci のクラスター管理のページ

2. **Create (作成)**をクリックします。そうすると 図3.3「luci のクラスター作成のダイアログボックス」 に示してあるように **Create New Cluster (新規クラスターの作成)** ダイアログボックスが表示されます。

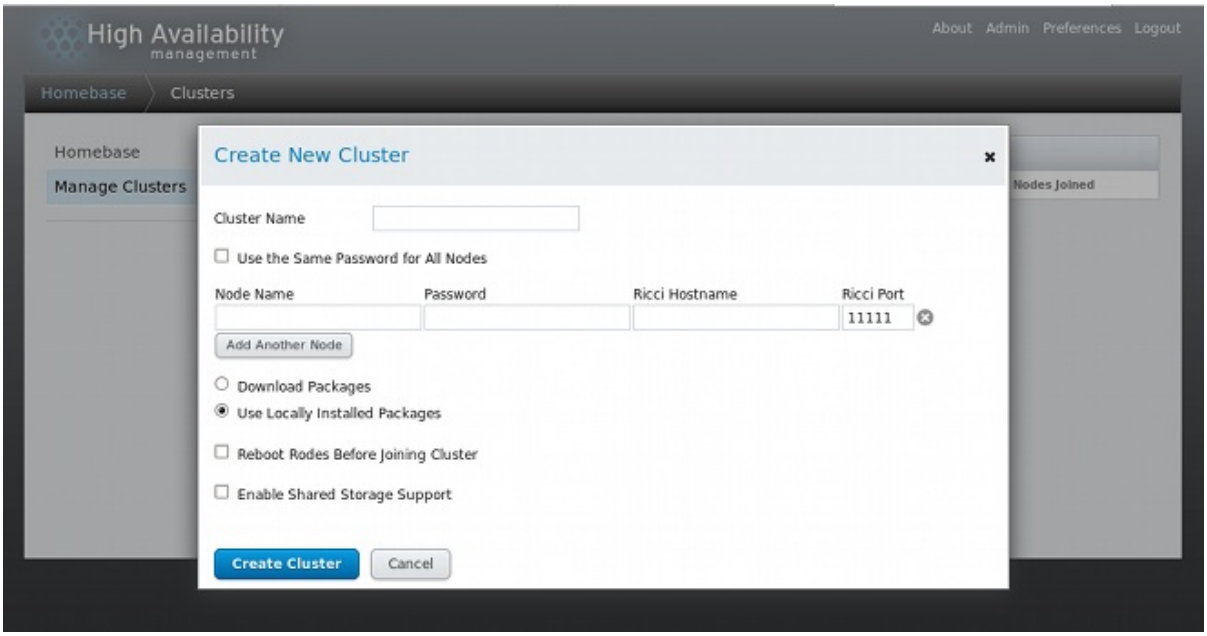


図3.3 luci のクラスター作成のダイアログボックス

3. 必要に応じて、**Create New Cluster** ダイアログボックスに以下のパラメーターを入力します：

   - **Cluster Name (クラスター名)** のテキストボックスに、クラスター名を入力します。クラスター名は 15 文字以下にしてください。

   - クラスター内の各ノードが同じ **ricci** パスワードである場合、**Use same password for all nodes (全てのノードに同じパスワードを使用)** にチェックを入れると、ノード追加時に **password (パスワード)** フィールドに自動入力できます。

   - クラスター内のノードの名前を **Node Name (ノード名)** カラムに入力して、そのノード用の **ricci** パスワードを **Password** カラムに入力します。ノード名の長さは、最大 255 バイトまでになります。

- システムをクラスタートラフィック用のみに使用される専用のプライベートネットワークで設定している場合、**luci** はクラスターノード名が解決するアドレスとは異なるアドレス上の **ricci** と通信するように設定すると良いでしょう。これを行うには、アドレスを **Ricci Hostname** として入力します。

- **ricci** エージェント用にデフォルトの 11111 とは異なるポートを使用している場合は、そのパラメーターを変更できます。

- **Add Another Node (別のノードを追加)** をクリックして、クラスター内の追加ノード毎にノード名と **ricci** パスワードを入力します。

- クラスターを作成する時点でノードに既にインストールされているクラスターソフトウェアパッケージをアップグレードしたくない場合は、**Use locally installed packages（ローカルにインストールされたパッケージを使用）** オプションを選択します。全てのクラスターソフトウェアパッケージをアップグレードしたい場合は、**Download Packages (パッケージのダウンロード)** オプションを選択します。

注記

Use locally installed packages か **Download Packages** のオプションのどちらを選択するにしても、ベースのクラスターコンポーネントの一部が欠如している場合 (**cman**、**rgmanager**、**modcluster** 及びそれら全ての依存関係)、それらはインストールされます。インストールできなければ、ノードの作成は失敗します。

- **Reboot nodes before joining cluster** の選択は任意です。

- クラスター化ストレージが必要な場合は、**Enable shared storage support (共有ストレージサポートを有効にする)** を選択します。これにより、クラスター化ストレージをサポートするパッケージがダウンロードされ、クラスター化 LVM が有効になります。ただし、これは Resilient Storage アドオン、又は Scalable File System アドオンへのアクセスがある場合にのみ選択してください。

4. **クラスターの作成 (Create Cluster)** をクリックします。**クラスターの作成 (Create Cluster)** をクリックすると以下のアクションが発生します：

   1. **パッケージのダウンロード (Download Packages)** を選択した場合は、クラスターソフトウェアパッケージがノード上にダウンロードされます。

   2. クラスターソフトウェアがノードにインストールされます (又は、適切なソフトウェアパッケージがインストールされたことが確認されます)。

   3. クラスター設定ファイルが更新され、クラスター内の各ノードに伝播します。

   4. 追加されたノード群がクラスターに参加します。

   クラスターが作成中であることを知らせるメッセージが表示されます。クラスターの準備ができると、図3.4「クラスターノードの表示」 にあるように新規作成されたクラスターのステータスが表示されます。**ricci** がどのノードでも稼働していない場合は、クラスター作成は失敗することに注意して下さい。

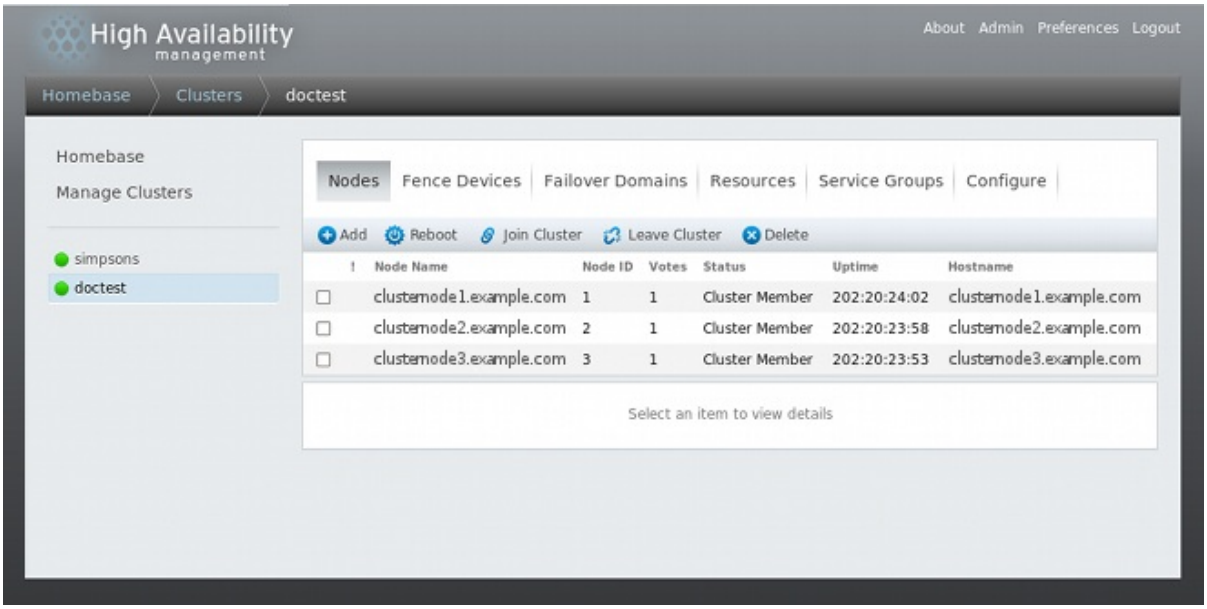


図3.4 クラスターノードの表示

5. クラスターを作成するために **Create Cluster（クラスターの作成）** をクリックした後に、クラスターノード表示ページの上部にあるメニューから **Add (追加)** 又は **Delete (削除)** の機能をクリックすることで、クラスターに対するノードの追加/削除ができます。クラスター全体を削除しない限り、ノードを削除するにはまずノードを停止する必要があります。現在稼働中の既存のクラスターからノードを削除する方法については 「クラスターからのメンバーの削除」を参照してください。

注記

クラスターからクラスターノードの削除は、元に戻すことが不可能な破壊的な操作です。

3.5. グローバルクラスタープロパティ

設定するクラスターを選択すると、クラスター固有のページが表示されます。このページには、クラスター単位でプロパティを設定できるインターフェースがあります。クラスター別のプロパティを設定するには、クラスター表示の上部にある **Configure** をクリックしてください。すると、インターフェースが **General**、**Fence Daemon**、**Network**、**Redundant Ring**、**QDisk**、**Logging** のようにタブに分けられます。これらのタブでパラメーターを設定するには、以下のセクションの手順に従います。パラメーター設定の必要がないタブがある場合は、そのタブのセクションは省略してください。

3.5.1. 全般プロパティの設定

General（全般） のタブをクリックすると **General Properties（全般プロパティ）** のページが表示され、設定バージョンを変更するためのインターフェースが表示されます。

- **Cluster Name（クラスター名）** テキストボックスは、クラスター名を表示します。ここでは、クラスター名を変更することはできません。クラスター名を変更する唯一の方法は、新しい名前で新規にクラスター設定を作成することです。

- **Configuration Version（設定バージョン）** の値は、クラスターの作成時に **1** にセットされており、クラスター設定を変更する度に自動的に増加します。ただし、別の値にセットする必要がある場合は、**Configuration Version** テキストボックスで指定できます。

Configuration Version の値を変更した場合は、**Apply（適用）** をクリックして変更を反映させます。

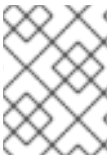
3.5.2. フェンスデーモンプロパティの設定

Fence Daemon（フェンスデーモン） タブをクリックすると、**Fence Daemon Properties** ページが表示されます。ここでは、**Post Fail Delay（失敗後の待機）** 及び **Post Join Delay（参加後の待機）** を設定するためのインターフェースを提供します。これらのパラメーターに設定する値はクラスターの一般的なフェンシングプロパティです。クラスター内のノード群に特定のフェンスデバイスを設定するには、「フェンスデバイスの設定」 で示してあるように、クラスターディスプレイの **Fence Devices（フェンスデバイス）** メニュー項目を使用します。

- **Post Fail Delay** パラメーターは、ノードが失敗後、ノード (フェンスドメインのメンバー) をフェンシングするまでにフェンスデーモン (**fenced**) が待機する秒数です。**Post Fail Delay** のデフォルト値は、**0** です。この値はクラスターとネットワークのパフォーマンスに合わせて変更できます。

- **Post Join Delay** パラメーターは、ノードが fence ドメインを結合してからノードをフェンシングするまでの fence デーモン (**fenced**) の待機時間 (秒数) です。**luci** は、**Post Join Delay** の値を **3** に設定します。一般的な **Post Join Delay** の値は 20 秒から 30 秒の間となっていますが、クラスターやネットワークのパフォーマンスに左右されます。

必要な値を入力して **Apply** をクリックすると変更が反映されます。



注記

Post Join Delay 及び **Post Fail Delay** についての詳細は、fenced(8) の man ページを参照してください。

3.5.3. ネットワークの設定

Network （ネットワーク） タブをクリックすると、**Network Configuration（ネットワークの設定）** ページが表示されます。ここでは、ネットワークのトランスポートタイプを設定するためのインターフェースを提供します。

このタブを使用して、以下のオプションから 1 つ選択します：

- **UDP Multicast and Let Cluster Choose the Multicast Address（UDP マルチキャスト、クラスターがマルチキャストアドレスを選択）**

  これがデフォルト設定です。このオプションが選択されている場合、Red Hat High Availability アドオンソフトウェアはクラスター ID を基にしてマルチキャストアドレスを作成します。アドレスの下位 16 ビットを生成して、それを IP プロトコルが IPV4 又は IPV6 であるかに応じてアドレスの上部に追加します：

  - IPV4 の場合 — 形成されるアドレスは、239.192 に Red Hat High Availability アドオンソフトウェアにより生成される下位 16 ビットが加わります。

  - IPV6 の場合 — 形成されるアドレスは、FF15:: に Red Hat High Availability アドオンソフトウェアにより生成される下位 16 ビットが加わります。

注記

クラスター ID は、各クラスター用に **cman** が生成する一意の識別子です。クラスター ID を表示するには、1 つのクラスターノードで **cman_tool status** コマンドを実行します。

- **UDP Multicast and Specify the Multicast Address Manually（UDP マルチキャスト、マルチキャストアドレスを手動で指定）**

  特定のマルチキャストアドレスを使用する必要がある場合は、このオプションを選択して **Multicast Address** テキストボックスにマルチキャストアドレスを入力します。

  マルチキャストアドレスを指定する場合は、**cman** が使用する 239.192.x.x シリーズ（IPv6 の場合は FF15::）を使用することをお勧めします。この範囲以外のマルチキャストアドレスを使用すると、予期できない結果が生じる場合があります。例えば、224.0.0.x (「ネットワーク上の全ホスト」) を使用した場合、正しくルートされないか、一部のハードウェアでは全くルートされない可能性さえあります。

  マルチキャストアドレスを指定/修正した場合は、それを反映させるためにクラスターを再起動する必要があります。**Conga** を使ったクラスターの起動/停止に関する詳細は、「クラスターの起動、停止、再起動、削除」 を参照してください。

  注記

  マルチキャストアドレスを指定する場合、クラスターパケットが通過するルーターの設定を確認するようにしてください。一部のルーターはアドレスを認識するのに長時間かかり、クラスターのパフォーマンスに重大な影響を与える場合があります。

- **UDP Unicast (UDPU)**

  Red Hat Enterprise Linux 6.2 リリースでは、クラスター内のノードは UDP ユニキャストトランスポートのメカニズムを使用することで通信可能となっています。ただし、クラスターネットワークについては IP マルチキャストの使用を推奨してます。UDP ユニキャストは、IP マルチキャストを使用できない場合に活用する代替手段となります。UDP ユニキャストを使った GFS2 のデプロイメントは推奨されません。

Apply（適用） をクリックします。トランスポートタイプを変更した場合は、変更を反映させるためにクラスターの再起動が必要です。

3.5.4. 冗長リングプロトコルの設定

Red Hat Enterprise Linux 6.4 では、Red Hat High Availability アドオンはリングプロトコルの冗長設定に対応しています。冗長リングプロトコルを使用する場合、「冗長リングプロトコルの設定」 の記載にあるようにさまざまな点を考慮する必要があります。

Redundant Ring タブをクリックすると、**Redundant Ring Protocol Configuration** ページが表示されます。このページでは、クラスターに対して現在設定されているノードがすべて表示されます。冗長リングプロトコルを使用するようにシステム設定をする場合、2 つ目のリングの各ノードに **Alternate Name** を指定する必要があります。

Redundant Ring Protocol Configuration ページでは、オプションで 2 つ目のリングに対して **Alternate Ring Multicast Address**、**Alternate Ring CMAN Port**、 **Alternate Ring Multicast Packet TTL** を指定できます。

2 つ目のリングにマルチキャストアドレスを指定した場合、代わりのマルチキャストアドレスもしくは代わりのポートは 1 つ目のリングのマルチキャストアドレスと異なるものにする必要があります。代替ポートを指定した場合、システム自体は操作実行にポートとポート 1 を使用するので、1 つ目のリングと 2 つ目のリングのポート番号は、2 つ以上異なるものである必要があります。別のマルチキャストアドレスを指定しなかった場合、システムは 2 つ目のリングに異なるマルチキャストアドレスを自動的に使用するようになっています。

3.5.5. Quorum Disk の設定

QDisk タブをクリックすると、**Quorum Disk Configuration（クォーラムディスクの設定）** ページが表示されます。ここには、quorum disk のパラメーターを設定するインターフェースがあり、quorum disk を使用する必要があるかどうかを設定します。

注記

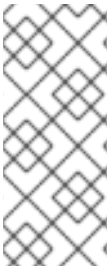
Quorum disk のパラメーター及びヒューリスティックは、サイトの環境や必要となる特別な要件によって異なります。quorum disk のパラメーターとヒューリスティックの使用について理解するには、qdisk(5) の man ページを参照してください。Quorum disk の理解と使用にサポートが必要な場合は、Red Hat 認定サポート担当者までご連絡ください。

デフォルトでは、**Do Not Use a Quorum Disk (クォーラムディスクを使用しない)** パラメーターが有効です。Quorum disk を使用する必要がある場合は、**Use a Quorum Disk (クォーラムディスクを使用する)** をクリックして、quorum disk のパラメーターを入力し、**Apply** をクリックします。変更を反映させるためにクラスターを再起動します。

表3.1「Quorum disk のパラメーター」 は、quorum disk のパラメーターについて説明しています。

表3.1 Quorum disk のパラメーター

パラメーター	説明
Specify Physical Device: By Device Label	**mkqdisk** ユーティリティによって作成される quorum disk のラベルを指定します。このフィールドが使用されると、quorum デーモンは **/proc/partitions** ファイルを読み取り、各ブロックデバイスの qdisk 署名を確認して、指定したラベルと比較します。これは、quorum デバイス名がノード間で異なる場合の設定に役立ちます。
Heuristics	**Path to Program** — このヒューリスティックが利用できるか判断する際に使用するプログラム。**/bin/sh -c** で実行可能なものなら何でも構いません。戻り値が 0 は成功で、それ以外の値は失敗を指します。このフォールドは必須フィールドです。 **Interval** — ヒューリスティックが投票される頻度 (秒単位) です。すべてのヒューリスティックのデフォルト間隔は 2 秒です。 **Score** — このヒューリスティックのウェイトです。ヒューリスティックのスコアを決定する時には注意が必要です。各ヒューリスティックのデフォルトのスコアは 1 です。 **TKO** — このヒューリスティックが利用不可能と宣言されるまでの連続する失敗数。
Minimum Total Score	ノードが「生存」と見なされるための最低限のスコア。スコアが省略されている場合や「0」にセットされていると、デフォルトの関数である **floor((*n*+1)/2)** が使用されます。*n* はヒューリスティックのスコアの合計です。**Minimum Total Score** の値は、ヒューリスティックのスコアの合計を超過してはいけません。超過した場合は、quorum disk は利用できません。



注記

QDisk Configuration タブの **Apply** をクリックすると、各クラスターノードのクラスター設定ファイル (**/etc/cluster/cluster.conf**) に変更を伝播します。ただし、quorum disk が動作する、あるいは quorum disk のパラメーターに行った変更が反映されるには、クラスターを再起動 (「クラスターの起動、停止、再起動、削除」 を参照) して、各ノードで **qdiskd** デーモンが再起動することを確認する必要があります。

3.5.6. ロギング設定

Logging（ロギング） タブをクリックすると、**Logging Configuration（ロギング設定）** ページが表示されます。このページは、ロギング設定を行うインターフェースを提供します。

グローバルロギング設定用に以下のセッティングを構成できます:

- **Log Debugging Messages** にチェックマークを入れると、ログファイル内のメッセージのデバッグを有効にします。
- **Log Messages to Syslog** にチェックマークを入れると、**syslog** へのメッセージを有効にします。**Syslog Message Facility** 及び **Syslog Message Priority** を選択できます。**Syslog Message Priority** 設定は、選択したレベル以上でメッセージが **syslog** に送信されることを意味しています。
- **Log Messages to Log File** にチェックマークを入れると、ログファイルへのメッセージを有効にします。**Log File Path** の名前を指定できます。**logfile message priority** 設定は、選択したレベル以上でメッセージがログファイルに書き込まれることを意味しています。

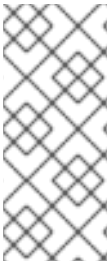
特定のデーモンに対するグローバルのロギング設定を上書きするには、**Logging Configuration** ページの最下部にある **Daemon-specific Logging Overrides** の見出しの下に一覧表示されているデーモンから 1 つ選択します。デーモンの選択後、そのデーモンにデバッグメッセージをログするかどうか確認できます。そのデーモンに関して、**syslog** とログファイルの設定を指定することも可能です。

Apply をクリックすると、指定したロギング設定の変更が反映されます。

3.6. フェンスデバイスの設定

フェンスデバイスの設定は、クラスター用のフェンスデバイスの作成、更新、削除で行います。クラスター内のノード群用にフェンシングを設定するには、まずクラスター内でフェンスデバイスを設定する必要があります。

フェンスデバイスを作成するには、フェンスデバイスタイプを選択し、そのフェンスデバイス用のパラメーター (例えば、名前、IP アドレス、ログイン、パスワード) を入力します。フェンスデバイスを更新するには、既存のフェンスデバイスを選択して、そのフェンスデバイス用のパラメーターを変更します。フェンスデバイスを削除するには、既存のフェンスデバイスを選択して削除します。



注記

各ノードに複数のフェンシングメカニズムを設定することが推奨されます。フェンシングデバイスが失敗する要因には、ネットワークの分割、電源異常、フェンシングデバイスそのものに問題がある場合、などがあります。複数のフェンシングメカニズムを設定することで、1 つのフェンシングデバイスの障害が致命的な結果になる可能性を低減することができます。

このセクションでは、以下のタスクを行うための手順を説明しています：

- フェンスデバイスの作成 —「フェンスデバイスの作成」 を参照してください。フェンスデバイスを作成して命名した後は、「クラスターメンバー用のフェンシングの設定」 に示してあるようにクラスター内の各ノード用にフェンスデバイスを設定できます。
- フェンスデバイスの更新 —「フェンスデバイスの修正」 を参照してください。
- フェンスデバイスの削除 —「フェンスデバイスの削除」 を参照してください。

クラスター固有のページから、クラスターディスプレイの上部にある **Fence Devices（フェンスデバイス）** をクリックすることで、クラスターのフェンスデバイスを設定できます。これにより、クラスター用のフェンスデバイスが表示され、フェンスデバイス設定用のメニュー項目である **Add** 及び **Delete** が表示されます。これが以下のセクションで説明する各手順の開始点となります。

注記

初めてクラスターを設定する場合、フェンスデバイスは作成されていないため、何も表示されません。

図3.5「luci フェンスデバイスの設定ページ」 は、フェンスデバイスが作成されていない時のフェンスデバイス設定画面を示しています。

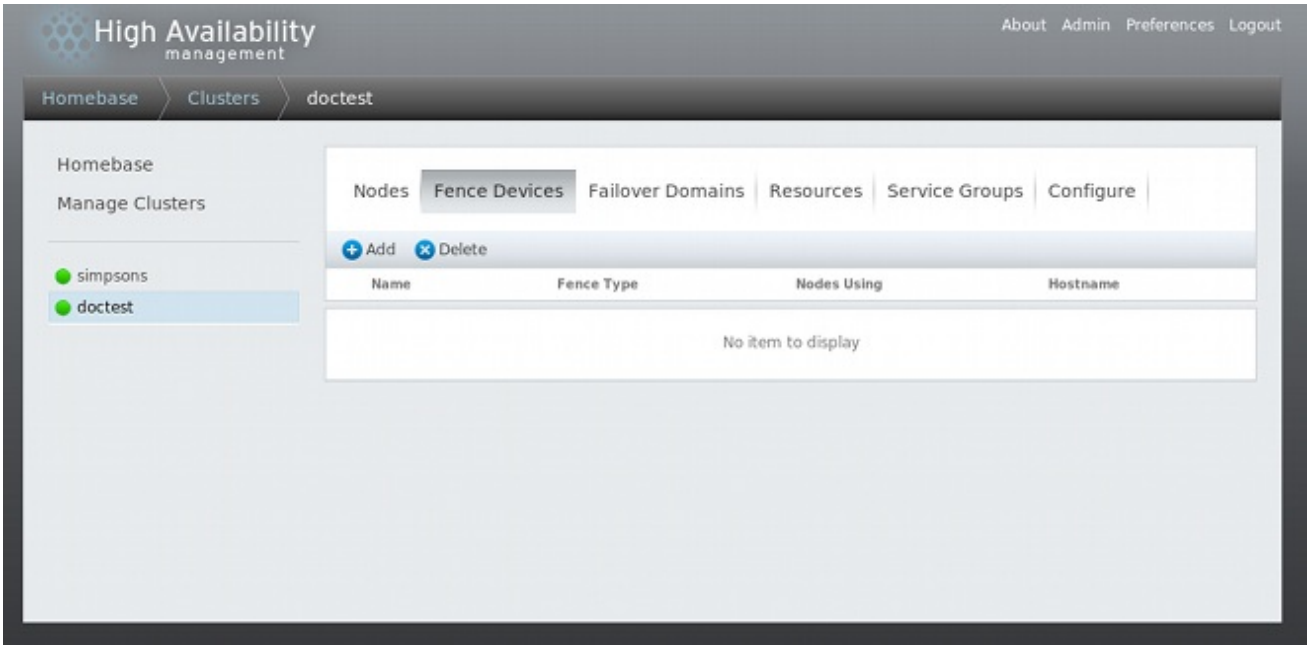


図3.5 luci フェンスデバイスの設定ページ

3.6.1. フェンスデバイスの作成

フェンスデバイスを作成するには、以下の手順に従います：

1. **Fence Devices（フェンスデバイス）** 設定ページから、**Add（追加）** をクリックします。**Add** をクリックすると、**Add Fence Device (Instance)（フェンスデバイス（インスタンス）の追加）** ダイアログボックスが表示されます。このダイアログボックスから、設定するフェンスデバイスのタイプを選択します。
2. フェンスデバイスのタイプに応じて、**Add Fence Device (Instance)** ダイアログボックス内の情報を指定します。フェンスデバイスパラメーターについての詳細情報は、付録A フェンスデバイスパラメーター を参照してください。「クラスターメンバー用のフェンシングの設

定」 で示してあるように、一部のケースでは、個別のノード用にフェンスを設定する時にはフェンスデバイスに対してノード特有のパラメーターを追加で指定する必要があります。

3. **Submit** をクリックします。

フェンスデバイスが追加されると、**Fence Devices（フェンスデバイス群）** 設定ページ上に表示されます。

3.6.2. フェンスデバイスの修正

フェンスデバイスを修正するには、以下の手順に従います：

1. **Fence Devices** 設定ページから、修正したいフェンスデバイスの名前をクリックします。そのデバイス用に設定された値が示されたフェンスデバイスのダイアログボックスが表示されます。

2. フェンスデバイスを修正するには、表示されたパラメーターへの変更を入力します。詳細は 付録A *フェンスデバイスパラメーター* を参照してください。

3. **Apply（適用）** をクリックして、設定が更新されるのを待ちます。

3.6.3. フェンスデバイスの削除

注記

使用中のフェンスデバイスは削除できません。ノードが現在使用しているフェンスデバイスを削除するには、そのデバイスを使用しているノードのノードフェンス設定を最初に更新してから、デバイスを削除してください。

フェンスデバイスを削除するには、以下の手順に従ってください：

1. **Fence Devices** 設定のページから、フェンスデバイスの左にあるボックスにチェックマークを入れて、削除するデバイスを選択します。

2. **Delete（削除）** をクリックして設定が更新されるのを待ちます。デバイスが削除中であることを示すメッセージが表示されます。

設定が更新されると、削除されたデバイスは表示されなくなります。

3.7. クラスターメンバー用のフェンシングの設定

クラスター作成とフェンスデバイス作成の最初の手順が完了した後は、クラスターノード用のフェンシングを設定する必要があります。新規のクラスターを作成してそのクラスター用のフェンスデバイスを設定した後にノード用のフェンシングを設定するには、このセクションの手順に従ってください。クラスター内の各ノード用にフェンシングを設定する必要がある点に注意して下さい。

以下のセクションでは、各ノード用の単一フェンスデバイスの設定、バックアップフェンスデバイスを持つノードの設定、冗長電源装置を持つノードの設定を行う手順を説明しています。

- 「単一ノードに対する単一フェンスデバイスの設定」
- 「バックアップフェンスデバイスの設定」
- 「冗長電源装置を持つノードの設定」

3.7.1. 単一ノードに対する単一フェンスデバイスの設定

以下の手順を使用して、単一のフェンスデバイスを持つノードを設定します。

1. クラスター固有のページから、クラスターディスプレイの上部にある **Nodes（ノード群）** をクリックすることで、クラスター内のノード群のフェンシングを設定できます。これにより、クラスターを構成するノード群が表示されます。これはまた、**luci Homebase** ページの左側のメニューから **Manage Clusters（クラスターの管理）** の下にあるクラスター名をクリックした時に表示されるデフォルトのページでもあります。

2. 任意のノード名をクリックします。ノードのリンクをクリックすると、そのノードが設定された方法を示すリンク先のページが表示されます。

   ノード固有のページは、ノード上で現在稼働しているサービスと、そのノードがメンバーとなっているフェイルオーバードメインを表示します。既存のフェイルオーバードメインを修正するには、その名前をクリックします。フェイルオーバードメインの設定についての詳細は、「フェイルオーバードメインの設定」 をご覧ください。

3. ノード固有のページで、**Fence Devices** の下にある **Add Fence Method（フェンスメソッドの追加）** をクリックします。これにより、**Add Fence Method to Node（フェンスメソッドをノードに追加）** ダイアログボックスが表示されます。

4. このノード用に設定しているフェンシングメソッドの **Method Name（メソッド名）** を入力します。 これは Red Hat High Availability アドオンで使用される任意の名前です。これはデバイスの DNS 名と同じではありません。

5. **Submit** をクリックします。そうすると **Fence Devices** の下に先ほど追加したメソッドを表示するノード固有の画面が表示されます。

6. このメソッド用のフェンスインスタンスを設定するには、フェンスメソッドの下に表示される **Add Fence Instance（フェンスインスタンスの追加）** ボタンをクリックします。これにより、「フェンスデバイスの作成」 の説明のとおり、以前に設定したフェンスデバイスを選択できる **Add Fence Device (Instance)** ドロップダウンメニューが表示されます。

7. このメソッド用にフェンスデバイスを選択します。このフェンスデバイスにノード固有のパラメーターを設定する必要がある場合は、設定すべきパラメーターがディスプレイに表示されます。フェンシングパラメーターの詳細については 付録A *フェンスデバイスパラメーター* を参照してください。

注記

パワーフェンス以外のメソッド (つまり、SAN/ストレージフェンシング) 用には、ノード特定のパラメーター画面で **Unfencing** がデフォルト選択されます。これにより、フェンス済みのノードのストレージへのアクセスは、再起動されるまでは再度有効になりません。アンフェンシングを必要とするデバイスを設定する際には、最初にクラスターを停止し、デバイスおよびアンフェンシングを含むすべての設定をクラスターが開始される前に追加する必要があります。ノードのアンフェンシングに関する詳細は、**fence_node**(8) の man ページを参照してください。

8. **Submit** をクリックします。これにより、フェンスメソッドとフェンスインスタンスを表示するノード固有の画面に戻ります。

3.7.2. バックアップフェンスデバイスの設定

1 つのノードに対して複数のフェンシングメソッドを定義できます。1 番目のメソッドを使用してフェンシングが失敗すると、システムは 2 番目のメソッドを使用してノードのフェンスを試行します。その後に、追加で設定したメソッドが使用されます。

以下の手順を使用して、単一のノードにバックアップフェンスデバイスを設定します。

1. 「単一ノードに対する単一フェンスデバイスの設定」 に記載されている手順に従って、ノードにプライマリフェンシングメソッドを設定します。

2. 定義したプライマリメソッドのディスプレイで、**Add Fence Method** をクリックします。

3. このノードに設定するバックアップフェンシングメソッドの名前を入力して、**Submit** をクリックします。これにより、先ほどプライマリフェンスメソッドの下に追加したメソッドを示すノード固有の画面が表示されます。

4. **Add Fence Instance** をクリックしてこのメソッドにフェンスインスタンスを設定します。これにより、「フェンスデバイスの作成」 の説明のとおり、以前に設定したフェンスデバイスを選択できるドロップダウンメニューが表示されます。

5. このメソッド用にフェンスデバイスを選択します。このフェンスデバイスにノード固有のパラメーターを設定する必要がある場合は、設定すべきパラメーターがディスプレイに表示されます。フェンシングパラメーターの詳細については 付録A *フェンスデバイスパラメーター* を参照してください。

6. **Submit** をクリックします。これにより、フェンスメソッドとフェンスインスタンスを表示するノード固有の画面に戻ります。

必要に応じて、引き続きフェンシングメソッドを追加できます。**Move Up（上に移動）** や **Move Down（下に移動）** をクリックすることで、このノードに使用されるフェンシングメソッドの順番を再編成できます。

3.7.3. 冗長電源装置を持つノードの設定

使用中のクラスターがノード群用に冗長電源装置で設定されている場合は、そのノード群がフェンスされる必要があるときに完全にシャットダウンされるようにフェンシングを設定する必要があります。各電源装置を別々のフェンスメソッドとして設定すると、各電源装置は別々にフェンスされます。つまり、1 つ目の電源装置がフェンスされると、2 つ目の電源装置によりシステムは引き続き実行可能なため、システムは全くフェンスされません。二重電源装置によってシステムを設定するには、両方の電源装置が遮断されてシステムが完全に停止するようにフェンスデバイスを設定する必要があります。**Conga** を使用してシステムを設定する場合は、単一のフェンシングメソッド内で 2 つのインスタンスを設定する必要があります。

二重電源装置を持つノードにフェンシングを設定するには、このセクションの手順に従ってください。

1. 冗長電源を持つノードにフェンシングを設定するには、まず各電源スイッチをクラスター用のフェンスデバイスとして設定する必要があります。フェンスデバイスの設定に関する詳細は、「フェンスデバイスの設定」 を参照してください。

2. クラスター固有のページから、クラスターディスプレイの上部にある **ノード群（Nodes）** をクリックします。これにより、クラスターを構成するノード群が表示されます。これはまた、**luci Homebase** ページの左側のメニューから **Manage Clusters（クラスターの管理）** の下にあるクラスター名をクリックした時に表示されるデフォルトのページでもあります。

3. 任意のノード名をクリックします。ノードのリンクをクリックすると、そのノードが設定された方法を示すリンク先のページが表示されます。

4. ノード固有のページで、**Add Fence Method** をクリックします。

5. このノードに設定するフェンシングメソッドの名前を入力します。

6. **Submit** をクリックします。そうすると **Fence Devices** の下に先ほど追加したメソッドを表示するノード固有の画面が表示されます。

7. **Add Fence Instance** をクリックすることで、このメソッド用のフェンスインスタンスとして 1 つ目の電源装置を設定します。これにより、「フェンスデバイスの作成」 の説明のとおり、以前に設定したパワーフェンシングデバイスを選択できるドロップダウンメニューが表示されます。

8. このメソッドにパワーフェンスデバイスの 1 つを選択して、このデバイス用に適切なパラメーターを入力します。

9. **Submit** をクリックします。これにより、フェンスメソッドとフェンスインスタンスを表示するノード固有の画面に戻ります。

10. 1 つ目のパワーフェンシングデバイスを設定した同じフェンスメソッドの下で、**Add Fence Instance** をクリックします。これにより、「フェンスデバイスの作成」 の説明のとおり、以前に設定した 2 つ目のパワーフェンシングデバイスを選択できるドロップダウンメニューが表示されます。

11. このメソッドに 2 つ目のパワーフェンスデバイスを選択して、このデバイス用に適切なパラメーターを入力します。

12. **Submit** をクリックします。これで、フェンスメソッドとフェンスインスタンスを表示するノード固有の画面に戻り、各デバイスがシステム電源を連続でオフ/オンにすることを表示します。これは 図3.6「二重パワーフェンシング設定」 で示しています。

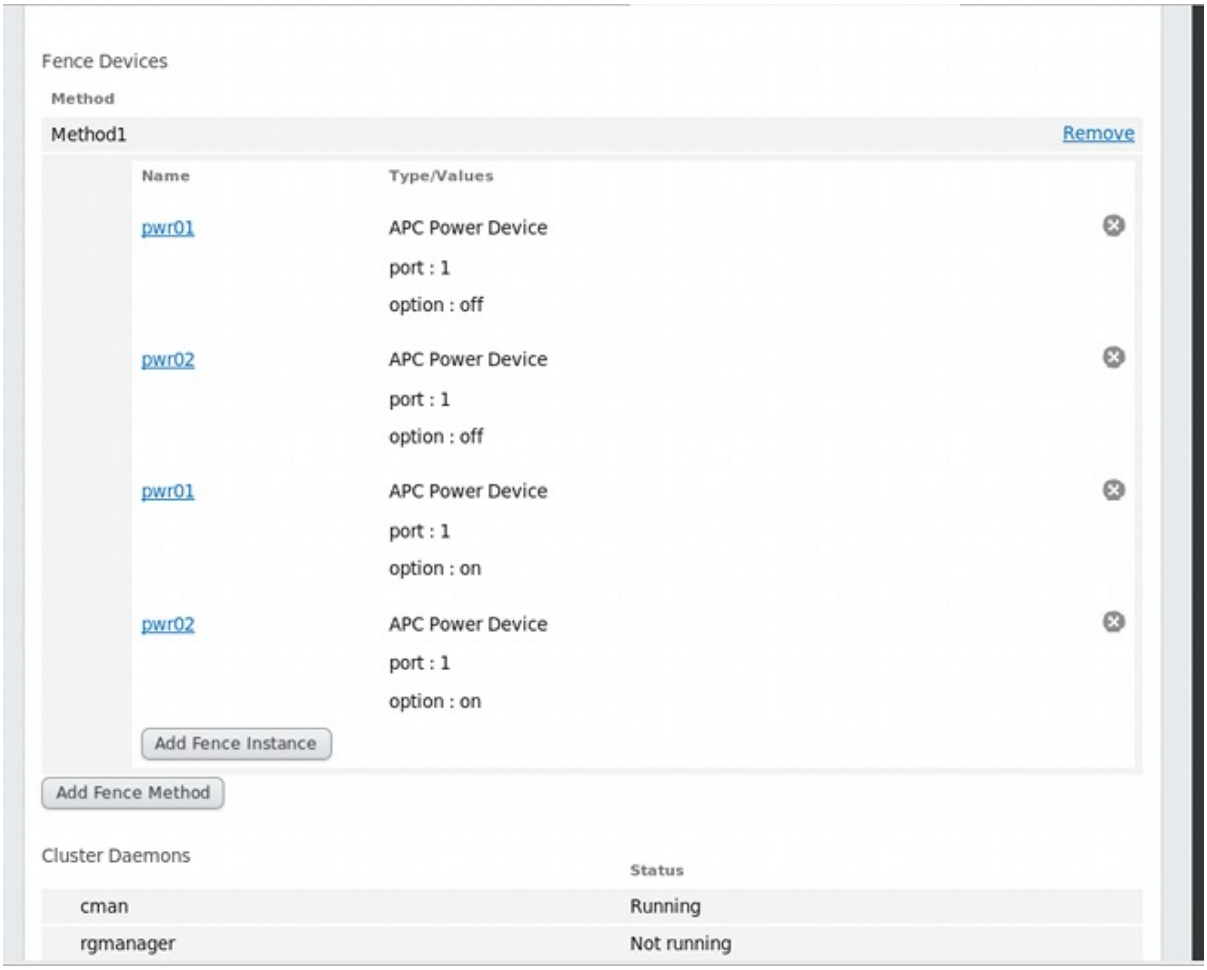


図3.6 二重パワーフェンシング設定

3.8. フェイルオーバードメインの設定

フェイルオーバードメインは、ノード障害時にクラスターサービスを実行するのに有効なクラスターノードの名前付きサブセットです。フェイルオーバードメインは以下の特性を持つことができます：

- Unrestricted（制限なし） — 優先するメンバーのサブセットを指定できるようにしますが、このドメインに割り当てられたクラスターサービスはいずれの利用可能なメンバーでも実行できます。

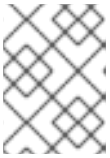
- Restricted（制限あり） — 特定のクラスターサービスを実行できるメンバーを制限できるようにします。制限されたフェイルオーバードメインのメンバーがどれも使用できない場合は、クラスターサービスを (手動あるいはクラスターソフトウェアによっても) 開始できません。

- Unordered（優先順なし） — クラスターサービスが優先順なしのフェイルオーバードメインへ割り当てられる場合、クラスターサービスを実行するメンバーは、優先順なしの利用可能なフェイルオーバードメインメンバーから選択されます。

- Ordered（優先順あり） — フェイルオーバードメインのメンバー間で優先順を指定できるようにします。一覧のトップにあるメンバーが最優先となり、次に一覧の 2 番目のメンバー、その後も順に続きます。

- Failback（フェイルバック） — フェイルオーバードメイン内のサービスが、ノード障害の前に元々実行していたノードへフェイルバックするかどうかを指定できるようにします。この特性の設定が役立つのは、ノードが繰り返し失敗し、それが優先順ありのフェイルオーバードメイ

ンの一部であるような状況です。この状況では、ノードがフェイルオーバードメイン内の優先ノードである場合には、優先ノードと別のノード間でサービスがフェイルオーバーとフェイルバックを繰り返す可能性があるため、パフォーマンスに重大な影響を与えることになります。



注記

優先順のあるフェイルオーバーが設定されている場合にのみ、フェイルバックの特性が利用できます。

注記

フェイルオーバードメイン設定を変更しても、現在実行中のサービスには影響しません。

注記

フェイルオーバードメインは、運用には必須 **ではありません。**

デフォルトでは、フェイルオーバードメインは制限なしで優先順がありません。

複数のメンバーを持つクラスター内では、制限ありのフェイルオーバードメインを使用することで、クラスターサービス (**httpd** など) を実行するためのクラスターを設定する作業を最小限にできます。このためには、クラスターサービスを実行する全てのメンバー上で、設定を同一にする必要があります。クラスターサービスを実行するようクラスター全体を設定する代わりに、クラスターサービスに関連付ける制限ありのフェイルオーバードメイン内のメンバーのみを設定するだけで済みます。

注記

優先するメンバーを設定するには、1 つだけのクラスターメンバーから成る制限なしのフェイルオーバードメインを作成します。そうすることで、クラスターサービスが主にそのクラスターメンバー (優先メンバー) 上で実行することになりますが、クラスターサービスは他のどのメンバーにでもフェイルオーバーできるようになります。

以下のセクションでは、フェイルオーバードメインの追加、修正、及び削除について説明しています：

- 「フェイルオーバードメインの追加」
- 「フェイルオーバードメインの修正」
- 「フェイルオーバードメインの削除」

3.8.1. フェイルオーバードメインの追加

フェイルオーバードメインを追加するには、このセクションの以下のステップに従ってください。

1. クラスター固有のページから、クラスターディスプレイの上部にある **Failover Domains（フェイルオーバードメイン）** をクリックすることで、クラスター用のフェイルオーバードメインを設定できます。これにより、このクラスター用に設定したフェイルオーバードメインが表示されます。

2. **Add（追加）** をクリックします。**Add** をクリックすると 図3.7「luci フェイルオーバードメイン設定のダイアログボックス」 で示してあるように、**Add Failover Domain to Cluster（フェイルオーバードメインをクラスターに追加）**ダイアログボックスが表示されま

す。

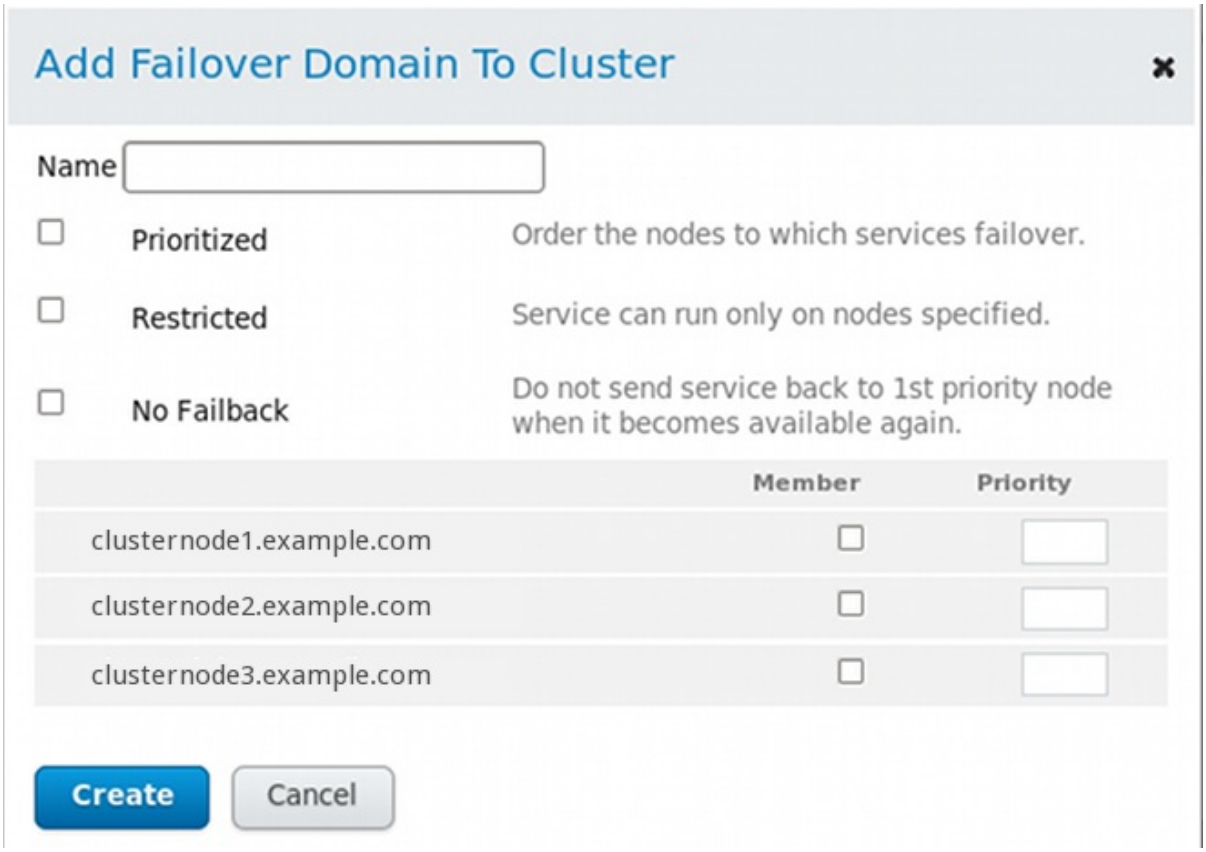


図3.7 luci フェイルオーバードメイン設定のダイアログボックス

3. **Add Failover Domain to Cluster** ダイアログボックスで、**Name（名前）** テキストボックスでフェイルオーバードメイン名を指定します。

注記

名前は、クラスター内で使用されている他の名前に対して、その目的を区別できるような説明的な名前にすることをお勧めします。

4. フェイルオーバードメイン内のメンバーのフェイルオーバー優先度のセッティングを有効にするには、**Prioritized（優先度設定）** チェックボックスをクリックします。**Prioritized** にチェックマークを入れると、その優先度の値 **Priority（優先度）** をフェイルオーバードメインのメンバーとして選択した各ノードに設定できます。

5. このフェイルオーバードメイン内のメンバーにフェイルオーバーを限定するには、**Restricted（制限あり）** チェックボックスをクリックします。**Restricted** にチェックマークを入れると、このフェイルオーバードメインに割り当てられたサービスはこのフェイルオーバードメイン内のノードのみにフェイルオーバーします。

6. ノードがこのフェイルオーバードメイン内にフェイルバックしないように指定するには、**No Failback（フェイルバックなし）** チェックボックスをクリックします。**No Failback** にチェックマークを入れた場合に、サービスが優先ノードからフェイルオーバーすると、サービスは回復後にそのオリジナルノードにフェイルバックしません。

7. このフェイルオーバードメインのメンバーを設定します。フェイルオーバードメインのメンバーになる予定の各ノードの **Member（メンバー）** チェックボックスをクリックします。**Prioritized** にチェックマークが入っている場合は、フェイルオーバードメインの各メンバー用の **Priority** テキストボックス内で優先度を設定します。

8. **Create（作成）** をクリックします。これにより、新規に作成したフェイルオーバードメインを表示する **Failover Domains** ページが表示されます。新規のドメインが作成されたことを示すメッセージが表示されます。ページをリフレッシュしてステータスを更新してください。

3.8.2. フェイルオーバードメインの修正

フェイルオーバードメインを修正するには、このセクションの以下のステップに従ってください。

1. クラスター固有のページから、クラスターディスプレイの上部にある **Failover Domains** をクリックすることで、クラスター用のフェイルオーバードメインを設定できます。これにより、このクラスター用に設定されたフェイルオーバードメインが表示されます。

2. フェイルオーバードメイン名をクリックします。フェイルオーバードメインの設定ページが表示されます。

3. フェイルオーバードメインの **Prioritized**、**Restricted**、又は **No Failback** のプロパティを修正するには、プロパティ横のチェックボックスにチェックマークを入れるか外してから、**Update Properties（プロパティの更新）** をクリックします。

4. フェイルオーバードメインのメンバーシップを修正するには、クラスターメンバーの横にあるチェックボックスにチェックマークを入れるか外します。フェイルオーバードメインの優先度が設定されている場合は、クラスターメンバーの優先度設定も修正できます。**Update Settings（設定の更新）** をクリックします。

3.8.3. フェイルオーバードメインの削除

フェイルオーバードメインを削除するには、このセクションの以下のステップに従ってください。

1. クラスター固有のページから、クラスターディスプレイの上部にある **Failover Domains** をクリックすることで、クラスター用のフェイルオーバードメインを設定できます。これにより、このクラスター用に設定されたフェイルオーバードメインが表示されます。

2. 削除するフェイルオーバードメインにチェックマークを入れます。

3. **Delete（削除）** をクリックします。

3.9. グローバルクラスターリソースの設定

クラスター内で実行しているサービスで使用できるグローバルリソースの設定、及び特定のサービスのみに利用可能なリソースの設定を行うことができます。

グローバルクラスターリソースを追加するには、このセクションの手順に従ってください。「クラスターへのクラスターサービスの追加」 の説明のとおり、サービスの設定時に、特定のサービスにローカルであるリソースを追加することができます。

1. クラスター固有のページから、クラスターディスプレイの上部にある **Resources（リソース）** をクリックすることで、クラスターにリソースを追加できます。これにより、そのクラスター用に設定されたリソースが表示されます。

2. **Add（追加）** をクリックします。**Add Resource to Cluster（リソースをクラスターに追加）** ドロップダウンメニューが表示されます。

3. **Add Resource to Cluster** の下にあるドロップダウンボックスをクリックして、設定するリソースタイプを選択します。

4. 追加するリソース用のリソースパラメーターを入力します。付録B *HA リソースパラメーター* はリソースパラメーターについて説明しています。

5. **Submit** をクリックします。**Submit** をクリックすると、**Resources** を表示するリソースページに戻ります。ここでは、追加されたリソース (及び他のリソース) が表示されます。

既存のリソースを修正するには、以下の手順を実行します。

1. **luci** の **Resources** ページから、修正するリソースの名前をクリックします。これで、リソースのパラメーターが表示されます。

2. リソースパラメーターを編集します。

3. **Apply** をクリックします。

既存のリソースを削除するには、以下の手順を実行します。

1. **luci** の **Resources** ページから、削除するリソースのチェックボックスをクリックします。

2. **Delete** をクリックします。

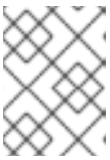
3.10. クラスターへのクラスターサービスの追加

クラスターにクラスターサービスを追加するには、このセクション内の手順に従ってください。

1. クラスター固有のページから、クラスターディスプレイの上部にある **Service Groups（サービスグループ）** をクリックすることで、クラスターへサービスを追加できます。これにより、そのクラスター用に設定されているサービスが表示されます (「高可用性サービスの管理」 で説明してあるように、**Service Groups** ページからサービスの起動/再起動/無効を行うことも可能です)。

2. **Add** をクリックします。**Add Service Group to Cluster** ダイアログボックスが表示されます。

3. **Add Service Group to Cluster** ダイアログボックスにある、**Service Name** テキストボックスにサービスの名前を入力します。



注記

名前は、クラスター内の他のサービスと明白に区別できる名前を使用してください。

4. クラスターが開始して稼働する時に、サービスが自動起動するようにしたい場合は、**Automatically Start This Service（自動的にこのサービスを開始）** チェックボックスにチェックマークを入れます。このチェックボックスにチェックマークが入って **いない** 場合は、クラスターが停止状態から復帰する時に手動でサービスを起動する必要があります。

5. **Run Exclusive（専用で実行）** チェックボックスにチェックマークを入れて、他のサービスが実行していないノード上でのみサービスが実行できるようにポリシーを設定します。

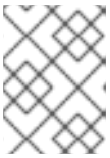
6. クラスターにフェイルオーバードメインを設定した場合、**Failover Domain** パラメーターのドロップダウンメニューを使用してこのサービス用にフェイルオーバードメインを選択できます。フェイルオーバードメインの設定についての詳細は、「フェイルオーバードメインの設定」 を参照してください。

7. **Recovery Policy** ドロップダウンボックスを使用して、サービスの回復ポリシーを選択します。オプションとしては、サービスの **Relocate（再配置）**、**Restart（再起動）**、**Restart-Disable（再起動後に無効）**、**Disable（無効）** があります。

   Restart オプションを選択すると、システムはサービスを再配置する前に、失敗したサービスの再起動を試行することを意味します。**Relocate** オプションを選択した場合、システムは別のノードでサービスの再起動を試行することを意味します。**Disable** オプションを選択すると、システムはコンポーネントに障害があった場合にリソースグループを無効にすることを意味します。**Restart-Disable** オプションを選択した場合は、システムはサービスに障害があった場合にサービスの再起動を試行することを意味します。ただし、サービスの再起動が失敗した場合、サービスはクラスター内の別のホストに移動する代わりに無効になります。

   サービスの回復ポリシーとして **Restart** 又は **Restart-Disable** を選択した場合、サービスの再配置/無効化までの再起動が失敗する最大回数と、再起動を破棄するまでの時間を秒単位で指定することができます。

8. サービスにリソースを追加するには、**`Add Resource`** をクリックします。**`Add Resource`** をクリックすると、**`Add Resource To Service`** ドロップダウンボックスが表示され、既存のグローバルリソースの追加、又はこのサービス ***のみ*** に利用可能な新規リソースの追加を行うことができます。

   注記

   フローティング IP アドレスリソースを含むクラスターサービスの設定時には、IP リソースを最初のエントリーとして設定する必要があります。

   - 既存のグローバルリソースを追加するには、**`Add Resource To Service`** ドロップダウンボックスから既存のリソース名をクリックします。設定するサービス用の **Service Groups** ページ上にリソースとそのパラメーターが表示されます。グローバルリソースの追加/修正についての詳細は 「グローバルクラスターリソースの設定」 をご覧ください。

   - このサービスにのみ利用可能な新規リソースを追加するには、**`Add Resource To Service`** ドロップダウンボックスから設定するリソースのタイプを選択して、追加するリソースのリソースパラメーターを入力します。付録B *HA リソースパラメーター* は、リソースパラメーターについて説明しています。

   - リソースをサービスに追加する場合、それが既存のグローバルリソース又はこのサービスのみに利用可能なリソースであるかに関わらず、リソースを **Independent Subtree（独立したサブツリー）** 又は **Non-Critical Resource (非クリティカルなリソース)** と指定できます。

     リソースを独立したサブツリーと指定した場合、リソースに障害が発生すると、システムが通常の回復を試行する前に (サービス全体ではなく) そのリソースのみが再起動します。そのサービス用に回復ポリシーを実装するまでに、ノード上のそのリソースに対して試行する再起動の最大回数を指定できます。また、システムがそのサービスに回復ポリシーを実装するまでの時間を秒単位で指定することもできます。

     リソースを非クリティカルなリソースと指定した場合、リソースに障害が発生すると、そのリソースのみが再起動します。リソースが失敗を繰り返す場合は、サービス全体ではなくそのリソースのみが無効になります。リソースを無効にするまでに、ノード上のそのリソースに対して試行する再起動の最大回数を指定できます。また、システムがそのリソースを無効にするまでの時間を秒単位で指定することもできます。

9. 定義するリソースに子リソースを追加したい場合は、**`Add Child Resource`（子リソースの追加）** をクリックします。**`Add Child Resource`** をクリックすると、**Add Resource To**

Service ドロップダウンボックスが表示され、既存のグローバルリソースの追加、又はこのサービスのみに利用可能な新規リソースの追加を行うことができます。使用している要件に合うように、リソースへ子リソースを追加し続けることも可能です。

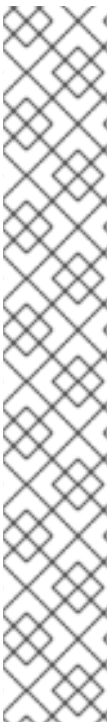
注記

Samba サービスのリソースを追加する場合は、別のリソースの子としてではなく、そのサービスに直接追加します。

注記

フローティング IP アドレスリソースを含むクラスターサービスの依存関係ツリー設定時には、IP リソースを別のリソースの子としてではなく、最初のエントリーとして設定する必要があります。

10. リソース群のサービスへの追加、子リソースのリソース群への追加が完了したら、**Submit** をクリックします。 **Submit** をクリックすると、追加済みのサービス（及び他のサービス群）を表示する **Service Groups（サービスグループ）** ページに戻ります。



注記

クラスターサービスで使用する IP サービスが存在するか確認するには、クラスターノードで (廃止された **ifconfig** コマンドではなく) **/sbin/ip addr show** コマンドを使用します。以下は、クラスターサービスを実行しているノードで **/sbin/ip addr show** コマンドを実行した場合の出力です。

```
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1356 qdisc pfifo_fast
qlen 1000
    link/ether 00:05:5d:9a:d8:91 brd ff:ff:ff:ff:ff:ff
    inet 10.11.4.31/22 brd 10.11.7.255 scope global eth0
    inet6 fe80::205:5dff:fe9a:d891/64 scope link
    inet 10.11.4.240/22 scope global secondary eth0
       valid_lft forever preferred_lft forever
```

既存のサービスを修正するには、以下の手順を実行します。

1. **Service Groups** ダイアログボックスから、修正するサービスの名前をクリックします。これにより、そのサービス用に設定されているパラメーターとリソースが表示されます。
2. サービスパラメーターを編集します。
3. **Submit** をクリックします。

既存のサービスを削除するには、以下の手順を実行します。

1. **luci Service Groups** ページから、削除するサービス用のチェックボックスをクリックします。
2. **Delete** をクリックします。

3. Red Hat Enterprise Linux 6.3 以降は、**luci** がサービスを削除する前に、サービスグループの削除を確認するメッセージが表示され、サービスを構成するリソースを停止します。サービスを削除せずにダイアログボックスを閉じるには、**Cancel** をクリックします。選択したサービスを削除するには、**Proceed** をクリックしてください。

第4章 CONGA を使用した RED HAT HIGH AVAILABILITY アドオンの管理

本章では Red Hat High Availability アドオンを管理する上での各種管理タスクについて説明しています。以下のセクションから構成されます：

- 「luci インターフェースへの既存クラスターの追加」
- 「luci インターフェースからのクラスターの削除」
- 「クラスターノードの管理」
- 「クラスターの起動、停止、再起動、削除」
- 「高可用性サービスの管理」
- 「luci 設定のバックアップと復元」

4.1. LUCI インターフェースへの既存クラスターの追加

以前に High Availability アドオンクラスターを作成したことがある場合は、そのクラスターを簡単に **luci** インターフェースに追加して、**Conga** を使用したクラスターの管理を行うことができます。

既存のクラスターを **luci** インターフェースに追加するには、以下の手順に従います：

1. **luci Homebase** ページの左側にあるメニューの **Manage Clusters（クラスターの管理）** をクリックします。**Clusters（クラスター）** 画面が表示されます。
2. **Add（追加）** をクリックします。**Add Existing Cluster（既存クラスターの追加）** 画面が表示されます。
3. 既存クラスター内にあるいずれかのノードのホスト名と **ricci** パスワードを入力します。クラスター内の各ノードにはクラスターの全ての設定情報が含まれているため、これでクラスターを **luci** インターフェースに追加するための十分な情報を提供できます。
4. **Connect（接続）** をクリックします。**Add Existing Cluster（既存クラスターの追加）** の画面にクラスター名とクラスター内の他のノード名が表示されます。
5. クラスター内の各ノード用に個別の **ricci**パスワードを入力するか、パスワードを 1 つ入力して **Use the same password for all nodes（全てのノードに同じパスワードを使用）** を選択します。
6. **Add Cluster（クラスターの追加）** をクリックします。ここで、以前に設定したクラスターが **Manage Clusters（クラスターの管理）** 画面に表示されます。

4.2. LUCI インターフェースからのクラスターの削除

クラスターサービスやクラスターのメンバーシップに影響を与えることなく、クラスターを **luci** 管理 GUI から削除できます。クラスターを削除した場合、後ほどそのクラスターを追加しなおすことができます。あるいは、「luci インターフェースへの既存クラスターの追加」 の説明のとおり、別の **luci** インスタンスに追加することもできます。

クラスターサービスやクラスターメンバーシップに影響を与えることなくクラスターを **luci** 管理 GUI から削除するには、以下の手順に従います。

1. **luci Homebase** ページの左側にあるメニューの **Manage Clusters（クラスターの管理）** をクリックします。**Clusters（クラスター）** 画面が表示されます。
2. 削除するクラスターを選択します。
3. **Remove** をクリックします。**luci** 管理 GUI からクラスターを削除するか確認されます。

完全にクラスターを削除する方法、全てのクラスターサービスを停止する方法、ノードからクラスター設定を削除する方法については、「クラスターの起動、停止、再起動、削除」 を参照してください。

4.3. クラスターノードの管理

このセクションでは、**Conga** の **luci** サーバーコンポーネントによって以下のノード管理機能を実行する方法について記載しています：

- 「クラスターノードの再起動」
- 「ノードのクラスターに対する離脱/参加」
- 「稼働しているクラスターへのメンバーの追加」
- 「クラスターからのメンバーの削除」

4.3.1. クラスターノードの再起動

クラスター内の 1 つのノードを再起動するには、以下の手順を実行します：

1. クラスター固有のページから、クラスターディスプレイの上部にある **Nodes（ノード群）** をクリックします。これにより、クラスターを構成するノード群が表示されます。これは、**luci Homebase** ページの左側にあるメニューから **Manage Clusters（クラスターの管理）** の下にあるクラスター名をクリックした時に表示されるデフォルトのページでもあります。
2. 再起動するノード用のチェックボックスをクリックすることでそのノードを選択します。
3. ページ上部にあるメニューから **Reboot（再起動）** 機能を選択します。これにより選択したノードが再起動して、そのノードが再起動中であることを示すメッセージがページ上部に表示されます。
4. ページをリフレッシュして、ノードが更新されていることを確認します。

再起動したいノード群を全て選択してから **Reboot** をクリックすることで、一度に複数のノードを再起動することも可能です。

4.3.2. ノードのクラスターに対する離脱/参加

Conga の **luci** サーバーコンポーネントを使用すると、ノード上の全てのクラスターサービスを停止することにより、ノードがアクティブなクラスターから離脱するようにできます。また、**Conga** の **luci** サーバーコンポーネントを使用して、離脱したノードがクラスターに再参加するようにもできます。

ノードをクラスターから離脱させることは、ノードからクラスター設定情報を削除することにはなりません。クラスターノードディスプレイには、ノードのステータスが **`Not a cluster member`（クラスターメンバーではない）** として引き続き表示されます。クラスター設定からノードを完全に削除するための情報は 「クラスターからのメンバーの削除」 をご覧ください。

ノードをクラスターから離脱させるには、以下の手順を実行します。これによりノード内のクラスターソフトウェアをシャットダウンします。ノードをクラスターから離脱させることで、クラスターの再起動時にノードが自動的に参加できないようにします。

1. クラスター固有のページから、クラスターディスプレイの上部にある **Nodes（ノード群）** をクリックします。これにより、クラスターを構成するノード群が表示されます。これは、**luci Homebase** ページの左側にあるメニューから **Manage Clusters（クラスターの管理）** の下にあるクラスター名をクリックした時に表示されるデフォルトのページでもあります。

2. クラスターから離脱させるノードを選択するには、そのチェックボックスをクリックします。

3. ページ上部のメニューから **Leave Cluster（クラスターから離脱）** 機能を選択します。これでページ上部にメッセージが表示され、ノードが停止中であることを示します。

4. ページをリフレッシュして、ノードが更新されていることを確認します。

クラスターから離脱させるノードを全て選択した後に **Leave Cluster** をクリックすることにより、一度に複数のノードをクラスターから離脱させることもできます。

ノードをクラスターに再参加させるには、再参加させるノード群のチェックボックスをクリックして **Join Cluster（クラスターに参加）** を選択することで、ノード群を選択します。これで選択したノード群がクラスターに参加することになり、ノード群は再起動時にクラスターに参加します。

4.3.3. 稼働しているクラスターへのメンバーの追加

稼働中のクラスターにメンバーを追加するには、このセクションの以下の手順に従ってください。

1. クラスター固有のページから、クラスターディスプレイの上部にある **Nodes（ノード群）** をクリックします。これにより、クラスターを構成するノード群が表示されます。これは、**luci Homebase** ページの左側にあるメニューから **Manage Clusters（クラスターの管理）** の下にあるクラスター名をクリックした時に表示されるデフォルトのページでもあります。

2. **Add（追加）** をクリックします。**Add** をクリックすると、**`Add Nodes To Cluster`（クラスターにノードを追加）** のダイアログボックスが表示されます。

3. **Node Hostname（ノードホスト名）** テキストボックスにノード名を入力します。**Password** テキストボックスに **ricci** パスワードを入力します。 **ricci** エージェント用にデフォルトの 11111 以外の異なるポートを使用している場合は、このパラメーターを使用しているポートに変更します。

4. クラスター化ストレージをサポートしてクラスター化 LVM を有効にするパッケージをダウンロードするためにクラスター化ストレージが必要な場合は、**Enable shared storage support（共有ストレージサポートを有効にする）** のチェックボックスにチェックマークを入れます。これは、Resilient Storage アドオン又は Scalable File System アドオンにアクセスできる場合にのみ選択してください。

5. さらにノードを追加するには、**`Add Another Node`（別のノードを追加）** をクリックして、追加ノード毎に名前とパスワードを入力します。

6. **`Add Nodes`（ノード群の追加）** をクリックします。**`Add Nodes`** をクリックすると、以下の動作が行われます：

    1. **Download Packages（パッケージのダウンロード）** を選択した場合は、クラスターソフトウェアパッケージが追加ノード上にダウンロードされます。

    2. クラスターソフトウェアがノードにインストールされます (又は、適切なソフトウェアパッケージがインストールされたことが確認されます)。

3. クラスター設定ファイルが更新され、クラスター内の各ノードに伝播します — 追加ノードも伝播します。

4. 追加されたノードがクラスターに参加します。

ノードがクラスターに追加されていることを示すメッセージとともに **Nodes** ページが表示されます。ページをリフレッシュして、ステータスを更新します。

7. ノード追加のプロセスが完了したら、「フェンスデバイスの設定」 に示してあるように新規追加ノードのノード名をクリックしてこのノード用にフェンシングを設定します。

4.3.4. クラスターからのメンバーの削除

現在動作中の既存クラスターからメンバーを削除するには、このセクションの手順に従ってください。なお、クラスター内の全ノードを一度に削除しない限りは、ノードを停止してから削除する必要があります。

1. クラスター固有のページから、クラスターディスプレイの上部にある **Nodes（ノード群）** をクリックします。これにより、クラスターを構成するノード群が表示されます。これは、**luci Homebase** ページの左側にあるメニューから **Manage Clusters（クラスターの管理）** の下にあるクラスター名をクリックした時に表示されるデフォルトのページでもあります。

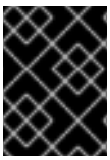
注記

ノードの削除時にノード上で実行中のサービスをフェイルオーバーさせるには、次の手順を省略してください。

2. 削除されるノード上で実行中の各サービスを無効/再配置します。サービスの無効と再配置の詳細については 「高可用性サービスの管理」 を参照してください。

3. 削除するノード、又はノード群を選択します。

4. **Delete（削除）** をクリックします。**Nodes** ページはノードが削除されていることを示します。ページをリフレッシュして、現在のステータスを確認します。



重要

クラスターからクラスターノードを削除することは、元に戻すことが不可能な破壊的な操作です。

4.4. クラスターの起動、停止、再起動、削除

クラスター内の各ノード上で以下の操作を実行することで、クラスターの起動/停止/再起動ができます。クラスター固有のページから、クラスターディスプレイ上部の **Nodes** をクリックします。これによりクラスターを構成するノード群が表示されます。

クラスターサービスが停止又は再起動されるノード上で実行しているために別のクラスターメンバーに移動させる必要がある場合、クラスターノード又はクラスター全体に起動、再起動の操作を行うことにより、クラスターサービスを短時間停止することができます。

クラスターを停止するには、以下の手順を実行します。これは、ノード群内のクラスターソフトウェアをシャットダウンしますが、ノードからクラスター設定情報を削除することはありません。ノード群は、**Not a cluster member** のステータスで引き続きクラスターノードのディスプレイに表示されます。

1. 各ノードの横にあるチェックボックスをクリックすることで、クラスター内の全てのノードを選択します。
2. ページ上部のメニューから **Leave Cluster** 機能を選択します。これにより、メッセージがページ上部に表示され各ノードが停止されることを示します。
3. ページをリフレッシュして、ノード群が更新されていることを確認します。

クラスターを起動するには、以下の手順を実行します：

1. 各ノードの横にあるチェックボックスをクリックすることで、クラスター内の全てのノードを選択します。
2. ページ上部のメニューから **Join Cluster** 機能を選択します。
3. ページをリフレッシュして、ノード群が更新されていることを確認します。

実行中のクラスターを再起動するには、まずクラスター内の全てのノードを停止して、前述のようにクラスター内の全てのノードを起動します。

クラスターを完全に削除するには、以下の手順を実行します。これにより、全てのクラスターサービスは停止し、ノードからクラスター設定情報、さらにはクラスターディスプレイからノードを削除します。後ほど削除したノードを使用して既存のクラスターを追加しようと試行すると、**luci** はノードがクラスターのメンバーでないことを示します。

重要

クラスターの削除は、元に戻すことが不可能な破壊的な操作です。削除したクラスターを復元するには、最初からクラスターを再度作成、定義する必要があります。

1. 各ノードの横にあるチェックボックスをクリックすることで、クラスター内の全てのノードを選択します。
2. ページ上部のメニューから **Delete** 機能を選択します。

クラスターサービスの停止やクラスターのメンバーシップの変更をせずに、**luci** インターフェースからクラスターを削除したい場合は、「luci インターフェースからのクラスターの削除」 に記載されているように **Manage Clusters** ページの **Remove** オプションを使用します。

4.5. 高可用性サービスの管理

「クラスターへのクラスターサービスの追加」 に示してあるようなサービスの追加と修正に加えて、**Conga** の **luci** サーバーコンポーネントを介して高可用性サービス用に以下のような機能を実行することもできます：

- サービスの起動
- サービスの再起動
- サービスの無効化
- サービスの削除
- サービスの再配置

クラスター固有のページから、クラスターディスプレイの上部にある **Service Groups（サービスグループ）** をクリックすることにより、そのクラスターのサービスを管理できます。これにより、そのクラスター用に設定されたサービス群が表示されます。

- **Starting a service（サービスの開始）** — 現在稼働していないいずれかのサービスを起動するには、起動したいサービスのチェックボックスをクリックしてそのサービスを選択し、 **Start** をクリックします。

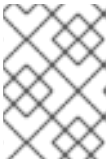
- **Restarting a service（サービスの再起動）** — 現在稼働中のいずれかのサービスを再起動するには、再起動したいサービスのチェックボックスをクリックしてそのサービスを選択し、**Restart** をクリックします。

- **Disabling a service（サービスの無効化）** — 現在稼働中のいずれかのサービスを無効にするには、無効にしたいサービスのチェックボックスをクリックしてそのサービスを選択し、**Disable** をクリックします。

- **Deleting a service（サービスの削除）** — 現在稼働していないいずれかのサービスを削除するには、削除したいサービスのチェックボックスをクリックしてそのサービスを選択し、**Delete** をクリックします。

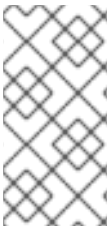
- **Relocating a service（サービスの再配置）** — 稼働中のサービスを再配置するには、サービス群ディスプレイ内でそのサービス名をクリックします。これにより、そのサービスのサービス設定ページが表示され、サービスが現在実行しているノードを示すディスプレイが表示されます。

  `Start on node...`（指定ノードで開始） ドロップダウンボックスから、サービスを再配置したいノードを選択して、**Start** アイコンをクリックします。画面上部にメッセージが表示され、サービスが起動していることを示します。選択したノード上でサービスが稼働していることを示す画面を確認するには、画面をリフレッシュする必要がある場合があります。



注記

選択した実行中のサービスが **`vm`** サービスの場合、ドロップダウンボックスは **`relocate`** オプションの代わりに **`migrate`** オプションを表示します。



注記

Services ページのサービス名をクリックしても個別のサービスを起動、再起動、無効化、あるいは削除することができます。これにより、サービス設定のページが表示されます。サービス設定ページの右上には、**Start**、**Restart**、**Disable**、及び **Delete** の同じアイコンがあります。

4.6. LUCI 設定のバックアップと復元

Red Hat Enterprise Linux 6.2 リリース以降、以下の手順を実行して **`/var/lib/luci/data/luci.db`** ファイルに格納されている **luci** データベースのバックアップをとることができます。これは **`cluster.conf`** ファイルに格納されているクラスター設定自体ではありません。代わりに、**luci** が管理するユーザー、クラスター、関連プロパティの一覧が含まれています。デフォルトでは、この手順で作成するバックアップは **`luci.db`** ファイルと同じディレクトリに書き込まれます。

1. **`service luci stop`** を実行します。

2. **`service luci backup-db`** を実行します。

   オプションとして、**`backup-db`** コマンドにパラメーターとしてファイル名を指定することがで

きます。これにより、そのファイルに **luci** データベースが書き込まれます。例えば、**luci** データベースを **/root/luci.db.backup** ファイルに書き込むには、**service luci backup-db /root/luci.db.backup** コマンドを実行します。ただし、**/var/lib/luci/data/** 以外の場所に書き込まれるバックアップファイル (**service luci backup-db** の使用時に指定するファイル名を持つバックアップ) は、**list-backups** コマンドの出力には表示されない点に注意してください。

3. **service luci start** を実行します。

以下の手順を実行して、**luci** データベースを復元します。

1. **service luci stop** を実行します。
2. **service luci list-backups** を実行します。復元するファイル名に注意してください。
3. **service luci restore-db /var/lib/luci/data/*lucibackupfile*** を実行します。ここでは、*lucibackupfile* が復元するバックアップファイルです。

   例えば、次のコマンドは **luci-backup20110923062526.db** バックアップファイルに格納された **luci** 設定情報を復元します：

   ```
   service luci restore-db /var/lib/luci/data/luci-
   backup20110923062526.db
   ```

4. **service luci start** を実行します。

luci データベースを復元する必要があるが、完全な再インストールのためバックアップを作成したマシンから **host.pem** ファイルを失ったような場合は、クラスターノードを再認証するために手動でクラスターを **luci** に追加しなおす必要があります。

以下の手順を実行することで、バックアップを作成したマシンではなく別のマシンに **luci** データベースを復元します。データベース自体の復元に加えて、SSL 証明書をコピーして **luci** が **ricci** ノードに対して認証されていることを確認する必要があります。この例では、バックアップの作成は **luci1** マシンで行われ、バックアップの復元は **luci2** マシンで行われます。

1. 以下の一連のコマンドを実行して、**luci1** で **luci** バックアップを作成します。また、SSL 証明書ファイルと **luci** バックアップを **luci2** にコピーします。

   ```
   [root@luci1 ~]# service luci stop
   [root@luci1 ~]# service luci backup-db
   [root@luci1 ~]# service luci list-backups
   /var/lib/luci/data/luci-backup20120504134051.db
   [root@luci1 ~]# scp /var/lib/luci/certs/host.pem
   /var/lib/luci/data/luci-backup20120504134051.db root@luci2:
   ```

2. **luci2** マシンで、**luci** がインストールされ実行中でないことを確認してください。インストールされていない場合はインストールしてください。
3. 以下の一連のコマンドを実行して、認証が適切であることを確認し、**luci1** から **luci2** への **luci** データベースの復元を行います。

   ```
   [root@luci2 ~]# cp host.pem /var/lib/luci/certs/
   [root@luci2 ~]# chown luci: /var/lib/luci/certs/host.pem
   [root@luci2 ~]# /etc/init.d/luci restore-db ~/luci-
   ```

```
backup20120504134051.db
[root@luci2 ~]# shred -u ~/host.pem ~/luci-backup20120504134051.db
[root@luci2 ~]# service luci start
```

第5章 CCS コマンドを使用した RED HAT HIGH AVAILABILITY アドオンの設定

Red Hat Enterprise Linux 6.1 リリース以降、Red Hat High Availability アドオンは **ccs** クラスター設定コマンドのサポートを提供しています。**ccs** コマンドにより、管理者は **cluster.conf** クラスター設定ファイルの作成/修正/表示が可能です。**ccs** コマンドを使用して、クラスター設定ファイルをローカルファイルシステム又はリモートノード上で設定できます。さらには、**ccs** コマンドにより、管理者は設定済のクラスター内にある 1 つ又はすべてのノードでクラスターサービスの起動と停止を行うこともできます。

本章は **ccs** コマンドを使用した Red Hat High Availability アドオンクラスター設定ファイルの設定方法について説明しています。実行中のクラスターを管理するための **ccs** コマンドの使用に関しては 6 章*Red Hat High Availability アドオンを使用した **ccs** の管理* をご覧ください。

本章は以下のセクションで構成されます。

- 「操作の概要」
- 「設定のタスク」
- 「**ricci** の起動」
- 「クラスターの作成」
- 「フェンスデバイスの設定」
- 「クラスターメンバー用のフェンシングの設定」
- 「フェイルオーバードメインの設定」
- 「グローバルクラスターリソースの設定」
- 「クラスターへのクラスターサービスの追加」
- 「Quorum disk の設定」
- 「その他のクラスター設定」
- 「その他のクラスター設定」
- 「クラスタノード群への設定ファイルの伝播」

注記

High Availability アドオンの導入がご使用のニーズに適合していて、サポート可能であることを確認してください。導入する前に、Red Hat 認定担当者に連絡して、設定を検証してください。さらに、設定のバーンイン期間を設けて障害モードのテストを実施してください。

注記

本章では、一般に使用されている **cluster.conf** の要素と属性を参照します。**cluster.conf** の要素と属性の総括的な一覧と説明は、**/usr/share/cluster/cluster.rng** にあるクラスタースキーマと **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** (例えば、**/usr/share/doc/cman-3.0.12/cluster_conf.html**) にある注釈付きスキーマを参照してください。

5.1. 操作の概要

このセクションでは、クラスターを設定するための **ccs** コマンドの使用に関する一般的な操作について説明します。

- 「ローカルシステム上でのクラスター設定ファイルの作成」
- 「現在のクラスター設定の表示」
- 「ccs コマンドを使用した ricci パスワードの指定」
- 「クラスター設定コンポーネントの修正」

5.1.1. ローカルシステム上でのクラスター設定ファイルの作成

ccs コマンドを使用すると、クラスターノードでクラスター設定ファイルを作成できます。あるいは、ローカルファイルシステムでクラスター設定ファイルを作成して、それをクラスター内のホストに送信することも可能です。これによりローカルマシンのファイルでの作業が可能になるため、バージョン管理下で維持管理ができます。別の方法として、ニーズに合わせてファイルをタグ付けすることも可能です。**ccs** コマンドを使用するには、root 権限は必要ありません。

ccs コマンドを使用してクラスターノード上でクラスター設定ファイルを作成/編集する場合、**-h** オプションを使用して、ホスト名を指定します。これは、ホストの **cluster.conf** ファイルを作成/編集することになります:

```
ccs -h host [options]
```

ローカルシステム上でクラスター設定ファイルを作成/編集するには、**ccs** コマンドで **-f** オプションを使用して、クラスターの操作を行う時に設定ファイル名を指定します。ファイル名は任意に指定できます。

```
ccs -f file [options]
```

ローカルでファイルを作成した後は **ccs** コマンドで **--setconf** オプションを使用して、ファイルをクラスターノードに送信することができます。クラスター内のホストマシン上で、送信したファイルは **cluster.conf** と名前が付けられ、**/etc/cluster** ディレクトリに配置されます。

```
ccs -h host -f file --setconf
```

ccs コマンドの **--setconf** オプションの使用方法については、 「クラスタノード群への設定ファイルの伝播」 をご覧ください。

5.1.2. 現在のクラスター設定の表示

クラスターの設定ファイルの作成時に現在ファイルを出力したい場合、以下のコマンドを使用して、クラスター内のノードをホストとして指定します。

```
ccs -h host --getconf
```

ローカルシステム上でクラスター設定ファイルを作成する場合は、「ローカルシステム上でのクラスター設定ファイルの作成」 に示してあるように、**-h** オプションではなく **-f** オプションを指定します。

5.1.3. ccs コマンドを使用した ricci パスワードの指定

ccs コマンドは **cluster.conf** ファイルのコピーをクラスターのノード群に配布します。このコマンドの実行には、「**ricci** の注意事項」 で説明してあるように、クラスターノード群に **ricci** がインストールされ稼働している必要があります。**ricci** を使用する場合は、いずれかのマシンから **ricci** とやりとりを行う初回時にパスワードが必要になります。

使用しているマシンからある特定のマシン上で **ricci** のインスタンスのパスワードを入力していない場合は、**ccs** コマンドが必要な場合にパスワードをプロンプトします。別の方法として、**-p** オプションを使用して、コマンドラインで **ricci** パスワードを指定することも可能です。

```
ccs -h host -p password --sync --activate
```

ccs コマンドの **--sync** オプションを使用して **cluster.conf** ファイルをクラスター内の全ノードに伝播し、そのコマンドに **ricci** パスワードを指定する場合、**ccs** コマンドはクラスター内の各ノードにそのパスワードを使用します。個々のノードで **ricci** に違うパスワードを設定する必要がある場合は、**--setconf** に **-p** オプションを付けて使用することで、設定ファイルを同時に 1 つのノードに配布できます。

5.1.4. クラスター設定コンポーネントの修正

ccs コマンドを使用して、クラスターコンポーネント及びそれらの属性をクラスター設定ファイル内で設定します。そのファイルにクラスターコンポーネントを追加した後に、そのコンポーネントの属性を修正するには、定義したコンポーネントを削除してから、属性を修正して再度コンポーネントを追加しなければなりません。各コンポーネントでこれを実行する方法の詳細は、本章の個別のセクションで紹介しています。

cman クラスターコンポーネントの属性は、クラスターコンポーネントの変更手順に例外を追加できます。この属性を変更するには、**ccs** コマンドの **--setcman** オプションを実行して、新しい属性を指定します。「以前の設定を上書きするコマンド」 に記載されているように、このオプションを指定すると、明示的に指定していない値はすべてデフォルト値にリセットされるので注意してください。

5.1.5. 以前の設定を上書きするコマンド

ccs コマンドの中には、プロパティを設定すると上書きセマンティクスの実装をするオプションが複数あります。つまり、設定を何も指定せずにオプションの 1 つをつけて **ccs** コマンドを実行できますが、全設定がデフォルト値にリセットされるということです。これらのオプションは以下のとおりです。

- **--settotem**
- **--setdlm**
- **--setrm**

- **--setcman**
- **--setmulticast**
- **--setaltmulticast**
- **--setfencedaemon**
- **--setlogging**
- **--setquorumd**

例えば、フェンスデーモンのプロパティをすべてリセットするには、以下のコマンドを実行します。

```
# ccs -h hostname --setfencedaemon
```

ただし、これらのコマンドのいずれかを使用してプロパティをリセットした場合、コマンドのその他のプロパティがデフォルト値にリセットされる点に注意してください。例えば、以下のコマンドを使用して **post_fail_delay** プロパティを 5 に設定することができます。

```
# ccs -h hostname --setfencedaemon post_fail_delay=5
```

コマンド実行後、以下のコマンドを実行して **post_join_delay** プロパティを 10 にリセットすると、**post_fail_delay** プロパティはデフォルト値にリセットされます。

```
# ccs -h hostname --setfencedaemon post_join_delay=10
```

post_fail_delay と **post_join_delay** の両方のプロパティをリセットするには、以下の例のように、同じコマンド内で両方を指定します。

```
# ccs -h hostname --setfencedaemon post_fail_delay=5 post_join_delay=10
```

フェンスデバイスの設定に関する詳細情報は、「フェンスデバイスの設定」 を参照してください。

5.1.6. 設定の妥当性検証

ccs コマンドを使用して、クラスター設定ファイルの作成/編集を行う場合、設定はクラスタースキーマに従って自動的に検証されます。Red Hat Enterprise Linux 6.3 リリース以降、**ccs** コマンドは **-h** オプションで指定するノードの **/usr/share/cluster/cluster.rng** にあるクラスタースキーマに従って設定を検証します。これまでは、**ccs** コマンドは **ccs** コマンドでパッケージ化されたローカルシステムのクラスタースキーマである **/usr/share/ccs/cluster.rng** を常に使用していました。**-f** オプションを使用してローカルシステムを指定する場合は、**ccs** コマンドはそのシステムで **ccs** コマンドでパッケージ化されたクラスタースキーマの **/usr/share/ccs/cluster.rng** を引き続き使用します。

5.2. 設定のタスク

ccs を使用して Red Hat High Availability アドオンソフトウェアを設定するには、以下のステップを実行します。

1. クラスター内の全てのノード上で **ricci** が稼働していることを確認。「**ricci** の起動」 を参照してください。

2. クラスターを作成。「クラスターの作成」 を参照してください。
3. フェンスデバイスを設定。「フェンスデバイスの設定」 を参照してください。
4. クラスターメンバー用にフェンシングを設定。「クラスターメンバー用のフェンシングの設定」 を参照してください。
5. フェイルオーバードメインを作成。「フェイルオーバードメインの設定」 を参照してください。
6. リソースを作成。「グローバルクラスターリソースの設定」 を参照してください。
7. クラスターサービスを作成。「クラスターへのクラスターサービスの追加」 を参照してください。
8. 必要に応じて quorum disk を設定。「Quorum disk の設定」 を参照してください。
9. グローバルクラスタープロパティを設定。「その他のクラスター設定」 を参照してください。
10. クラスター設定ファイルを全てのクラスターノードに伝播。「クラスタノード群への設定ファイルの伝播」 を参照してください。

5.3. RICCI の起動

クラスターのノード上でクラスター設定ファイルを作成/配布するには、**ricci** サービスが各ノードで実行されている必要があります。**ricci** を起動する前に、使用しているシステムが次のとおりに設定されていることを確認してください。

1. 使用しているクラスターノードの IP ポートは、**ricci** に対して有効である必要があります。クラスターノードの IP ポートを有効にする方法については 「クラスターノードでの IP ポートの有効化」 をご覧ください。
2. 「**ricci** の注意事項」 の説明のとおり、**ricci** サービスがクラスター内の全てのノード上にインストールされ、**ricci** パスワードが割り当てられている必要があります。

各ノードで **ricci** がインストール、設定された後に、各ノード上で **ricci** サービスを開始します:

```
# service ricci start
Starting ricci:                                            [  OK  ]
```

5.4. クラスターの作成

このセクションでは、**ccs** コマンドを使用した、フェンシング、フェイルオーバードメイン、HA サービスのないスケルトンクラスター設定の作成/修正/削除の方法を説明しています。後続のセクションでは、これらの設定方法を説明します。

スケルトンクラスターの設定ファイルを作成するには、まずクラスターを作成して名前を付け、それから以下の手順のようにクラスターにノード群を追加します:

1. クラスターの 1 つのノード上でクラスター設定ファイルを作成するには、**ccs** コマンドを使用します。これに、**-h** パラメーターを付けるとファイルを作成するノードを指定でき、**createcluster** オプションを付けるとクラスターの名前を指定できます:

```
ccs -h host --createcluster clustername
```

例えば、以下のコマンドでは **mycluster** と呼ばれる設定ファイルを **node-01.example.com** に作成します:

```
ccs -h node-01.example.com --createcluster mycluster
```

クラスター名は 15 文字以内にしてください。

指定するホスト上に既に **cluster.conf** ファイルが存在する場合は、次のコマンドを実行してその既存ファイルを入れ替えます。

お使いのローカルシステムにクラスター設定ファイルを作成したい場合、**-h** オプションではなく **-f** オプションを指定してください。ファイルのローカル作成に関する情報は、「ローカルシステム上でのクラスター設定ファイルの作成」 を参照してください。

2. クラスターに含まれるノードを設定するには、クラスター内の各ノードに対して以下のコマンドを実行します。ノード名の長さは、最大 255 バイトまでになります。

```
ccs -h host --addnode node
```

例えば、以下の 3 つのコマンドは **node-01.example.com**、**node-02.example.com**、及び **node-03.example.com** のノードを **node-01.example.com** 上にある設定ファイルに追加します:

```
ccs -h node-01.example.com --addnode node-01.example.com
ccs -h node-01.example.com --addnode node-02.example.com
ccs -h node-01.example.com --addnode node-03.example.com
```

クラスター用に設定されているノード群の一覧を表示するには、以下のコマンドを実行します:

```
ccs -h host --lsnodes
```

例5.1「3 つのノードを追加した後の **cluster.conf** ファイル」 は、クラスター **mycluster** を作成した後の **node-01.example.com**、**node-02.example.com**、及び **node-03.example.com** のノードを含む **cluster.conf** 設定ファイルを示しています。

例5.1 3 つのノードを追加した後の cluster.conf ファイル

```
<cluster name="mycluster" config_version="2">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
```

```
    </fencedevices>
    <rm>
    </rm>
</cluster>
```

ノードをクラスターに追加する時に、定足数があるかどうか判定するのにノードが貢献する投票数を指定できます。クラスターノードの投票数をセットするには以下のコマンドを使用します:

```
ccs -h host --addnode host --votes votes
```

ノードの追加時に、**ccs** はノード識別子として使用される一意の整数をノードに割り当てます。ノードの作成時に手動でノード識別子を指定する場合は、以下のコマンドを使用します:

```
ccs -h host --addnode host --nodeid nodeid
```

クラスターからノードを削除するには、次のコマンドを使用します:

```
ccs -h host --rmnode node
```

「クラスタノード群への設定ファイルの伝播」 に説明してあるように、クラスターの全コンポーネントの設定が終了した時点で、クラスター設定ファイルを全てのノードに対して同期する必要があります。

5.5. フェンスデバイスの設定

フェンスデバイスの設定は、クラスター用のフェンスデバイスの作成、更新、削除で行います。クラスター内のノードにフェンシングを設定するには、まずクラスター内でフェンスデバイスを作成して、名前をつけます。クラスター内の個別ノードのフェンシング設定に関する情報は、「クラスターメンバー用のフェンシングの設定」 を参照してください。

フェンスデバイスを設定する前に、使用しているシステムのフェンスデーモンプロパティの一部をデフォルト値から変更してください。フェンスデーモンに設定する値はクラスター用の一般的な値です。以下に、変更する可能性があるクラスターの一般的なフェンシングプロパティをまとめています:

- **post_fail_delay** 属性は、ノードに障害が起こった後にノード (フェンスドメインのメンバー) をフェンスするまでフェンスデーモン (**fenced**) が待機する秒数です。**post_fail_delay** のデフォルト値は **0** です。この値はクラスター及びネットワークパフォーマンスに合わせて変更できます。

- **post-join_delay** 属性は、ノードが fence ドメインを結合してからノードをフェンシングするまでの fence daemon (**fenced**) の待機時間 (秒数) です。**post_join_delay** のデフォルト値は、**6**で、**Post Join Delay** の一般的な設定は 20 秒から 30 秒の間となっていますが、クラスターやネットワークのパフォーマンスにより左右されます。

post_fail_delayおよび **post_join_delay** 属性の値を **ccs** コマンドの**--setfencedaemon** オプションでリセットします。ただし、**ccs --setfencedaemon** コマンドを実行すると、明示的に設定された既存のフェンスデーモンプロパティをすべて上書きして、デフォルト値にリストアする点に注意してください。

例えば、**post_fail_delay** 属性の値を設定するには、以下のコマンドを実行します。すると、この属性値以外で、このコマンドで設定した既存のフェンスデーモンプロパティが、デフォルト値にリストアします。

```
ccs -h host --setfencedaemon post_fail_delay=value
```

post_join_delay 属性の値を設定するには、以下のコマンドを実行します。すると、この属性値以外で、このコマンドで設定した既存のフェンスデーモンプロパティが、デフォルト値にリストアします。

```
ccs -h host --setfencedaemon post_join_delay=value
```

post_join_delay 属性と **post_fail_delay** 属性両方の値を設定するには、以下のコマンドを実行します。

```
ccs -h host --setfencedaemon post_fail_delay=value post_join_delay=value
```

注記

post_join_delay 属性と **post_fail_delay** 属性に関する詳細、及び変更可能な追加のフェンスデーモンプロパティの詳細については、fenced(8) の man ページ、**/usr/share/cluster/cluster.rng** にあるクラスタースキーマ、**/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** の注釈付きスキーマを参照してください。

クラスターにフェンスデバイスを設定するには、以下のコマンドを実行します:

```
ccs -h host --addfencedev devicename [fencedeviceoptions]
```

例えば、クラスターノード **node1** の設定ファイルに、IP アドレスが **apc_ip_example**、ログイン名が **login_example**、パスワードが **password_example** である **my_apc** と呼ばれる APC フェンスデバイスを設定するには、次のコマンドを実行します。

```
ccs -h node1 --addfencedev myfence agent=fence_apc ipaddr=apc_ip_example
login=login_example passwd=password_example
```

次の例は、この APC フェンスデバイスを追加した後の **cluster.conf** 設定ファイルの **fencedevices** セクションを示しています。

```
<fencedevices>
      <fencedevice agent="fence_apc" ipaddr="apc_ip_example"
login="login_example" name="my_apc" passwd="password_example"/>
</fencedevices>
```

クラスターにフェンスデバイスを設定する場合、クラスターに利用可能なデバイスや各デバイスに使用できるオプションの一覧を確認すると役立つ場合があります。また、使用しているクラスターに現在設定されているフェンスデバイスの一覧を確認することもよいでしょう。利用可能なフェンスデバイスの一覧及び使用しているクラスターに現在設定されているフェンスデバイスの一覧を表示するための **ccs** コマンドの使用方法については、「フェンスデバイスとフェンスデバイスオプションの一覧表示」 を参照してください。

クラスター設定からフェンスデバイスを削除するには、以下のコマンドを実行します:

```
ccs -h host --rmfencedev fence_device_name
```

例えば、**myfence** と呼ばれるフェンスデバイスをクラスターノード **node1** 上にあるクラスター設定ファイルから削除するには、以下のコマンドを実行します:

```
ccs -h node1 --rmfencedev myfence
```

既に設定済のフェンスデバイスの属性を修正する必要があれば、まずそのフェンスデバイスを削除して、その後に修正した属性を持つフェンスデバイスを再度追加します。

「クラスタノード群への設定ファイルの伝播」 に説明してあるように、クラスターの全コンポーネントの設定を終了した時点で、クラスター設定ファイルを全てのノードに対して同期する必要がある点に注意してください。

5.6. フェンスデバイスとフェンスデバイスオプションの一覧表示

ccs コマンドを使用すると、利用可能なフェンスデバイスの一覧及び各フェンスタイプのオプション一覧を表示することができます。また、**ccs** コマンドでは、使用しているクラスターに現在設定されているフェンスデバイスの一覧も表示できます。

使用しているクラスターに現在利用できるフェンスデバイスの一覧を表示するには、次のコマンドを実行してください:

```
ccs -h host --lsfenceopts
```

例えば、次のコマンドはクラスターノード **node1** で利用可能なフェンスデバイスを一覧表示します。サンプルの出力を表示します。

```
[root@ask-03 ~]# ccs -h node1 --lsfenceopts
fence_rps10 - RPS10 Serial Switch
fence_vixel - No description available
fence_egenera - No description available
fence_xcat - No description available
fence_na - Node Assassin
fence_apc - Fence agent for APC over telnet/ssh
fence_apc_snmp - Fence agent for APC over SNMP
fence_bladecenter - Fence agent for IBM BladeCenter
fence_bladecenter_snmp - Fence agent for IBM BladeCenter over SNMP
fence_cisco_mds - Fence agent for Cisco MDS
fence_cisco_ucs - Fence agent for Cisco UCS
fence_drac5 - Fence agent for Dell DRAC CMC/5
fence_eps - Fence agent for ePowerSwitch
fence_ibmblade - Fence agent for IBM BladeCenter over SNMP
fence_ifmib - Fence agent for IF MIB
fence_ilo - Fence agent for HP iLO
fence_ilo_mp - Fence agent for HP iLO MP
fence_intelmodular - Fence agent for Intel Modular
fence_ipmilan - Fence agent for IPMI over LAN
fence_kdump - Fence agent for use with kdump
fence_rhevm - Fence agent for RHEV-M REST API
fence_rsa - Fence agent for IBM RSA
fence_sanbox2 - Fence agent for QLogic SANBox2 FC switches
```

```
fence_scsi - fence agent for SCSI-3 persistent reservations
fence_virsh - Fence agent for virsh
fence_virt - Fence agent for virtual machines
fence_vmware - Fence agent for VMware
fence_vmware_soap - Fence agent for VMware over SOAP API
fence_wti - Fence agent for WTI
fence_xvm - Fence agent for virtual machines
```

特定のフェンスタイプに指定できるオプションの一覧を表示するには、以下のコマンドを実行します:

```
ccs -h host --lsfenceopts fence_type
```

例えば、次のコマンドは **fence_wti** フェンスエージェントのフェンスオプションを一覧表示します。

```
[root@ask-03 ~]# ccs -h node1 --lsfenceopts fence_wti
fence_wti - Fence agent for WTI
  Required Options:
  Optional Options:
    option: No description available
    action: Fencing Action
    ipaddr: IP Address or Hostname
    login: Login Name
    passwd: Login password or passphrase
    passwd_script: Script to retrieve password
    cmd_prompt: Force command prompt
    secure: SSH connection
    identity_file: Identity file for ssh
    port: Physical plug number or name of virtual machine
    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
    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
```

クラスター用に現在設定されているフェンスデバイスの一覧を表示するには、次のコマンドを実行します:

```
ccs -h host --lsfencedev
```

5.7. クラスターメンバー用のフェンシングの設定

クラスターの作成とフェンスデバイスの作成の初期ステップを完了したら、クラスターノード用にフェンシングを設定する必要があります。新規のクラスターを作成してそのクラスター用にフェンスデバイスを設定した後に、ノード群にフェンシングを設定するには、このセクションの以下のステップに従っ

てください。フェンシングはクラスター内のそれぞれのノードに設定する必要があることに注意してください。

注記

各ノードに複数のフェンシングメカニズムを設定することが推奨されます。フェンシングデバイスが失敗する要因には、ネットワークの分割、電源異常、フェンシングデバイスそのものに問題がある場合、などがあります。複数のフェンシングメカニズムを設定することで、1 つのフェンシングデバイスの障害が致命的な結果になる可能性を低減することができます。

このセクションでは、以下の手順について説明します。

- 「単一パワーベースのフェンスデバイスによるノードの設定」
- 「単一のストレージベースのフェンスデバイスによるノードの設定」
- 「バックアップフェンスデバイスの設定」
- 「冗長電源を持つノードの設定」
- 「フェンスメソッドとフェンスインスタンスの削除」

5.7.1. 単一パワーベースのフェンスデバイスによるノードの設定

以下の手順にしたがって、単一のパワーベースのフェンスデバイスでノードを設定します。これは **fence_apc** フェンシングエージェントを使用する **my_apc** と呼ばれるフェンスデバイスを使います。この例では、「フェンスデバイスの設定」 にあるように、これまで **my_apc** という名前のデバイスが **--addfencedev** オプションで設定されていました。

1. ノード用のフェンスメソッドを追加して、そのフェンスメソッドの名前を入力します。

   ```
   ccs -h host --addmethod method node
   ```

   例えば、クラスターノード **node-01.example.com** にある設定ファイル内でノード **node-01.example.com** 用に **APC** と呼ばれるフェンスメソッドを設定するには、以下のコマンドを実行します:

   ```
   ccs -h node01.example.com --addmethod APC node01.example.com
   ```

2. メソッド用のフェンスインスタンスを追加します。ノードに使用するフェンスデバイス、このインスタンスの適用先であるノード、メソッドの名前、及びこのノードに特有であるこのメソッド用のオプションを指定する必要があります:

   ```
   ccs -h host --addfenceinst fencedevicename node method [options]
   ```

   たとえば、クラスターノード **node-01.example.com** 上の設定ファイル内にフェンスインスタンスを設定するには、以下のコマンドを実行します。ここでは、**APC** と呼ばれるメソッドを使用してクラスターノード **node-01.example.com** をフェンスするために、**my_apc** と呼ばれるフェンスデバイス上の APC スイッチパワーポート 1 を使用するとします。

   ```
   ccs -h node01.example.com --addfenceinst my_apc node01.example.com
   APC port=1
   ```

クラスター内の各ノード用にフェンスを追加する必要があります。以下のコマンドを使用すると、**APC** と呼ばれるメソッドで各ノードにフェンスメソッドが設定されます。フェンスメソッド用のデバイスは **my_apc** をデバイス名として指定しますが、これは 「フェンスデバイスの設定」 で説明してあるように、以前に **--addfencedev** オプションで設定したデバイスです。各ノードは、一意の APC スイッチパワーポート番号で設定されます。**node-01.example.com** のポート番号は **1** で、**node-02.example.com** のポート番号は **2**、**node-03.example.com** のポート番号は **3** となります。

```
ccs -h node01.example.com --addmethod APC node01.example.com
ccs -h node01.example.com --addmethod APC node02.example.com
ccs -h node01.example.com --addmethod APC node03.example.com
ccs -h node01.example.com --addfenceinst my_apc node01.example.com APC
port=1
ccs -h node01.example.com --addfenceinst my_apc node02.example.com APC
port=2
ccs -h node01.example.com --addfenceinst my_apc node03.example.com APC
port=3
```

例5.2「パワーベースのフェンスメソッドを追加した後の **cluster.conf**」 は、クラスター内の各ノードにこれらのフェンシングメソッドとインスタンスを追加した後の **cluster.conf** 設定ファイルを示しています。

例5.2 パワーベースのフェンスメソッドを追加した後の cluster.conf

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC">
              <device name="my_apc" port="1"/>
             </method>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="APC">
              <device name="my_apc" port="2"/>
            </method>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
            <method name="APC">
              <device name="my_apc" port="3"/>
            </method>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
         <fencedevice agent="fence_apc" ipaddr="apc_ip_example"
login="login_example" name="my_apc" passwd="password_example"/>
   </fencedevices>
   <rm>
```

```
    </rm>
</cluster>
```

「クラスタノード群への設定ファイルの伝播」 に説明してあるように、クラスターの全コンポーネントの設定を終了した時点で、クラスター設定ファイルを全てのノードに対して同期する必要がある点に注意してください。

5.7.2. 単一のストレージベースのフェンスデバイスによるノードの設定

ノードのフェンスにパワーフェンス以外のメソッド (つまり、SAN/ストレージフェンシング) を使用する場合は、フェンスデバイスに *unfencing* を設定する必要があります。これにより、フェンス済みのノードは再起動されるまでは再度有効になりません。アンフェンシングを必要とするデバイスを設定する際には、最初にクラスターを停止し、デバイスおよびアンフェンシングを含むすべての設定をクラスターが開始される前に追加する必要があります。

ノードにアンフェンシングを設定する場合、**on** または **enable** の明示的なアクションが意図的に追加されたノードに対して設定を行った該当するフェンスデバイスをミラーするデバイスを指定します。

ノードのアンフェンシングに関する詳細は、**fence_node**(8) の man ページを参照してください。

以下の手順に従って、**sanswitch1** と呼ばれるフェンスデバイスを使用する単一のストレージベースのフェンスデバイスでノードを設定します。このフェンスデバイスは **fence_sanbox2** フェンシングエージェントを使用します。

1. ノード用のフェンスメソッドを追加して、そのフェンスメソッドの名前を入力します。

   ```
   ccs -h host --addmethod method node
   ```

   例えば、クラスターノード **node-01.example.com** にある設定ファイル内でノード **node-01.example.com** 用に **SAN** と呼ばれるフェンスメソッドを設定するには、以下のコマンドを実行します:

   ```
   ccs -h node01.example.com --addmethod SAN  node01.example.com
   ```

2. メソッド用のフェンスインスタンスを追加します。ノードに使用するフェンスデバイス、このインスタンスの適用先であるノード、メソッドの名前、及びこのノードに特有であるこのメソッド用のオプションを指定する必要があります:

   ```
   ccs -h host --addfenceinst fencedevicename node method [options]
   ```

   例えば、クラスターノード **node-01.example.com** 上の設定ファイル内にフェンスインスタンスを設定するには、以下のコマンドを実行します。ここでは、**SAN** と呼ばれるメソッドを使用してクラスターノード **node-01.example.com** をフェンスするために、**sanswitch1** と呼ばれるフェンスデバイス上の SAN スイッチパワーポート 11 を使用するとします:

   ```
   ccs -h node01.example.com --addfenceinst sanswitch1
   node01.example.com SAN port=11
   ```

3. このノード上でストレージベースのフェンスデバイスにアンフェンシングを設定するには、以下のコマンドを実行します。

```
ccs -h host --addunfence fencedevicename node action=on|off
```

クラスター内の各ノードにフェンスメソッドを追加する必要があります。以下のコマンドは **SAN** と呼ばれるフェンスメソッドを各ノードに設定します。フェンスメソッド用のデバイスは **sanswitch** をデバイス名として指定しますが、これは 「フェンスデバイスの設定」 で説明してあるように、以前に --addfencedev オプションで設定したデバイスです。各ノードは一意の SAN 物理ポート番号で設定されます。**node-01.example.com** 用のポート番号は **11** で、**node-02.example.com** のポート番号は **12**、**node-03.example.com** のポート番号は **13** となります。

```
ccs -h node01.example.com --addmethod SAN node01.example.com
ccs -h node01.example.com --addmethod SAN node02.example.com
ccs -h node01.example.com --addmethod SAN node03.example.com
ccs -h node01.example.com --addfenceinst sanswitch1 node01.example.com SAN
port=11
ccs -h node01.example.com --addfenceinst sanswitch1 node02.example.com SAN
port=12
ccs -h node01.example.com --addfenceinst sanswitch1 node03.example.com SAN
port=13
ccs -h node01.example.com --addunfence sanswitch1 node01.example.com
port=11 action=on
ccs -h node01.example.com --addunfence sanswitch1 node02.example.com
port=12 action=on
ccs -h node01.example.com --addunfence sanswitch1 node03.example.com
port=13 action=on
```

例5.3「ストレージベースのフェンスメソッドを追加した後の **cluster.conf**」 は、フェンシングメソッド、フェンシングインスタンス、及びアンフェンシングをクラスター内の各ノードに追加した後の **cluster.conf** 設定ファイルを示しています。

例5.3 ストレージベースのフェンスメソッドを追加した後の cluster.conf

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="SAN">
       <device name="sanswitch1" port="11"/>
             </method>
         </fence>
         <unfence>
             <device name="sanswitch1" port="11" action="on"/>
         </unfence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="SAN">
       <device name="sanswitch1" port="12"/>
             </method>
         </fence>
         <unfence>
             <device name="sanswitch1" port="12" action="on"/>
         </unfence>
     </clusternode>
```

```
      <clusternode name="node-03.example.com" nodeid="3">
          <fence>
             <method name="SAN">
        <device name="sanswitch1" port="13"/>
              </method>
          </fence>
          <unfence>
              <device name="sanswitch1" port="13" action="on"/>
          </unfence>
      </clusternode>
   </clusternodes>
   <fencedevices>
        <fencedevice agent="fence_sanbox2" ipaddr="san_ip_example"
login="login_example" name="sanswitch1" passwd="password_example"/>
   </fencedevices>
   <rm>
   </rm>
</cluster>
```

「クラスタノード群への設定ファイルの伝播」 に説明してあるように、クラスターの全コンポーネントの設定を終了した時点で、クラスター設定ファイルを全てのノードに対して同期する必要がある点に注意してください。

5.7.3. バックアップフェンスデバイスの設定

1 つのノードに対して複数のフェンシングメソッドを定義できます。1 番目のメソッドを使用してフェンシングが失敗すると、システムは 2 番目のメソッドを使用してノードのフェンスを試行します。その後に、追加で設定したメソッドが使用されます。ノードにバックアップのフェンシングメソッドを設定するには、ノードに 2 つのメソッドを設定し、各メソッドにフェンスインスタンスを設定します。

注記

設定したフェンシングメソッドをシステムが使用する順序は、クラスター設定ファイル内の順番に従います。**ccs** コマンドで設定する最初のメソッドがプライマリのフェンシングメソッドとなり、2 番目に設定するメソッドがバックアップのフェンシングメソッドとなります。この順序を変更するには、設定ファイルからプライマリのフェンシングメソッドを削除後、そのメソッドを再度追加しなおします。

以下のコマンドを実行すると、ノード用に現在設定されているフェンスメソッドとインスタンスの一覧をいつでも表示することができます。ノードを指定しないと、このコマンドは全てのノードで現在設定されているフェンスメソッドとインスタンスを一覧表示します。

```
ccs -h host --lsfenceinst [node]
```

次の手順にしたがい、ノードにプライマリのフェンシングメソッドを設定します。 使用するフェンスデバイスは **my_apc**、フェンスエージェントは **fence_apc** です。また、 バックアップ用のフェンスデバイスは **sanswitch1**、フェンスエージェントは **fence_sanbox2** を使用します。**sanswitch1** デバイスはストレージベースのフェンスエージェントのため、このデバイスにアンフェンシングを設定する必要もあります。

1. ノード用にプライマリのフェンスメソッドを追加して、そのフェンスメソッドに名前を付けます。

```
ccs -h host --addmethod method node
```

例えば、クラスターノード **node-01.example.com** にある設定ファイル内でノード **node-01.example.com** 用のプライマリメソッドとして **APC** と呼ばれるフェンスメソッドを設定するには、以下のコマンドを実行します:

```
ccs -h node01.example.com --addmethod APC node01.example.com
```

2. プライマリメソッド用にフェンスインスタンスを追加します。ノード用に使用するフェンスデバイス、このインスタンスの適用先となるノード、メソッドの名前、及びこのノードに特有であるこのメソッド用のオプションを指定する必要があります:

```
ccs -h host --addfenceinst fencedevicename node method [options]
```

たとえば、クラスターノード **node-01.example.com** 上の設定ファイル内にフェンスインスタンスを設定するには、以下のコマンドを実行します。ここでは、**APC** と呼ばれるメソッドを使用してクラスターノード **node-01.example.com** をフェンスするために、**my_apc** と呼ばれるフェンスデバイス上の APC スイッチパワーポート 1 を使用するとします。

```
ccs -h node01.example.com --addfenceinst my_apc node01.example.com
APC port=1
```

3. ノード用のバックアップフェンスメソッドを追加してフェンスメソッドに名前を付けます。

```
ccs -h host --addmethod method node
```

例えば、クラスターノード **node-01.example.com** にある設定ファイル内でノード **node-01.example.com** 用に **SAN** と呼ばれるバックアップフェンスメソッドを設定するには、以下のコマンドを使用します:

```
ccs -h node01.example.com --addmethod SAN  node01.example.com
```

4. バックアップメソッド用にフェンスインスタンスを追加します。ノード用に使用するフェンスデバイス、このインスタンスの適用先となるノード、メソッドの名前、及びこのノードに特有であるこのメソッド用のオプションを指定する必要があります:

```
ccs -h host --addfenceinst fencedevicename node method [options]
```

例えば、クラスターノード **node-01.example.com** 上の設定ファイル内にフェンスインスタンスを設定するには、以下のコマンドを実行します。ここでは、**SAN** と呼ばれるメソッドを使用してクラスターノード **node-01.example.com** をフェンスするために、**sanswitch1** と呼ばれるフェンスデバイス上の SAN スイッチパワーポート 11 を使用するとします:

```
ccs -h node01.example.com --addfenceinst sanswitch1
node01.example.com SAN port=11
```

5. **sanswitch1** デバイスはストレージベースのデバイスであるため、このデバイス用にアンフェンシングを設定する必要があります。

```
ccs -h node01.example.com --addunfence sanswitch1 node01.example.com
port=11 action=on
```

必要に応じてフェンシングメソッドを追加し続けることが可能です。

この手順は、クラスター内の 1 つのノードにフェンスデバイスとバックアップフェンスデバイスを設定します。クラスター内の他のノードにもフェンシングを設定する必要があります。

例5.4「バックアップのフェンスメソッドを追加した後の **cluster.conf**」 は、パワーベースのプライマリのフェンシングメソッド及びストレージベースのバックアップのフェンシングメソッドを、クラスター内の各ノードに追加した後の **cluster.conf** 設定ファイルを示しています。

例5.4 バックアップのフェンスメソッドを追加した後の cluster.conf

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC">
              <device name="my_apc" port="1"/>
             </method>
            <method name="SAN">
       <device name="sanswitch1" port="11"/>
             </method>
         </fence>
         <unfence>
             <device name="sanswitch1" port="11" action="on"/>
         </unfence
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="APC">
              <device name="my_apc" port="2"/>
            </method>
            <method name="SAN">
       <device name="sanswitch1" port="12"/>
             </method>
         </fence>
         <unfence>
             <device name="sanswitch1" port="12" action="on"/>
         </unfence
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
            <method name="APC">
              <device name="my_apc" port="3"/>
            </method>
            <method name="SAN">
       <device name="sanswitch1" port="13"/>
             </method>
         </fence>
         <unfence>
             <device name="sanswitch1" port="13" action="on"/>
         </unfence
     </clusternode>
   </clusternodes>
   <fencedevices>
        <fencedevice agent="fence_apc" ipaddr="apc_ip_example"
```

```
login="login_example" name="my_apc" passwd="password_example"/>
        <fencedevice agent="fence_sanbox2" ipaddr="san_ip_example"
login="login_example" name="sanswitch1" passwd="password_example"/>
   </fencedevices>
   <rm>
   </rm>
</cluster>
```

「クラスタノード群への設定ファイルの伝播」 に説明してあるように、クラスターの全コンポーネントの設定を終了した時点で、クラスター設定ファイルを全てのノードに対して同期する必要がある点に注意してください。

注記

設定したフェンシングメソッドをシステムが使用する順序は、クラスター設定ファイル内の順番に従います。最初に設定するメソッドがプライマリのフェンシングメソッドとなり、2 番目に設定するメソッドがバックアップのフェンシングメソッドとなります。この順序を変更するには、設定ファイルからプライマリのフェンシングメソッドを削除後、そのメソッドを再度追加しなおします。

5.7.4. 冗長電源を持つノードの設定

使用中のクラスターがノード群用に冗長電源装置で設定されている場合は、そのノード群がフェンスされる必要があるときに完全にシャットダウンされるようにフェンシングを設定する必要があります。各電源装置を別々のフェンスメソッドとして設定すると、各電源装置は別々にフェンスされます。つまり、1 つ目の電源装置がフェンスされると、2 つ目の電源装置によりシステムは引き続き実行可能なため、システムは全くフェンスされません。二重電源装置のシステムを設定するには、両方の電源装置が遮断されてシステムが完全に停止するようにフェンスデバイスを設定する必要があります。そのためには単一のフェンシングメソッド内に 2 つのインスタンスを設定する必要があります。また、各インスタンスに対して、両方のフェンスデバイスの **action** 属性を **off** に設定してから、各デバイスの**action** 属性を **on** に設定する必要があります。

二重電源装置のノードにフェンシングを設定するには、このセクションの以下の手順に従ってください。

1. 冗長電源を持つノードにフェンシングを設定するには、各電源スイッチをクラスターのフェンスデバイスとして設定する必要があります。フェンスデバイスの設定に関する詳細は、「フェンスデバイスの設定」 を参照してください。

   クラスター用に現在設定されているフェンスデバイスの一覧を表示するには、次のコマンドを実行します:

   ```
   ccs -h host --lsfencedev
   ```

2. ノード用のフェンスメソッドを追加して、そのフェンスメソッドの名前を入力します。

   ```
   ccs -h host --addmethod method node
   ```

   例えば、クラスターノード **node-01.example.com** にある設定ファイル内でノード **node-01.example.com** 用に **APC-dual** と呼ばれるフェンスメソッドを設定するには、以下のコマンドを実行します:

```
ccs -h node01.example.com --addmethod APC-dual node01.example.com
```

3. 1 つ目の電源装置用のフェンスインスタンスをフェンスメソッドに追加します。ノードに使用するフェンスデバイス、このインスタンスの適用先となるノード、メソッドの名前、及びこのノードに特有であるこのメソッド用のオプションを指定する必要があります。ここで **action** 属性を **off** として設定します。

```
ccs -h host --addfenceinst fencedevicename node method [options]
action=off
```

例えば、クラスターノード **node-01.example.com** 上の設定ファイル内にフェンスインスタンスを設定するには、以下のコマンドを実行します。ここでは、**APC-dual** と呼ばれるメソッドを使用してクラスターノード **node-01.example.com** をフェンスするために、**apc1** と呼ばれるフェンスデバイス上の APC スイッチパワーポート 1 を使用するとします。また、**action** 属性を **off** に設定します。

```
ccs -h node01.example.com --addfenceinst apc1 node01.example.com
APC-dual port=1 action=off
```

4. 2 つ目の電源装置用のフェンスインスタンスをフェンスメソッドに追加します。ノードに使用するフェンスデバイス、このインスタンスの適用先となるノード、メソッドの名前、及びこのノードに特有であるこのメソッド用のオプションを指定する必要があります。ここで、このインスタンスにも **action** 属性を **off** として設定します:

```
ccs -h host --addfenceinst fencedevicename node method [options]
action=off
```

例えば、クラスターノード **node-01.example.com** 上の設定ファイル内に 2 つ目のフェンスインスタンスを設定するには、以下のコマンドを実行します。ここでは、**APC-dual** と呼ばれる 1 つ目のインスタンスに指定した時と同じメソッドを使用してクラスターノード **node-01.example.com** をフェンスするために、**apc2** と呼ばれるフェンスデバイス上の APC スイッチパワーポート 1 を使用するとします。また、**action** 属性を **off** に設定します。

```
ccs -h node01.example.com --addfenceinst apc2 node01.example.com
APC-dual port=1 action=off
```

5. ここで、1 つ目の電源装置用に別のフェンスインスタンスをフェンスメソッドに追加して、**action** 属性を **on** として設定します。ノードに使用するフェンスデバイス、このインスタンスの適用先となるノード、メソッドの名前、及びこのノードに特有であるこのメソッド用のオプションを指定する必要があります。また、**action** 属性を **on** として指定します:

```
ccs -h host --addfenceinst fencedevicename node method [options]
action=on
```

例えば、クラスターノード **node-01.example.com** 上の設定ファイル内にフェンスインスタンスを設定するには、以下のコマンドを実行します。ここでは、**APC-dual** と呼ばれるメソッドを使用してクラスターノード **node-01.example.com** をフェンスするために、**apc1** と呼ばれるフェンスデバイス上の APC スイッチパワーポート 1 を使用するとします。また、**action** 属性を **on** に設定します。

```
ccs -h node01.example.com --addfenceinst apc1 node01.example.com
APC-dual port=1 action=on
```

6. 2 つ目の電源装置用のフェンスインスタンスをフェンスメソッドに追加し、このインスタンス用に **action** 属性を **on** に指定します。ノードに使用するフェンスデバイス、このインスタンスの適用先となるノード、メソッドの名前、及びこのノードに特有であるこのメソッド用のオプションを指定する必要があります。また、**action** 属性を **on** に指定します。

```
ccs -h host --addfenceinst fencedevicename node method [options]
action=on
```

例えば、クラスターノード **node-01.example.com** 上の設定ファイル内に 2 つ目のフェンスインスタンスを設定するには、以下のコマンドを実行します。ここでは、**APC-dual** と呼ばれる 1 つ目のインスタンスに指定した時と同じメソッドを使用してクラスターノード **node-01.example.com** をフェンスするために、**apc2** と呼ばれるフェンスデバイス上の APC スイッチパワーポート 1 を使用するとします。また、**action** 属性を **on** に設定します。

```
ccs -h node01.example.com --addfenceinst apc2 node01.example.com
APC-dual port=1 action=on
```

例5.5「二重パワーフェンシングを追加した後の **cluster.conf**」 は、2 つの電源装置用のフェンシングをクラスターの各ノードに追加した後の **cluster.conf** 設定ファイルを示しています。

例5.5 二重パワーフェンシングを追加した後の cluster.conf

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC-dual">
              <device name="apc1" port="1"action="off"/>
              <device name="apc2" port="1"action="off"/>
              <device name="apc1" port="1"action="on"/>
              <device name="apc2" port="1"action="on"/>
             </method>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="APC-dual">
              <device name="apc1" port="2"action="off"/>
              <device name="apc2" port="2"action="off"/>
              <device name="apc1" port="2"action="on"/>
              <device name="apc2" port="2"action="on"/>
            </method>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
            <method name="APC-dual">
              <device name="apc1" port="3"action="off"/>
              <device name="apc2" port="3"action="off"/>
              <device name="apc1" port="3"action="on"/>
              <device name="apc2" port="3"action="on"/>
            </method>
```

```
          </fence>
      </clusternode>
   </clusternodes>
   <fencedevices>
       <fencedevice agent="fence_apc" ipaddr="apc_ip_example"
login="login_example" name="apc1" passwd="password_example"/>
       <fencedevice agent="fence_apc" ipaddr="apc_ip_example"
login="login_example" name="apc2" passwd="password_example"/>
   </fencedevices>
   <rm>
   </rm>
</cluster>
```

「クラスタノード群への設定ファイルの伝播」 に説明してあるように、クラスターの全コンポーネントの設定を終了した時点で、クラスター設定ファイルを全てのノードに対して同期する必要がある点に注意してください。

5.7.5. フェンスメソッドとフェンスインスタンスの削除

フェンスメソッドをクラスター設定から削除するには、以下のコマンドを実行します:

```
ccs -h host --rmmethod method node
```

例えば、**node01.example.com** 用に設定した **APC** と呼ばれるフェンスメソッドを、クラスターノード **node01.example.com** 上のクラスター設定ファイルから削除するには、以下のコマンドを実行します:

```
ccs -h node01.example.com  --rmmethod APC node01.example.com
```

フェンスデバイスの全てのフェンスインスタンスをフェンスメソッドから削除するには、以下のコマンドを実行します:

```
ccs -h host --rmfenceinst fencedevicename node method
```

例えば、 **node01.example.com** 用に設定された **APC-dual** と呼ばれるメソッドの **apc1** と呼ばれるフェンスデバイスの全てのインスタンスを、クラスターノード **node01.example.com** 上のクラスター設定ファイルから削除するには、以下のコマンドを実行します:

```
ccs -h node01.example.com --rmfenceinst apc1 node01.example.com APC-dual
```

5.8. フェイルオーバードメインの設定

フェイルオーバードメインは、ノードに障害が発生した場合にクラスターサービスを実行するのに有効なクラスターノードの名前付きサブセットです。フェイルオーバードメインは、以下の特性を持つことができます：

- Unrestricted（制限なし） — 優先するメンバーのサブセットを指定できるようにしますが、このドメインに割り当てられたクラスターサービスはいずれの利用可能なメンバーでも実行できます。

- Restricted（制限あり） — 特定のクラスターサービスを実行できるメンバーを制限できるようにします。制限されたフェイルオーバードメインのメンバーがどれも使用できない場合は、クラスターサービスを (手動あるいはクラスターソフトウェアによっても) 開始できません。

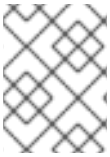
- Unordered（優先順なし） — クラスターサービスが優先順なしのフェイルオーバードメインへ割り当てられる場合、クラスターサービスを実行するメンバーは、優先順なしの利用可能なフェイルオーバードメインメンバーから選択されます。

- Ordered（優先順あり） — フェイルオーバードメインのメンバー間で優先順を指定できるようにします。一覧のトップにあるメンバーが最優先となり、次に一覧の 2 番目のメンバー、その後も順に続きます。

- Failback（フェイルバック） — フェイルオーバードメイン内のサービスが、ノード障害の前に元々実行していたノードへフェイルバックするかどうかを指定できるようにします。この特性の設定が役立つのは、ノードが繰り返し失敗し、それが優先順ありのフェイルオーバードメインの一部であるような状況です。この状況では、ノードがフェイルオーバードメイン内の優先ノードである場合には、優先ノードと別のノード間でサービスがフェイルオーバーとフェイルバックを繰り返す可能性があるため、パフォーマンスに重大な影響を与えることになります。

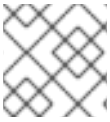
注記

優先順のあるフェイルオーバーが設定されている場合にのみ、フェイルバックの特性が利用できます。



注記

フェイルオーバードメイン設定を変更しても、現在実行中のサービスには影響しません。

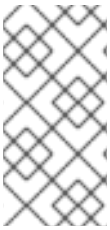


注記

フェイルオーバードメインは、運用には必須 **ではありません。**

デフォルトでは、フェイルオーバードメインは制限なしで優先順がありません。

複数のメンバーを持つクラスター内では、制限ありのフェイルオーバードメインを使用することで、クラスターサービス (**httpd** など) を実行するためのクラスターを設定する作業を最小限にできます。このためには、クラスターサービスを実行する全てのメンバー上で、設定を同一にする必要があります。クラスターサービスを実行するようクラスター全体を設定する代わりに、クラスターサービスに関連付ける制限ありのフェイルオーバードメイン内のメンバーのみを設定するだけで済みます。



注記

優先するメンバーを設定するには、1 つだけのクラスターメンバーから成る制限なしのフェイルオーバードメインを作成します。そうすることで、クラスターサービスが主にそのクラスターメンバー (優先メンバー) 上で実行することになりますが、クラスターサービスは他のどのメンバーにでもフェイルオーバーできるようになります。

フェイルオーバードメインを設定するには、以下の手順を実行します:

1. フェイルオーバードメインを追加するには、以下のコマンドを実行します:

```
ccs -h host --addfailoverdomain name [restricted] [ordered]
[nofailback]
```

注記

名前は、クラスター内で使用されている他の名前に対して、その目的を区別できるような説明的な名前にすることをお勧めします。

例として次のコマンドでは、**node-01.example.com** 上で制限なし、優先順あり、フェイルバックが可能な **example_pri** と呼ばれるフェイルオーバードメインを設定します:

```
ccs -h node-01.example.com --addfailoverdomain example_pri ordered
```

2. ノードをフェイルオーバードメインに追加するには、以下のコマンドを実行します:

```
ccs -h host --addfailoverdomainnode failoverdomain node priority
```

例えば、**node-01.example.com** 上の設定ファイルでフェイルオーバードメイン **example_pri** を設定して、優先順位 1 を持つ **node-01.example.com**、優先順位 2 を持つ **node-02.example.com**、及び優先順位 3 を持つ **node-03.example.com** を含むようにするには、以下のコマンド群を実行します:

```
ccs -h node-01.example.com --addfailoverdomainnode example_pri node-
01.example.com 1
ccs -h node-01.example.com --addfailoverdomainnode example_pri node-
02.example.com 2
ccs -h node-01.example.com --addfailoverdomainnode example_pri node-
03.example.com 3
```

以下のコマンドで、クラスター内で設定された全てのフェイルオーバードメインとフェイルオーバードメインノードを一覧表示することができます:

```
ccs -h host --lsfailoverdomain
```

フェイルオーバードメインを削除するには、以下のコマンドを実行します:

```
ccs -h host --rmfailoverdomain name
```

フェイルオーバードメインからノードを削除するには、以下のコマンドを実行します:

```
ccs -h host --rmfailoverdomainnode failoverdomain node
```

「クラスタノード群への設定ファイルの伝播」 に説明してあるように、クラスターの全コンポーネントの設定を終了した時点で、クラスター設定ファイルを全てのノードに対して同期する必要がある点に注意してください。

5.9. グローバルクラスターリソースの設定

2 つのタイプのリソースを設定できます:

- Global （グローバル） — クラスター内の全てのサービスに利用可能なリソースです。

- Service-specific（サービス特有） — 1 つのサービスにのみ利用可能なリソースです。

クラスター内で現在設定されているリソースとサービスを一覧表示するには、以下のコマンドを実行します:

```
ccs -h host --lsservices
```

グローバルクラスターリソースを追加するには、以下のコマンドを実行します。「クラスターへのクラスターサービスの追加」 で説明してあるように、サービスの設定時に、特定のサービスにローカルであるリソースを追加することができます。

```
ccs -h host --addresource resourcetype [resource options]
```

例えば、以下のコマンドはグローバルファイルシステムリソースを **node01.example.com** 上のクラスター設定ファイルに追加します。リソースの名前は **web_fs** で、ファイルシステムデバイスは **/dev/sdd2**、ファイルシステムのマウントポイントは **/var/www**、ファイルシステムタイプは **ext3** となります。

```
ccs -h node01.example.com --addresource fs name=web_fs device=/dev/sdd2
mountpoint=/var/www fstype=ext3
```

利用可能なリソースタイプとリソースオプションに関する詳細は 付録B *HA リソースパラメーター* をご覧ください。

グローバルリソースを削除するには、以下のコマンドを実行します:

```
ccs -h host --rmresource resourcetype [resource options]
```

既存のグローバルリソースのパラメーターを変更する必要がある場合は、そのリソースを削除してから再度設定し直します。

「クラスタノード群への設定ファイルの伝播」 に説明してあるように、クラスターの全コンポーネントの設定を終了した時点で、クラスター設定ファイルを全てのノードに対して同期する必要がある点に注意してください。

5.10. クラスターへのクラスターサービスの追加

クラスター内にクラスターサービスを設定するには、以下の手順を実行します:

1. 以下のコマンドを使用してクラスターにサービスを追加します:

   ```
   ccs -h host --addservice servicename [service options]
   ```

   注記

   名前は、クラスター内の他のサービスと明白に区別できる名前を使用してください。

   クラスター設定にサービスを追加する時、以下の属性を設定します。

   - **autostart** — クラスターの開始時にサービスを自動起動するかどうかを指定します。有効にするには「1」を、無効にするには「0」を使用します。デフォルトは有効です。

- **domain** — フェイルオーバードメインを指定します（必要な場合）。
- **exclusive** — 他のサービスが稼働していないノード上でのみサービスを稼働するポリシーを指定します。
- **recovery** — サービスの回復ポリシーを指定します。オプションとしては、サービスの再配置、再起動、無効、再起動後に無効、があります。再起動の回復ポリシーは、システムが障害が発生したサービスを別のノードに再配置する前に、そのサービスの再起動を試行することを示しています。再配置のポリシーは、異なるノードでサービスの再起動を試行することを意味します。無効のポリシーは、いずれかのコンポーネントに障害が発生した場合にシステムがリソースグループを無効にすることを示しています。再起動後に無効のポリシーでは、サービスに障害が発生した場合システムがサービスの再起動を試行することを意味します。ただし、そのサービスの再起動が失敗すると、サービスはクラスター内の別のホストに移る代わりに無効になります。

  サービスの回復ポリシーとして **Restart** 又は **Restart-Disable** を選択した場合、サービスの再配置/無効化までに再起動が失敗する最大回数と、再起動を破棄するまでの時間を秒単位で指定することができます。

例えば、フェイルオーバードメイン **example_pri** を使用し、回復ポリシー **relocate** を持つ **example_apache** と呼ばれるクラスターノード **node-01.example.com** にある設定ファイルにサービスを追加するには、以下のコマンドを使用します:

```
ccs -h node-01.example.com --addservice example_apache
domain=example_pri recovery=relocate
```

クラスターにサービスを設定する場合、クラスターに利用可能なサービスや各サービスに使用できるオプションの一覧を確認すると役立つ場合があります。利用可能なサービスとオプションの一覧を表示するための **ccs** コマンドの使用方法については、「利用可能なクラスターサービスおよびリソースの一覧表示」 を参照してください。

2. 以下のコマンドを使用してリソースをサービスに追加します:

```
ccs -h host --addsubservice servicename subservice [service options]
```

使用したいリソースのタイプに応じて、サービスにグローバル又はサービス特有のリソースを入力します。グローバルリソースを追加するには、**ccs** コマンドに **--addsubservice** オプションを使用してリソースを追加します。例えば、**web_fs** と呼ばれるグローバルファイルシステムリソースを **node-01.example.com** にあるクラスター設定ファイル上の **example_apache** と呼ばれるサービスに追加するには、以下のコマンドを実行します:

```
ccs -h node01.example.com --addsubservice example_apache fs
ref=web_fs
```

サービス特有のリソースをサービスに追加するには、全てのサービスオプションを指定する必要があります。例えば、以前に **web_fs** をグローバルサービスとして定義していない場合は、次のコマンドを使ってそれをサービス特有のリソースとして追加できます:

```
ccs -h node01.example.com --addsubservice example_apache fs
name=web_fs device=/dev/sdd2 mountpoint=/var/www fstype=ext3
```

3. 子サービスをサービスに追加するには、**ccs** コマンドに **--addsubservice** オプションを使用して、サービスオプションを指定します。

依存関係のツリー構造の中でサービスを追加する必要がある場合は、コロン (:) を使用して要素を区切り、括弧を使用して同じタイプのサブサービスを指定します。以下の例では、3 つ目の **nfsclient** サービスを **nfsclient** サービスのサブサービスとして追加します。これは **nfsclient** サービスのサブサービスであり、その **nfsclient** サービスは **service_a** と呼ばれるサービスのサブサービスです。

```
ccs -h node01.example.com --addsubservice service_a
nfsclient[1]:nfsclient[2]:nfsclient
```

注記

Samba-service リソースを追加している場合は、それを他のリソースの子としてではなく、直接サービスに追加します。

注記

フローティング IP アドレスリソースを含むクラスターサービスの依存関係ツリー設定時には、IP リソースを最初のエントリーとして設定する必要があります。

注記

クラスターサービスで使用する IP サービスが存在するか確認するには、クラスターノードで (廃止された **ifconfig** コマンドではなく) **/sbin/ip addr show** コマンドを使用します。以下は、クラスターサービスを実行しているノードで **/sbin/ip addr show** コマンドを実行した場合の出力です。

```
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1356 qdisc pfifo_fast
qlen 1000
    link/ether 00:05:5d:9a:d8:91 brd ff:ff:ff:ff:ff:ff
    inet 10.11.4.31/22 brd 10.11.7.255 scope global eth0
    inet6 fe80::205:5dff:fe9a:d891/64 scope link
    inet 10.11.4.240/22 scope global secondary eth0
       valid_lft forever preferred_lft forever
```

サービスとそのサブサービスの全てを削除するには、以下のコマンドを実行します:

```
ccs -h host --rmservice servicename
```

サブサービスを削除するには、以下のコマンドを実行します:

```
ccs -h host --rmsubservice servicename subservice [service options]
```

「クラスタノード群への設定ファイルの伝播」 に説明してあるように、クラスターの全コンポーネントの設定を終了した時点で、クラスター設定ファイルを全てのノードに対して同期する必要がある点に注意してください。

5.11. 利用可能なクラスターサービスおよびリソースの一覧表示

ccs コマンドを使用すると、クラスターに利用可能なサービスおよびリソースの一覧を表示できます。**ccs** コマンドを使用して、特定のサービスまたはリソースタイプに指定可能なオプションを一覧表示することもできます。

使用中のクラスターに利用可能なクラスターサービスの一覧を表示するには、次のコマンドのいずれかを実行します (**--lsresourceopts** は **--lsserviceopts** のエイリアス)。

```
ccs -h host --lsserviceopts
ccs -h host --lsresourceopts
```

たとえば、次のコマンドはクラスターノード **node1** で利用可能なクラスターサービスおよびリソースを一覧表示します。サンプルの出力を表示します。

```
[root@ask-03 ~]# ccs -h node1 --lsserviceopts
service - Defines a service (resource group).
ASEHAagent - Sybase ASE Failover Instance
SAPDatabase - SAP database resource agent
SAPInstance - SAP instance resource agent
apache - Defines an Apache web server
clusterfs - Defines a cluster file system mount.
fs - Defines a file system mount.
ip - This is an IP address.
lvm - LVM Failover script
mysql - Defines a MySQL database server
named - Defines an instance of named server
netfs - Defines an NFS/CIFS file system mount.
nfsclient - Defines an NFS client.
nfsexport - This defines an NFS export.
nfsserver - This defines an NFS server resource.
openldap - Defines an Open LDAP server
oracledb - Oracle 10g Failover Instance
orainstance - Oracle 10g Failover Instance
oralistener - Oracle 10g Listener Instance
postgres-8 - Defines a PostgreSQL server
samba - Dynamic smbd/nmbd resource agent
script - LSB-compliant init script as a clustered resource.
tomcat-6 - Defines a Tomcat server
vm - Defines a Virtual Machine
action - Overrides resource action timings for a resource instance.
```

特定のタイプのサービスに指定できるオプションの一覧を表示するには、以下のコマンドを実行します。

```
ccs -h host --lsserviceopts service_type
```

例えば、次のコマンドは **vm** サービス用のサービスオプションを一覧表示します。

```
[root@ask-03 ~]# ccs -f node1 --lsserviceopts vm
vm - Defines a Virtual Machine
  Required Options:
    name: Name
  Optional Options:
```

```
    domain: Cluster failover Domain
    autostart: Automatic start after quorum formation
    exclusive: Exclusive resource group
    recovery: Failure recovery policy
    migration_mapping: memberhost:targethost,memberhost:targethost ..
    use_virsh: If set to 1, vm.sh will use the virsh command to manage
virtual machines instead of xm. This is required when using non-Xen
virtual machines (e.g. qemu / KVM).
    xmlfile: Full path to libvirt XML file describing the domain.
    migrate: Migration type (live or pause, default = live).
    path: Path to virtual machine configuration files.
    snapshot: Path to the snapshot directory where the virtual machine
image will be stored.
    depend: Top-level service this depends on, in service:name format.
    depend_mode: Service dependency mode (soft or hard).
    max_restarts: Maximum restarts for this service.
    restart_expire_time: Restart expiration time; amount of time before a
restart is forgotten.
    status_program: Additional status check program
    hypervisor: Hypervisor
    hypervisor_uri: Hypervisor URI (normally automatic).
    migration_uri: Migration URI (normally automatic).
    __independent_subtree: Treat this and all children as an independent
subtree.
    __enforce_timeouts: Consider a timeout for operations as fatal.
    __max_failures: Maximum number of failures before returning a failure
to a status check.
    __failure_expire_time: Amount of time before a failure is forgotten.
    __max_restarts: Maximum number restarts for an independent subtree
before giving up.
    __restart_expire_time: Amount of time before a failure is forgotten
for an independent subtree.
```

5.12. 仮想マシンのリソース

仮想マシンのリソースを設定する方法は、特にサービスの定義にグループ化されない点など、他のクラスターリソースとは異なります。Red Hat Enterprise Linux 6.2 リリース以降、**ccs** コマンドを使用してクラスター内で仮想マシンを設定する場合、**--addvm** (**addservice** オプションではなく) を使用できます。これにより、クラスター設定ファイルの **rm** 設定ノードで直接 **vm** リソースが定義されることになります。

仮想マシンのリソースには、少なくとも **name** 及び **path** 属性が必要です。**name** 属性は **libvirt** ドメインの名前と一致し、**path** 属性は共有の仮想マシン定義が格納されるディレクトリを指定します。

注記

クラスター設定ファイルの **path** 属性は、個々のファイルへのパスではなく、パス指定又はディレクトリの名前です。

仮想マシンの定義が **/mnt/vm_defs** と呼ばれる共有ディレクトリに格納されている場合、次のコマンドは **guest1** と呼ばれる仮想マシンを定義することになります。

```
# ccs -h node1.example.com --addvm guest1 path=/mnt/vm_defs
```

このコマンドを実行すると、次の行が **cluster.conf** ファイルの **rm** 設定ノードに追加されます。

```
<vm name="guest1" path="/mnt/vm_defs"/>
```

5.13. QUORUM DISK の設定

注記

Quorum-disk のパラメーター及びヒューリスティックは、サイトの環境及び必要となる特別な要件によって異なります。quorum-disk のパラメーター及びヒューリスティックの使用について理解するには、qdisk(5) の man ページを参照してください。Quorum disk の理解と使用にサポートが必要な場合は、Red Hat 認定サポート担当者までご連絡ください。

以下のコマンドを使用して、quorum disk を使用するためにシステムを設定します:

```
ccs -h host --setquorumd [quorumd options]
```

「以前の設定を上書きするコマンド」 に記載されているように、このコマンドは、**--setquorumd** オプションで設定したその他のプロパティをすべて、デフォルト値にリセットする点に注意してください。

表5.1「Quorum disk のオプション」 は、設定する必要のある quorum disk のオプションの意味を要約しています。quorum disk のパラメーターの完全な一覧は **/usr/share/cluster/cluster.rng** にあるクラススキーマと **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** にある注釈付きスキーマを参照してください。

表5.1 Quorum disk のオプション

パラメーター	説明
interval	秒単位の読み取り/書き込みサイクルの頻度
votes	十分に高いスコアがある時に quorum デーモンが **cman** に広告する投票数
tko	ノードが dead と宣言されるまでにミスするサイクル数
min_score	ノードが「生存」と見なされるための最低限スコア。スコアが省略されている場合や「0」にセットされていると、デフォルトの関数である **floor((*n*+1)/2)** が使用されます。*n* はヒューリスティックのスコアの合計です。**Minimum Score** の値は、ヒューリスティックのスコアの合計を超過してはいけません。超過した場合は、quorum disk は利用できません。
device	quorum デーモンが使用するストレージデバイスです。このデバイスは全てのノードで同一でなければなりません。

パラメーター	説明
label	**mkqdisk** ユーティリティで作成される quorum disk のラベルを指定します。このフィールドにエントリがある場合は、ラベルは **Device** フィールドを上書きします。このフィールドが使用されると、 quorum デーモンは **/proc/partitions** を読み取り、見付かった全てのブロックデバイス上の qdisk 署名をチェックし、指定されたラベルと比較します。これは quorum デバイス名がノード間で異なる場合の設定に役立ちます。

以下のコマンドを使用して quorum disk のヒューリスティックを設定します:

```
ccs -h host --addheuristic [heuristic options]
```

表5.2「Quorum disk のヒューリスティック」 はセットする必要がある quorum disk のヒューリスティックの意味を要約しています。

表5.2 Quorum disk のヒューリスティック

パラメーター	説明
program	このヒューリスティックが利用可能か判断する際に使用されるプログラムへのパス。**/bin/sh -c** で実行可能なものなら何でも構いません。戻り値が 0 の場合は成功で、それ以外の値は失敗を意味します。このパラメーターは quorum ディスクを使用するために必要です。
interval	ヒューリスティックが投票される頻度 (秒数) です。いずれのヒューリスティックもデフォルトの間隔は 2 秒です。
score	このヒューリスティックのウェイトです。ヒューリスティックのスコアを決定する時には注意が必要です。各ヒューリスティックのデフォルトのスコアは 1 です。
tko	ヒューリスティックが利用不可能と宣言されるまでの連続する失敗数。

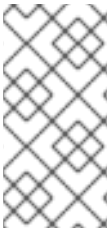
システムで設定されている quorum disk のオプションとヒューリスティックの一覧を表示するには、以下のコマンドを実行します:

```
ccs -h host --lsquorum
```

ヒューリスティックオプションで指定したヒューリスティックを削除するには、以下のコマンドを実行します:

```
ccs -h host rmheuristic [heuristic options]
```

「クラスタノード群への設定ファイルの伝播」 に説明してあるように、クラスターの全コンポーネントの設定を終了した時点で、クラスター設定ファイルを全てのノードに対して同期する必要がある点に注意してください。



注記

同期とアクティベートにより、更新されたクラスター設定ファイルの伝播とアクティベートが行われます。ただし、quorum disk が動作するには、クラスターを再起動する必要があります (「クラスターの起動と停止」 を参照)。これで、各ノードで **qdiskd** デーモンを確実に再起動できます。

5.14. その他のクラスター設定

このセクションでは、以下を設定するための **ccs** コマンドの使用方法について説明します:

- 「クラスター設定のバージョン」
- 「マルチキャスト設定」
- 「2 ノードクラスターの設定」
- 「ロギング」
- 「冗長リングプロトコルの設定」

ccs コマンドを使用すると、**totem** オプション、**dlm** オプション、**rm** オプション、及び **cman** オプションを含む高度なクラスター設定のパラメーターを設定することもできます。これらのパラメーターを設定する方法についての詳細は **ccs**(8) の man ページ、及び **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** にある注釈付きクラスター設定ファイルのスキーマをご覧ください。

クラスターに既に設定されているその他のクラスター属性の一覧を表示するには、以下のコマンドを実行します:

```
ccs -h host --lsmisc
```

5.14.1. クラスター設定のバージョン

クラスター設定ファイルには、クラスター設定のバージョン値が含まれます。設定バージョン値は、クラスター設定ファイルを作成した時点でデフォルトで **1** にセットされ、そのクラスター設定を修正する度に自動的に増加します。ただし、他の値に設定する必要がある場合は、以下のコマンドで値を指定できます:

```
ccs -h host --setversion n
```

既存のクラスター設定ファイル上の現在の設定バージョン値を表示するには、以下のコマンドを使用します:

```
ccs -h host --getversion
```

現在の設定バージョン値をクラスター内の全てのノード上のクラスター設定ファイル内で 1 つ増加させるには、以下のコマンドを実行します:

```
ccs -h host --incversion
```

5.14.2. マルチキャスト設定

クラスター設定ファイルでマルチキャストアドレスが指定されていない場合、Red Hat High Availability アドオンソフトウェアがクラスターの ID に基づいて作成します。アドレスの下位 16 ビットを生成し、それを IP プロトコルが IPv4 又は IPv6 であるかに応じてアドレスの上部に追記します:

- IPv4 の場合 — 形成されるアドレスは、239.192 に Red Hat High Availability アドオンソフトウェアにより生成される下位 16 ビットが加わります。
- IPv6 の場合 — 形成されるアドレスは、FF15:: に Red Hat High Availability アドオンソフトウェアにより生成される下位 16 ビットが加わります。

注記

クラスター ID は、各クラスター用に **cman** が生成する一意の識別子です。クラスター ID を表示するには、クラスターノードで **cman_tool status** コマンドを実行します。

クラスター設定ファイル内でマルチキャストアドレスを手動で指定するには、以下のコマンドを実行します:

```
ccs -h host --setmulticast multicastaddress
```

「以前の設定を上書きするコマンド」 に記載されているように、このコマンドは、**--setmulticast** オプションで設定したその他のプロパティをすべて、デフォルト値にリセットする点に注意してください。

マルチキャストアドレスを指定する場合は、**cman** が使用する 239.192.x.x シリーズ（又は IPv6 の場合は FF15::）を使用することをお勧めします。この範囲以外のマルチキャストアドレスを使用すると、予期できない結果が生じる場合があります。例えば、224.0.0.x (「ネットワーク上の全ホスト」) を使用した場合、正しくルートされないか、一部のハードウェアでは全くルートされない可能性さえあります。

マルチキャストアドレスを指定/修正した場合は、それを反映させるためにクラスターを再起動する必要があります。**ccs** コマンドを使ったクラスターの起動/停止に関する詳細は、「クラスターの起動と停止」 を参照してください。

注記

マルチキャストアドレスを指定する場合、クラスターパケットが通過するルーターの設定を確認するようにしてください。一部のルーターはアドレスを認識するのに長時間かかり、クラスターのパフォーマンスに重大な影響を与える場合があります。

設定ファイルからマルチキャストアドレスを削除するには、**ccs** コマンドに **--setmulticast** オプションを使用します。マルチキャストアドレスは指定しないでください。

```
ccs -h host --setmulticast
```

5.14.3. 2 ノードクラスターの設定

2 ノードクラスターを設定している場合は、以下のコマンドを実行すると単一ノードが定足数を維持できるようになります (例えば 1 つのノードが失敗した場合など):

```
ccs -h host --setcman two_node=1 expected_votes=1
```

「以前の設定を上書きするコマンド」 に記載されているように、このコマンドは、**--setcman** オプションで設定したその他のプロパティをすべて、デフォルト値にリセットする点に注意してください。

ccs --setcman コマンドを使用して **two_node** オプションの追加/削除/修正を行った場合、変更を反映させるためにクラスターを再起動する必要があります。**ccs** コマンドを使ったクラスターの起動/停止に関する詳細は、「クラスターの起動と停止」 を参照してください。

5.14.4. ロギング

クラスターの全デーモンにデバッグを有効にできます。あるいは、個々のクラスタープロセスに対してロギングを有効にすることも可能です。

全デーモンに対してデバッグを有効にするには、次のコマンドを実行します。デフォルトでは、**/var/log/cluster/*daemon*.log** ファイルにログ記録されます:

```
ccs -h host --setlogging [logging options]
```

例えば、以下のコマンドは全デーモンに対してデバッグを有効にします:

```
# ccs -h node1.example.com --setlogging debug=on
```

「以前の設定を上書きするコマンド」 に記載されているように、このコマンドは、**--setlogging** オプションで設定したその他のプロパティをすべて、デフォルト値にリセットする点に注意してください。

個々のクラスタープロセスに対してデバッグを有効にするには、次のコマンドを実行します。デーモン毎のロギング設定は、グローバル設定を上書きします:

```
ccs -h host --addlogging [logging daemon options]
```

例えば、以下のコマンドは **corosync** 及び **fenced** デーモンのデバッグを有効にします:

```
# ccs -h node1.example.com --addlogging name=corosync debug=on
# ccs -h node1.example.com --addlogging name=fenced debug=on
```

個々のデーモンに対するログ設定を削除するには、次のコマンドを使用します:

```
ccs -h host --rmlogging name=clusterprocess
```

例えば、次のコマンドは **fenced** デーモンのデーモン固有のログ設定を削除します。

```
ccs -h host --rmlogging name=fenced
```

グローバル及びデーモン毎のロギングに対して設定可能な、ロギング及び追加のロギングオプションを有効にできるロギングデーモンの一覧については、**cluster.conf**(5) の man ページを参照してください。

「クラスタノード群への設定ファイルの伝播」 に説明してあるように、クラスターの全コンポーネントの設定を終了した時点で、クラスター設定ファイルを全てのノードに対して同期する必要がある点に注意してください。

5.14.5. 冗長リングプロトコルの設定

Red Hat Enterprise Linux 6.4 では、Red Hat High Availability アドオンはリングプロトコルの冗長設定に対応しています。冗長リングプロトコルを使用する場合、「冗長リングプロトコルの設定」 の記載にあるようにさまざまな点を考慮する必要があります。

冗長リングプロトコルを使用するために 2 つ目のネットワークインターフェースを指定するには、**ccs** コマンドの **--addalt** オプションを使用してノードに別の名前を追加します。

```
ccs -h host --addalt node_name alt_name
```

例えば、以下のコマンドは、クラスターノード **clusternet-node1-eth1** に別の名前 **clusternet-node1-eth2** を設定します。

```
# ccs -h clusternet-node1-eth1 --addalt clusternet-node1-eth1 clusternet-
node1-eth2
```

オプションで、2 つ目のリングに対して、マルチキャストアドレス、ポート、TTL を手動指定できます。2 つ目のリングにマルチキャストアドレスを指定した場合、代わりのマルチキャストアドレスまたは代わりのポートは 1 つ目のマルチキャストアドレスと異なる必要があります。代替ポートを指定した場合、システム自体は操作実行にポートとポート 1 を使用するので、1 つ目のリングと 2 つ目のリングのポート番号は、2 つ以上異なるものである必要があります。別のマルチキャストアドレスを指定しなかった場合は、システムは 2 つ目のリングに別のマルチキャストアドレスを自動的に使用するようになっています。

2 つ目のリングに別のマルチキャストアドレス、ポート、TTL を指定するには、**ccs** コマンドの **--setaltmulticast** オプションを使用します。

```
ccs -h host --setaltmulticast [alt_multicast_address]
[alt_multicast_options].
```

例えば、以下のコマンドは、**clusternet-node1-eth1** ノードに、**cluster.conf** ファイルで定義したクラスターに対する別のマルチキャストアドレス 239.192.99.88、ポート 888、TTL 3 を設定します。

```
ccs -h clusternet-node1-eth1 --setaltmulticast 239.192.99.88 port=888
ttl=3
```

代替のマルチキャストアドレスを削除するには、**ccs** コマンドの **--setaltmulticast** オプションを指定しますが、マルチキャストアドレスは指定しないでください。「以前の設定を上書きするコマンド」 で記載されているように、このコマンドを実行すると、**--setaltmulticast** オプションで設定可能なその他のプロパティがすべて、デフォルト値にリセットされる点に注意してください。

「クラスタノード群への設定ファイルの伝播」 に説明してあるように、クラスターの全コンポーネントの設定が終了した時点で、クラスター設定ファイルを全てのノードに対して同期する必要があります。

5.15. クラスタノード群への設定ファイルの伝播

クラスター内のノードの 1 つでクラスターファイルを作成/編集した後には、同じファイルを全てのクラスターのノードに伝播してその設定をアクティベートする必要があります。

以下のコマンドを使用して、クラスター設定ファイルを伝播してアクティベートします:

```
ccs -h host --sync --activate
```

ホストのクラスター設定ファイル内で指定されている全てのノードに全く同一のクラスター設定ファイルがあることを確認するには、以下のコマンドを実行します:

```
ccs -h host --checkconf
```

ローカルノード上で設定ファイルを作成/編集した場合は、以下のコマンドを使用してそのファイルをクラスター内の 1 つのノードに送信します:

```
ccs -f file -h host --setconf
```

ローカルファイル内で指定されている全てのノードに全く同一のクラスター設定ファイルがあることを確認するには、以下のコマンドを実行します:

```
ccs -f file --checkconf
```

第6章 RED HAT HIGH AVAILABILITY アドオンを使用した CCS の管理

本章では、**ccs** コマンドを使用して Red Hat High Availability アドオンを管理する上での各種管理タスクについて説明しています。**ccs** コマンドは Red Hat Enterprise Linux 6.1 リリース以降のバージョンでサポートされています。本章は以下のようなセクションで構成されます。

- 「クラスターノードの管理」
- 「クラスターの起動と停止」
- 「クラスター内の問題の診断と是正」

6.1. クラスターノードの管理

このセクションは、以下のような **ccs** コマンドを使用したノード管理機能を実行する方法を説明しています:

- 「ノードのクラスターに対する離脱/参加」
- 「稼働しているクラスターへのメンバーの追加」

6.1.1. ノードのクラスターに対する離脱/参加

ccs コマンドを使用すると、ノード上でクラスターサービスを停止することによりノードがクラスターから離脱するようにできます。ノードがクラスターから離脱しても、そのノードからクラスター設定情報を削除することにはなりません。ノードをクラスターから離脱させることで、クラスターの再起動時にノードが自動的に参加できないようにします。

ノードをクラスターから離脱させるには、以下のコマンドを実行します。この操作により、**-h** オプションで指定されたノード上のクラスターサービスが停止します:

```
ccs -h host --stop
```

ノード上のクラスターサービスを停止すると、そのノード上で実行中のサービスはいずれもフェイルオーバーします。

クラスター設定からノードを完全に削除するには、「クラスターの作成」 で説明してあるように、**ccs** コマンドで **--rmnode** オプションを使用します。

ノードをクラスターに再参加させるには、以下のコマンドを実行します。この操作は **-h** オプションで指定されたノード上でクラスターサービスを起動します:

```
ccs -h host --start
```

6.1.2. 稼働しているクラスターへのメンバーの追加

メンバーを実行中のクラスターに追加するには、 「クラスターの作成」 で説明されている方法でノードをクラスターに追加します。設定ファイルを更新した後に、 「クラスタノード群への設定ファイルの伝播」 の説明のとおりにそのファイルをクラスターの全ノードに伝播して、新しいクラスター設定ファイルをアクティベートします。

6.2. クラスターの起動と停止

ccs コマンドを使用してクラスターを停止するには、以下のコマンドを使ってクラスター内の全てのノード上でクラスターサービスを停止します:

```
ccs -h host --stopall
```

ccs コマンドを使用して稼働していないクラスターを起動するには、以下のコマンドを使ってクラスター内の全てのノード上でクラスターサービスを起動します:

```
ccs -h host --startall
```

6.3. クラスター内の問題の診断と是正

クラスター内の問題の診断と是正についての詳細は 9章 *クラスター内の問題の診断と修正* をご覧ください。ただし、**ccs** コマンドを使用してもいくつかの簡単なチェックを実行できます。

ホストのクラスター設定ファイル内で指定されている全てのノードに全く同一のクラスター設定ファイルがあることを検証するには、以下のコマンドを実行します：

```
ccs -h host --checkconf
```

ローカルノードで設定ファイルを作成/編集した場合に、ローカルファイル内で指定された全てのノードが全く同一のクラスター設定ファイルを持っていることを検証するには、次のコマンドを使用します：

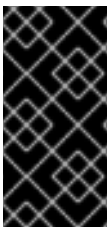
```
ccs -f file --checkconf
```

第7章 RED HAT HIGH AVAILABILITY の手動での設定

本章では、クラスター設定ファイル (**/etc/cluster/cluster.conf**) を直接編集してコマンドラインツールを使用することにより Red Hat High Availability アドオンを設定する方法を説明しています。本章は、まず提供されるサンプルのファイルから一度に 1 セクションずつ設定ファイルを構築する手順を紹介します。提供されているサンプルファイルから始める方法以外に、**cluster.conf** の man ページからスケルトンの設定ファイルをコピーすることもできます。しかし、それを行った場合、本章で後ほど示される手順の情報と必ずしも合うとは限りません。クラスター設定ファイルを作成して設定する方法は他にもありますが、本章では、一度に 1 セクションずつ設定ファイルを構築する手順を提供します。また、この手順はあくまで使用しているクラスタリングニーズに適合するための設定ファイルを開発する出発点であることも注意してください。

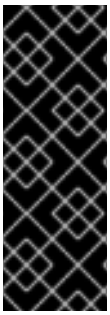
本章は以下のセクションで構成されます。

- 「設定のタスク」
- 「基本的なクラスター設定ファイルの作成」
- 「フェンシングの設定」
- 「フェイルオーバードメインの設定」
- 「HA サービスの設定」
- 「デバッグオプションの設定」
- 「冗長リングプロトコルの設定」
- 「設定の確認」



重要

High Availability アドオンの導入がご使用のニーズに適合していて、サポート可能であることを確認してください。導入する前に、Red Hat 認定担当者に連絡して、設定を検証してください。さらに、設定のバーンイン期間を設けて障害モードのテストを実施してください。

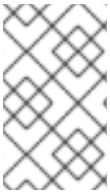


重要

本章では、一般に使用されている **cluster.conf** の要素と属性を参照します。**cluster.conf** の要素と属性の総括的な一覧と説明は、**/usr/share/cluster/cluster.rng** にあるクラスタースキーマと **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** (例えば、**/usr/share/doc/cman-3.0.12/cluster_conf.html**) にある注釈付きスキーマを参照してください。

重要

本章にある一部の手順では、クラスター設定をクラスター全域に伝播するために **cman_tool version -r** コマンドを使用する必要があります。このコマンドを使用するには、**ricci** が稼働していなければなりません。**ricci** を使用する場合は、いずれかのマシンから **ricci** とやりとりをする初回時にパスワードが必要となります。**ricci** サービスについての詳細情報は、「**ricci** の注意事項」 を参照してください。



注記

本章にある手順には 付録E *コマンドラインツールの要約* に一覧表示されているコマンドラインツールの特定のコマンドが含まれている場合があります。全てのコマンドと変数についての詳細は、各コマンドラインツール用の man ページを参照してください。

7.1. 設定のタスク

コマンドラインツールを使用した Red Hat High Availability アドオンソフトウェアを設定する方法は、以下の手順で行います：

1. クラスターを作成。「基本的なクラスター設定ファイルの作成」 を参照してください。
2. フェンシングを設定。「フェンシングの設定」 を参照してください。
3. フェイルオーバードメインを設定。「フェイルオーバードメインの設定」 を参照してください。
4. HA サービスを設定。「HA サービスの設定」 を参照してください。
5. 設定を確認。「設定の確認」 を参照してください。

7.2. 基本的なクラスター設定ファイルの作成

クラスターハードウェア、Red Hat Enterprise Linux、および High Availability アドオンソフトウェアがインストールされていれば、クラスター設定ファイル (**/etc/cluster/cluster.conf**) を作成して High Availability アドオンの実行を開始することができます。このセクションでは、単に開始点として、フェンシング、フェイルオーバードメイン、および HA サービスのないスケルトンクラスター設定ファイルを作成する方法を説明します。これ以降のセクションでは、設定ファイルでのこれらの部分を設定する方法を説明しています。

重要

この手順は、クラスター設定ファイルを作成するための暫定的なステップにすぎません。得られるファイルにはフェンシングがなく、サポート対象の設定とは見なされません。

次の手順は、スケルトンクラスター設定ファイルを作成して設定する方法を説明しています。最終的には、使用しているクラスターの設定ファイルは、ノード数、フェンシングのタイプ、及び HA サービスのタイプと数、サイト特有の他の要件によって変化します。

1. クラスター内のいずれかのノードで、例7.1「**cluster.conf** の例： 基本設定」 内のサンプルのテンプレートを使用して **/etc/cluster/cluster.conf** を作成します。
2. **(オプション)** 2 ノードクラスターを設定している場合は、設定ファイルに以下の行を追加して単一ノードでも定足数を維持できるようにします (例えば、1 つのノードが失敗した場合など)：

   ```
   <cman two_node="1" expected_votes="1"/>
   ```

   cluster.conf ファイルから **two_node** オプションを追加/削除した時に、設定を更新した場合は、変更を反映させるためにクラスターを再起動する必要があります。クラスターの設定を更新する詳細については、「設定の更新」 を参照してください。**two_node** オプションを指定する例は、例7.2「**cluster.conf** の例： 基本的な 2 ノード設定」 を参照してください。

3. **cluster** 属性を使用してクラスター名と設定のバージョン番号である **name** と **config_version** を指定します。(例7.1「**cluster.conf** の例： 基本設定」 又は 例7.2「**cluster.conf** の例： 基本的な 2 ノード設定」 を参照)。

4. **clusternodes** セクションで、**clusternode** 属性を使用して各ノードのノード名とノード ID である **name** と **nodeid** を指定します。ノード名の長さは、最大 255 バイトまでになります。

5. **/etc/cluster/cluster.conf** を保存します。

6. **ccs_config_validate** コマンドを実行することで、クラスタースキーマ (**cluster.rng**) に対するファイルの妥当性を検証します。例えば、

```
[root@example-01 ~]# ccs_config_validate
Configuration validates
```

7. 設定ファイルを各クラスターノード内の **/etc/cluster/** に伝播します。例えば、**scp** コマンドを使用するとファイルを他のクラスターノードへ伝播できます。

注記

クラスターの初回作成時には、この方法でクラスター設定ファイルを伝達する必要があります。クラスターがインストールされて稼働すると、クラスター設定ファイルは **cman_tool version -r** コマンドを使用して伝達できるようになります。更新した設定ファイルの伝達には **scp** コマンドを使用することもできますが、**scp** コマンドの使用中は、全てのノードでクラスターソフトウェアが停止する必要があります。さらに、**scp** で更新済みの設定ファイルを伝達する場合は、 **ccs_config_validate** を実行することをお勧めします。

注記

サンプルの設定ファイル内に他の要素と属性がありますが (例えば、**fence** と **fencedevices**)、それらは今すぐ設定する必要はありません。本章の後半の手順では、他の要素と属性の指定に関する情報が提供されています。

8. クラスターを開始します。各クラスターノードで以下のコマンドを実行します：

   service cman start

   例えば：

```
[root@example-01 ~]# service cman start
Starting cluster:
   Checking Network Manager...                             [  OK
]
   Global setup...                                         [  OK
]
   Loading kernel modules...                               [  OK
]
   Mounting configfs...                                    [  OK
]
   Starting cman...                                        [  OK
]
   Waiting for quorum...                                   [  OK
```

```
]
   Starting fenced...                                         [  OK
]
   Starting dlm_controld...                                   [  OK
]
   Starting gfs_controld...                                   [  OK
]
   Unfencing self...                                          [  OK
]
   Joining fence domain...                                    [  OK
]
```

9. いずれかのクラスターノードで **cman_tool nodes** を実行して、ノード群がクラスター内でメンバーとして機能していることを確認します (ステータスカラム "Sts" で "M" として表示)。例えば：

```
[root@example-01 ~]# cman_tool nodes
Node  Sts   Inc   Joined               Name
   1   M    548   2010-09-28 10:52:21  node-01.example.com
   2   M    548   2010-09-28 10:52:21  node-02.example.com
   3   M    544   2010-09-28 10:52:21  node-03.example.com
```

10. クラスターが稼働していれば、「フェンシングの設定」 に進みます。

基本設定の例

例7.1「**cluster.conf** の例： 基本設定」 と 例7.2「**cluster.conf** の例： 基本的な 2 ノード設定」 (2 ノードクラスター用) はそれぞれ、開始点として非常に基本的なクラスター設定ファイルのサンプルを提供します。本章の後半にある手順には、フェンシングと HA サービスに関する情報が記載されています。

例7.1 cluster.conf の例： 基本設定

```
<cluster name="mycluster" config_version="2">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
   </fencedevices>
   <rm>
   </rm>
</cluster>
```

例7.2 `cluster.conf` の例： 基本的な 2 ノード設定

```
<cluster name="mycluster" config_version="2">
   <cman two_node="1" expected_votes="1"/>
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
   </fencedevices>
   <rm>
   </rm>
</cluster>
```

2 ノードクラスター内の `totem` の `consensus` 値

2 ノードクラスターを作成して、後でそのクラスターにノードを追加する意図がない場合は、**`cluster.conf`** ファイル内にある **`totem`** タグの **`consensus`** 値を省略して、**`consensus`** 値が自動的に算出されるようにします。**`consensus`** 値が自動的に算出されると、以下のルールが適用されます:

- 2 つ、又はそれ以下のノードしかない場合、**`consensus`** 値は 上限 2000 msec で 下限 200 msec を持つ (token * 0.2) となります。
- 3 つ、又はそれ以上のノードがある場合、**`consensus`** 値は (token + 2000 msec) となります。

このようにして **`cman`** ユーティリティに consensus タイムアウトを設定させると、後で 2 ノードから 3 ノード (又はそれ以上) に移動するにはクラスターの再起動が必要になります。これは token タイムアウトを基にしてconsensus タイムアウトをより大きな値に変更する必要があるためです。

2 ノードクラスターを設定中であり、今後それ以上のノードにアップグレードする予定がある場合は、consensus タイムアウトを上書きできるため 2 つから 3 つ (又はそれ以上) のノードに移動する時にクラスターの再起動が不要になります。これは、**`cluster.conf`** 内で以下のように実行できます:

```
<totem token="X" consensus="X + 2000" />
```

設定パーサーは X + 2000 を自動的に算出しないことに注意して下さい。数式ではなく整数値を使用する必要があります。

2 ノードクラスターに対して最適化した consensus タイムアウトを使用する利点は、全体のフェイルオーバータイムが 2 ノードのケースでは短縮できる点です。これは consensus が token タイムアウトの関数ではないからです。

cman 内の 2 ノード検出機能の場合、重要な点は物理ノードの数であり、 **cluster.conf** ファイル内の **two_node=1** 指示文の存在ではないことに注意して下さい。

7.3. フェンシングの設定

フェンシングを設定するには、(a) クラスター内に 1 つ以上のデバイスを指定、(b) 各ノードに 1 つ以上のフェンスメソッドを指定 (任意のフェンスデバイス又は指定されたフェンスデバイスを使用) します。

注記

各ノードに複数のフェンシングメカニズムを設定することが推奨されます。フェンシングデバイスが失敗する要因には、ネットワークの分割、電源異常、フェンシングデバイスそのものに問題がある場合、などがあります。複数のフェンシングメカニズムを設定することで、1 つのフェンシングデバイスの障害が致命的な結果になる可能性を低減することができます。

使用する設定に必要となるフェンスデバイスとフェンスメソッドのタイプを基にして、以下のように **cluster.conf** を設定します。

1. **fencedevices** セクションでは、**fencedevice** 要素とフェンスデバイス従属属性を使用して各フェンスデバイスを指定します。例7.3「**cluster.conf** に追加された APC フェンスデバイス」 は APC フェンスデバイスを追加した設定ファイルの例を示しています。

2. **clusternodes** セクションでは、各 **clusternode** セクションの **fence** 要素内でノードの各フェンスメソッドを指定します。**method** 属性である、**name** を使用してフェンスメソッドの名前を指定します。**device** 要素とその属性である、**name** とフェンスデバイス特有のパラメーターを使用して各フェンスメソッド用のフェンスデバイスを指定します。例 7.4「**cluster.conf** に追加されたフェンスメソッド」 はクラスター内の各ノード用に 1 つのフェンスデバイスを持つフェンスメソッドの例を示しています。

3. パワーフェンス以外のメソッド (つまり、SAN/ストレージフェンシング) 用には、**clusternodes** セクションで **unfence** セクションを追加します。これにより、フェンス済みのノードは再起動されるまでは再度有効になりません。アンフェンシングを必要とするデバイスを設定する際には、最初にクラスターを停止し、デバイスおよびアンフェンシングを含むすべての設定をクラスターが開始される前に追加する必要があります。ノードのアンフェンシングに関する詳細は、**fence_node**(8) の man ページを参照してください。

   unfence セクションは、**fence** セクションとは異なり、**method** セクションが含まれていませんが、**device** 参照を直接含んでいます。これは **fence** の該当するデバイスセクションをミラーし、"on" 又は "enable" の明示的なアクション (**action**) がはっきりと追加されています。**fence** と **unfence device** の行によって同じ **fencedevice** が参照され、同じノード毎の引数が繰り返されます。

   action 属性を "on" 又は "enable" に指定すると、再起動時にノードを有効にします。例 7.4「**cluster.conf** に追加されたフェンスメソッド」 と 例7.5「**cluster.conf**： ノード毎に複数のフェンスメソッド」 には **unfence** の要素と属性の例が含まれています。

   unfence についての詳細は、**fence_node** の man ページを参照してください。

4. 値を増加することにより **config_version** 属性を更新します (例えば、**config_version="2"** から **config_version="3">** へ変更)。

5. **/etc/cluster/cluster.conf** を保存します。

6. **(オプション) ccs_config_validate** コマンドを実行することでクラススキーマ (**cluster.rng**) に対して更新されたファイルを検証します。例えば：

```
[root@example-01 ~]# ccs_config_validate
Configuration validates
```

7. **cman_tool version -r** コマンドを実行してクラスターノードの他の部分に設定を伝播します。このコマンドは追加の検証も実行します。更新されたクラスターの設定情報を伝播できるようにするには、各ノード内で **ricci** が稼働している必要があります。

8. 更新した設定ファイルが伝播されたことを確認します。

9. 「フェイルオーバードメインの設定」 へ進んでください。

必要であれば、ノード毎に複数のフェンスメソッドとフェンスメソッド毎に複数のフェンスデバイスを持つ複雑な構成を設定することができます。ノード毎に複数のフェンスメソッドを指定する時、最初のメソッドを使用してフェンシングが失敗した場合は、フェンスデーモンである **fenced** は次のメソッドを試行し、1 つが成功するまでメソッド群を繰り返します。

時には、ノードのフェンシングには 2 つの I/O パス、又は 2 つのパワーポートを無効にする必要があります。これを行うにはフェンスメソッド内の 2 つ以上のデバイスを指定します。**fenced** は各フェンス/デバイス行に対してフェンスエージェントを 1 回実行しますが、フェンシングが成功と見なされるにはすべてが成功しなければなりません。

より複雑な設定は 「フェンシング設定の例」 に記載されています。

フェンスデバイスの設定に関する詳細情報は、フェンスデバイスエージェントの man ページ (例えば、**fence_apc** の man ページ) でご覧になれます。さらには、付録A *フェンスデバイスパラメーター* ではフェンシングパラメーターの情報、**/usr/sbin/** ではフェンスエージェントの情報、**/usr/share/cluster/cluster.rng** ではクラススキーマの情報、**/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** (例えば、**/usr/share/doc/cman-3.0.12/cluster_conf.html**) では注釈付きスキーマの情報を取得できます。

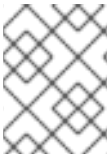
フェンシング設定の例

以下の例では、ノード毎に 1 つのフェンスメソッド、フェンスメソッド毎に 1 つのフェンスデバイスがある単純な設定を示しています。

- 例7.3「**cluster.conf** に追加された APC フェンスデバイス」
- 例7.4「**cluster.conf** に追加されたフェンスメソッド」

以下は、より複雑な設定を示した例です：

- 例7.5「**cluster.conf**： ノード毎に複数のフェンスメソッド」
- 例7.6「**cluster.conf**： マルチパス複数ポートのフェンシング」
- 例7.7「**cluster.conf**： デュアル電源供給を持つノードのフェンシング」



注記

このセクション内の例ではすべてが網羅されているわけではありません。つまり、要件によっては他のフェンシング設定方法がある場合があります。

例7.3 cluster.conf に追加された APC フェンスデバイス

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
         <fencedevice agent="fence_apc" ipaddr="apc_ip_example"
login="login_example" name="apc" passwd="password_example"/>
   </fencedevices>
   <rm>
   </rm>
</cluster>
```

この例では、フェンスデバイス (**fencedevice**) は **fencedevices** 要素に追加されています。また、フェンスエージェント (**agent**) を **fence_apc** として、IP アドレス (**ipaddr**) を **apc_ip_example** として、ログイン (**login**) を **login_example** として、フェンスデバイス名 (**name**) を **apc** として、パスワード (**passwd**) を **password_example** として指定しています。

例7.4 cluster.conf に追加されたフェンスメソッド

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC">
              <device name="apc" port="1"/>
             </method>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="APC">
              <device name="apc" port="2"/>
```

```
            </method>
          </fence>
      </clusternode>
      <clusternode name="node-03.example.com" nodeid="3">
          <fence>
             <method name="APC">
               <device name="apc" port="3"/>
             </method>
          </fence>
      </clusternode>
    </clusternodes>
    <fencedevices>
          <fencedevice agent="fence_apc" ipaddr="apc_ip_example"
login="login_example" name="apc" passwd="password_example"/>
    </fencedevices>
    <rm>
    </rm>
</cluster>
```

この例では、フェンスメソッド (**method**) が各ノードに追加されています。各ノードのフェンスメソッド名 (**name**) は **APC** です。各ノードのフェンスメソッドのデバイス (**device**) は、名前 (**name**) を **apc** 及び各ノード用に一意の APC スイッチパワーポート番号 (**port**) として指定します。ここでは例として、node-01.example.com のポート番号を **1** (**port="1"**) とします。各ノードのデバイス名 (**device name="apc"**) は、**fencedevices** 要素のこの行にある**apc** の名前 **apc** によってフェンスデバイスをポイントします: **fencedevice agent="fence_apc" ipaddr="apc_ip_example" login="login_example" name="apc" passwd="password_example"**

例7.5 cluster.conf： ノード毎に複数のフェンスメソッド

```
<cluster name="mycluster" config_version="3">
    <clusternodes>
      <clusternode name="node-01.example.com" nodeid="1">
          <fence>
             <method name="APC">
               <device name="apc" port="1"/>
              </method>
             <method name="SAN">
        <device name="sanswitch1" port="11"/>
              </method>
          </fence>
          <unfence>
              <device name="sanswitch1" port="11" action="on"/>
          </unfence>
      </clusternode>
      <clusternode name="node-02.example.com" nodeid="2">
          <fence>
             <method name="APC">
               <device name="apc" port="2"/>
             </method>
             <method name="SAN">
        <device name="sanswitch1" port="12"/>
```

```
            </method>
        </fence>
        <unfence>
            <device name="sanswitch1" port="12" action="on"/>
        </unfence>
    </clusternode>
    <clusternode name="node-03.example.com" nodeid="3">
        <fence>
           <method name="APC">
             <device name="apc" port="3"/>
           </method>
           <method name="SAN">
      <device name="sanswitch1" port="13"/>
            </method>
        </fence>
        <unfence>
            <device name="sanswitch1" port="13" action="on"/>
        </unfence>
    </clusternode>
  </clusternodes>
  <fencedevices>
       <fencedevice agent="fence_apc" ipaddr="apc_ip_example"
login="login_example" name="apc" passwd="password_example"/>
       <fencedevice agent="fence_sanbox2" ipaddr="san_ip_example"
login="login_example" name="sanswitch1" passwd="password_example"/>
  </fencedevices>
  <rm>
  </rm>
</cluster>
```

例7.6 `cluster.conf`： マルチパス複数ポートのフェンシング

```
<cluster name="mycluster" config_version="3">
  <clusternodes>
    <clusternode name="node-01.example.com" nodeid="1">
        <fence>
           <method name="SAN-multi">
      <device name="sanswitch1" port="11"/>
      <device name="sanswitch2" port="11"/>
    </method>
        </fence>
        <unfence>
            <device name="sanswitch1" port="11" action="on"/>
            <device name="sanswitch2" port="11" action="on"/>
        </unfence>
    </clusternode>
    <clusternode name="node-02.example.com" nodeid="2">
        <fence>
           <method name="SAN-multi">
      <device name="sanswitch1" port="12"/>
      <device name="sanswitch2" port="12"/>
```

```
            </method>
          </fence>
          <unfence>
              <device name="sanswitch1" port="12" action="on"/>
              <device name="sanswitch2" port="12" action="on"/>
          </unfence>
      </clusternode>
      <clusternode name="node-03.example.com" nodeid="3">
          <fence>
             <method name="SAN-multi">
        <device name="sanswitch1" port="13"/>
        <device name="sanswitch2" port="13"/>
             </method>
          </fence>
          <unfence>
              <device name="sanswitch1" port="13" action="on"/>
              <device name="sanswitch2" port="13" action="on"/>
          </unfence>
      </clusternode>
    </clusternodes>
    <fencedevices>
         <fencedevice agent="fence_sanbox2" ipaddr="san_ip_example"
login="login_example" name="sanswitch1" passwd="password_example"/>
         <fencedevice agent="fence_sanbox2" ipaddr="san_ip_example"
login="login_example" name="sanswitch2" passwd="password_example"/>
    </fencedevices>
    <rm>
    </rm>
</cluster>
```

例7.7 `cluster.conf`： デュアル電源供給を持つノードのフェンシング

```
<cluster name="mycluster" config_version="3">
    <clusternodes>
      <clusternode name="node-01.example.com" nodeid="1">
          <fence>
             <method name="APC-dual">
               <device name="apc1" port="1"action="off"/>
               <device name="apc2" port="1"action="off"/>
               <device name="apc1" port="1"action="on"/>
               <device name="apc2" port="1"action="on"/>
              </method>
          </fence>
      </clusternode>
      <clusternode name="node-02.example.com" nodeid="2">
          <fence>
             <method name="APC-dual">
               <device name="apc1" port="2"action="off"/>
               <device name="apc2" port="2"action="off"/>
               <device name="apc1" port="2"action="on"/>
               <device name="apc2" port="2"action="on"/>
```

```
            </method>
          </fence>
      </clusternode>
      <clusternode name="node-03.example.com" nodeid="3">
          <fence>
             <method name="APC-dual">
               <device name="apc1" port="3"action="off"/>
               <device name="apc2" port="3"action="off"/>
               <device name="apc1" port="3"action="on"/>
               <device name="apc2" port="3"action="on"/>
             </method>
          </fence>
      </clusternode>
    </clusternodes>
    <fencedevices>
        <fencedevice agent="fence_apc" ipaddr="apc_ip_example"
login="login_example" name="apc1" passwd="password_example"/>
        <fencedevice agent="fence_apc" ipaddr="apc_ip_example"
login="login_example" name="apc2" passwd="password_example"/>
    </fencedevices>
    <rm>
    </rm>
</cluster>
```

パワースイッチを使用してデュアル電源装置があるノードをフェンスする時、どちらかのポートに電力を回復する前に両方の電源ポートをオフにするようにエージェントに伝える必要があります。デフォルトによるエージェントのオフ/オン動作により、電力がノードに対して完全に無効にならない場合もあります。

7.4. フェイルオーバードメインの設定

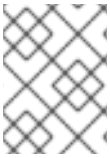
フェイルオーバードメインは、ノードに障害が発生した場合にクラスターサービスを実行するのに有効なクラスターノードの名前付きサブセットです。フェイルオーバードメインは、以下の特性を持つことができます：

- Unrestricted（制限なし） — 優先するメンバーのサブセットを指定できるようにしますが、このドメインに割り当てられたクラスターサービスはいずれの利用可能なメンバーでも実行できます。
- Restricted（制限あり） — 特定のクラスターサービスを実行できるメンバーを制限できるようにします。制限されたフェイルオーバードメインのメンバーがどれも使用できない場合は、クラスターサービスは (手動あるいはクラスターソフトウェアによっても) 開始できません。
- Unordered（優先順なし） — クラスターサービスが優先順なしのフェイルオーバードメインへ割り当てられる場合、クラスターサービスを実行するメンバーは、優先順なしの利用可能なフェイルオーバードメインメンバーから選択されます。
- Ordered（優先順あり） — フェイルオーバードメインのメンバー内で優先順を指定できるようにします。優先順フェイルオーバードメインは、優先順位が一番低い番号を持つノードを最初に選択します。すなわち、フェイルオーバードメイン内で優先番号 "1" を持つノードの優先度が最も高くなるため、フェイルオーバードメイン内で最優先すべきノードとなります。そのノードの次に優先されるノードは、次に高い優先番号を持つノードとなります。

- Failback（フェイルバック） — フェイルオーバードメイン内のサービスが、ノード障害の前に元々実行していたノードへフェイルバックするかどうかを指定できるようにします。この特性の設定が役立つのは、ノードが繰り返し失敗し、それが優先順ありのフェイルオーバードメインの一部であるような状況です。この状況では、ノードがフェイルオーバードメイン内の優先ノードである場合には、優先ノードと別のノード間でサービスがフェイルオーバーとフェイルバックを繰り返す可能性があるため、パフォーマンスに重大な影響を与えることになります。

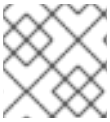
注記

優先順のあるフェイルオーバーが設定されている場合にのみ、フェイルバックの特性が利用できます。



注記

フェイルオーバードメイン設定を変更しても、現在実行中のサービスには影響しません。



注記

フェイルオーバードメインは、運用には必須 **ではありません。**

デフォルトでは、フェイルオーバードメインは制限なしで優先順がありません。

複数のメンバーを持つクラスター内では、制限ありのフェイルオーバードメインを使用することで、クラスターサービス (**httpd** など) を実行するためのクラスターを設定する作業を最小限にできます。このためには、クラスターサービスを実行する全てのメンバー上で、設定を同一にする必要があります。クラスターサービスを実行するようクラスター全体を設定する代わりに、クラスターサービスに関連付ける制限ありのフェイルオーバードメイン内のメンバーのみを設定するだけで済みます。

注記

優先するメンバーを設定するには、1 つだけのクラスターメンバーから成る制限なしのフェイルオーバードメインを作成します。そうすることで、クラスターサービスが主にそのクラスターメンバー (優先メンバー) 上で実行することになりますが、クラスターサービスは他のどのメンバーにでもフェイルオーバーできるようになります。

フェイルオーバードメインを設定するには、以下の手順を使用します：

1. クラスター内のいずれかのノードで **/etc/cluster/cluster.conf** を開きます。
2. 使用する各フェイルオーバードメイン用の **rm** 要素内に以下のスケルトンセクションを追加します：

```
        <failoverdomains>
            <failoverdomain name="" nofailback="" ordered=""
restricted="">
                <failoverdomainnode name="" priority=""/>
                <failoverdomainnode name="" priority=""/>
                <failoverdomainnode name="" priority=""/>
            </failoverdomain>
        </failoverdomains>
```

-

注記

failoverdomainnode 属性の数はフェイルオーバードメイン内のノード数で変化します。前述したテキスト内のスケルトン **failoverdomain** セクションは 3 つの **failoverdomainnode** 要素を示しており、フェイルオーバードメインに 3 つのノードが存在することを示唆しています。

3. **failoverdomain** セクションでは、要素と属性の値を提供します。要素と属性の説明については、注釈付きクラススキーマの 『failoverdomain』 セクションを参照してください。注釈付きクラススキーマはいずれかのクラスターノード内の **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** (例えば、**/usr/share/doc/cman-3.0.12/cluster_conf.html**) にあります。**failoverdomains** セクションの例は、例7.8「**cluster.conf** に追加されたフェイルオーバードメイン」 を参照してください。

4. 値を増加することにより **config_version** 属性を更新します (例えば、**config_version="2"** から **config_version="3">** へ変更)。

5. **/etc/cluster/cluster.conf** を保存します。

6. **(オプション) ccs_config_validate** コマンドを実行して、クラススキーマ (**cluster.rng**) に対するファイルの妥当性を検証します。たとえば、

```
[root@example-01 ~]# ccs_config_validate
Configuration validates
```

7. **cman_tool version -r** コマンドを実行して、設定をクラスターノードの残り部分へ伝播します。

8. 「HA サービスの設定」 へ進みます。

例7.8「**cluster.conf** に追加されたフェイルオーバードメイン」 は優先順のある制限なしのフェイルオーバードメインを持つ設定の例を示しています。

例7.8 cluster.conf に追加されたフェイルオーバードメイン

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC">
              <device name="apc" port="1"/>
             </method>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="APC">
              <device name="apc" port="2"/>
            </method>
         </fence>
     </clusternode>
```

```
      <clusternode name="node-03.example.com" nodeid="3">
          <fence>
             <method name="APC">
               <device name="apc" port="3"/>
             </method>
          </fence>
      </clusternode>
   </clusternodes>
   <fencedevices>
         <fencedevice agent="fence_apc" ipaddr="apc_ip_example"
login="login_example" name="apc" passwd="password_example"/>
   </fencedevices>
   <rm>
       <failoverdomains>
           <failoverdomain name="example_pri" nofailback="0"
ordered="1" restricted="0">
               <failoverdomainnode name="node-01.example.com"
priority="1"/>
               <failoverdomainnode name="node-02.example.com"
priority="2"/>
               <failoverdomainnode name="node-03.example.com"
priority="3"/>
           </failoverdomain>
       </failoverdomains>
   </rm>
</cluster>
```

failoverdomains セクションには、クラスター内の各フェイルオーバードメイン用の **failoverdomain** セクションが含まれています。この例には、フェイルオーバードメインが 1 つあります。**failoverdomain** の行では、名前 (**name**) が **example_pri** として指定されています。さらに、フェイルバックなし (**failback="0"**)、フェイルオーバーは優先順あり (**ordered="1"**)、フェイルオーバードメインは制限なし (**restricted="0"**) と指定されています。

7.5. HA サービスの設定

HA (High Availability) サービスを設定するには、リソースを設定して、それらをサービスに割り当てます。

以下のセクションでは、リソースとサービスを追加するための **/etc/cluster/cluster.conf** を編集する方法を説明しています。

- 「クラスターリソースの追加」
- 「クラスターへのクラスターサービスの追加」

重要

High Availability のリソースとサービスを使用すると、設定が広範囲で可能になります。リソースパラメーターとリソース動作の理解を深めるためには、付録B *HA リソースパラメーター* 及び 付録C *HA リソースの動作* を参照してください。使用している設定がサポート可能であることを確認し、最適なパフォーマンスを実現するには、Red Hat 認定サポート担当者に連絡してください。

7.5.1. クラスターリソースの追加

2 種類のリソースを設定できます:

- Global（グローバル） — クラスター内のどのサービスにも利用可能なリソース。これらは、設定ファイル (**rm** 要素内) の **resources** セクションで設定されています。
- Service-specific（サービス特有） — 1 つのサービスのみに利用可能なリソース。これらは、設定ファイル (**rm** 要素内) の各 **service** セクションで設定されています。

このセクションでは、グローバルリソースの追加方法を説明しています。サービス特有のリソースを設定する手順については、「クラスターへのクラスターサービスの追加」 を参照してください。

グローバルクラスターリソースを追加するには、このセクションの手順に従ってください。

1. クラスター内のいずれかのノードで **/etc/cluster/cluster.conf** を開きます。
2. **rm** 要素内に **resources** セクションを追加します。例えば：

```
    <rm>
        <resources>

        </resources>
    </rm>
```

3. これに作成したいサービスに応じてリソースを入力します。例えば、Apache サービス内で使用したいリソースがあると想定します。それらがファイルシステム (**fs**) リソース、IP (**ip**) リソース、及び Apache (**apache**) リソースで構成されているとします。

```
    <rm>
        <resources>
           <fs name="web_fs" device="/dev/sdd2"
mountpoint="/var/www" fstype="ext3"/>
           <ip address="127.143.131.100" monitor_link="yes"
sleeptime="10"/>
           <apache config_file="conf/httpd.conf"
name="example_server" server_root="/etc/httpd" shutdown_wait="0"/>
        </resources>
    </rm>
```

   例7.9「リソースが追加された **cluster.conf** ファイル」 では、**resources** セクションを追加した **cluster.conf** ファイルのサンプルを示しています。

4. 値を増加することにより **config_version** 属性を更新します (例えば、**config_version="2"** から **config_version="3"** へ変更)。
5. **/etc/cluster/cluster.conf** を保存します。
6. **(オプション) ccs_config_validate** コマンドを実行して、クラススキーマ (**cluster.rng**) に対するファイルの妥当性を検証します。たとえば、

```
[root@example-01 ~]# ccs_config_validate
Configuration validates
```

7. **cman_tool version -r** コマンドを実行して、設定をクラスターノードの残り部分へ伝播します。
8. 更新した設定ファイルが伝播されたことを確認します。
9. 「クラスターへのクラスターサービスの追加」 へ進みます。

例7.9 リソースが追加された cluster.conf ファイル

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC">
              <device name="apc" port="1"/>
             </method>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="APC">
              <device name="apc" port="2"/>
            </method>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
            <method name="APC">
              <device name="apc" port="3"/>
            </method>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
         <fencedevice agent="fence_apc" ipaddr="apc_ip_example"
login="login_example" name="apc" passwd="password_example"/>
   </fencedevices>
   <rm>
       <failoverdomains>
           <failoverdomain name="example_pri" nofailback="0"
ordered="1" restricted="0">
               <failoverdomainnode name="node-01.example.com"
priority="1"/>
               <failoverdomainnode name="node-02.example.com"
priority="2"/>
               <failoverdomainnode name="node-03.example.com"
priority="3"/>
           </failoverdomain>
       </failoverdomains>
       <resources>
           <fs name="web_fs" device="/dev/sdd2" mountpoint="/var/www"
```

```
fstype="ext3"/>
          <ip address="127.143.131.100" monitor_link="yes"
sleeptime="10"/>
          <apache config_file="conf/httpd.conf" name="example_server"
server_root="/etc/httpd" shutdown_wait="0"/>
        </resources>

   </rm>
</cluster>
```

7.5.2. クラスターへのクラスターサービスの追加

クラスターにクラスターサービスを追加するには、このセクション内の手順に従ってください。

注記

このセクションの例では、すべてのリソースが同一レベルにあるクラスターサービスを示しています。リソース階層に依存関係があるサービスの定義に関する情報および親と子のリソース動作を管理するルールについては、付録C *HA リソースの動作* を参照してください。

1. クラスター内のいずれかのノードで **/etc/cluster/cluster.conf** を開きます。
2. 各サービス用の **rm** 要素内に **service** セクションを追加します。例えば：

```
    <rm>
        <service autostart="1" domain="" exclusive="0" name=""
recovery="restart">

        </service>
    </rm>
```

3. **service** 要素内で以下のパラメーター (属性) を設定します：
   - **autostart** — クラスターの開始時にサービスを自動起動するかどうかを指定します。有効にするには「1」を、無効にするには「0」を使用します。デフォルトは有効です。
   - **domain** — フェイルオーバードメイン（必要である場合）を指定します。
   - **exclusive** — 他のサービスが稼働していないノード上でのみサービスを稼働するポリシーを指定します。
   - **recovery** — サービスの回復ポリシーを指定します。オプションとしては、サービスの再配置、再起動、無効、再起動後に無効、があります。
4. 使用したいリソースのタイプに応じて、サービスにグローバル又はサービス特有のリソースを入力します。

   例えば、グローバルリソースを使用する Apache サービスは以下のようになります：

   ■

```
    <rm>
        <resources>
                <fs name="web_fs" device="/dev/sdd2"
mountpoint="/var/www" fstype="ext3"/>
                <ip address="127.143.131.100" monitor_link="yes"
sleeptime="10"/>
                <apache config_file="conf/httpd.conf"
name="example_server" server_root="/etc/httpd" shutdown_wait="0"/>
        </resources>
        <service autostart="1" domain="example_pri" exclusive="0"
name="example_apache" recovery="relocate">
                <fs ref="web_fs"/>
                <ip ref="127.143.131.100"/>
                <apache ref="example_server"/>
        </service>
    </rm>
```

例えば、サービス特有のリソースを使用する Apache サービスは以下のようになります：

```
    <rm>
        <service autostart="0" domain="example_pri" exclusive="0"
name="example_apache2" recovery="relocate">
                <fs name="web_fs2" device="/dev/sdd3"
mountpoint="/var/www2" fstype="ext3"/>
                <ip address="127.143.131.101" monitor_link="yes"
sleeptime="10"/>
                <apache config_file="conf/httpd.conf"
name="example_server2" server_root="/etc/httpd" shutdown_wait="0"/>
        </service>
    </rm>
```

例7.10「サービスが追加された **cluster.conf**：1 つはグローバルリソースを使用、もう 1 つはサービス特有のリソースを使用」 では、2 つのサービスを持つ **cluster.conf** ファイルの例を示してします。

- **example_apache** — このサービスはグローバルリソースである **web_fs**、**127.143.131.100** 及び **example_server** を使用します。
- **example_apache2** — このサービスはサービス特有のリソースである **web_fs2**、**127.143.131.101**、及び **example_server2** を使用します。

5. 値を増加することにより **config_version** 属性を更新します (例えば、**config_version="2"** から **config_version="3">** へ変更)。
6. **/etc/cluster/cluster.conf** を保存します。
7. **(オプション) ccs_config_validate** コマンドを実行することでクラスタースキーマ (**cluster.rng**) に対して更新されたファイルを検証します。例えば：

```
[root@example-01 ~]# ccs_config_validate
Configuration validates
```

8. **cman_tool version -r** コマンドを実行して、設定をクラスターノードの残り部分へ伝播します。
9. 更新した設定ファイルが伝播されたことを確認します。
10. 「設定の確認」 へ進みます。

例7.10 サービスが追加された cluster.conf：1 つはグローバルリソースを使用、もう 1 つはサービス特有のリソースを使用

```
<cluster name="mycluster" config_version="3">
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC">
              <device name="apc" port="1"/>
             </method>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="APC">
              <device name="apc" port="2"/>
            </method>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
            <method name="APC">
              <device name="apc" port="3"/>
            </method>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
         <fencedevice agent="fence_apc" ipaddr="apc_ip_example"
login="login_example" name="apc" passwd="password_example"/>
   </fencedevices>
   <rm>
       <failoverdomains>
           <failoverdomain name="example_pri" nofailback="0"
ordered="1" restricted="0">
               <failoverdomainnode name="node-01.example.com"
priority="1"/>
               <failoverdomainnode name="node-02.example.com"
priority="2"/>
               <failoverdomainnode name="node-03.example.com"
priority="3"/>
           </failoverdomain>
       </failoverdomains>
       <resources>
           <fs name="web_fs" device="/dev/sdd2" mountpoint="/var/www"
fstype="ext3"/>
           <ip address="127.143.131.100" monitor_link="yes"
sleeptime="10"/>
```

```
            <apache config_file="conf/httpd.conf" name="example_server"
server_root="/etc/httpd" shutdown_wait="0"/>
        </resources>
        <service autostart="1" domain="example_pri" exclusive="0"
name="example_apache" recovery="relocate">
            <fs ref="web_fs"/>
            <ip ref="127.143.131.100"/>
            <apache ref="example_server"/>
        </service>
        <service autostart="0" domain="example_pri" exclusive="0"
name="example_apache2" recovery="relocate">
            <fs name="web_fs2" device="/dev/sdd3" mountpoint="/var/www2"
fstype="ext3"/>
            <ip address="127.143.131.101" monitor_link="yes"
sleeptime="10"/>
            <apache config_file="conf/httpd.conf" name="example_server2"
server_root="/etc/httpd" shutdown_wait="0"/>
        </service>
    </rm>
</cluster>
```

7.6. 冗長リングプロトコルの設定

Red Hat Enterprise Linux 6.4 では、Red Hat High Availability アドオンはリングプロトコルの冗長設定に対応しています。

冗長リングプロトコルを使用するようにシステムを設定する場合、以下の事項を考慮する必要があります。

- リングを 3 つ以上指定しないでください。
- 各リングで同じプロトコルを使用すること、IPv4 と IPv6 を混同しないようにしてください。
- 必要であれば、2 つ目のリングにマルチキャストアドレスを手動で指定可能です。2 つ目のリングにマルチキャストアドレスを指定した場合、代わりのマルチキャストアドレスまたは代わりのポートは 1 つ目のリングのマルチキャストアドレスと異なる必要があります。別のマルチキャストアドレスを指定しなかった場合、システムは 2 つ目のリングに別のマルチキャストアドレスを自動的に使用するようになっています。

  別のポートを指定した場合、操作の実行にポートとポート 1 をシステム自体で使用するため、1 つ目と 2 つ目のポート番号は 2 つ以上違うものでなければなりません。
- 同一サブネット上で 2 つの異なるインターフェースを使用しないでください。
- 一般的に、NIC またはスイッチのいずれかに問題が発生したときのために、2 種の NIC および 2 種のスイッチで冗長リングプロトコルを設定すると良いでしょう。
- **ifdown** コマンドまたは **service network stop** コマンドを使用して、ネットワークの問題のシミュレーションを行わないでください。このコマンドでは、クラスター全体が破壊され、クラスター内の全ノードを再起動しない限り回復しなくなります。
- ケーブルが抜けると **ifdown** が実行されるため、**NetworkManager** を使用しないでください。

- NIC の 1 ノードに問題が発生すると、リング全体に問題があるとマーキングされます。
- 問題の発生したリングを回復するために手動介入は必要ありません。回復には、問題の NIC やスイッチなどのように、問題の原因となる部分を修正するだけで結構です。

冗長リングプロトコルに 2 番目のネットワークインターフェースを指定するには **cluster.conf** ファイルの **clusternode** のセクションに **altname** コンポーネントを追加します。**altname** を指定するには、**name** 属性を指定してノードの 2 つ目のホスト名または IP アドレスを指定します。

以下の例では、クラスターノード **clusternet-node1-eth1** の代わりの名前として **clusternet-node1-eth2** を指定しています。

```
<cluster name="mycluster" config_version="3" >
  <logging debug="on"/>
  <clusternodes>
    <clusternode name="clusternet-node1-eth1" votes="1" nodeid="1">
      <fence>
        <method name="single">
          <device name="xvm" domain="clusternet-node1"/>
        </method>
      </fence>
      <altname name="clusternet-node1-eth2"/>
    </clusternode>
```

clusternode ブロックの **altname** セクションは、位置独立コードであるため、**fence** セクションの前にも後ろにも置くことが可能です。クラスターノードに複数の **altname** コンポーネントを指定しないでください。複数指定すると、システムが起動しなくなります。

オプションで、**cluster.conf** 設定ファイルの **cman** セクションに **altmulticast** コンポーネントを含めることで、2 つ目のマルチキャストアドレス、ポート、TTL を手動で指定することができます。**altmulticast** コンポーネントでは、**addr**、**port**、**ttl** パラメーターを使用できます。

以下は、クラスターの設定ファイルの **cman** セクションで、2 つ目のリンクのマルチキャストアドレス、ポート、TTL を設定している例です。

```
<cman>
   <multicast addr="239.192.99.73" port="666" ttl="2"/>
   <altmulticast addr="239.192.99.88" port="888" ttl="3"/>
</cman>
```

7.7. デバッグオプションの設定

クラスターの全デーモンにデバッグを有効にできます。あるいは、個々のクラスタープロセスに対してロギングを有効にすることも可能です。

全デーモンに対してデバッグを有効にするには、以下を **/etc/cluster/cluster.conf** に追加します。デフォルトでは、**/var/log/cluster/*daemon*.log** ファイルにログ記録されます。

```
<cluster config_version="7" name="rh6cluster">
```

```
  <logging debug="on"/>
   ...
</cluster>
```

個々のクラスタープロセスに対してデバッグを有効にするには、以下の行を **/etc/cluster/cluster.conf** ファイルに追加します。デーモン毎のロギング設定は、グローバル設定を上書きします。

```
<cluster config_version="7" name="rh6cluster">
   ...
  <logging>
       <!-- turning on per-subsystem debug logging -->
       <logging_daemon name="corosync" debug="on" />
       <logging_daemon name="fenced" debug="on" />
       <logging_daemon name="qdiskd" debug="on" />
       <logging_daemon name="rgmanager" debug="on" />
       <logging_daemon name="dlm_controld" debug="on" />
       <logging_daemon name="gfs_controld" debug="on" />
  </logging>
   ...
</cluster>
```

ロギングだけでなくグローバル及びデーモン毎のロギングの両方に設定可能な追加のロギングオプションを有効にできるロギングデーモンの一覧については、**cluster.conf**(5) の man ページを参照してください。

7.8. NFSEXPORT および NFSSERVER リソースの設定

このセクションでは、**nfsexport** または **nfsserver** リソースの設定時に考慮すべき点および問題点を説明します。

nfsexport リソースエージェントは、NFSv2 および NFSv3 クライアントで機能します。**nfsexport** の使用時には、以下のことを実行する必要があります。

- **nfs** および **nfslock** がブート時に有効となっていることを確認。
- 全クラスターノードで **RPCNFSDARGS="-N 4"** を **/etc/sysconfig/nfs** ファイルに追加。
- **nfslock="1"** を **cluster.conf** ファイルの **service** コンポーネントに追加。
- サービスを以下のように構成します。

```
<service nfslock="1" ... >
  <fs name="myfs" ... >
    <nfsexport name="exports">
      <nfsclient ref="client1" />
      <nfsclient ref="client2" />
      ...
    </nfsexport>
  </fs>
  <ip address="10.1.1.2" />
```

```
    ...
  </service>
```

nfsserver リソースエージェントは、NFSv3 および NFSv4 クライアントで機能します。**nfsserver** の使用時には、以下のことを実行する必要があります。

- **nfs** および **nfslock** がブート時に無効となっていることを確認。
- サービスで **nfslock="1"** と設定されていないことを確認。
- サービスを以下のように構成します。

```
<service ... >
  <fs name="myfs" ... >
    <nfsserver name="server">
      <nfsclient ref="client1" />
      <nfsclient ref="client2" />
      ...
    </nfsexport>
  </fs>
  <ip address="10.1.1.2" />
  ...
</service>
```

NFSv3 および NFSv4 との使用で **nfsserver** リソースエージェントを用いるようにシステムを設定する際は、以下の制限を理解しておく必要があります。

- クラスター 1 つあたり 1 つの **nfsserver** リソースのみを設定すること。さらに必要な場合は、問題となっている 2 つのサービスが**決して**同一ホスト上でスタートしないようにするために制限されたフェイルオーバードメインを使用する必要があります。
- 2 つ以上のサービスでグローバルに設定された **nfsserver** リソースを参照しないこと。
- 古いスタイルの NFS サービスを同一クラスター内で新しい **nfsserver** と混在させないこと。旧式の NFS サービスは NFS デーモンの稼働を必要としていましたが、**nfsserver** はサービス開始時にデーモンが停止している必要があります。
- 複数のファイルシステムを使用する場合は、エクスポートに継承を使用することができません。つまり、複数のファイルシステムでのサービスにおける **nfsclient** リソースの再利用は制限されることになります。しかし、ターゲットおよびパス属性の明示的定義は、**nfsclients** の数に制限されずに行うことができます。

7.9. 設定の確認

クラスター設定ファイルを作成した後は、以下の手順を実行して、正しく稼働することを確認します：

1. 各ノードで、クラスターソフトウェアを起動します。このアクションにより、開始時のみにチェックされる追加設定が実行中の設定に含まれていることを確認します。**service cman restart** を実行すると、クラスターソフトウェアを再起動できます。例えば：

```
[root@example-01 ~]# service cman restart
```

```
Stopping cluster:
   Leaving fence domain...                                 [  OK
]
   Stopping gfs_controld...                                [  OK
]
   Stopping dlm_controld...                                [  OK
]
   Stopping fenced...                                      [  OK
]
   Stopping cman...                                        [  OK
]
   Waiting for corosync to shutdown:                       [  OK  ]
   Unloading kernel modules...                             [  OK
]
   Unmounting configfs...                                  [  OK
]
Starting cluster:
   Checking Network Manager...                             [  OK
]
   Global setup...                                         [  OK
]
   Loading kernel modules...                               [  OK
]
   Mounting configfs...                                    [  OK
]
   Starting cman...                                        [  OK
]
   Waiting for quorum...                                   [  OK
]
   Starting fenced...                                      [  OK
]
   Starting dlm_controld...                                [  OK
]
   Starting gfs_controld...                                [  OK
]
   Unfencing self...                                       [  OK
]
   Joining fence domain...                                 [  OK
]
```

2. CLVM がクラスター化ボリュームの作成に使用されている場合は、**service clvmd start** を実行します。例えば：

```
[root@example-01 ~]# service clvmd start
Activating VGs:                                            [  OK
]
```

3. Red Hat GFS2 を使用している場合は、**service gfs2 start** を実行します。例えば：

```
[root@example-01 ~]# service gfs2 start
Mounting GFS2 filesystem (/mnt/gfsA):                      [  OK  ]
Mounting GFS2 filesystem (/mnt/gfsB):                      [  OK  ]
```

4. 高可用性 (HA) サービスを使用している場合は、**service rgmanager start** を実行します。例えば：

```
[root@example-01 ~]# service rgmanager start
Starting Cluster Service Manager:                          [  OK  ]
```

5. いずれかのクラスターノードで **cman_tool nodes** を実行して、ノード群がクラスター内でメンバーとして機能していることを確認します (ステータスカラム "Sts" で "M" として表示)。例えば：

```
[root@example-01 ~]# cman_tool nodes
Node  Sts   Inc   Joined               Name
   1   M    548   2010-09-28 10:52:21  node-01.example.com
   2   M    548   2010-09-28 10:52:21  node-02.example.com
   3   M    544   2010-09-28 10:52:21  node-03.example.com
```

6. いずれかのノードで **clustat** ユーティリティを使用して、HA サービスが期待どおりに稼働していることを確認します。さらに、**clustat** はクラスターノード群のステータスを表示します。例えば：

```
[root@example-01 ~]#clustat
Cluster Status for mycluster @ Wed Nov 17 05:40:00 2010
Member Status: Quorate

 Member Name                             ID   Status
 ------ ----                             ---- ------
 node-03.example.com                         3 Online, rgmanager
 node-02.example.com                         2 Online, rgmanager
 node-01.example.com                         1 Online, Local,
rgmanager

 Service Name                   Owner (Last)
State
 ------- ----                   ----- ------                    --
---
 service:example_apache         node-01.example.com
started
 service:example_apache2        (none)
disabled
```

7. クラスターが期待通りに動作している場合は、設定ファイルの作成は終了です。*8章コマンドラインツールを使用した Red Hat High Availability アドオンの管理* 内に説明してあるコマンドラインツールを使用してクラスターを管理できます。

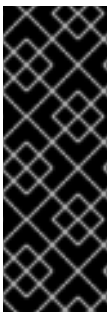
第8章 コマンドラインツールを使用した RED HAT HIGH AVAILABILITY アドオンの管理

本章では、Red Hat High Availability アドオンを管理する上での各種管理タスクについて説明しています。以下のセクションから構成されます：

- 「クラスターソフトウェアの起動と停止」
- 「ノードの削除と追加」
- 「高可用性 (HA) サービスの管理」
- 「設定の更新」

重要

Red Hat High Availability アドオンの導入がご使用のニーズに適合していて、サポート可能であることを確認してください。導入する前に、Red Hat 認定担当者に連絡して、設定を検証してください。さらに、設定のバーンイン期間を設けて障害モードのテストを実施してください。



重要

本章では、一般に使用されている **cluster.conf** の要素と属性を参照します。**cluster.conf** の要素と属性の総括的な一覧と説明は、**/usr/share/cluster/cluster.rng** にあるクラスタースキーマと **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** (例えば、**/usr/share/doc/cman-3.0.12/cluster_conf.html**) にある注釈付きスキーマを参照してください。

重要

本章にある一部の手順は、クラスター設定をクラスター全域に伝播するために **cman_tool -r** コマンドを使用する必要があります。このコマンドを使用するには、**ricci** が稼働していなければなりません。

注記

本章にある手順には 付録E *コマンドラインツールの要約* に一覧表示されているコマンドラインツールの特定のコマンドが含まれている場合があります。全てのコマンドと変数についての詳細は、各コマンドラインツール用の man ページを参照してください。

8.1. クラスターソフトウェアの起動と停止

ノード上のクラスターソフトウェアを起動/停止するには、「クラスターソフトウェアの起動」 と 「クラスターソフトウェアの停止」 に従います。ノード上のクラスターソフトウェアを起動するとそれがクラスターに参加し、ノード上のクラスターソフトウェアを停止するとそれがクラスターから離脱します。

8.1.1. クラスターソフトウェアの起動

ノード上のクラスターソフトウェアを起動するには、以下のコマンドを次の順序で入力します：

1. **service cman start**

2. **service clvmd start**：クラスター化ボリュームの作成に CLVM が使用されている場合

3. **service gfs2 start**：Red Hat GFS2 を使用している場合

4. **service rgmanager start**：HA（高可用性）サービス (**rgmanager**) を使用している場合

例えば：

```
[root@example-01 ~]# service cman start
Starting cluster:
   Checking Network Manager...                             [  OK  ]
   Global setup...                                         [  OK  ]
   Loading kernel modules...                               [  OK  ]
   Mounting configfs...                                    [  OK  ]
   Starting cman...                                        [  OK  ]
   Waiting for quorum...                                   [  OK  ]
   Starting fenced...                                      [  OK  ]
   Starting dlm_controld...                                [  OK  ]
   Starting gfs_controld...                                [  OK  ]
   Unfencing self...                                       [  OK  ]
   Joining fence domain...                                 [  OK  ]
[root@example-01 ~]# service clvmd start
Starting clvmd:                                            [  OK  ]
Activating VG(s):   2 logical volume(s) in volume group "vg_example" now
active
                                                           [  OK  ]
[root@example-01 ~]# service gfs2 start
Mounting GFS2 filesystem (/mnt/gfsA):                      [  OK  ]
Mounting GFS2 filesystem (/mnt/gfsB):                      [  OK  ]
[root@example-01 ~]# service rgmanager start
Starting Cluster Service Manager:                          [  OK  ]
[root@example-01 ~]#
```

8.1.2. クラスターソフトウェアの停止

ノード上のクラスターソフトウェアを停止するには、以下のコマンドを次の順序で入力します：

1. **service rgmanager stop**：HA（高可用性）サービス (**rgmanager**) を使用している場合

2. **service gfs2 stop**：Red Hat GFS2 を使用している場合

3. **umount -at gfs2**：**rgmanager** と Red Hat GFS2 を使用している場合に、**rgmanager** の起動時に、マウントされていた（しかしシャットダウン時にアンマウントされていない）GFS2 ファイルも全てアンマウントされるようにするためです。

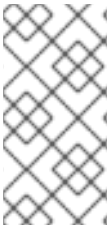
4. **service clvmd stop**：クラスター化ボリュームの作成に CLVM が使用されている場合

5. **service cman stop**

例えば：

```
[root@example-01 ~]# service rgmanager stop
Stopping Cluster Service Manager:                          [  OK  ]
```

```
[root@example-01 ~]# service gfs2 stop
Unmounting GFS2 filesystem (/mnt/gfsA):                    [  OK  ]
Unmounting GFS2 filesystem (/mnt/gfsB):                    [  OK  ]
[root@example-01 ~]# umount -at gfs2
[root@example-01 ~]# service clvmd stop
Signaling clvmd to exit                                    [  OK  ]
clvmd terminated                                           [  OK  ]
[root@example-01 ~]# service cman stop
Stopping cluster:
   Leaving fence domain...                                 [  OK  ]
   Stopping gfs_controld...                                [  OK  ]
   Stopping dlm_controld...                                [  OK  ]
   Stopping fenced...                                      [  OK  ]
   Stopping cman...                                        [  OK  ]
   Waiting for corosync to shutdown:                       [  OK  ]
   Unloading kernel modules...                             [  OK  ]
   Unmounting configfs...                                  [  OK  ]
[root@example-01 ~]#
```



注記

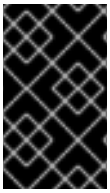
ノード上のクラスターソフトウェアを停止すると、HA サービスが別のノードにフェイルオーバーする要因となります。別の方法として、クラスターソフトウェアを停止する前に HA サービスを別のノードに再配置/移行することを検討してみてください。HA サービスの管理についての詳細は 「高可用性 (HA) サービスの管理」 を参照してください。

8.2. ノードの削除と追加

このセクションでは、クラスターに対してノードを削除/追加する方法を説明しています。クラスターからノードを削除するには、「クラスターからのノードの削除」 に従います。クラスターへノードを追加するには、「クラスターへのノードの追加」 に沿って行います。

8.2.1. クラスターからのノードの削除

クラスターからノードを削除するには、削除するノード上のクラスターソフトウェアをシャットダウンして、その変更を反映するためにクラスター設定を更新します。



重要

クラスターからノードを削除することで 2 つを超えるノードから 2 つのノードへ遷移する場合、クラスター設定ファイルの更新後、各ノードでクラスターソフトウェアを再起動しなければなりません。

クラスターからノードを削除するには、以下の手順を実行します。

1. いずれかのノードで **clusvcadm** ユーティリティを使用して、クラスターから削除されるノード上で実行している各 HA サービスの再配置/移行/停止を行います。**clusvcadm** の使用についての詳細は 「高可用性 (HA) サービスの管理」 を参照してください。

2. クラスターから削除するノード上で 「クラスターソフトウェアの停止」 に従ってクラスターソフトウェアを停止します。例えば：

   ```
   [root@example-01 ~]# service rgmanager stop
   ```

```
Stopping Cluster Service Manager:                          [  OK  ]
[root@example-01 ~]# service gfs2 stop
Unmounting GFS2 filesystem (/mnt/gfsA):                    [  OK  ]
Unmounting GFS2 filesystem (/mnt/gfsB):                    [  OK  ]
[root@example-01 ~]# service clvmd stop
Signaling clvmd to exit                                    [  OK
]
clvmd terminated                                           [  OK
]
[root@example-01 ~]# service cman stop
Stopping cluster:
   Leaving fence domain...                                 [  OK
]
   Stopping gfs_controld...                                [  OK
]
   Stopping dlm_controld...                                [  OK
]
   Stopping fenced...                                      [  OK
]
   Stopping cman...                                        [  OK
]
   Waiting for corosync to shutdown:                       [  OK  ]
   Unloading kernel modules...                             [  OK
]
   Unmounting configfs...                                  [  OK
]
[root@example-01 ~]#
```

3. クラスター内のいずれかのノードで **/etc/cluster/cluster.conf** を編集して、削除するノードの **clusternode** セクションを削除します。例えば、例8.1「3 ノードクラスターの設定」 で、node-03.example.com が削除されることになっている場合、そのノードの **clusternode** セクションを削除します。ノード (群) の削除によりクラスターが 2 ノードクラスターになる場合、設定ファイルに以下の行を追加することで単一ノードが定足数を維持できるようにします (例えば 1 つのノードが失敗した場合) :

   <cman two_node="1" expected_votes="1"/>

   3 ノードと 2 ノードの設定の比較は、 「3 ノードと 2 ノードの設定サンプル」 を参照してください。

4. **config_version** 属性を更新するには、値を増加します (例えば、**config_version="2"** から **config_version="3">** へ変更)。

5. **/etc/cluster/cluster.conf** を保存します。

6. **(オプション) ccs_config_validate** コマンドを実行してクラススキーマ (**cluster.rng**) に対して更新したファイルを検証します。例えば :

   ```
   [root@example-01 ~]# ccs_config_validate
   Configuration validates
   ```

7. **cman_tool version -r** コマンドを実行して、設定をクラスターノードの残り部分へ伝播します。

8. 更新した設定ファイルが伝播されたことを確認します。

9. クラスターのノード数を 2 つを超えるノードから 2 つのノードに変更した場合は、以下のようにしてクラスターソフトウェアを再起動しなければなりません：

    1. 各ノードで「クラスターソフトウェアの停止」 に従ってクラスターソフトウェアを停止します。例えば：

```
[root@example-01 ~]# service rgmanager stop
Stopping Cluster Service Manager:                          [  OK
]
[root@example-01 ~]# service gfs2 stop
Unmounting GFS2 filesystem (/mnt/gfsA):                    [  OK
]
Unmounting GFS2 filesystem (/mnt/gfsB):                    [  OK
]
[root@example-01 ~]# service clvmd stop
Signaling clvmd to exit                                    [
OK  ]
clvmd terminated                                           [
OK  ]
[root@example-01 ~]# service cman stop
Stopping cluster:
   Leaving fence domain...                                 [
OK  ]
   Stopping gfs_controld...                                [
OK  ]
   Stopping dlm_controld...                                [
OK  ]
   Stopping fenced...                                      [
OK  ]
   Stopping cman...                                        [
OK  ]
   Waiting for corosync to shutdown:                       [  OK
]
   Unloading kernel modules...                             [
OK  ]
   Unmounting configfs...                                  [
OK  ]
[root@example-01 ~]#
```

    2. 各ノードで「クラスターソフトウェアの起動」 に従ってクラスターソフトウェアを起動します。例えば：

```
[root@example-01 ~]# service cman start
Starting cluster:
   Checking Network Manager...                             [
OK  ]
   Global setup...                                         [
OK  ]
   Loading kernel modules...                               [
OK  ]
   Mounting configfs...                                    [
OK  ]
   Starting cman...                                        [
OK  ]
   Waiting for quorum...                                   [
OK  ]
```

```
   Starting fenced...                                         [
OK  ]
   Starting dlm_controld...                                   [
OK  ]
   Starting gfs_controld...                                   [
OK  ]
   Unfencing self...                                          [
OK  ]
   Joining fence domain...                                    [
OK  ]
[root@example-01 ~]# service clvmd start
Starting clvmd:                                               [
OK  ]
Activating VG(s):   2 logical volume(s) in volume group
"vg_example" now active
                                                              [
OK  ]
[root@example-01 ~]# service gfs2 start
Mounting GFS2 filesystem (/mnt/gfsA):                         [  OK
]
Mounting GFS2 filesystem (/mnt/gfsB):                         [  OK
]
[root@example-01 ~]# service rgmanager start
Starting Cluster Service Manager:                             [  OK
]
[root@example-01 ~]#
```

3. いずれかのクラスターノードで **cman_tool nodes** を実行して、ノード群がクラスター内のメンバーとして機能していることを確認します (ステータスカラム "Sts" 内で "M" として表示)。例えば：

```
[root@example-01 ~]# cman_tool nodes
Node  Sts   Inc   Joined               Name
   1   M    548   2010-09-28 10:52:21  node-01.example.com
   2   M    548   2010-09-28 10:52:21  node-02.example.com
```

4. いずれかのノードで **clustat** ユーティリティを使用して、HA サービスが期待どおりに稼働していることを確認します。さらに、**clustat** はクラスターノード群のステータスを表示します。例えば：

```
[root@example-01 ~]#clustat
Cluster Status for mycluster @ Wed Nov 17 05:40:00 2010
Member Status: Quorate

 Member Name                             ID   Status
 ------ ----                             ---- ------
 node-02.example.com                         2 Online, rgmanager
 node-01.example.com                         1 Online, Local,
rgmanager

 Service Name                   Owner (Last)
State
 ------- ----                   ----- ------
-----
 service:example_apache         node-01.example.com
```

```
started
 service:example_apache2        (none)
disabled
```

8.2.2. クラスターへのノードの追加

ノードをクラスターに追加するには、クラスター設定の更新、追加するノードへ更新した設定の伝播、ノード上でのクラスターソフトウェアの起動を行います。ノードをクラスターに追加するには、以下の手順を実行します：

1. クラスター内のいずれかのノードで **/etc/cluster/cluster.conf** を編集して、追加されるノード用に **clusternode** セクションを追加します。例えば、例8.2「2 ノードクラスターの設定」 で、node-03.example.com が追加されることになっている場合、そのノードに **clusternode** を追加します。ノード (群) の追加により 2 ノードクラスターから 3 つ以上のノードを持つクラスターへクラスターが遷移する場合、以下の **cman** 属性を **/etc/cluster/cluster.conf** から削除します：

   - **cman two_node="1"**
   - **expected_votes="1"**

   3 ノードと 2 ノードの設定の比較は、「3 ノードと 2 ノードの設定サンプル」 を参照してください。

2. **config_version** 属性を更新するには、値を増加します (例えば、**config_version="2"** から **config_version="3">** へ変更)。

3. **/etc/cluster/cluster.conf** を保存します。

4. **(オプション) ccs_config_validate** コマンドを実行してクラススキーマ (**cluster.rng**) に対して更新したファイルを検証します。例えば：

   ```
   [root@example-01 ~]# ccs_config_validate
   Configuration validates
   ```

5. **cman_tool version -r** コマンドを実行して、設定をクラスターノードの残り部分へ伝播します。

6. 更新した設定ファイルが伝播されたことを確認します。

7. 更新した設定ファイルを、クラスターに追加する各ノード内の **/etc/cluster/** に伝播します。例えば、**scp** コマンドを使用して、更新した設定ファイルをクラスターに追加する各ノードへ送信します。

8. クラスターのノード数が 2 つのノードから 2 つを超えるノードに遷移した場合は、以下のようにして既存のクラスターノード群内のクラスターソフトウェアを再起動しなければなりません：

   1. 各ノードで 「クラスターソフトウェアの停止」 に従ってクラスターソフトウェアを停止します。例えば：

      ```
      [root@example-01 ~]# service rgmanager stop
      Stopping Cluster Service Manager:                          [  OK
      ]
      [root@example-01 ~]# service gfs2 stop
      ```

```
Unmounting GFS2 filesystem (/mnt/gfsA):                    [  OK
]
Unmounting GFS2 filesystem (/mnt/gfsB):                    [  OK
]
[root@example-01 ~]# service clvmd stop
Signaling clvmd to exit                                    [
OK  ]
clvmd terminated                                           [
OK  ]
[root@example-01 ~]# service cman stop
Stopping cluster:
   Leaving fence domain...                                 [
OK  ]
   Stopping gfs_controld...                                [
OK  ]
   Stopping dlm_controld...                                [
OK  ]
   Stopping fenced...                                      [
OK  ]
   Stopping cman...                                        [
OK  ]
   Waiting for corosync to shutdown:                       [  OK
]
   Unloading kernel modules...                             [
OK  ]
   Unmounting configfs...                                  [
OK  ]
[root@example-01 ~]#
```

2. 各ノードで「クラスターソフトウェアの起動」に従ってクラスターソフトウェアを起動します。例えば：

```
[root@example-01 ~]# service cman start
Starting cluster:
   Checking Network Manager...                             [
OK  ]
   Global setup...                                         [
OK  ]
   Loading kernel modules...                               [
OK  ]
   Mounting configfs...                                    [
OK  ]
   Starting cman...                                        [
OK  ]
   Waiting for quorum...                                   [
OK  ]
   Starting fenced...                                      [
OK  ]
   Starting dlm_controld...                                [
OK  ]
   Starting gfs_controld...                                [
OK  ]
   Unfencing self...                                       [
OK  ]
   Joining fence domain...                                 [
OK  ]
```

```
[root@example-01 ~]# service clvmd start
Starting clvmd:                                            [
OK  ]
Activating VG(s):   2 logical volume(s) in volume group
"vg_example" now active
                                                           [
OK  ]
[root@example-01 ~]# service gfs2 start
Mounting GFS2 filesystem (/mnt/gfsA):                      [  OK
]
Mounting GFS2 filesystem (/mnt/gfsB):                      [  OK
]
[root@example-01 ~]# service rgmanager start
Starting Cluster Service Manager:                          [  OK
]
[root@example-01 ~]#
```

9. クラスターに追加する各ノードで「クラスターソフトウェアの起動」に従って、クラスターソフトウェアを起動します。例えば：

```
[root@example-01 ~]# service cman start
Starting cluster:
   Checking Network Manager...                             [  OK
]
   Global setup...                                         [  OK
]
   Loading kernel modules...                               [  OK
]
   Mounting configfs...                                    [  OK
]
   Starting cman...                                        [  OK
]
   Waiting for quorum...                                   [  OK
]
   Starting fenced...                                      [  OK
]
   Starting dlm_controld...                                [  OK
]
   Starting gfs_controld...                                [  OK
]
   Unfencing self...                                       [  OK
]
   Joining fence domain...                                 [  OK
]
[root@example-01 ~]# service clvmd start
Starting clvmd:                                            [  OK
]
Activating VG(s):   2 logical volume(s) in volume group "vg_example"
now active
                                                           [  OK
]
[root@example-01 ~]# service gfs2 start
Mounting GFS2 filesystem (/mnt/gfsA):                      [  OK  ]
Mounting GFS2 filesystem (/mnt/gfsB):                      [  OK  ]
```

```
[root@example-01 ~]# service rgmanager start
Starting Cluster Service Manager:                          [  OK  ]
[root@example-01 ~]#
```

10. いずれかのノードで **clustat** ユーティリティを使用して、それぞれ追加したノードがクラスターの一部として稼働していることを確認します。例えば：

```
[root@example-01 ~]#clustat
Cluster Status for mycluster @ Wed Nov 17 05:40:00 2010
Member Status: Quorate

 Member Name                                ID   Status
 ------ ----                                ---- ------
 node-03.example.com                            3 Online, rgmanager
 node-02.example.com                            2 Online, rgmanager
 node-01.example.com                            1 Online, Local,
rgmanager

 Service Name                     Owner (Last)
State
 ------- ----                     ----- ------                    --
---
 service:example_apache           node-01.example.com
started
 service:example_apache2          (none)
disabled
```

clustat の使用についての詳細は、「高可用性 (HA) サービスの管理」 を参照してください。

また、**cman_tool status** を使用して、ノードの投票数、ノード数、定足数を確認することもできます。例えば：

```
[root@example-01 ~]#cman_tool status
Version: 6.2.0
Config Version: 19
Cluster Name: mycluster
Cluster Id: 3794
Cluster Member: Yes
Cluster Generation: 548
Membership state: Cluster-Member
Nodes: 3
Expected votes: 3
Total votes: 3
Node votes: 1
Quorum: 2
Active subsystems: 9
Flags:
Ports Bound: 0 11 177
Node name: node-01.example.com
Node ID: 3
Multicast addresses: 239.192.14.224
Node addresses: 10.15.90.58
```

11. いずれかのノードで **clusvcadm** ユーティリティを使用して、実行中のサービスを新規に参加したノードに移行/再配置できます。また、無効になっているサービスも有効にすることができ

ます。**clusvcadm** の使用についての詳細は 「高可用性 (HA) サービスの管理」 をご覧ください。

8.2.3. 3 ノードと 2 ノードの設定サンプル

3 ノードと 2 ノードの設定の比較については、以下のサンプルを参照してください。

例8.1 3 ノードクラスターの設定

```
<cluster name="mycluster" config_version="3">
   <cman/>
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC">
              <device name="apc" port="1"/>
             </method>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="APC">
              <device name="apc" port="2"/>
            </method>
         </fence>
     </clusternode>
     <clusternode name="node-03.example.com" nodeid="3">
         <fence>
            <method name="APC">
              <device name="apc" port="3"/>
            </method>
         </fence>
     </clusternode>
   </clusternodes>
   <fencedevices>
         <fencedevice agent="fence_apc" ipaddr="apc_ip_example"
login="login_example" name="apc" passwd="password_example"/>
   </fencedevices>
   <rm>
       <failoverdomains>
           <failoverdomain name="example_pri" nofailback="0"
ordered="1" restricted="0">
               <failoverdomainnode name="node-01.example.com"
priority="1"/>
               <failoverdomainnode name="node-02.example.com"
priority="2"/>
               <failoverdomainnode name="node-03.example.com"
priority="3"/>
           </failoverdomain>
       </failoverdomains>
       <resources>
           <ip address="127.143.131.100" monitor_link="yes"
sleeptime="10">
             <fs name="web_fs" device="/dev/sdd2"
```

```
mountpoint="/var/www" fstype="ext3">
                  <apache config_file="conf/httpd.conf"
name="example_server" server_root="/etc/httpd" shutdown_wait="0"/>
              </fs>
           </ip>
       </resources>
       <service autostart="0" domain="example_pri" exclusive="0"
name="example_apache" recovery="relocate">
           <fs ref="web_fs"/>
           <ip ref="127.143.131.100"/>
           <apache ref="example_server"/>
       </service>
       <service autostart="0" domain="example_pri" exclusive="0"
name="example_apache2" recovery="relocate">
           <fs name="web_fs2" device="/dev/sdd3" mountpoint="/var/www"
fstype="ext3"/>
           <ip address="127.143.131.101" monitor_link="yes"
sleeptime="10"/>
           <apache config_file="conf/httpd.conf" name="example_server2"
server_root="/etc/httpd" shutdown_wait="0"/>
       </service>
   </rm>
</cluster>
```

例8.2 2 ノードクラスターの設定

```
<cluster name="mycluster" config_version="3">
   <cman two_node="1" expected_votes="1"/>
   <clusternodes>
     <clusternode name="node-01.example.com" nodeid="1">
         <fence>
            <method name="APC">
              <device name="apc" port="1"/>
             </method>
         </fence>
     </clusternode>
     <clusternode name="node-02.example.com" nodeid="2">
         <fence>
            <method name="APC">
              <device name="apc" port="2"/>
            </method>
         </fence>
   </clusternodes>
   <fencedevices>
         <fencedevice agent="fence_apc" ipaddr="apc_ip_example"
login="login_example" name="apc" passwd="password_example"/>
   </fencedevices>
   <rm>
       <failoverdomains>
           <failoverdomain name="example_pri" nofailback="0"
ordered="1" restricted="0">
              <failoverdomainnode name="node-01.example.com"
```

```
priority="1"/>
                <failoverdomainnode name="node-02.example.com"
priority="2"/>
            </failoverdomain>
        </failoverdomains>
        <resources>
            <ip address="127.143.131.100" monitor_link="yes"
sleeptime="10">
               <fs name="web_fs" device="/dev/sdd2"
mountpoint="/var/www" fstype="ext3">
                  <apache config_file="conf/httpd.conf"
name="example_server" server_root="/etc/httpd" shutdown_wait="0"/>
               </fs>
            </ip>
        </resources>
        <service autostart="0" domain="example_pri" exclusive="0"
name="example_apache" recovery="relocate">
            <fs ref="web_fs"/>
            <ip ref="127.143.131.100"/>
            <apache ref="example_server"/>
        </service>
        <service autostart="0" domain="example_pri" exclusive="0"
name="example_apache2" recovery="relocate">
            <fs name="web_fs2" device="/dev/sdd3" mountpoint="/var/www"
fstype="ext3"/>
            <ip address="127.143.131.101" monitor_link="yes"
sleeptime="10"/>
            <apache config_file="conf/httpd.conf" name="example_server2"
server_root="/etc/httpd" shutdown_wait="0"/>
        </service>
    </rm>
</cluster>
```

8.3. 高可用性 (HA) サービスの管理

高可用性サービスを管理するには、**Cluster Status Utility** である **clustat** と **Cluster User Service Administration Utility** である **clusvcadm** を使用します。**clustat** はクラスターのステータスを表示し、**clusvcadm** は高可用性サービスを管理する手段を提供します。

このセクションでは、**clustat** と **clusvcadm** コマンドを使用した HA サービスの管理についての基本情報を提供しています。以下のようなサブセクションで構成されています：

- 「**clustat** を使用した HA サービスステータスの表示」
- 「**clusvcadm** を使用した HA サービスの管理」

8.3.1. clustat を使用した HA サービスステータスの表示

clustat はクラスター全域のステータスを表示します。メンバーシップ情報、定足数表示、全ての高可用性サービスの状態を表示して、**clustat** コマンドが実行されているノード (ローカル) を示します。表8.1「サービスのステータス」 はサービスがなり得る状態と **clustat** を実行している時に表示

される状態を説明しています。例8.3「clustat の表示」 は、clustat 表示の例を示しています。clustat コマンドの実行に関する詳細情報は、clustat の man ページを参照してください。

表8.1 サービスのステータス

サービスのステータス	説明
Started	サービスリソースは設定されており、サービスを所有するクラスターシステム上で利用可能です。
Recovering	サービスは別のノード上で開始待ちです。
Disabled	サービスは無効になっており、割り当てられている所有者はいません。クラスターにより無効なサービスが自動的に再起動されることはありません。
Stopped	停止した状態で、サービスは、次のサービス又はノードの遷移後に起動するかどうか評価されます。これは一時的な状態です。この状態からサービスを無効/有効にすることができます。
Failed	サービスは dead と判定されます。リソースの **停止** 動作が失敗するとサービスは常にこの状態になります。サービスがこの状態になった後は、**disable** 要求を発行する前に、割り振られたリソース (例えば、マウント済みのファイルシステム) が存在しないことを確認しなければなりません。サービスがこの状態になった時に実行できる唯一の操作は **disable** です。
Uninitialized	この状態はスタートアップと **clustat -f** の実行中に特定のケースで出現します。

例8.3 clustat の表示

```
[root@example-01 ~]#clustat
Cluster Status for mycluster @ Wed Nov 17 05:40:15 2010
Member Status: Quorate

 Member Name                             ID   Status
 ------ ----                             ---- ------
 node-03.example.com                         3 Online, rgmanager
 node-02.example.com                         2 Online, rgmanager
 node-01.example.com                         1 Online, Local,
rgmanager

 Service Name                   Owner (Last)                   State
 ------- ----                   ----- ------                   -----
 service:example_apache         node-01.example.com            started
 service:example_apache2        (none)
disabled
```

8.3.2. clusvcadm を使用した HA サービスの管理

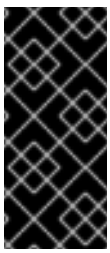
clusvcadm コマンドを使用して HA サービスを管理できます。これを使うと以下の操作を実行できます：

- サービスの有効と起動
- サービスの無効化
- サービスの停止
- サービスの凍結
- サービスの凍結解除
- サービスの移行（仮想マシンサービス用のみ）
- サービスの再配置
- サービスの再起動

表8.2「サービスの操作」 は、上記の操作について詳しく説明しています。これらの操作方法についての総括的な説明は、**clusvcadm** ユーティリティの man ページを参照してください。

表8.2 サービスの操作

サービスの操作	説明	コマンドの構文
Enable	サービスを起動します。オプションとして優先ターゲット上、及びフェイルオーバードメインのルールに従って行うことができます。どちらのオプションもない場合は、**clusvcadm** が実行するローカルホストがサービスを起動します。元々の **起動** が失敗した場合、サービスは **再配置** 操作が要求されたかのように動作します (この表の **Relocate** を参照)。動作が成功すると、サービスは起動した状態になります。	**clusvcadm -e <service_name>** 又は **clusvcadm -e <service_name> -m <member>** (-m オプションを使用すると、サービスを開始する優先ターゲットメンバーを指定します。
Disable	サービスを停止して無効の状態にします。これは、サービスが **停止** 状態にある時に許可される唯一の操作です。	**clusvcadm -d <service_name>**

サービスの操作	説明	コマンドの構文
Relocate	サービスを別のノードに移動します。オプションとして、サービスを受信するための優先ノードを指定できます。しかし、サービスがホスト上で実行できないこと（例えば、サービスが起動に失敗、又はホストがオフライン）により再配置が起こらないわけではなく、別のノードが選択されます。**rgmanager** はクラスター内の全ての許可されたノード上でサービスの起動を試みます。クラスター内の許可されたターゲットノードがサービスを正しく起動できない場合は、再配置は失敗となりサービスはオリジナルの所有者で再起動を試行します。オリジナルの所有者がサービスを再起動できない場合は、サービスは **停止** 状態になります。	**clusvcadm -r <service_name>** 又は **clusvcadm -r <service_name> -m <member>** (-m オプションを使用すると、サービスを起動する優先ターゲットメンバーを指定します。)
Stop	サービスを停止して、**停止** 状態にします。	**clusvcadm -s <service_name>**
Freeze	現在稼働しているノード上のサービスを凍結します。これにより、ノードが失敗又は rgmanager が停止した場合に、サービスのステータスチェックとフェイルオーバーは阻止されます。この機能を使用することで、サービスをサスペンドして基となるリソースをメンテナンスできます。**freeze** と **unfreeze** の操作を使用する上での重要な情報は、「Freeze と Unfreeze の操作時の注意事項」 を参照してください。	**clusvcadm -Z <service_name>**
Unfreeze	サービスを **凍結** 状態から解除します。これによりステータスチェックを再度有効にします。**freeze** と **unfreeze** の操作を使用する上で重要な情報は、「Freeze と Unfreeze の操作時の注意事項」 を参照してください。	**clusvcadm -U <service_name>**
Migrate	仮想マシンを別のノードに移行します。ターゲットノードを指定してください。移行時の障害の内容によって、仮想マシンは **失敗** 状態になるか、オリジナルの所有者上で起動した状態になる場合があります。	**clusvcadm -M <service_name> -m <member>**  **重要** **移行** 操作には、 **-m <member>** オプションを使用してターゲットノードを **指定しなければなりません。**
Restart	現在稼働しているノード上でサービスを再起動します。	**clusvcadm -R <service_name>**

Freeze と Unfreeze の操作時の注意事項

freeze 操作を行うと **rgmanager** サービスの一部をメンテナンスできます。例えば、1 つの **rgmanager** サービス内にデータベースとウェブサーバーがある場合、**rgmanager** サービスの凍結、データベースの停止、メンテナンス、データベースの再起動、サービスの凍結解除が可能になります。

サービスは凍結している時、以下のように動作します：

- **Status** チェックは無効です。
- **Start** 操作は無効です。
- **Stop** 操作は無効です。
- (サービス所有者を電源オフにしても) フェイルオーバーは起こりません

重要

以下のガイドラインに従わなかった場合、リソースが複数ホスト上で割り振られることになります：

- rgmanager の再起動の前にホストをリブートする予定でない限りは、サービスの凍結時に rgmanager の全てのインスタンスを停止 **しないでください。**
- 報告を受けたサービス所有者がクラスターに再参加して rgmanager を再起動するまでは、サービスを凍結解除 **しないでください。**

8.4. 設定の更新

クラスター設定を更新するには、クラスター設定ファイル (**/etc/cluster/cluster.conf**) を編集して、それをクラスター内の各ノードへ伝播します。設定を更新するには、以下のいずれかの手順を使用します：

- 「**cman_tool version -r** を使用した設定の更新」
- 「**scp** を使用した設定の更新」

8.4.1. cman_tool version -r を使用した設定の更新

cman_tool version -r コマンドを使用して設定を更新するには、以下の手順を実行します：

1. クラスター内のいずれかのノードで **/etc/cluster/cluster.conf** ファイルを編集します。
2. 値を増加することにより **config_version** 属性を更新します (例えば、**config_version="2"** から **config_version="3"** へ変更)。
3. **/etc/cluster/cluster.conf** を保存します。
4. **cman_tool version -r** コマンドを実行して、設定をクラスターノードの残り部分へ伝播します。更新したクラスター設定情報を伝播できるようにするには、 各クラスターノードで **ricci** が稼働している必要があります。
5. 更新された **cluster.conf** 設定が伝達されていることを確認します。伝達されていない場合、**scp** コマンドを使用してそれを各クラスターノード内の **/etc/cluster/** に伝達します。

6. 以下の設定のみを変更した場合は、このステップ (クラスターソフトウェアの再起動) を省略できます：

   - クラスター設定からノードを削除 — 2 つを超えるノードから 2 つのノードに変更する場合**以外**。クラスターからノードを削除する方法と 2 つを超えるノードから 2 つのノードに遷移する方法についての詳細は、「ノードの削除と追加」 を参照してください。
   - クラスター設定にノードを追加 — 2 つのノードから 2 つを超えるノードに変更する場合 **以外**。クラスターにノードを追加する方法と 2 ノードから 2 つを超えるノードに遷移する方法についての詳細は、「クラスターへのノードの追加」 を参照してください。
   - デーモンが情報をログする方法の変更
   - HA サービス/VM メンテナンス (追加、編集、又は削除)
   - リソースメンテナンス (追加、編集、又は削除)
   - フェイルオーバードメインのメンテナンス (追加、編集、又は削除)

   これら以外は、以下のようにしてクラスターソフトウェアを再起動する必要があります：

   1. 各ノードで 「クラスターソフトウェアの停止」 にしたがってクラスターソフトウェアを停止します。
   2. 各ノードで 「クラスターソフトウェアの起動」 にしたがってクラスターソフトウェアを起動します。

   クラスターソフトウェアを停止して起動することで、スタートアップ時にのみチェックされる設定の変更が確実に実行中の設定に含まれるようになります。

7. いずれかのクラスターノードで **cman_tool nodes** を実行して、ノード群がクラスター内のメンバーとして機能していることを確認します (ステータスカラム "Sts" 内で "M" として表示)。例えば：

```
[root@example-01 ~]# cman_tool nodes
Node  Sts   Inc   Joined               Name
   1   M    548   2010-09-28 10:52:21  node-01.example.com
   2   M    548   2010-09-28 10:52:21  node-02.example.com
   3   M    544   2010-09-28 10:52:21  node-03.example.com
```

8. いずれかのノードで **clustat** ユーティリティを使用して、HA サービスが期待どおりに稼働していることを確認します。さらに、**clustat** はクラスターノード群のステータスを表示します。例えば：

```
[root@example-01 ~]#clustat
Cluster Status for mycluster @ Wed Nov 17 05:40:00 2010
Member Status: Quorate

 Member Name                             ID   Status
 ------ ----                             ---- ------
 node-03.example.com                         3 Online, rgmanager
 node-02.example.com                         2 Online, rgmanager
 node-01.example.com                         1 Online, Local,
rgmanager

 Service Name                    Owner (Last)
```

```
State
 ------- ----                    -----  ------                    --
---
 service:example_apache          node-01.example.com
started
 service:example_apache2         (none)
disabled
```

9. クラスターが期待通りに動作している場合は、設定の更新は終了です。

8.4.2. scp を使用した設定の更新

scp コマンドを使用して設定を更新するには、以下の手順を実行します：

1. クラスター内のいずれかのノードで **/etc/cluster/cluster.conf** ファイルを編集します。
2. **config_version** 属性を更新するには、値を増加します (例えば、**config_version="2"** から **config_version="3">** へ変更)。
3. **/etc/cluster/cluster.conf** を保存します。
4. **ccs_config_validate** コマンドを実行することで、クラススキーマ (**cluster.rng**) に対して更新ファイルの妥当性検証をします。例えば：

```
[root@example-01 ~]# ccs_config_validate
Configuration validates
```

5. 更新したファイルが有効である場合、**scp** コマンドを使用してそれを各クラスターノード内の **/etc/cluster/** に伝播します。
6. 更新した設定ファイルが伝播されたことを確認します。
7. 新しい設定をリロードするには、クラスターノードのいずれかで以下のコマンドを実行します。

```
cman_tool version -r
```

ricci がインストールされていない場合、以下のコマンドを使うことができます。

```
cman_tool version -s
```

8. 以下の設定のみを変更した場合は、このステップ (クラスターソフトウェアの再起動) を省略できます：
   - クラスター設定からノードを削除 — 2 つを超えるノードから 2 つのノードに変更する場合 **以外**。クラスターからノードを削除する方法と 2 つを超えるノードから 2 つのノードに遷移する方法についての詳細は、「ノードの削除と追加」 を参照してください。
   - クラスター設定にノードを追加 — 2 つのノードから 2 つを超えるノードに変更する場合 **以外**。クラスターにノードを追加する方法と 2 ノードから 2 つを超えるノードに遷移する方法についての詳細は、「クラスターへのノードの追加」 を参照してください。
   - デーモンが情報をログする方法の変更

- HA サービス/VM メンテナンス (追加、編集、又は削除)
- リソースメンテナンス (追加、編集、又は削除)
- フェイルオーバードメインのメンテナンス (追加、編集、又は削除)

これら以外は、以下のようにしてクラスターソフトウェアを再起動する必要があります：

1. 各ノードで「クラスターソフトウェアの停止」にしたがってクラスターソフトウェアを停止します。
2. 各ノードで「クラスターソフトウェアの起動」にしたがってクラスターソフトウェアを起動します。

クラスターソフトウェアを停止して起動することで、スタートアップ時にのみチェックされる設定の変更が確実に実行中の設定に含まれるようになります。

9. ノードがクラスター内でメンバーとして機能しており、HA サービスが期待どおりに稼働していることを確認します。

   1. いずれかのクラスターノードで **cman_tool nodes** を実行して、ノード群がクラスター内のメンバーとして機能していることを確認します (ステータスカラム "Sts" 内で "M" として表示)。例えば：

      ```
      [root@example-01 ~]# cman_tool nodes
      Node  Sts   Inc   Joined               Name
         1   M    548   2010-09-28 10:52:21  node-01.example.com
         2   M    548   2010-09-28 10:52:21  node-02.example.com
         3   M    544   2010-09-28 10:52:21  node-03.example.com
      ```

   2. いずれかのノードで **clustat** ユーティリティを使用して、HA サービスが期待どおりに稼働していることを確認します。さらに、**clustat** はクラスターノード群のステータスを表示します。例えば：

      ```
      [root@example-01 ~]#clustat
      Cluster Status for mycluster @ Wed Nov 17 05:40:00 2010
      Member Status: Quorate

       Member Name                             ID   Status
       ------ ----                             ---- ------
       node-03.example.com                         3 Online, rgmanager
       node-02.example.com                         2 Online, rgmanager
       node-01.example.com                         1 Online, Local,
      rgmanager

       Service Name                   Owner (Last)
      State
       ------- ----                   ----- ------
      -----
       service:example_apache         node-01.example.com
      started
       service:example_apache2        (none)
      disabled
      ```

   クラスターが期待通りに動作している場合は、設定の更新は終了です。

第9章 クラスター内の問題の診断と修正

クラスターの問題は、その性質上トラブルシューティングが難しいことがあります。これは単一システムでの問題診断と比較して、システムのクラスターはより複雑なためです。しかし、システム管理者がクラスターの導入/管理を行う場合に遭遇する可能性が高い共通の問題があります。これらの問題に対処する方法を理解することにより、クラスターの導入と管理をより簡単に行うことができます。

本章は、クラスターに関するよくある問題とそれらのトラブルシューティングの方法について説明しています。別途ヘルプが必要な場合は、ナレッジベースを参照するか、Red Hat 認定サポート担当者にご連絡ください。問題が GFS2 ファイルシステム特有の場合、GFS2 に関するよくある問題は 『Global File System 2』 ドキュメントに記載されています。

9.1. 設定の変更が反映されない

クラスターの設定に変更を行った場合、クラスターの全てのノードに変更を伝播する必要があります。

- **Conga** を使用してクラスターを設定する場合、変更を適用すると自動的に **Conga** が変更を伝播します。
- **ccs** コマンドを使用してクラスター設定に変更を伝播する詳細については、「クラスタノード群への設定ファイルの伝播」 を参照してください。
- コマンドラインツールを使用してクラスター設定に変更を伝播する詳細については、「設定の更新」 を参照してください。

使用しているクラスターに以下のいずれかの設定変更を行う場合、変更を反映させるために変更を伝播した後クラスターを再起動する必要はありません。

- クラスター設定からノードを削除 — 2 つを超えるノードから 2 つのノードに変更する場合 **以外**
- クラスター設定にノードを追加 — 2 つのノードから 2 つを超えるノードに変更する場合 **以外**
- ロギング設定を変更
- HA サービス又は VM コンポーネントを追加、編集、又は削除
- クラスターリソースを追加、編集、又は削除
- フェイルオーバードメインを追加、修正、及び削除

ただし、使用しているクラスターに他の設定変更を行う場合は、変更を反映させるためにクラスターを再起動する必要があります。クラスターの再起動が必要な場合は、クラスターに以下の設定変更を行った場合です。

- クラスター設定ファイルから **two_node** オプションを追加又は削除
- クラスター名を変更
- **corosync** 又は **openais** のタイマーを変更
- quorum disk に対するヒューリスティックの追加/変更/削除、quorum disk タイマーの変更、quorum disk デバイスの変更。これらの変更を反映させるには、**qdiskd** デーモンをグローバルで再起動する必要があります。

- **rgmanager** の **central_processing** モードを変更。この変更を反映させるには、**rgmanager** をグローバルで再起動する必要があります。
- マルチキャストアドレスを変更
- トランスポートモードを UDP マルチキャストから UDP ユニキャストへ切り替え、又は UDP ユニキャストから UDP マルチキャストへ切り替え

Conga、**ccs** コマンド、又はコマンドラインツールを使用してクラスターを再起動することができます。

- **Conga** を使用したクラスターの再起動については、「クラスターの起動、停止、再起動、削除」 を参照してください。
- **ccs** コマンドを使用したクラスターの再起動については、「クラスターの起動と停止」 を参照してください。
- コマンドラインツールを使用したクラスターの再起動については、「クラスターソフトウェアの起動と停止」 を参照してください。

9.2. クラスターを構成できない

新規のクラスター構成で問題がある場合は、以下の項目をチェックしてください。

- 名前解決が正しく設定されていることを確認します。**cluster.conf** ファイル内のクラスターノード名は、そのクラスターが通信に使用するネットワーク上でクラスターのアドレスを解決するために使用する名前と一致する必要があります。例えば、クラスターノード名が **nodea** と **nodeb** である場合、両方のノードが **/etc/cluster/cluster.conf** ファイルと **/etc/hosts** ファイル内にそれらの名前にマッチするエントリを持っていることを確認してください。
- クラスターがノード間の通信にマルチキャストを使用する場合は、マルチキャストトラフィックがブロックされていないこと、遅延されていないこと、クラスターが通信に使用するネットワークで妨害されていないことを確認してください。一部の Cisco スイッチには、マルチキャストトラフィックで遅延の原因になる機能がある点に注意してください。
- **telnet** 又は **SSH** を使用して、リモートノードに到達するかどうか確認します。
- **ethtool eth1 | grep link** コマンドを実行して、イーサネットリンクが有効であるかをチェックします。
- 各ノードで **tcpdump** コマンドを使用して、ネットワークトラフィックをチェックします。
- ノード間でファイアウォールルールが通信をブロックしていないことを確認します。
- インターノード通信にクラスターが使用するインターフェースが 0、1、2 以外のボンディングモードを使用していないことを確認します (ボンディングモード 0 と 2 は Red Hat Enterprise Linux 6.4 よりサポートされています)。

9.3. フェンス後/再起動後にノードがクラスターに再参加できない

フェンスや再起動の後にノードがクラスターに再参加しない場合は、以下の項目をチェックしてください。

- Cisco Catalyst スイッチを経由してトラフィックを送っているクラスターはこの問題に遭遇する可能性があります。
- 全てのクラスターノードに同じバージョンの **cluster.conf** ファイルがあることを確認します。 **cluster.conf** ファイルがいずれかのノードで異なる場合は、それらのノードはフェンス後にクラスターに参加できない可能性があります。

  Red Hat Enterprise Linux 6.1 以降、以下のコマンドを使用すると、ホストのクラスター設定ファイル内で指定された全てのノードに同一のクラスター設定ファイルがあることを確認できます:

  ```
  ccs -h host --checkconf
  ```

  ccs コマンドに関する詳細情報は、*5章ccs コマンドを使用した Red Hat High Availability アドオンの設定* と *6章Red Hat High Availability アドオンを使用した ccs の管理* をご覧ください。
- クラスターへの参加を試行しているノード内のクラスターサービスに **chkconfig on** を設定していることを確認してください。
- ファイアウォールルールによって、ノードとクラスター内の他のノードとの通信がブロックされていないことを確認します。

9.4. クラスターデーモンがクラッシュする

RGManager には、メインの **rgmanager** プロセスが予期せず失敗した場合にホストを再起動するウォッチドッグプロセスがあります。これにより、クラスターノードはフェンスされ、**rgmanager** は別のホスト上でサービスを回復します。ウォッチドッグデーモンがメインの **rgmanager** プロセスがクラッシュしたことを検出すると、クラスターノードを再起動します。そして、アクティブなクラスターノードはそのクラスターノードが離脱したことを検出して、それをクラスターから削除します。

プロセス ID (PID) の比較的小さい数字は、監視プロセスで、子 (PID 番号が高いプロセス) がクラッシュした場合にアクションを取ります。**gcore** を使用して数値の高い PID 番号を持つプロセスのコアをキャプチャーして、クラッシュしたデーモンのトラブルシューティングのサポートをします。

コアをキャプチャ/表示するために必要なパッケージをインストールし、**rgmanager** 及び **rgmanager-debuginfo** が同一バージョンであることを確認してください。これが実行されないと、キャプチャされたアプリケーションコアが使用できない場合があります。

```
$ yum -y --enablerepo=rhel-debuginfo install gdb rgmanager-debuginfo
```

9.4.1. ランタイムでの rgmanager コアのキャプチャ

起動時に実行する **rgmanager** プロセスが 2 つあります。番号が大きい PID を持つ **rgmanager** プロセスのコアをキャプチャする必要があります。

以下は、**ps** コマンドの出力の例です。**rgmanager** の 2 つのプロセスを示しています。

```
$ ps aux | grep rgmanager | grep -v grep

root     22482  0.0  0.5  23544  5136 ?        S<Ls Dec01   0:00 rgmanager
root     22483  0.0  0.2  78372  2060 ?        S<l  Dec01   0:47 rgmanager
```

以下の例で、**pidof** プログラムは、コアを作成するために適切な PID である、番号が大きい PID を自動的に決定するために使用されます。このコマンドは、PID 番号が大きいプロセス 22483 のアプリケーションコアをキャプチャします。

```
$ gcore -o /tmp/rgmanager-$(date '+%F_%s').core $(pidof -s rgmanager)
```

9.4.2. デーモンクラッシュ時のコアのキャプチャ

デフォルトでは、**/etc/init.d/functions** スクリプトは **/etc/init.d/rgmanager** により呼び出されたデーモンからのコアファイルをブロックします。デーモンがアプリケーションコアを作成する場合は、そのオプションを有効にする必要があります。この手順は、キャプチャされたアプリケーションコアを必要とする全クラスターノード上で行わなければなりません。

rgmanager デーモンのクラッシュ時にコアファイルを作成するには、**/etc/sysconfig/cluster** ファイルを編集します。***DAEMONCOREFILELIMIT*** パラメーターで、プロセスがクラッシュした場合にデーモンがコアファイルを作成できるようにします。**-w** オプションというものがあり、これは監視プロセスが実行されないようにします。監視デーモンは、**rgmanager** がクラッシュした場合、(場合によっては) 監視デーモンが実行されているのにコアファイルが作成されない場合にクラスターノードを再起動するため、コアファイルのキャプチャーは無効にする必要があります。

```
DAEMONCOREFILELIMIT="unlimited"
RGMGR_OPTS="-w"
```

rgmanager を再起動して、新しい設定オプションをアクティベートします。

```
service rgmanager restart
```

注記

クラスターサービスがこのクラスターノードで実行している場合、実行中のサービスを不適切な状態にする可能性があります。

コアファイルは、**rgmanager** プロセスのクラッシュにより生成された場合に書き込まれます。

```
ls /core*
```

出力は以下のようになります：

```
/core.11926
```

アプリケーションコアをキャプチャするために **rgmanager** を再起動する前に、/ ディレクトリ下にある全ての古いファイルを移動もしくは削除してください。**rgmanager** クラッシュを経験したクラスターノードは、ウォッチドッグプロセスが実行していないことを確認するためにコアがキャプチャされた後、再起動又はフェンスされることになります。

9.4.3. gdb バックトレースセッションの記録

コアファイルをキャプチャしたら、GNU Debugger である **gdb** を使用してその内容を閲覧できます。影響を受けたシステムからコアファイルの **gdb** のスクリプトセッションを記録するには、以下のコマンドを実行します。

```
$ script /tmp/gdb-rgmanager.txt
$ gdb /usr/sbin/rgmanager /tmp/rgmanager-.core.
```

これにより **gdb** セッションが開始され、**script** によりそれを適切なテキストファイルに記録します。**gdb** で、以下のコマンドを実行します。

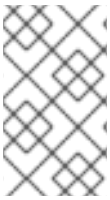
```
(gdb) thread apply all bt full
(gdb) quit
```

ctrl-D を押すと、スクリプトセッションを停止し、それをテキストファイルに保存します。

9.5. クラスターサービスがハングする

クラスターサービスがノードのフェンスを試みる場合、クラスターサービスはフェンス動作が正しく完了するまで停止しています。そのため、クラスターが制御するストレージ、又はサービスがハングして、クラスターノードが異なるクラスターメンバーシップを表示する場合、あるいはノードのフェンスを試行する時にクラスターがハングして、ノードを再起動して復元する必要がある場合には、以下の状況をチェックしてください。

- クラスターがノードのフェンスを試行して、フェンス動作が失敗した可能性があります。
- 全てのノード上で **/var/log/messages** ファイルを検証して、フェンス障害のメッセージがあるかどうかチェックします。もしあれば、クラスター内のノード群を再起動して、フェンシングを正しく設定します。
- 「2 ノードクラスターの各ノードが片方のノードの停止を報告する」 の説明のように、ネットワークパーティションが発生しなかったことを確認します。また、ノード間の通信が引き続き可能であり、ネットワークが有効であることを確認します。
- 一部のノードがクラスターから離脱すると、残りのノード群は定足数に足りなくなる場合があります。クラスターは機能するために定足数が必要です。ノードが削除されてクラスターが定足数を満たさなくなると、サービスとストレージはハングします。期待投票数を調節するか、必要な数のノードをクラスターに戻します。



注記

fence_node コマンド又は **Conga** を使用して手動でノードをフェンスすることができます。詳細は **fence_node** の man ページ及び 「ノードのクラスターに対する離脱/参加」 を参照してください。

9.6. クラスターサービスが起動しない

クラスターが制御するサービスが起動しない場合は、以下の状況をチェックしてください。

- **cluster.conf** ファイル内のサービス設定に構文エラーがある可能性があります。**rg_test** コマンドを使用すると、設定の構文を検証できます。設定や構文に間違いがある場合は、**rg_test** が問題を知らせてくれます。

  ```
  $ rg_test test /etc/cluster/cluster.conf start service servicename
  ```

  rg_test コマンドに関する詳細情報は、「サービスとリソース順序へのデバッグとテスト」をご覧ください。

設定が有効ならば、リソースグループマネージャーのロギングを増加してメッセージログを読むことにより、サービスの失敗となる原因を判定します。ログレベルを増加するには、**loglevel="7"** パラメーターを **cluster.conf** ファイル内の **rm** タグに追加します。そうすると、クラスター化サービスの起動、停止、移行に関するメッセージログをより詳細に表示することができます。

9.7. クラスターが制御するサービスが移行に失敗する

クラスターが制御するサービスが別のノードへの移行に失敗しても、サービスが一部のノード上で起動する場合は、以下の状況をチェックしてください。

- サービス実行に必要なクラスター内の全ノード上に、そのサービスの実行に必要なリソースが存在することを確認します。例えば、クラスター化したサービスが、スクリプトファイルが特定の場所にある、あるいはファイルシステムが特定のマウントポイントでマウントされていると仮定する場合、これらのリソースがクラスター内の全ノード上の予期される位置で利用可能であることを確認しなければなりません。
- フェイルオーバードメイン、サービスの依存関係、サービスの独占性が、期待通りにサービスをノードに移行できない方法で設定されていないことを確認します。
- 問題となるサービスが仮想マシンのリソースである場合、ドキュメントをチェックして、正しい設定作業がすべて完了していることを確認します。
- 「クラスターサービスが起動しない」 で説明してあるように、リソースグループマネージャーのログを増やします。その後、メッセージログを読み取り、移行する際にサービス起動が失敗する原因を割り出します。

9.8. 2 ノードクラスターの各ノードが片方のノードの停止を報告する

使用しているクラスターが 2 ノードクラスターであり、各ノードが自身は稼働しているが他方のノードが停止していると報告する場合、そのクラスターノード群がクラスターハートビートネットワーク上でマルチキャストを介して相互に通信できていないことを示しています。これは「スプリットブレイン」、又は「ネットワークパーティション」と呼ばれます。これに対処するには、「クラスターを構成できない」 で説明してある状況をチェックしてください。

9.9. LUN パス障害でノードがフェンスされる

LUN パスに障害が発生する度に、クラスター内のノードがフェンスされる場合は、マルチパス化したストレージ上で quorum disk を使用したことが原因である場合があります。quorum disk を使用していて、その quorum disk がマルチパス化したストレージ上にある場合は、パス障害を許容するために全てのタイミングが正しくセットされていることを確認してください。

9.10. QUORUM DISK がクラスターメンバーとして表示されない

quorum disk を使用するようにシステムを設定していても quorum disk がクラスターのメンバーとして表示されない場合は、以下の状況をチェックしてください。

- **qdisk** サービスに **chkconfig on** がセットされていることを確認します。
- **qdisk** サービスを開始したことを確認します。
- なお、quorum disk がクラスターに登録されるには数分かかります。これは通常の起こりうる動作です。

9.11. 異常なフェイルオーバーの動作

クラスターサーバーでよくある問題は異常なフェイルオーバーの動作です。あるサービスが起動した時に別のサービスが停止する、あるいはサービスがフェイルオーバー時に起動を拒否することがあります。これはフェイルオーバードメイン、サービスの依存関係、サービスの独占性などから構成される、フェイルオーバーの複雑なシステムが原因です。より簡易なサービスやフェイルオーバードメイン設定に戻して、問題が引き続き発生するかどうか確認してください。全条件下でこれらの機能がフェイルオーバーに及ぼす影響を完全に把握していない限りは、サービス専任性や依存関係などの機能は使用しないことをお勧めします。

9.12. フェンシングがランダムに発生する

ノードが無作為にフェンスされる場合は、以下の状況をチェックしてください。

- フェンスの根本的な原因は **常に** トークンを紛失するノードです。これは、ノードがクラスターの残りと通信ができなくなり、ハートビートの送信を停止するという意味です。
- 指定されたトークンの間隔内にシステムがハートビートを返さない状況は、いずれもフェンスにつながります。デフォルトではトークンの間隔は 10 秒です。これを指定するには、任意の値 (ミリ秒単位) を **cluster.conf** ファイル内の totem タグのトークンパラメーターに追加します (例えば 30 秒の場合は **totem token="30000"** とセット)。
- ネットワークが健全で期待通りに機能していることを確認します。
- インターノード通信にクラスターが使用するインターフェースが 0、1、2 以外のボンディングモードを使用していないことを確認します (ボンディングモード 0 と 2 は Red Hat Enterprise Linux 6.4 よりサポートされています)。
- システムが「凍結」又はカーネルパニックを起こしていないかどうか判定する手段を取ります。**kdump** ユーティリティをセットアップして、フェンスの発生時にコアを取得するかどうか確認します。
- 誤ってフェンスに原因があるというような状況を作り出さないようにしてください。例えば、ストレージ障害が原因で quorum disk がノードを取り出す場合や、何らかの外部条件が理由で Oracle RAC のようなサードパーティ製品がノードを再起動するような場合です。多くの場合、そのような問題の特定にはメッセージログが非常に役立ちます。フェンス又はノードの再起動が発生すると常に、発生した時点からクラスター内の全ノードのメッセージログを検査することを標準的な作業として行ってください。
- 予定している時にシステムのハートビートの応答がない原因となるハードウェア欠陥がないか、システムを徹底的に検査します。

9.13. DLM (分散ロックマネージャー) のデバッグログは有効にする

DLM (分散ロックマネージャー) には必要に応じて有効にできる 2 つのデバッグオプションがあります。DLM カーネルデバッグと POSIX ロックデバッグです。

DLM デバッグを有効にするには、**/etc/cluster/cluster.conf** ファイルを編集して、設定オプションを **dlm** タグに追加します。**log_debug** オプションは DLM カーネルデバッグメッセージを有効にし、**plock_debug** オプションは POSIX ロックデバッグメッセージを有効にします。

以下の例にある **/etc/cluster/cluster.conf** ファイル内のセクションは、両方の DLM デバッグオプションを有効にする **dlm** タグを示しています。

```
<cluster config_version="42" name="cluster1">
  ...
  <dlm log_debug="1" plock_debug="1"/>
  ...
</cluster>
```

/etc/cluster/cluster.conf ファイルの編集後、**cman_tool version -r** コマンドを実行して残りのクラスターノードに設定を伝播します。

第10章 RED HAT HIGH AVAILABILITY アドオンを使用した SNMP の設定

Red Hat Enterprise Linux 6.1 リリース以降、Red Hat High Availability アドオンは SNMP トラップへのサポートを提供します。本章では、SNMP 用にシステムを設定する方法を説明した後に、Red Hat High Availability アドオンがクラスターイベントに送信するトラップについてまとめています。

10.1. SNMP と RED HAT HIGH AVAILABILITY アドオン

Red Hat High Availability アドオンの SNMP サブエージェントは **foghorn** で、SNMP トラップを送信します。**foghorn** サブエージェントは、AgentX プロトコルを介して **snmpd** デーモンと通信します。**foghorn** サブエージェントは SNMP トラップを作成するだけで、**get** や **set** など他の SNMP 動作はサポートしません。

現時点では、**foghorn** サブエージェントに **config** オプションはなく、ある特定のソケットを使用するための設定を行うことができません。デフォルトの AgentX ソケットのみが現在サポートされています。

10.2. RED HAT HIGH AVAILABILITY アドオンを使用した SNMP の設定

Red Hat High Availability アドオンを使用して SNMP を設定するには、クラスター内の各ノード上で以下の手順を使用して、必要なサービスが有効かつ稼働しているようにします。

1. Red Hat High Availability アドオンで SNMP トラップを使用するには、**snmpd** サービスが必要となり、これはマスターエージェントの役割をします。**foghorn** サービスはサブエージェントであり、AgentX プロトコルを使用するため、以下の行を **/etc/snmp/snmpd.conf** ファイルに追加して、AgentX サポートを有効にする必要があります。

   ```
   master agentx
   ```

2. SNMP トラップ通知の送信先となるホストを指定するには、以下の行を **/etc/snmp/snmpd.conf** ファイルに追加します:

   ```
   trap2sink host
   ```

   通知の処理方法に関する詳細は **snmpd.conf** の man ページをご覧ください。

3. 以下のコマンド群を実行して **snmpd** デーモンが有効かつ稼働していることを確認します:

   ```
   # chkconfig snmpd on
   # service snmpd start
   ```

4. **messagebus** デーモンがまだ有効でなく稼働していない場合は、以下のコマンド群を実行します:

   ```
   # chkconfig messagebus on
   # service messagebus start
   ```

5. 以下のコマンド群を実行して **foghorn** デーモンが有効で稼働していることを確認します:

```
# chkconfig foghorn on
# service foghorn start
```

6. 以下のコマンドを実行して、**COROSYNC-MIB** が SNMP トラップを生成するようにシステムを設定し、**corosync-notifyd** デーモンが有効かつ実行しているようにします。

```
# echo "OPTIONS=\"-d\" " > /etc/sysconfig/corosync-notifyd
# chkconfig corosync-notifyd on
# service corosync-notifyd start
```

クラスター内の各ノードを SNMP 用に設定し、必要なサービスが稼働中であることを確認すると、**foghorn** サービスから D-bus 信号を受信し、SNMPv2 トラップに解釈されます。その後、これらのトラップは、**trapsink** エントリで定義しているホストに渡されて SNMPv2 トラップを受信します。

10.3. SNMP トラップの転送

SNMP トラップをクラスターの一部ではないマシンに転送することが可能です。外部マシン上で **snmptrapd** デーモンを使用して、通知に対する応答方法をカスタマイズできます。

以下の手順に従って、クラスター内の SNMP トラップをクラスターノード群の一部でないマシンに転送します:

1. クラスター内の各ノードに対して、「Red Hat High Availability アドオンを使用した SNMP の設定」 で説明してある手順を実行します。**/etc/snmp/snmpd.conf** ファイル内の **trap2sink *host*** エントリをセットして、**snmptrapd** デーモンを実行することになる外部ホストを指定します。

2. トラップを受信する外部ホスト上で、**/etc/snmp/snmptrapd.conf** 設定ファイルを編集して使用しているコミュニティ文字列を指定します。例えば、以下のエントリを使用して、**snmptrapd** デーモンが **public** コミュニティ文字列を使用して通知を処理できるようにします。

```
authCommunity log,execute,net public
```

3. トラップを受信する外部ホスト上で以下のコマンド群を実行して、 **snmptrapd** デーモンが有効で稼働中であることを確認します:

```
# chkconfig snmptrapd on
# service snmptrapd start
```

SNMP 通知の処理方法に関する詳細については **snmptrapd.conf** の man ページをご覧ください。

10.4. RED HAT HIGH AVAILABILITY アドオンにより生成される SNMP トラップ

foghorn デーモンは以下のトラップを生成します:

- **fenceNotifyFenceNode**

このトラップは、フェンス済みのノードが別のノードへのフェンスを試行する度に発生します。このトラップは 1 つのノード上（フェンス操作を試みたノード）でのみ生成される点に注意してください。この通知には以下のフィールドが含まれます:

- **fenceNodeName** - フェンス済みノードの名前
- **fenceNodeID** - フェンス済みノードのノード id
- **fenceResult** - フェンス操作の結果（0 は成功、-1 は問題あり、-2 はフェンシングメソッド定義なし）

- **rgmanagerServiceStateChange**

  このトラップは、クラスターサービスの状態が変化した時に発生します。この通知には以下のフィールドが含まれます:

  - **rgmanagerServiceName** - サービスの名前、これにはサービスタイプ（例えば **service:foo** や **vm:foo**）が含まれます。
  - **rgmanagerServiceState** - サービスの状態。トラップが複雑にならないように、**starting** や **stopping** などの遷移状態は除きます。
  - **rgmanagerServiceFlags** - サービスのフラグ。現在、サポートされているフラグは 2 つです。**frozen** は **clusvcadm -Z** を使用して凍結されているサービスを意味します。**partial** は、失敗したリソースに **non-critical** のフラグが付けられ、リソースが失敗してもそのコンポーネントはサービス全体に影響を与えずに手動で再開始できることを示しています。
  - **rgmanagerServiceCurrentOwner** - サービスのオーナー。サービスが稼働していない場合は、これは **(none)** になります。
  - **rgmanagerServicePreviousOwner** - 既知の場合、前回のサービスオーナー。前回のオーナーが不明である場合は、これは **(none)** を示します。

corosync-nodifyd デーモンは以下のトラップを生成します:

- **corosyncNoticesNodeStatus**

  このトラップはノードがクラスターに参加/離脱する時に発生します。通知には以下のフィールドが含まれます:

  - **corosyncObjectsNodeName** - ノード名
  - **corosyncObjectsNodeID** - ノード id
  - **corosyncObjectsNodeAddress** - ノード IP アドレス
  - **corosyncObjectsNodeStatus** - ノードのステータス (**joined** 又は **left**)

- **corosyncNoticesQuorumStatus**

  このトラップは quorum の状態が変化した時に発生します。この通知には以下のフィールドが含まれます:

  - **corosyncObjectsNodeName** - ノード名

- **corosyncObjectsNodeID** - ノード id
- **corosyncObjectsQuorumStatus** - quorum の新しい状態 (**quorate** 又は **NOT quorate**)

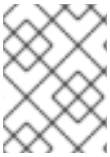
- **corosyncNoticesAppStatus**

  このトラップはクライアントアプリケーションが Corosync に対して接続/切断した時に発生します。

  - **corosyncObjectsNodeName** - ノード名
  - **corosyncObjectsNodeID** - ノード id
  - **corosyncObjectsAppName** - アプリケーション名
  - **corosyncObjectsAppStatus** - アプリケーションの新しい状態 (**connected** 又は **disconnected**)

第11章 CLUSTERED SAMBA の設定

Red Hat Enterprise Linux 6.2 リリースでは、Red Hat High Availability アドオンは Clustered Samba を active/active 設定で実行できるようになっています。これには、クラスターの全ノードに CTDB をインストールして設定する必要があり、これとあわせてGFS2 クラスターファイルシステムを使用します。



注記

Red Hat Enterprise Linux 6 は Clustered Samba を実行するノードを最大で 4 つサポートします。

本章では、サンプルのシステムを設定することにより CTDB を設定する手順を説明します。GFS2 ファイルシステムの設定については 『Global File System 2』 を参照してください。論理ボリュームの設定は、『論理ボリュームマネージャの管理』 をご覧ください。

注記

Samba 外から Samba シェア内にあるデータへの同時アクセスは、サポートされていません。

11.1. CTDB の概要

CTDB は Samba により使用される TDB データベースのクラスター実装です。CTDB を使用するには、クラスター化ファイルシステムが利用可能、かつクラスター内の全ノードで共有されている必要があります。CTDB は、このクラスター化ファイルシステムに加えクラスター化機能を提供します。また、Red Hat Enterprise Linux 6.2 リリース以降、CTDB はクラスタースタックを Red Hat Enterprise Linux クラスタリングにより提供されるクラスタースタックと同時に実行します。CTDB はノードメンバーシップ、リカバリ/フェイルオーバー、IP の再配置、Samba サービスを管理します。

11.2. 必要なパッケージ

Red Hat Enterprise Linux クラスタリングで Samba を実行するには、Red Hat High Availability アドオン及び Red Hat Resilient Storage アドオンを実行するために必要な標準パッケージに加えて、以下のパッケージが必要です。

- **`ctdb`**
- **`samba`**
- **`samba-common`**
- **`samba-winbind-clients`**

11.3. GFS2 の設定

Red Hat Enterprise Linux クラスタリングで Samba を設定するには、2 つの GFS2 ファイルシステムが必要です。1 つは CTDB 用の小規模なファイルシステム、もう 1 つは Samba 共有用のファイルシステムです。以下の例は、これら 2 つの GFS2 ファイルシステムを作成する方法を示しています。

GFS2 ファイルシステムを作成する前に、最初に LVM 論理ボリュームを各ファイルシステム用に作成してください。LVM 論理ボリュームの作成については、『論理ボリュームマネージャの管理』 を参照してください。この例では、以下の論理ボリュームを使用します。

- Samba 共有でエクスポートされるユーザーデータを保持し、そのデータに合うサイズにする **/dev/csmb_vg/csmb_lv**。この例では、100GB サイズの論理ボリュームを作成します。
- 共有の CTDB 状態の情報を格納し、1GB サイズが必要な **/dev/csmb_vg/ctdb_lv**。

クラスター化されたボリュームグループ及び論理ボリュームをクラスターの 1 ノードのみに作成します。

論理ボリュームに GFS2 ファイルシステムを作成するには、**mkfs.gfs2** コマンドを実行します。このコマンドは 1 クラスターノードのみで実行します。

/dev/csmb_vg/csmb_lv の論理ボリュームで Samba 共有をホストするファイルシステムを作成するには、次のコマンドを実行します。

```
[root@clusmb-01 ~]# mkfs.gfs2 -j3 -p lock_dlm -t csmb:gfs2
/dev/csmb_vg/csmb_lv
```

パラメーターの意味は以下のとおりです。

-j

ファイルシステムで作成するジャーナルの数を指定します。この例では、3 つのノードから成るクラスターを作成します。つまり、ノードごとに 1 つのジャーナルを作成します。

-p

ロッキングプロトコルを指定します。**lock_dlm** は GFS2 がノード間通信に使用するロッキングプロトコルです。

-t

ロックテーブル名を *cluster_name:fs_name* の形式で指定します。この例では、**cluster.conf** ファイルで指定されたクラスター名は **csmb** です。ファイルシステム名には **gfs2** を使用します。

このコマンドの出力は以下のように表示されます。

```
This will destroy any data on /dev/csmb_vg/csmb_lv.
  It appears to contain a gfs2 filesystem.

Are you sure you want to proceed? [y/n] y

Device:
/dev/csmb_vg/csmb_lv
Blocksize:  4096
Device Size  100.00 GB (26214400 blocks)
Filesystem Size: 100.00 GB (26214398 blocks)
Journals:  3
Resource Groups:  400
Locking Protocol:   "lock_dlm"
Lock Table:   "csmb:gfs2"
UUID:
  94297529-ABG3-7285-4B19-182F4F2DF2D7
```

この例では、**/dev/csmb_vg/csmb_lv** ファイルシステムは全ノードの **/mnt/gfs2** にマウントされます。このマウントポイントは、**/etc/samba/smb.conf** ファイル内で **path =** オプションを使って

share ディレクトリの場所として指定する値と一致する必要があります。詳細は 「Samba の設定」 に記載されています。

/dev/csmb_vg/ctdb_lv 論理ボリュームの CTDB 状態の情報をホストするためにファイルシステムを作成するには、次のコマンドを実行します。

```
[root@clusmb-01 ~]# mkfs.gfs2 -j3 -p lock_dlm -t csmb:ctdb_state
/dev/csmb_vg/ctdb_lv
```

なお、このコマンドで指定するロックテーブル名は **/dev/csmb_vg/csmb_lv** 上にファイルシステムを作成した例のロックテーブル名とは異なります。これで、ファイルシステムに使用される様々なデバイス用のロックテーブル名と区別します。

mkfs.gfs2 の出力は以下のように表示されます。

```
This will destroy any data on /dev/csmb_vg/ctdb_lv.
  It appears to contain a gfs2 filesystem.

Are you sure you want to proceed? [y/n] y

Device:
/dev/csmb_vg/ctdb_lv
Blocksize:    4096
Device Size   1.00 GB (262144 blocks)
Filesystem Size:  1.00 GB (262142 blocks)
Journals:  3
Resource Groups:  4
Locking Protocol:  "lock_dlm"
Lock Table:   "csmb:ctdb_state"
UUID:
  BCDA8025-CAF3-85BB-B062-CC0AB8849A03
```

この例では、**/dev/csmb_vg/ctdb_lv** ファイルシステムは全ノードの **/mnt/ctdb** にマウントされます。このマウントポイントは、**/etc/sysconfig/ctdb** ファイル内で **CTDB_RECOVERY_LOCK** オプションを使って **.ctdb.lock** ファイルの場所として指定する値と一致する必要があります。詳細は 「CTDB の設定」 に記載されています。

11.4. CTDB の設定

CTDB 設定ファイルは **/etc/sysconfig/ctdb** にあります。CTDB が動作するよう設定する必要がある必須フィールドは、以下のとおりです。

- **CTDB_NODES**
- **CTDB_PUBLIC_ADDRESSES**
- **CTDB_RECOVERY_LOCK**
- **CTDB_MANAGES_SAMBA** (有効にする必要あり)
- **CTDB_MANAGES_WINBIND** (メンバーサーバーで実行する場合は有効にする必要あり)

以下の例は、サンプルのパラメーターを使って設定した CTDB の動作の必須フィールドが入力された設定ファイルを示しています。

```
CTDB_NODES=/etc/ctdb/nodes
CTDB_PUBLIC_ADDRESSES=/etc/ctdb/public_addresses
CTDB_RECOVERY_LOCK="/mnt/ctdb/.ctdb.lock"
CTDB_MANAGES_SAMBA=yes
CTDB_MANAGES_WINBIND=yes
```

上記のパラメーターの意味は以下のとおりです。

CTDB_NODES

クラスターノードの一覧を含むファイルの場所を指定します。

CTDB_NODES が参照する **/etc/ctdb/nodes** ファイルには、次の例のようにクラスターノードの IP アドレスが一覧表示されているだけです。

```
192.168.1.151
192.168.1.152
192.168.1.153
```

この例では、クラスター/CTDB 通信とクライアントサービスの両方に使用されるインターフェース/IP は各ノードに 1 つだけあります。しかし、各クラスターノードは 2 つのネットワークインターフェースを持つことが強く推奨されます。これにより、インターフェースの 1 セットはクラスター/CTDB 通信専用に、別のインターフェースのセットはパブリッククライアントアクセス専用にできます。クラスターネットワークの適切な IP アドレスを使用し、**cluster.conf** ファイル内で使用されているホスト名/IP アドレスが同じであることを確かめてください。同様に、**public_addresses** ファイル内のクライアントアクセスに対してもパブリックネットワークの適切なインターフェースを使用するようにしてください。

ここで、極めて重要なことは、**/etc/ctdb/nodes** ファイルが全ノードで全く同一である点です。順番が重要であり、別々のノードで違う情報が見つかると CTDB は失敗します。

CTDB_PUBLIC_ADDRESSES

このクラスターによってエクスポートされる Samba 共有にアクセスするために使用可能な IP アドレスを一覧表示するファイルの場所を指定します。これらは、Clustered Samba サーバー名用に DNS で設定すべき IP アドレスで、CIFS クライアントが接続するアドレスです。Clustered Samba サーバー名を複数の IP アドレスを持つ 1 つの DNS の A レコードタイプとして設定し、ラウンドロビン DNS がクラスターノード全体にクライアントを分散するようにします。

この例では、ラウンドロビン DNS エントリ **csmb-server** を **/etc/ctdb/public_addresses** ファイルに一覧表示されている全アドレスで設定しました。DNS はこのエントリを使用するクライアントをクラスター全体にラウンドロビン式で分散します。

各ノードの **/etc/ctdb/public_addresses** ファイルの内容は次のとおりです。

```
192.168.1.201/0 eth0
192.168.1.202/0 eth0
192.168.1.203/0 eth0
```

この例では、現在ネットワークで使用されていない 3 つのアドレスを使用します。実際に使用する設定では、対象となるクライアントがアクセスできるアドレスを選択してください。

もう一つの方法として、この例では、合計 4 つのパブリックアドレスを除いた 3 つのノードがあるクラスター内の **/etc/ctdb/public_addresses** ファイルのコンテンツを表示しています。 以下の例では、IP アドレス 198.162.2.1 は、ノード 0 又はノード 1 のいずれかによってホストでき、これらのノードのうち少なくとも 1 つが利用可能な限りクライアントは使用することができます。

ノード 0 とノード 1 の両方に障害があった場合に限り、クライアントはこのパブリックアドレスを使用できなくなります。他のすべてのパブリックアドレスは、それぞれ単一のノードによって機能するため、それぞれ対応するノードも利用可能な場合にのみ使用できます。

ノード 0 の **/etc/ctdb/public_addresses** ファイルには次のコンテンツが含まれます。

```
198.162.1.1/24 eth0
198.162.2.1/24 eth1
```

ノード 1 の **/etc/ctdb/public_addresses** ファイルには次のコンテンツが含まれます。

```
198.162.2.1/24 eth1
198.162.3.1/24 eth2
```

ノード 2 の **/etc/ctdb/public_addresses** ファイルには次のコンテンツが含まれます。

```
198.162.3.2/24 eth2
```

CTDB_RECOVERY_LOCK

CTDB がリカバリ用に内部で使用するロックファイルを指定します。このファイルは、すべてのクラスターノードがアクセスできるように共有ストレージに存在する必要があります。このセクションの例は、全ノードで **/mnt/ctdb** にマウントされる GFS2 ファイルシステムを使用します。これは、エクスポートされる Samba 共有をホストする GFS2 ファイルシステムとは異なります。このリカバリロックファイルは、スプリットブレインシナリオを防ぐために使用されます。CTDB (1.0.112 以降) の新しいバージョンでは、このファイルがスプリットブレインを防ぐ別の仕組みで置換されていれば、このファイルの指定はオプションとなります。

CTDB_MANAGES_SAMBA

このフィールドを **yes** に設定して有効にする場合、CTDB が Samba サービスの起動/停止を行うことができると指定します。これは、CTDB がサービスの移行/フェイルオーバーを提供するために必要だと考えられているためです。

CTDB_MANAGES_SAMBA を有効にしたら、以下のコマンドを実行して、**smb** 及び **nmb** デーモンの **init** の自動スタートアップを無効にしてください。

```
[root@clusmb-01 ~]# chkconfig snb off
[root@clusmb-01 ~]# chkconfig nmb off
```

CTDB_MANAGES_WINBIND

このフィールドを **yes** に設定して有効にする場合、必要に応じて CTDB が **winbind** デーモンの起動/停止を行うことができると指定します。Windows ドメイン又は Active Directory セキュリティモードで CTDB を使用している場合には、これを有効にすることをお勧めします。

CTDB_MANAGES_WINBIND を有効にしたら、次のコマンドを実行して、 **winbind** デーモンの **init** の自動スタートアップを無効にしてください。

```
[root@clusmb-01 ~]# chkconfig windinbd off
```

11.5. SAMBA の設定

この例では、Samba の設定ファイルである **smb.conf** は **/etc/samba/smb.conf** にあります。このファイルには、以下のパラメーターが含まれています。

```
[global]
 guest ok = yes
 clustering = yes
 netbios name = csmb-server
[csmb]
 comment = Clustered Samba
  public = yes
 path = /mnt/gfs2/share
 writeable = yes
 ea support = yes
```

この例では、**/mnt/gfs2/share** にある **csmb** と呼ばれる共有をエクスポートします。これは、**/etc/sysconfig/ctdb** の CTDB 設定ファイル内で **CTDB_RECOVERY_LOCK** パラメーターとして指定した **/mnt/ctdb/.ctdb.lock** の GFS2 共有ファイルシステムとは異なります。

この例では、初回マウント時に **/mnt/gfs2** 内に **share** ディレクトリを作成します。**clustering = yes** エントリは、Samba に CTDB を使用するよう指示します。**netbios name = csmb-server** エントリは、すべてのノードが NetBIOS の共通名を持つように明示的に設定します。**ea support** パラメーターは、拡張属性の使用を計画している場合に必要です。

smb.conf 設定ファイルは、全クラスターノードで全く同一でなければなりません。

Samba は **net conf** コマンドを使用したレジストリベースの設定も提供します。これにより、クラスターノード間で設定ファイルを手動でコピーしなくてもクラスターメンバー間で設定が自動的に一致するようにします。 **net conf** コマンドの詳細については、**net**(8) の man ページを参照してください。

11.6. CTDB と SAMBA サービスの起動

クラスターの起動後、「GFS2 の設定」 の説明のとおり、GFS2 ファイルシステムをマウントする必要があります。Samba **share** ディレクトリのパーミッション及びクラスターノードのユーザーアカウントを、クライアントアクセス用に設定することをお勧めします。

全ノードで次のコマンドを実行して、**ctdbd** デーモンを起動します。この例では CTDB で **CTDB_MANAGES_SAMBA=yes** と設定したため、CTDB は全ノードで Samba サービスを起動して、設定済みのすべての Samba 共有をエクスポートします。

```
[root@clusmb-01 ~]# service ctdb start
```

CTDB が Samba を起動し、共有をエクスポートし、安定化するには数分かかる場合があります。**ctdb status** を実行すると、以下のように CTDB のステータスが表示されます。

```
[root@clusmb-01 ~]# ctdb status
Number of nodes:3
pnn:0 192.168.1.151     OK (THIS NODE)
pnn:1 192.168.1.152     OK
pnn:2 192.168.1.153     OK
Generation:1410259202
Size:3
hash:0 lmaster:0
hash:1 lmaster:1
```

```
hash:2 lmaster:2
Recovery mode:NORMAL (0)
Recovery master:0
```

全ノードが "OK" になっていることを確認したら、「Clustered Samba サーバーの使用」 の説明のとおり、Clustered Samba サーバーを安全に使用することができます。

11.7. CLUSTERED SAMBA サーバーの使用

/etc/ctdb/public_addresses ファイルで指定した IP アドレスの 1 つに接続するか、以前設定した **csmb-server** DNS エントリを使用することにより、以下のようにクライアントはエクスポートされた Samba 共有に接続できます。

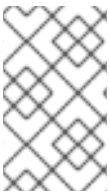
```
[root@clusmb-01 ~]# mount -t cifs //csmb-server/csmb /mnt/sambashare -o
user=testmonkey
```

又は

```
[user@clusmb-01 ~]$ smbclient //csmb-server/csmb
```

付録A フェンスデバイスパラメーター

本付録は、フェンスデバイスのパラメーターに関する詳細を表にまとめたものです。パラメーターの設定は、**luci** の使用、**ccs** コマンドの使用、又は **etc/cluster/cluster.conf** ファイルの編集により行うことができます。各フェンスエージェントのフェンスデバイスパラメーターの完全な一覧と詳細については、各エージェントの man ページを参照してください。



注記

フェンスデバイスの **Name（名前）** パラメーターは、Red Hat High Availability アドオンで使用されるデバイス用の任意の名前を指定します。これはデバイス用の DNS 名と同じではありません。

注記

一部のフェンスデバイスにはオプションの **Password Script** パラメーターがあります。**Password Script** パラメーターを使用すると、フェンスデバイスのパスワードが **Password** パラメーターからではなく、スクリプトから提供されるように指定できます。**Password Script** パラメーターは、**Password** パラメーターを上書きすることでクラスター設定ファイル (**/etc/cluster/cluster.conf**) 内のパスワードが可視化しないようにします。

表A.1「フェンスデバイスのサマリ」 には、フェンスデバイス、それらに関連するフェンスデバイスエージェント、フェンスデバイスのパラメーターについて文書化した表の参照が一覧表示されています。

表A.1 フェンスデバイスのサマリ

フェンスデバイス	フェンスエージェント	パラメーターの詳細についての参照
APC Power Switch (telnet/SSH)	fence_apc	表A.2「APC Power Switch (telnet/SSH)」
APC Power Switch over SNMP	fence_apc_snmp	表A.3「APC Power Switch over SNMP」
Brocade Fabrid Switch	fence_brocade	表A.4「Brocade Fabric Switch」
Cisco MDS	fence_cisco_mds	表A.5「Cisco MDS」
Cisco UCS	fence_cisco_ucs	表A.6「Cisco UCS」
Dell DRAC 5	fence_drac5	表A.7「Dell DRAC 5」

フェンスデバイス	フェンスエージェント	パラメーターの詳細についての参照
Eaton Network Power Switch (SNMP Interface)	fence_eaton_snmp	表A.8「Eaton Network Power Controller (SNMP Interface) (Red Hat Enterprise Linux 6.4 以降)」
Egenera SAN Controller	fence_egenera	表A.9「Egenera SAN Controller」
ePowerSwitch	fence_eps	表A.10「ePowerSwitch」
Fence virt	fence_virt	表A.11「Fence virt」
Fujitsu Siemens Remoteview Service Board (RSB)	fence_rsb	表A.12「Fujitsu Siemens Remoteview Service Board (RSB)」
HP BladeSystem	fence_hpblade	表A.13「HP BladeSystem (Red Hat Enterprise Linux 6.4 以降)」
HP iLO (Integrated Lights Out)、HP iLO2	fence_ilo、fence_ilo2	表A.14「HP iLO (Integrated Lights Out) および HP iLO2」
HP iLO (Integrated Lights Out) MP	fence_ilo_mp	表A.15「HP iLO (Integrated Lights Out) MP」
IBM BladeCenter	fence_bladecenter	表A.16「IBM BladeCenter」
IBM BladeCenter SNMP	fence_ibmblade	表A.17「IBM BladeCenter SNMP」
IBM iPDU	fence_ipdu	表A.18「IBM iPDU (Red Hat Enterprise Linux 6.4 以降)」
IF MIB	fence_ifmib	表A.19「IF MIB」
Intel Modular	fence_intelmodular	表A.20「Intel Modular」

フェンスデバイス	フェンスエージェント	パラメーターの詳細についての参照
IPMI (Intelligent Platform Management Interface) LAN、IBM Integrated Management Module、Dell iDRAC、HPiLO3、HPiLO4.	fence_ipmilan、fence_imm、fence_idrac、fence_ilo3、fence_ilo4	表A.21「IPMI (Intelligent Platform Management Interface) LAN、Dell iDrac、IBM Integrated Management Module、HPiLO3、HPiLO4」
RHEV-M REST API	fence_rhevm	表A.22「RHEV-M REST API (RHEL 6.2 以降及び RHEV 3.0 以降)」
SCSI Fencing	fence_scsi	表A.23「SCSI 保存フェンシング」
VMware Fencing (SOAP Interface)	fence_vmware_soap	表A.24「VMware フェンシング (SOAP インターフェース) (Red Hat Enterprise Linux 6.2 以降)」
WTI Power Switch	fence_wti	表A.25「WTI Power Switch」

表A.2「APC Power Switch (telnet/SSH)」 には、telnet/SSH による APC のフェンスエージェントである **fence_apc** により使用されるフェンスデバイスパラメーターが一覧表示されています。

表A.2 APC Power Switch (telnet/SSH)

luci フィールド	cluster.conf 属性	説明
Name	**name**	フェンスデーモンが telnet/ssh 経由でログインするクラスターに接続されている APC デバイスの名前
IP Address or Hostname	**ipaddr**	デバイスに割り当てられた IP アドレス、又はホスト名
IP Port (optional)	**ipport**	デバイスへの接続に使用する TCP ポート
Login	**login**	デバイスのアクセスに使用するログイン名
Password	**passwd**	デバイスへの接続の認証に使用するパスワード

luci フィールド	cluster.conf 属性	説明
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
Power Wait (seconds)	**power_wait**	power off 又は power on コマンドを発行後に待機する秒数
Power Timeout (seconds)	**power_timeout**	power off または power on コマンドを発行後にステータス変更をテストするまで待機する秒数。デフォルト値は 20 です。
Shell Timeout (seconds)	**shell_timeout**	コマンド発行後にコマンドプロンプトを待機する秒数。デフォルト値は 3 です。
Login Timeout (seconds)	**login_timeout**	ログイン後にコマンドプロンプトを待機する秒数。デフォルト値は 5 です。
Times to Retry Power On Operation	**retry_on**	power on 動作を再試行する回数。デフォルト値は 1 です。
Port	**port**	TCP ポート
Switch (optional)	**switch**	複数のデイジーチェーンで接続されたスイッチを使用する時に、ノードに接続する APC スイッチのスイッチ番号
Delay (optional)	**delay**	フェンシング開始まで待機する秒数。デフォルト値は 0 です。
Use SSH	**secure**	システムが SSH を使用してデバイスにアクセスすることを示します。SSH の使用時には、パスワード、パスワードスクリプト、ID ファイルのいずれかを指定する必要があります。
Path to SSH Identity File	**identity_file**	SSH の識別ファイル

表A.3「APC Power Switch over SNMP」 には、SNMP プロトコルを介して SNP デバイスにログインする API のフェンスエージェントである **fence_apc_snmp** により使用されるフェンスデバイスパラメーターが一覧表示されています。

表A.3 APC Power Switch over SNMP

luci フィールド	cluster.conf 属性	説明
Name	**name**	フェンスデーモンが SNMP プロトコルを介してログインするクラスターに接続されている APC デバイスの名前

luci フィールド	cluster.conf 属性	説明
IP Address or Hostname	**ipaddr**	デバイスに割り当てられた IP アドレス、又はホスト名
UDP/TCP port	**udpport**	デバイスへの接続に使用する UDP/TCP ポート。デフォルト値は 161 です。
Login	**login**	デバイスのアクセスに使用するログイン名
Password	**passwd**	デバイスへの接続の認証に使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
SNMP Version	**snmp_version**	使用する SNMP バージョン (1、2c、3)。デフォルト値は 1 です。
SNMP Community	**community**	SNMP コミュニティストリング。デフォルト値は **private** です。
SNMP Security Level	**snmp_sec_level**	SNMP セキュリティレベル (noAuthNoPriv、authNoPriv、authPriv)
SNMP Authentication Protocol	**snmp_auth_prot**	SNMP 認証プロトコル (MD5、SHA)
SNMP Privacy Protocol	**snmp_priv_prot**	SNMP プライバシープロトコル (DES、AES)
SNMP Privacy Protocol Password	**snmp_priv_passwd**	SNMP プライバシープロトコルパスワード
SNMP Privacy Protocol Script	**snmp_priv_passwd_script**	SNMP プライバシープロトコル用のパスワードを提供するスクリプト。これを使用すると、**SNMP プライバシープロトコルパスワード** のパラメーターを上書きします。
Power Wait (seconds)	**power_wait**	power off 又は power on コマンドを発行後に待機する秒数
Power Timeout (seconds)	**power_timeout**	power off または power on コマンドを発行後にステータス変更をテストするまで待機する秒数。デフォルト値は 20 です。
Shell Timeout (seconds)	**shell_timeout**	コマンド発行後にコマンドプロンプトを待機する秒数。デフォルト値は 3 です。

luci フィールド	cluster.conf 属性	説明
Login Timeout (seconds)	**login_timeout**	ログイン後にコマンドプロンプトを待機する秒数。デフォルト値は 5 です。
Times to Retry Power On Operation	**retry_on**	power on 動作を再試行する回数。デフォルト値は 1 です。
Port (Outlet) Number	**port**	TCP ポート
Delay (optional)	**delay**	フェンシング開始まで待機する秒数。デフォルト値は 0 です。

表A.4「Brocade Fabric Switch」 には、Brocade FC スイッチのフェンスエージェントである **fence_brocade** により使用されるフェンスデバイスパラメーターが一覧表示されています。

表A.4 Brocade Fabric Switch

luci フィールド	cluster.conf 属性	説明
Name	**name**	クラスターに接続された Brocade デバイスの名前
IP Address or Hostname	**ipaddr**	デバイスに割り当てられた IP アドレス
Login	**login**	デバイスのアクセスに使用するログイン名
Password	**passwd**	デバイスへの接続の認証に使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
Port	**port**	スイッチアウトレット番号

luci フィールド	cluster.conf 属性	説明
Unfencing	**unfence** section of the cluster configuration file	これを有効にすると、フェンス済みのノードは再起動されるまでは再度有効になりません。これは、パワーフェンス以外のメソッド (つまり、SAN/ストレージフェンシング) に必要となります。アンフェンシングを必要とするデバイスを設定する際には、最初にクラスターを停止し、デバイスおよびアンフェンシングを含むすべての設定をクラスターが開始される前に追加する必要があります。ノードのアンフェンシングに関する詳細は、**fence_node**(8) の man ページを参照してください。クラスター設定ファイルでのアンフェンシング設定に関する詳細情報は、「フェンシングの設定」 を参照してください。**ccs** コマンドによるアンフェンシング設定についての情報は、「単一のストレージベースのフェンスデバイスによるノードの設定」 を参照してください。

表A.5「Cisco MDS」 には、Cisco MDS のフェンスエージェントである **fence_cisco_mds** により使用されるフェンスデバイスパラメーターが一覧表示されています。

表A.5 Cisco MDS

luci フィールド	cluster.conf 属性	説明
Name	**name**	SNMP が有効になっている Cisco MDS 9000 シリーズデバイスの名前
IP Address or Hostname	**ipaddr**	デバイスに割り当てられた IP アドレス、又はホスト名
UDP/TCP port (optional)	**udpport**	デバイスへの接続に使用する UDP/TCP ポート。デフォルト値は 161 です。
Login	**login**	デバイスのアクセスに使用するログイン名
Password	**passwd**	デバイスへの接続の認証に使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
SNMP Version	**snmp_version**	使用する SNMP バージョン (1、2c、3)
SNMP Community	**community**	SNMP コミュニティストリング
SNMP Security Level	**snmp_sec_level**	SNMP セキュリティレベル (noAuthNoPriv、authNoPriv、authPriv)

luci フィールド	cluster.conf 属性	説明
SNMP Authentication Protocol	**snmp_auth_prot**	SNMP 認証プロトコル (MD5、SHA)
SNMP Privacy Protocol	**snmp_priv_prot**	SNMP プライバシープロトコル (DES、AES)
SNMP Privacy Protocol Password	**snmp_priv_passwd**	SNMP プライバシープロトコルパスワード
SNMP Privacy Protocol Script	**snmp_priv_passwd_script**	SNMP プライバシープロトコル用のパスワードを提供するスクリプト。これを使用すると、**SNMP プライバシープロトコルパスワード** のパラメーターを上書きします。
Power Wait (seconds)	**power_wait**	power off 又は power on コマンドを発行後に待機する秒数
Power Timeout (seconds)	**power_timeout**	power off または power on コマンドを発行後にステータス変更をテストするまで待機する秒数。デフォルト値は 20 です。
Shell Timeout (seconds)	**shell_timeout**	コマンド発行後にコマンドプロンプトを待機する秒数。デフォルト値は 3 です。
Login Timeout (seconds)	**login_timeout**	ログイン後にコマンドプロンプトを待機する秒数。デフォルト値は 5 です。
Times to Retry Power On Operation	**retry_on**	power on 動作を再試行する回数。デフォルト値は 1 です。
Port (Outlet) Number	**port**	TCP ポート
Delay (optional)	**delay**	フェンシング開始まで待機する秒数。デフォルト値は 0 です。

表A.6「Cisco UCS」 には、Cisco UCS のフェンスエージェントである **fence_cisco_ucs** により使用されるフェンスデバイスパラメーターが一覧表示されています。

表A.6 Cisco UCS

luci フィールド	cluster.conf 属性	説明
Name	**name**	Cisco UCS デバイス用の名前

luci フィールド	cluster.conf 属性	説明
IP Address or Hostname	**ipaddr**	デバイスに割り当てられた IP アドレス、又はホスト名
IP port (optional)	**ipport**	デバイスへの接続に使用する TCP ポート
Login	**login**	デバイスのアクセスに使用するログイン名
Password	**passwd**	デバイスへの接続の認証に使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
Use SSL	**ssl**	デバイスとの通信に SSL 接続を使用する
Sub-Organization	**suborg**	サブ組織へのアクセスに必要な追加のパス
Power Wait (seconds)	**power_wait**	power off 又は power on コマンドを発行後に待機する秒数
Power Timeout (seconds)	**power_timeout**	power off または power on コマンドを発行後にステータス変更をテストするまで待機する秒数。デフォルト値は 20 です。
Shell Timeout (seconds)	**shell_timeout**	コマンド発行後にコマンドプロンプトを待機する秒数。デフォルト値は 3 です。
Login Timeout (seconds)	**login_timeout**	ログイン後にコマンドプロンプトを待機する秒数。デフォルト値は 5 です。
Times to Retry Power On Operation	**retry_on**	power on 動作を再試行する回数。デフォルト値は 1 です。
Port (Outlet) Number	**port**	仮想マシンの名前
Delay (optional)	**delay**	フェンシング開始まで待機する秒数。デフォルト値は 0 です。

表A.7「Dell DRAC 5」 には、Dell DRAC 5 のフェンスエージェントである **fence_drac5** により使用されるフェンスデバイスパラメーターが一覧表示されています。

表A.7 Dell DRAC 5

luci フィールド	cluster.conf 属性	説明
Name	**name**	DRAC に割り当てられた名前
IP Address or Hostname	**ipaddr**	DRAC に割り当てられた IP アドレス、又はホスト名
IP Port (optional)	**ipport**	デバイスへの接続に使用する TCP ポート
Login	**login**	DRAC へのアクセスに使用するログイン名
Password	**passwd**	DRAC への接続を認証するために使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
Use SSH	**secure**	システムが SSH を使用してデバイスにアクセスすることを示します。SSH の使用時には、パスワード、パスワードスクリプト、ID ファイルのいずれかを指定する必要があります。
Path to SSH Identity File	**identity_file**	SSH の識別ファイル
Module Name	**module_name**	複数の DRAC モジュールを使用する時の DRAC のモジュール名 (オプション)
Force Command Prompt	**cmd_prompt**	使用するコマンドプロンプト。デフォルト値は '\$' です。
Power Wait (seconds)	**power_wait**	power off 又は power on コマンドを発行後に待機する秒数
Delay (seconds)	**delay**	フェンシング開始まで待機する秒数。デフォルト値は 0 です。
Power Timeout (seconds)	**power_timeout**	power off または power on コマンドを発行後にステータス変更をテストするまで待機する秒数。デフォルト値は 20 です。
Shell Timeout (seconds)	**shell_timeout**	コマンド発行後にコマンドプロンプトを待機する秒数。デフォルト値は 3 です。
Login Timeout (seconds)	**login_timeout**	ログイン後にコマンドプロンプトを待機する秒数。デフォルト値は 5 です。

luci フィールド	cluster.conf 属性	説明
Times to Retry Power On Operation	**retry_on**	power on 動作を再試行する回数。デフォルト値は 1 です。

表A.8「Eaton Network Power Controller (SNMP Interface) (Red Hat Enterprise Linux 6.4 以降)」 は、Eaton over SNMP ネットワークパワースイッチ向けのフェンスエージェント **fence_eaton_snmp** が使用するフェンスデバイスパラメーターを記載しています。

表A.8 Eaton Network Power Controller (SNMP Interface) (Red Hat Enterprise Linux 6.4 以降)

luci フィールド	cluster.conf 属性	説明
Name	**name**	クラスターに接続された Eaton ネットワークパワースイッチの名前
IP Address or Hostname	**ipaddr**	デバイスに割り当てられた IP アドレス、又はホスト名
UDP/TCP Port (optional)	**udpport**	デバイスへの接続に使用する UDP/TCP ポート。デフォルト値は 161 です。
Login	**login**	デバイスのアクセスに使用するログイン名
Password	**passwd**	デバイスへの接続の認証に使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
SNMP Version	**snmp_version**	使用する SNMP バージョン (1、2c、3)。デフォルト値は 1 です。
SNMP Community	**community**	SNMP コミュニティストリング。デフォルト値は **private** です。
SNMP Security Level	**snmp_sec_level**	SNMP セキュリティレベル (noAuthNoPriv、authNoPriv、authPriv)
SNMP Authentication Protocol	**snmp_auth_prot**	SNMP 認証プロトコル (MD5、SHA)
SNMP Privacy Protocol	**snmp_priv_prot**	SNMP プライバシープロトコル (DES、AES)

luci フィールド	cluster.conf 属性	説明
SNMP Privacy Protocol Password	**snmp_priv_passwd**	SNMP プライバシープロトコルパスワード
SNMP Privacy Protocol Script	**snmp_priv_passwd_script**	SNMP プライバシープロトコル用のパスワードを提供するスクリプト。これを使用すると、**SNMP プライバシープロトコルパスワード** のパラメーターを上書きします。
Power wait (seconds)	**power_wait**	power off 又は power on コマンドを発行後に待機する秒数
Power Timeout (seconds)	**power_timeout**	power off または power on コマンドを発行後にステータス変更をテストするまで待機する秒数。デフォルト値は 20 です。
Shell Timeout (seconds)	**shell_timeout**	コマンド発行後にコマンドプロンプトを待機する秒数。デフォルト値は 3 です。
Login Timeout (seconds)	**login_timeout**	ログイン後にコマンドプロンプトを待機する秒数。デフォルト値は 5 です。
Times to Retry Power On Operation	**retry_on**	power on 動作を再試行する回数。デフォルト値は 1 です。
Port (Outlet) Number	**port**	仮想マシンの物理的なプラグ番号または名前。このパラメーターは常に必須となっています。
Delay (optional)	**delay**	フェンシング開始まで待機する秒数。デフォルト値は 0 です。

表A.9「Egenera SAN Controller」 には、Egenera BladeFrame のフェンスエージェントである **fence_egenera** により使用されるフェンスデバイスパラメーターが一覧表示されています。

表A.9 Egenera SAN Controller

luci フィールド	cluster.conf 属性	説明
Name	**name**	クラスターに接続されている Egenera BladeFrame デバイスの名前
CServer	**cserver**	デバイスに割り当てられているホスト名 (及び、オプションとして **username@hostname** 形式のユーザー名)。 詳細情報は、fence_egenera(8) の man ページを参照してください。
ESH Path (optional)	**esh**	cserver 上の esh コマンドへのパス (デフォルトは /opt/panmgr/bin/esh です)。

luci フィールド	cluster.conf 属性	説明
Username	**user**	ログイン名。デフォルト値は **root** です。
lpan	**lpan**	デバイスの論理プロセスエリアネットワーク (LPAN)
pserver	**pserver**	デバイスのプロセッシングブレード (pserver) の名前
Delay (optional)	**delay**	フェンシング開始まで待機する秒数。デフォルト値は 0 です。
Unfencing	**unfence** section of the cluster configuration file	これを有効にすると、フェンス済みのノードは再起動されるまでは再度有効になりません。これは、パワーフェンス以外のメソッド (つまり、SAN/ストレージフェンシング) に必要となります。アンフェンシングを必要とするデバイスを設定する際には、最初にクラスターを停止し、デバイスおよびアンフェンシングを含むすべての設定をクラスターが開始される前に追加する必要があります。ノードのアンフェンシングに関する詳細は、**fence_node**(8) の man ページを参照してください。クラスター設定ファイルでのアンフェンシング設定に関する詳細情報は、「フェンシングの設定」 を参照してください。**ccs** コマンドによるアンフェンシング設定についての情報は、「単一のストレージベースのフェンスデバイスによるノードの設定」 を参照してください。

表A.10「ePowerSwitch」 には、ePowerSwitch のフェンスエージェントである **fence_eps** により使用されるフェンスデバイスパラメーターが一覧表示されています。

表A.10 ePowerSwitch

luci フィールド	cluster.conf 属性	説明
Name	**name**	クラスターに接続されている ePowerSwitch デバイスの名前
IP Address or Hostname	**ipaddr**	デバイスに割り当てられた IP アドレス、又はホスト名
Login	**login**	デバイスのアクセスに使用するログイン名
Password	**passwd**	デバイスへの接続の認証に使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
Name of Hidden Page	**hidden_page**	デバイス用の非表示ページの名前

luci フィールド	cluster.conf 属性	説明
Times to Retry Power On Operation	**retry_on**	power on 動作を再試行する回数。デフォルト値は 1 です。
Port (Outlet) Number	**port**	仮想マシンの物理的なプラグ番号又は名前
Delay (optional)	**delay**	フェンシング開始まで待機する秒数。デフォルト値は 0 です。

表A.11「Fence virt」 には、Fence virt フェンスデバイスのフェンスエージェントである **fence_virt** により使用されるフェンスデバイスパラメーターが一覧表示されています。

表A.11 Fence virt

luci フィールド	cluster.conf 属性	説明
Name	**name**	Fence virt フェンスデバイスの名前
Serial Device	**serial_device**	ホスト上では、シリアルデバイスは各ドメインの設定ファイル内にマップされる必要があります。詳細情報は、**fence_virt.conf** の man ページをご覧ください。このフィールドを指定した場合は、**fence_virt** フェンシングエージェントがシリアルモードで実行するようになります。値を指定しないと、**fence_virt** フェンシングエージェントが VM チャンネルモードで実行するようになります。
Serial Parameters	**serial_params**	シリアルパラメーター。デフォルトは 115200, 8N1 です。
VM Channel IP Address	**channel_address**	チャンネル IP。デフォルト値は 10.0.2.179 です。
Port or Domain (deprecated)	**port**	フェンスする仮想マシン (ドメイン UUID 又は名前)
	ipport	チャンネルポート。デフォルト値は 1229 で、**luci** でこのフェンスデバイスを設定する時に使用される値です。

表A.12「Fujitsu Siemens Remoteview Service Board (RSB)」 には、Fujitsu-Siemens RSB のフェンスエージェントである **fence_rsb** により使用されるフェンスデバイスパラメーターが一覧表示されています。

表A.12 Fujitsu Siemens Remoteview Service Board (RSB)

luci フィールド	cluster.conf 属性	説明
Name	**name**	フェンスデバイスとして使用する RSB の名前
IP Address or Hostname	**ipaddr**	デバイスに割り当てられたホスト名
Login	**login**	デバイスのアクセスに使用するログイン名
Password	**passwd**	デバイスへの接続の認証に使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
TCP Port	**ipport**	telnet サービスがリッスンするポート番号。デフォルト値は 3172 です。
Force Command Prompt	**cmd_prompt**	使用するコマンドプロンプト。デフォルト値は '\$' です。
Power Wait (seconds)	**power_wait**	power off 又は power on コマンドを発行後に待機する秒数
Delay (seconds)	**delay**	フェンシング開始まで待機する秒数。デフォルト値は 0 です。
Power Timeout (seconds)	**power_timeout**	power off または power on コマンドを発行後にステータス変更をテストするまで待機する秒数。デフォルト値は 20 です。
Shell Timeout (seconds)	**shell_timeout**	コマンド発行後にコマンドプロンプトを待機する秒数。デフォルト値は 3 です。
Login Timeout (seconds)	**login_timeout**	ログイン後にコマンドプロンプトを待機する秒数。デフォルト値は 5 です。
Times to Retry Power On Operation	**retry_on**	power on 動作を再試行する回数。デフォルト値は 1 です。

表A.13「HP BladeSystem (Red Hat Enterprise Linux 6.4 以降)」 には、HP Bladesystem のフェンスエージェント **fence_hpblade** により使用されるフェンスデバイスパラメーターが記載されています。

表A.13 HP BladeSystem (Red Hat Enterprise Linux 6.4 以降)

luci フィールド	cluster.conf 属性	説明
Name	**name**	クラスターに接続された HP Bladesystem デバイスに割り当てられた名前
IP Address or Hostname	**ipaddr**	HP BladeSystem デバイスに割り当てられた IP アドレスまたはホスト名
IP Port (optional)	**ipport**	デバイスへの接続に使用する TCP ポート
Login	**login**	HP BladeSystem デバイスにアクセスする際に使用するログイン名。このパラメーターは必須です。
Password	**passwd**	フェンスデバイスへの接続認証に使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
Force Command Prompt	**cmd_prompt**	使用するコマンドプロンプト。デフォルト値は '\$' です。
Missing port returns OFF instead of failure	**missing_as_off**	ポートがない場合に、問題が発生する代わりに OFF を返します。
Power Wait (seconds)	**power_wait**	power off 又は power on コマンドを発行後に待機する秒数
Power Timeout (seconds)	**power_timeout**	power off または power on コマンドを発行後にステータス変更をテストするまで待機する秒数。デフォルト値は 20 です。
Shell Timeout (seconds)	**shell_timeout**	コマンド発行後にコマンドプロンプトを待機する秒数。デフォルト値は 3 です。
Login Timeout (seconds)	**login_timeout**	ログイン後にコマンドプロンプトを待機する秒数。デフォルト値は 5 です。
Times to Retry Power On Operation	**retry_on**	power on 動作を再試行する回数。デフォルト値は 1 です。
Use SSH	**secure**	システムが SSH を使用してデバイスにアクセスすることを示します。SSH の使用時には、パスワード、パスワードスクリプト、ID ファイルのいずれかを指定する必要があります。

luci フィールド	cluster.conf 属性	説明
Path to SSH Identity File	**identity_file**	SSH の識別ファイル

HP iLO デバイス **fence_ilo** および HP iLO2 デバイス **fence_ilo2** のフェンスエージェントは、同一実装を共有します。表A.14「HP iLO (Integrated Lights Out) および HP iLO2」 は、これらのエージェントが使用するフェンスデバイスパラメーターを一覧表示しています。

表A.14 HP iLO (Integrated Lights Out) および HP iLO2

luci フィールド	cluster.conf 属性	説明
Name	**name**	HP iLO サポートがあるサーバーの名前
IP Address or Hostname	**ipaddr**	デバイスに割り当てられた IP アドレス、又はホスト名
IP Port (optional)	**ipport**	デバイスへの接続に使用する TCP ポート
Login	**login**	デバイスのアクセスに使用するログイン名
Password	**passwd**	デバイスへの接続の認証に使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
Power Wait (seconds)	**power_wait**	power off 又は power on コマンドを発行後に待機する秒数
Delay (seconds)	**delay**	フェンシング開始まで待機する秒数。デフォルト値は 0 です。
Power Timeout (seconds)	**power_timeout**	power off または power on コマンドを発行後にステータス変更をテストするまで待機する秒数。デフォルト値は 20 です。
Shell Timeout (seconds)	**shell_timeout**	コマンド発行後にコマンドプロンプトを待機する秒数。デフォルト値は 3 です。
Login Timeout (seconds)	**login_timeout**	ログイン後にコマンドプロンプトを待機する秒数。デフォルト値は 5 です。
Times to Retry Power On Operation	**retry_on**	power on 動作を再試行する回数。デフォルト値は 1 です。

表A.15「HP iLO (Integrated Lights Out) MP」 には、HP iLO MP デバイスのフェンスエージェントである **fence_ilo_mp** により使用されるフェンスデバイスパラメーターが一覧表示されています。

表A.15 HP iLO (Integrated Lights Out) MP

luci フィールド	cluster.conf 属性	説明
Name	**name**	HP iLO サポートがあるサーバーの名前
IP Address or Hostname	**ipaddr**	デバイスに割り当てられた IP アドレス、又はホスト名
IP Port (optional)	**ipport**	デバイスへの接続に使用する TCP ポート
Login	**login**	デバイスのアクセスに使用するログイン名
Password	**passwd**	デバイスへの接続の認証に使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
Use SSH	**secure**	システムが SSH を使用してデバイスにアクセスすることを示します。SSH の使用時には、パスワード、パスワードスクリプト、ID ファイルのいずれかを指定する必要があります。
Path to SSH Identity File	**identity_file**	SSH の識別ファイル
Force Command Prompt	**cmd_prompt**	使用するコマンドプロンプト。デフォルト値は 'MP>', 'hpiLO->' です。
Power Wait (seconds)	**power_wait**	power off 又は power on コマンドを発行後に待機する秒数
Delay (seconds)	**delay**	フェンシング開始まで待機する秒数。デフォルト値は 0 です。
Power Timeout (seconds)	**power_timeout**	power off または power on コマンドを発行後にステータス変更をテストするまで待機する秒数。デフォルト値は 20 です。
Shell Timeout (seconds)	**shell_timeout**	コマンド発行後にコマンドプロンプトを待機する秒数。デフォルト値は 3 です。
Login Timeout (seconds)	**login_timeout**	ログイン後にコマンドプロンプトを待機する秒数。デフォルト値は 5 です。

luci フィールド	cluster.conf 属性	説明
Times to Retry Power On Operation	**retry_on**	power on 動作を再試行する回数。デフォルト値は 1 です。

表A.16「IBM BladeCenter」 には、IBM BladeCenter のフェンスエージェントである **fence_bladecenter** により使用されるフェンスデバイスパラメーターが一覧表示されています。

表A.16 IBM BladeCenter

luci フィールド	cluster.conf 属性	説明
Name	**name**	クラスターに接続された IBM BladeCenter デバイスの名前
IP Address or Hostname	**ipaddr**	デバイスに割り当てられた IP アドレス、又はホスト名
IP port (optional)	**ipport**	デバイスへの接続に使用する TCP ポート
Login	**login**	デバイスのアクセスに使用するログイン名
Password	**passwd**	デバイスへの接続の認証に使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
Power Wait (seconds)	**power_wait**	power off 又は power on コマンドを発行後に待機する秒数
Power Timeout (seconds)	**power_timeout**	power off または power on コマンドを発行後にステータス変更をテストするまで待機する秒数。デフォルト値は 20 です。
Shell Timeout (seconds)	**shell_timeout**	コマンド発行後にコマンドプロンプトを待機する秒数。デフォルト値は 3 です。
Login Timeout (seconds)	**login_timeout**	ログイン後にコマンドプロンプトを待機する秒数。デフォルト値は 5 です。
Times to Retry Power On Operation	**retry_on**	power on 動作を再試行する回数。デフォルト値は 1 です。

luci フィールド	cluster.conf 属性	説明
Use SSH	**secure**	システムが SSH を使用してデバイスにアクセスすることを示します。SSH の使用時には、パスワード、パスワードスクリプト、ID ファイルのいずれかを指定する必要があります。
Path to SSH Identity File	**identity_file**	SSH の識別ファイル

表A.17「IBM BladeCenter SNMP」 には、SNMP による IBM BladeCenter のフェンスエージェントである **fence_ibmblade** により使用されるフェンスデバイスパラメーターが一覧表示されています。

表A.17 IBM BladeCenter SNMP

luci フィールド	cluster.conf 属性	説明
Name	**name**	クラスターに接続された IBM BladeCenter SNMP デバイスの名前
IP Address or Hostname	**ipaddr**	デバイスに割り当てられた IP アドレス、又はホスト名
UDP/TCP Port (optional)	**udpport**	デバイスへの接続に使用する UDP/TCP ポート。デフォルト値は 161 です。
Login	**login**	デバイスのアクセスに使用するログイン名
Password	**passwd**	デバイスへの接続の認証に使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
SNMP Version	**snmp_version**	使用する SNMP バージョン (1、2c、3)。デフォルト値は 1 です。
SNMP Community	**community**	SNMP コミュニティストリング
SNMP Security Level	**snmp_sec_level**	SNMP セキュリティレベル (noAuthNoPriv、authNoPriv、authPriv)
SNMP Authentication Protocol	**snmp_auth_prot**	SNMP 認証プロトコル (MD5、SHA)
SNMP Privacy Protocol	**snmp_priv_prot**	SNMP プライバシープロトコル (DES、AES)

luci フィールド	cluster.conf 属性	説明
SNMP privacy protocol password	snmp_priv_passwd	SNMP プライバシープロトコルパスワード
SNMP Privacy Protocol Script	snmp_priv_passwd_script	SNMP プライバシープロトコル用のパスワードを提供するスクリプト。これを使用すると、**SNMP プライバシープロトコルパスワード** のパラメーターを上書きします。
Power Wait (seconds)	power_wait	power off 又は power on コマンドを発行後に待機する秒数
Power Timeout (seconds)	power_timeout	power off または power on コマンドを発行後にステータス変更をテストするまで待機する秒数。デフォルト値は 20 です。
Shell Timeout (seconds)	shell_timeout	コマンド発行後にコマンドプロンプトを待機する秒数。デフォルト値は 3 です。
Login Timeout (seconds)	login_timeout	ログイン後にコマンドプロンプトを待機する秒数。デフォルト値は 5 です。
Times to Retry Power On Operation	retry_on	power on 動作を再試行する回数。デフォルト値は 1 です。
Port (Outlet) Number	port	仮想マシンの物理的なプラグ番号又は名前
Delay (optional)	delay	フェンシング開始まで待機する秒数。デフォルト値は 0 です。

表A.18「IBM iPDU (Red Hat Enterprise Linux 6.4 以降)」 には、iPDU over SNMP デバイスのフェンスエージェント **fence_ipdu** で使用されるフェンスデバイスパラメーターを記載しています。

表A.18 IBM iPDU (Red Hat Enterprise Linux 6.4 以降)

luci フィールド	cluster.conf 属性	説明
Name	name	フェンスデーモンが SNMP プロトコルを介してログインするクラスターに接続されている IBM iPDU デバイスの名前
IP Address or Hostname	ipaddr	デバイスに割り当てられた IP アドレス、又はホスト名
UDP/TCP Port	udpport	デバイスへの接続に使用する UDP/TCP ポート。デフォルト値は 161 です。

luci フィールド	cluster.conf 属性	説明
Login	**login**	デバイスのアクセスに使用するログイン名
Password	**passwd**	デバイスへの接続の認証に使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
SNMP Version	**snmp_version**	使用する SNMP バージョン (1、2c、3)。デフォルト値は 1 です。
SNMP Community	**community**	SNMP コミュニティストリング。デフォルト値は **private** です。
SNMP Security Level	**snmp_sec_level**	SNMP セキュリティレベル (noAuthNoPriv、authNoPriv、authPriv)
SNMP Authentication Protocol	**snmp_auth_prot**	SNMP 認証プロトコル (MD5、SHA)
SNMP Privacy Protocol	**snmp_priv_prot**	SNMP プライバシープロトコル (DES、AES)
SNMP Privacy Protocol Password	**snmp_priv_passwd**	SNMP プライバシープロトコルパスワード
SNMP Privacy Protocol Script	**snmp_priv_passwd_script**	SNMP プライバシープロトコル用のパスワードを提供するスクリプト。これを使用すると、**SNMP プライバシープロトコルパスワード** のパラメーターを上書きします。
Power Wait (seconds)	**power_wait**	power off 又は power on コマンドを発行後に待機する秒数
Power Timeout (seconds)	**power_timeout**	power off または power on コマンドを発行後にステータス変更をテストするまで待機する秒数。デフォルト値は 20 です。
Shell Timeout (seconds)	**shell_timeout**	コマンド発行後にコマンドプロンプトを待機する秒数。デフォルト値は 3 です。
Login Timeout (seconds)	**login_timeout**	ログイン後にコマンドプロンプトを待機する秒数。デフォルト値は 5 です。
Times to Retry Power On Operation	**retry_on**	power on 動作を再試行する回数。デフォルト値は 1 です。

luci フィールド	cluster.conf 属性	説明
Port (Outlet) Number	**port**	仮想マシンの物理的なプラグ番号又は名前
Delay (optional)	**delay**	フェンシング開始まで待機する秒数。デフォルト値は 0 です。

表A.19「IF MIB」 には、IF-MIB デバイスのフェンスエージェントである **fence_ifmib** により使用されるフェンスデバイスパラメーターが一覧表示されています。

表A.19 IF MIB

luci フィールド	cluster.conf 属性	説明
Name	**name**	クラスターに接続された IF MIB デバイスの名前
IP Address or Hostname	**ipaddr**	デバイスに割り当てられた IP アドレス、又はホスト名
UDP/TCP Port (optional)	**udpport**	デバイスへの接続に使用する UDP/TCP ポート。デフォルト値は 161 です。
Login	**login**	デバイスのアクセスに使用するログイン名
Password	**passwd**	デバイスへの接続の認証に使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
SNMP Version	**snmp_version**	使用する SNMP バージョン (1、2c、3)。デフォルト値は 1 です。
SNMP Community	**community**	SNMP コミュニティストリング
SNMP Security Level	**snmp_sec_level**	SNMP セキュリティレベル (noAuthNoPriv、authNoPriv、authPriv)
SNMP Authentication Protocol	**snmp_auth_prot**	SNMP 認証プロトコル (MD5、SHA)

luci フィールド	cluster.conf 属性	説明
SNMP Privacy Protocol	snmp_priv_prot	SNMP プライバシープロトコル (DES、AES)
SNMP Privacy Protocol Password	snmp_priv_passwd	SNMP プライバシープロトコルパスワード
SNMP Privacy Protocol Script	snmp_priv_passwd_script	SNMP プライバシープロトコル用のパスワードを提供するスクリプト。これを使用すると、**SNMP プライバシープロトコルパスワード** のパラメーターを上書きします。
Power Wait (seconds)	power_wait	power off 又は power on コマンドを発行後に待機する秒数
Power Timeout (seconds)	power_timeout	power off または power on コマンドを発行後にステータス変更をテストするまで待機する秒数。デフォルト値は 20 です。
Shell Timeout (seconds)	shell_timeout	コマンド発行後にコマンドプロンプトを待機する秒数。デフォルト値は 3 です。
Login Timeout (seconds)	login_timeout	ログイン後にコマンドプロンプトを待機する秒数。デフォルト値は 5 です。
Times to Retry Power On Operation	retry_on	power on 動作を再試行する回数。デフォルト値は 1 です。
Port (Outlet) Number	port	仮想マシンの物理的なプラグ番号又は名前
Delay (optional)	delay	フェンシング開始まで待機する秒数。デフォルト値は 0 です。

表A.20「Intel Modular」 には、Intel Modular のフェンスエージェントである **fence_intelmodular** により使用されるフェンスデバイスパラメーターが一覧表示されています。

表A.20 Intel Modular

luci フィールド	cluster.conf 属性	説明
Name	name	クラスターに接続された Intel Modular デバイスの名前
IP Address or Hostname	ipaddr	デバイスに割り当てられた IP アドレス、又はホスト名

luci フィールド	cluster.conf 属性	説明
UDP/TCP Port (optional)	**udpport**	デバイスへの接続に使用する UDP/TCP ポート。デフォルト値は 161 です。
Login	**login**	デバイスのアクセスに使用するログイン名
Password	**passwd**	デバイスへの接続の認証に使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
SNMP Version	**snmp_version**	使用する SNMP バージョン (1、2c、3)。デフォルト値は 1 です。
SNMP Community	**community**	SNMP コミュニティストリング。デフォルト値は **private** です。
SNMP Security Level	**snmp_sec_level**	SNMP セキュリティレベル (noAuthNoPriv、authNoPriv、authPriv)
SNMP Authentication Protocol	**snmp_auth_prot**	SNMP 認証プロトコル (MD5、SHA)
SNMP Privacy Protocol	**snmp_priv_prot**	SNMP プライバシープロトコル (DES、AES)
SNMP Privacy Protocol Password	**snmp_priv_passwd**	SNMP プライバシープロトコルパスワード
SNMP Privacy Protocol Script	**snmp_priv_passwd_script**	SNMP プライバシープロトコル用のパスワードを提供するスクリプト。これを使用すると、**SNMP プライバシープロトコルパスワード** のパラメーターを上書きします。
Power Wait (seconds)	**power_wait**	power off 又は power on コマンドを発行後に待機する秒数
Power Timeout (seconds)	**power_timeout**	power off または power on コマンドを発行後にステータス変更をテストするまで待機する秒数。デフォルト値は 20 です。
Shell Timeout (seconds)	**shell_timeout**	コマンド発行後にコマンドプロンプトを待機する秒数。デフォルト値は 3 です。
Login Timeout (seconds)	**login_timeout**	ログイン後にコマンドプロンプトを待機する秒数。デフォルト値は 5 です。

luci フィールド	cluster.conf 属性	説明
Times to Retry Power On Operation	**retry_on**	power on 動作を再試行する回数。デフォルト値は 1 です。
Port (Outlet) Number	**port**	仮想マシンの物理的なプラグ番号又は名前
Delay (optional)	**delay**	フェンシング開始まで待機する秒数。デフォルト値は 0 です。

IPMI over LAN (**fence_ipmilan**)、Dell iDRAC (**fence_idrac**)、IBM Integrated Management Module (**fence_imm**)、HP iLO3 デバイス **fence_ilo3**、および HP iLO4 デバイス **fence_ilo4** のフェンスエージェントは、同一実装を共有します。表A.21「IPMI (Intelligent Platform Management Interface) LAN、Dell iDrac、IBM Integrated Management Module、HPiLO3、HPiLO4」 は、これらのエージェントが使用するフェンスデバイスパラメーターを一覧表示しています。

表A.21 IPMI (Intelligent Platform Management Interface) LAN、Dell iDrac、IBM Integrated Management Module、HPiLO3、HPiLO4

luci フィールド	cluster.conf 属性	説明
Name	**name**	クラスターに接続されるフェンスデバイスの名前
IP Address or Hostname	**ipaddr**	デバイスに割り当てられた IP アドレス、又はホスト名
Login	**login**	所定のポートに対し power on/off コマンドを発行できるユーザーのログイン名
Password	**passwd**	ポートへの接続の認証に使用されるパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
Authentication Type	**auth**	認証タイプ: **none**、**password** または **MD5**
Use Lanplus	**lanplus**	**True** または **1**。空白の場合、値は **False** になります。ハードウェアが対応している場合、Lanplus を有効にして接続のセキュリティを高めることが推奨されます。
Ciphersuite to use	**cipher**	IPMIv2 lanplus 接続用に使用するリモートサーバー認証、整合性、及び暗号化アルゴリズム

luci フィールド	cluster.conf 属性	説明
Privilege level	**privlvl**	デバイスの権限レベル
IPMI Operation Timeout	**timeout**	IPMI オペレーションのタイムアウト (秒単位)
Power Wait (seconds)	**power_wait**	power off 又は power on コマンドを発行後に待機する秒数
Delay (optional)	**delay**	フェンシング開始まで待機する秒数。デフォルト値は 0 です。

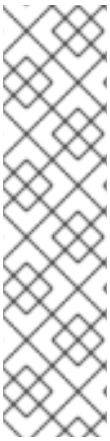
表A.22「RHEV-M REST API (RHEL 6.2 以降及び RHEV 3.0 以降)」 には、RHEV-M REST API のフェンスエージェントである **fence_rhevm** により使用されるフェンスデバイスパラメーターが一覧表示されています。

表A.22 RHEV-M REST API (RHEL 6.2 以降及び RHEV 3.0 以降)

luci フィールド	cluster.conf 属性	説明
Name	**name**	RHEV-M REST API フェンシングデバイス用の名前
IP Address or Hostname	**ipaddr**	デバイスに割り当てられた IP アドレス、又はホスト名
IP Port (optional)	**ipport**	デバイスへの接続に使用する TCP ポート
Login	**login**	デバイスのアクセスに使用するログイン名
Password	**passwd**	デバイスへの接続の認証に使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
Use SSL	**ssl**	デバイスとの通信に SSL 接続を使用する
Power Wait (seconds)	**power_wait**	power off 又は power on コマンドを発行後に待機する秒数
Power Timeout (seconds)	**power_timeout**	power off または power on コマンドを発行後にステータス変更をテストするまで待機する秒数。デフォルト値は 20 です。
Shell Timeout (seconds)	**shell_timeout**	コマンド発行後にコマンドプロンプトを待機する秒数。デフォルト値は 3 です。

luci フィールド	cluster.conf 属性	説明
Login Timeout (seconds)	**login_timeout**	ログイン後にコマンドプロンプトを待機する秒数。デフォルト値は 5 です。
Times to Retry Power On Operation	**retry_on**	power on 動作を再試行する回数。デフォルト値は 1 です。
Port (Outlet) Number	**port**	仮想マシンの物理的なプラグ番号又は名前
Delay (optional)	**delay**	フェンシング開始まで待機する秒数。デフォルト値は 0 です。

表A.23「SCSI 保存フェンシング」 には、SCSI 永続予約のフェンスエージェントである **fence_scsi** により使用されるフェンスデバイスパラメーターが一覧表示されています。



注記

フェンスメソッドとして SCSI 永続予約の使用には、以下の制限が伴います。

- SCSI フェンシングの使用時は、クラスター内の全てのノードは同じデバイスに登録する必要があります。それにより、各ノードが登録されているすべてのデバイスから他のノードの登録キーを削除できます。
- クラスターボリュームに使用されるデバイスは、パーティションではなく、完全な LUN でなければなりません。SCSI 永続予約は LUN 全体で機能するため、アクセスは個別のパーティションではなく各 LUN に対し制御されることになります。

表A.23 SCSI 保存フェンシング

luci フィールド	cluster.conf 属性	説明
Name	**name**	SCSI フェンスデバイスの名前
Unfencing	**unfence** section of the cluster configuration file	これを有効にすると、フェンス済みのノードは再起動されるまでは再度有効になりません。これは、パワーフェンス以外のメソッド (つまり、SAN/ストレージフェンシング) に必要となります。アンフェンシングを必要とするデバイスを設定する際には、最初にクラスターを停止し、デバイスおよびアンフェンシングを含むすべての設定をクラスターが開始される前に追加する必要があります。ノードのアンフェンシングに関する詳細は、**fence_node**(8) の man ページを参照してください。クラスター設定ファイルでのアンフェンシング設定に関する詳細情報は、「フェンシングの設定」 を参照してください。**ccs** コマンドによるアンフェンシング設定についての情報は、「単一のストレージベースのフェンスデバイスによるノードの設定」 を参照してください。

luci フィールド	cluster.conf 属性	説明
Node name	**nodename**	このノード名を使って、現行オペレーションに使用するキー値を生成します。
Key for current action	**key**	(ノード名に優先) 現行オペレーションに使用するキー。このキーはノードに対して一意であること。"on" アクションに対しては、キーはローカルノードを登録するキー使用を指定します。"off" アクションに対しては、デバイスから削除されるキーを指定します。
Delay (optional)	**delay**	フェンシング開始まで待機する秒数。デフォルト値は 0 です。

表A.24「VMware フェンシング (SOAP インターフェース) (Red Hat Enterprise Linux 6.2 以降)」 には、SOAP API による VMWare のフェンスエージェントである **fence_vmware_soap** により使用されるフェンスデバイスパラメーターが一覧表示されています。

表A.24 VMware フェンシング (SOAP インターフェース) (Red Hat Enterprise Linux 6.2 以降)

luci フィールド	cluster.conf 属性	説明
Name	**name**	仮想マシンフェンシングデバイスの名前
IP Address or Hostname	**ipaddr**	デバイスに割り当てられた IP アドレス、又はホスト名
IP Port (optional)	**ipport**	デバイスへの接続に使用する TCP ポート
Login	**login**	デバイスのアクセスに使用するログイン名
Password	**passwd**	デバイスへの接続の認証に使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
Power Wait (seconds)	**power_wait**	power off 又は power on コマンドを発行後に待機する秒数
Power Timeout (seconds)	**power_timeout**	power off または power on コマンドを発行後にステータス変更をテストするまで待機する秒数。デフォルト値は 20 です。
Shell Timeout (seconds)	**shell_timeout**	コマンド発行後にコマンドプロンプトを待機する秒数。デフォルト値は 3 です。

luci フィールド	cluster.conf 属性	説明
Login Timeout (seconds)	**login_timeout**	ログイン後にコマンドプロンプトを待機する秒数。デフォルト値は 5 です。
Times to Retry Power On Operation	**retry_on**	power on 動作を再試行する回数。デフォルト値は 1 です。
VM name	**port**	インベントリパス形式での仮想マシンの名前 (例えば /datacenter/vm/Discovered_virtual_machine/myMachine)
VM UUID	**uuid**	フェンスする仮想マシンの UUID
Delay (optional)	**delay**	フェンシング開始まで待機する秒数。デフォルト値は 0 です。
Use SSL	**ssl**	デバイスとの通信に SSL 接続を使用する

表A.25「WTI Power Switch」 には、WTI ネットワークパワースイッチのフェンスエージェントである **fence_wti** により使用されるフェンスデバイスパラメーターが一覧表示されています。

表A.25 WTI Power Switch

luci フィールド	cluster.conf 属性	説明
Name	**name**	クラスターに接続されている WTI パワースイッチの名前
IP Address or Hostname	**ipaddr**	デバイスに割り当てられた IP 又はホスト名のアドレス
IP Port (optional)	**ipport**	デバイスへの接続に使用する TCP ポート
Login	**login**	デバイスのアクセスに使用するログイン名
Password	**passwd**	デバイスへの接続の認証に使用するパスワード
Password Script (optional)	**passwd_script**	フェンスデバイスへのアクセス用パスワードを提供するスクリプト。これを使用すると **Password** パラメーターが上書きされます。
Force command prompt	**cmd_prompt**	使用するコマンドプロンプト。デフォルト値は、 ['RSM>', '>MPC', 'IPS>', 'TPS>', 'NBB>', 'NPS>', 'VMR>']

luci フィールド	cluster.conf 属性	説明
Power Wait (seconds)	**power_wait**	power off 又は power on コマンドを発行後に待機する秒数
Power Timeout (seconds)	**power_timeout**	power off または power on コマンドを発行後にステータス変更をテストするまで待機する秒数。デフォルト値は 20 です。
Shell Timeout (seconds)	**shell_timeout**	コマンド発行後にコマンドプロンプトを待機する秒数。デフォルト値は 3 です。
Login Timeout (seconds)	**login_timeout**	ログイン後にコマンドプロンプトを待機する秒数。デフォルト値は 5 です。
Times to Retry Power On Operation	**retry_on**	power on 動作を再試行する回数。デフォルト値は 1 です。
Use SSH	**secure**	システムが SSH を使用してデバイスにアクセスすることを示します。SSH の使用時には、パスワード、パスワードスクリプト、ID ファイルのいずれかを指定する必要があります。
Path to SSH Identity File	**identity_file**	SSH の識別ファイル
Port	**port**	仮想マシンの物理的なプラグ番号又は名前

付録B HA リソースパラメーター

本付録は、HA リソースパラメーターについて説明しています。パラメーターの設定は、**luci** の使用、**ccs** コマンドの使用、**etc/cluster/cluster.conf** ファイルの編集により行うことができます。表B.1「HA リソースの要約」 には、リソース、その該当するリソースエージェント、パラメーターの説明を含む他の表への参照が一覧表示されています。リソースエージェントについて理解を深めるには、いずれかのクラスターノードの **/usr/share/cluster** 内にあるリソースエージェントを参照してください。

本付録で説明されているリソースエージェントに加えて、**/usr/share/cluster** ディレクトリにはリソースグループ **service.sh** 用のダミーの OCF スクリプトが含まれています。このスクリプトに含まれているパラメーターの詳細については、**service.sh** スクリプトを参照してください。

cluster.conf の要素と属性についての総括的な一覧と説明は、**/usr/share/cluster/cluster.rng** にあるクラススキーマ、及び **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** (例えば、**/usr/share/doc/cman-3.0.12/cluster_conf.html**) にある注釈付きスキーマを参照してください。

表B.1 HA リソースの要約

リソース	リソースエージェント	パラメーターの説明への参照
Apache	apache.sh	表B.2「Apache (**apache** リソース)」
Condor Instance	condor.sh	表B.3「Condor インスタンス (**condor** リソース)」
Filesystem	fs.sh	表B.4「ファイルシステム (**fs** リソース)」
GFS2	clusterfs.sh	表B.5「GFS2 (**clusterfs** リソース)」
IP Address	ip.sh	表B.6「IP アドレス (**ip** リソース)」
HA LVM	lvm.sh	表B.7「HA LVM (**lvm** リソース)」
MySQL	mysql.sh	表B.8「MySQL (**mysql** リソース)」
NFS/CIFS Mount	netfs.sh	表B.9「NFS/CIFS マウント (**netfs** resource)」
NFS Client	nfsclient.sh	表B.10「NFS クライアント (**nfsclient** Resource)」
NFS v3 Export	nfsexport.sh	表B.11「NFS v3 エクスポート (**nfsexport** リソース)」
NFS Server	nfsserver.sh	表B.12「NFS サーバー (**nfsserver** リソース)」
Oracle 10g/11g Failover Instance	oracledb.sh	表B.14「Oracle 10g/11g フェイルオーバーインスタンス (**oracledb** リソース)」

リソース	リソースエージェント	パラメーターの説明への参照
Oracle 10g/11g Instance	orainstance.sh	表B.15「Oracle 10g/11g フェイルオーバーインスタンス (**orainstance** リスナー)」
Oracle 10g/11g Listener	oralistener.sh	表B.16「Oracle 10g/11g リスナー (**oralistener** リソース)」
Open LDAP	openldap.sh	表B.13「Open LDAP (**openldap** リソース)」
PostgreSQL 8	postgres-8.sh	表B.17「PostgreSQL 8 (**postgrest-8** リソース)」
SAP Database	SAPDatabase	表B.18「SAP データベース (**SAPDatabase** リソース)」
SAP Instance	SAPInstance	表B.19「SAP インスタンス (**SAPInstance** リソース)」
Samba Server	samba.sh	表B.20「Samba サーバー (**samba** リソース)」
Script	script.sh	表B.21「スクリプト (**script** リソース)」
Sybase ASE Failover Instance	ASEHAagent.sh	表B.22「Sybase ASE フェイルオーバーインスタンス (**ASEHAagent** リソース)」
Tomcat 6	tomcat-6.sh	表B.23「Tomcat 6 (**tomcat-6** リソース)」
Virtual Machine	vm.sh	表B.24「仮想マシン (**vm** リソースResource)」 注記： ホストクラスターが仮想マシンをサポートできる場合は、**luci** がこれを仮想サービスとして表示します。

表B.2 Apache (apache リソース)

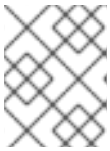
luci フィールド	cluster.conf 属性	説明
Name	**name**	Apache サービスの名前です。
Server Root	**server_root**	デフォルト値は **/etc/httpd** です。
Config File	**config_file**	Apache 設定ファイルを指定します。デフォルト値は **/etc/httpd/conf** です。
httpd Options	**httpd_options**	**httpd** 用の他のコマンドラインオプションです。

luci フィールド	cluster.conf 属性	説明
Shutdown Wait (seconds)	**shutdown_wait**	サービスシャットダウンの正確な終了までの待機秒数を指定します。

表B.3 Condor インスタンス (condor リソース)

フィールド	luci フィールド	cluster.conf 属性
Instance Name	**name**	Condor インスタンスに一意の名前を指定します。
Condor Subsystem Type	**type**	このインスタンスの Condor サブシステムのタイプである、**schedd**、**job_server**、又は **query_server** を指定します。

表B.4 ファイルシステム (fs リソース)

luci フィールド	cluster.conf 属性	説明
Name	**name**	ファイルシステムリソース用の名前を指定します。
Filesystem Type	**fstype**	指定がないと、**mount** がファイルシステムタイプを決定する試みをします。
Mount Point	**mountpoint**	このファイルシステムをマウントするためのファイルシステム階層内のパス
Device, FS Label, or UUID	**device**	ファイルシステムリソースと関連のあるデバイスを指定します。これは、ブロックデバイス、ファイルシステムラベル、又はファイルシステムの UUID のいずれかです。
Mount Options	**options**	マウントオプション。これはファイルシステムがマウントされる時に使用するオプションであり、ファイルシステム特有である場合があります。サポートされているマウントオプションについては、『**mount**(8)』 の man ページを参照してください。
File System ID (optional)	**fsid**	 **注記** ***File System ID*** は NFS サービスでのみ使用されます。 新規のファイルシステムリソースを作成する時、このフィールドは空白のままにできます。フィールドを空白にすると、パラメーターをコミットした後、設定中にファイルシステム ID は自動的に割り当てられます。ファイルシステム ID を明示的に割り当てる必要がある場合は、このフィールド内で指定してください。

luci フィールド	cluster.conf 属性	説明
Force Unmount	**force_unmount**	有効になっていると、ファイルシステムのアンマウントを強制します。デフォルトのセッティングは ***disabled***（無効） です。***Force Unmount*** はアンマウント時にマウントポイントの全てのプロセスをキルしてマウントを開放します。
Force fsck	**force_fsck**	有効になっていると、ファイルシステムをマウントする前に **fsck** を実行することになります。デフォルトセッティングは ***disabled*** です。
Enable NFS and lockd workaround (Red Hat Enterprise Linux 6.4 以降)	**nfsrestart**	ファイルシステムが NFS 経由でエクスポートされ、アンマウントの際に問題が発生することがある場合 (シャットダウン時やサービスのリロケーション時)、このオプションを設定するとアンマウントの操作の前にファイルシステムの参照をすべてドロップします。このオプションを設定するには、**Force unmount** オプションを有効にする必要があります。ただし、**NFS Server** リソースとあわせて使用することはできません。ファイルシステムのアンマウントが大変になるため、最終手段としてこのオプションを設定するようにしてください。
Use Quick Status Checks	**quick_status**	有効の場合は、クイックステータスチェックを実行します。
Reboot Host Node if Unmount Fails	**self_fence**	有効にすると、このファイルシステムのアンマウントに問題が発生した場合にノードを再起動します。**filesystem** リソースエージェントは、1、**yes**、**on** または **true** を認識してこのパラメーターを有効にします。また、0、**no**、**off** または **false** で無効にします。デフォルト設定は ***disabled*** です。

表B.5 GFS2 (clusterfs リソース)

luci フィールド	cluster.conf 属性	説明
Name	**name**	ファイルシステムリソースの名前
Mount Point	**mountpoint**	ファイルシステムリソースがマウントされるパスです。
Device, FS Label, or UUID	**device**	ファイルシステムリソースに関連しているデバイスのファイルです。
Filesystem Type	**fstype**	**luci** で GFS2 に設定します。
Mount Options	**options**	マウントオプションです。

luci フィールド	cluster.conf 属性	説明
File System ID (optional)	**fsid**	 **注記** ***File System ID*** は NFS サービスでのみ使用されます。 新規の GFS2 リソースを作成する時、このフィールドは空白のままにできます。フィールドを空白にすると、パラメーターをコミットした後、設定中にファイルシステム ID は自動的に割り当てられます。ファイルシステム ID を明示的に割り当てる必要がある場合は、このフィールド内で指定してください。
Force Unmount	**force_unmount**	有効になっていると、ファイルシステムのアンマウントを強制します。デフォルトのセッティングは ***disabled***（無効） です。***Force Unmount*** はアンマウント時にマウントポイントの全てのプロセスをキルしてマウントを開放します。GFS2 リソースでは、***Force Unmount*** が **有効**になっていないと、サービスの停止でマウントポイントはアンマウント**されません。**
Enable NFS and lockd workaround (Red Hat Enterprise Linux 6.4 以降)	**nfsrestart**	ファイルシステムが NFS 経由でエクスポートされ、アンマウントの際に問題が発生することがある場合 (シャットダウン時やサービスのリロケーション時)、このオプションを設定するとアンマウントの操作の前にファイルシステムの参照をすべてドロップします。このオプションを設定するには、**Force unmount** オプションを有効にする必要があります。ただし、**NFS Server** リソースとあわせて使用することはできません。ファイルシステムのアンマウントが大変になるため、最終手段としてこのオプションを設定するようにしてください。
Reboot Host Node if Unmount Fails	**self_fence**	これを有効にしていてファイルシステムのアンマウントに失敗すると、ノードはすぐに再起動します。通常、force-unmount サポートとあわせて使用しますが、必須ではありません。**GFS2** リソースエージェントは、1、**yes**、**on** または **true** を認識してこのパラメーターを有効にします。また、0、**no**、**off** または **false** で無効にします。

表B.6 IP アドレス (ip リソース)

luci フィールド	cluster.conf 属性	説明
IP Address, Netmask Bits	**address**	リソースの IP アドレス (オプションでネットマスクビット)。ネットマスクビット (またはネットワークプレフィックスの長さ) は、アドレスと区切り文字であるスラッシュの後ろに来ます。これは、CIDR 表記に準拠しています (例：10.1.1.1/8)。この IP アドレスは仮想アドレスです。IPv4、IPv6 アドレス、そして各 IP アドレスの NIC リンクモニタリングに対応しています。

luci フィールド	cluster.conf 属性	説明
Monitor Link	**monitor_link**	これを有効にすると、この IP アドレスがバインドされている NIC のリンクが存在しない場合にはステータスチェックが失敗します。
Disable Updates to Static Routes	**disable_rdisc**	RDISC プロトコルを使用したルーティングの更新を無効にします。
Number of Seconds to Sleep After Removing an IP Address	**sleeptime**	スリープする時間 (秒単位) を指定します。

表B.7 HA LVM (lvm リソース)

luci フィールド	cluster.conf 属性	説明
Name	**name**	この LVM リソースの一意の名前です。
Volume Group Name	**vg_name**	管理されているボリュームグループの説明的な名前です。
Logical Volume Name	**lv_name**	管理されている論理ボリュームの名前です。このパラメーターは、管理されているボリュームグループ内に複数の論理ボリュームが存在する場合はオプションとなります。
Fence the Node if It is Unable to Clean UP LVM Tags	**self_fence**	LVM タグを削除できない場合にノードをフェンシングします。LVM リソースエージェントは、**yes** か 1 の値でこのパラメーターを有効にするか、**no** か 0 の値でこのパラメーターを無効にします。

表B.8 MySQL (mysql リソース)

luci フィールド	cluster.conf 属性	説明
Name	**name**	MySQL サーバーリソースの名前を指定します。
Config File	**config_file**	設定ファイルを指定します。デフォルト値は **/etc/my.cnf** です。
Listen Address	**listen_address**	MySQL サーバーの IP アドレスを指定します。IP アドレスが指定されていない場合は、サービスからの最初の IP アドレスが採用されます。

luci フィールド	cluster.conf 属性	説明
mysqld Options	**mysqld_options**	**mysqld** 用の他のコマンドラインオプションです。
Startup Wait (seconds)	**startup_wait**	サービス起動の正確な終了までの待機秒数を指定します。
Shutdown Wait (seconds)	**shutdown_wait**	サービスシャットダウンの正確な終了までの待機秒数を指定します。

表B.9 NFS/CIFS マウント (netfs resource)

luci フィールド	cluster.conf 属性	説明
Name	**name**	sNFS 又は CIFS のマウント用シンボリック名です。  **注記** このリソースは、クラスターサービスが NFS クライアントになるように設定する時に必要です。
Mount Point	**mountpoint**	ファイルシステムリソースのマウント先へのパスです。
Host	**host**	NFS/CIFS サーバーの IP アドレス、又はホスト名です。
NFS Export Directory Name or CIFS share	**export**	NFS エクスポートディレクトリ名、又は CIFS 共有名です。
Filesystem Type	**fstype**	ファイルシステムタイプ: • ***NFS*** — デフォルト NFS バージョンの使用を指定します。 これがデフォルトのセッティングです。 • ***NFS v4*** — NFSv4 プロトコルの使用を指定します。 • ***CIFS*** — CIFS プロトコルの使用を指定します。
Do Not Unmount the Filesystem During a Stop of Relocation Operation.	**no_unmount**	有効の場合は、ファイルシステムが停止/再配置の動作中にアンマウントされないように指定します。

luci フィールド	cluster.conf 属性	説明
Force Unmount	**force_unmount**	***Force Unmount*** が有効の場合、サービスの停止時にクラスターはこのファイルシステムを使用している全てのプロセスを kill します。ファイルシステムを使用している全てのプロセスを kill するとファイルシステムの領域が使えるようになります。そうでない場合は、アンマウントは失敗してサービスが再起動します。
Options	**options**	マウントオプションです。マウントオプションの一覧を指定します。指定されていないと、ファイルシステムは **-o sync** としてマウントされます。

表B.10 NFS クライアント (nfsclient Resource)

luci フィールド	cluster.conf 属性	説明
Name	**name**	リソースツリー内でクライアントを参照するために使用されるそのシンボリック名です。これは、***Target*** オプションと同じ **ではありません。**
Target Hostname, Wildcard, or Netgroup	**target**	マウント元のサーバーです。ホスト名、ワイルドカード (IP アドレスか、ホスト名ベース)、又はエクスポート先のホストを定義する netgroup を使用して指定できます。
Allow Recovery of This NFS Client	**allow_recover**	回復を行います。
Options	**options**	このクライアント用のオプション一覧 — 例えば、追加のクライアントアクセス権を定義します。詳細情報は、『**exports** (5)』 の man ページの 『General Options』 を参照してください。

表B.11 NFS v3 エクスポート (nfsexport リソース)

luci フィールド	cluster.conf 属性	説明

luci フィールド	cluster.conf 属性	説明
Name	**name**	リソースの説明的な名前です。NFS エクスポートリソースは確実に NFS デーモンが稼働するようにします。完全に再利用可能で、通常は 1 つの NFS エクスポートのみが必要です。**nfsexport** リソース設定の詳細情報は、「nfsexport および nfsserver リソースの設定」 を参照してください。  **注記** NFS エクスポートリソースは、他の NFS リソースと区別できるような明確な名前にしてください。

表B.12 NFS サーバー (nfsserver リソース)

luci フィールド	cluster.conf 属性	説明
Name	**name**	NFS サーバーリソースの説明的な名前です。NFS サーバーリソースは、NFSv4 ファイルシステムをクライアントにエクスポートする際に便利です。NFSv4 の機能的な理由により、サーバー上に一度に存在することが可能なのは、1 つの NFSv4 リソースのみです。また、各クラスターノード上で NFS のローカルインスタンスを使用する際は、NFS サーバーリソースを使用することはできません。**nfsserver** リソース設定の詳細は、「nfsexport および nfsserver リソースの設定」 を参照してください。

表B.13 Open LDAP (openldap リソース)

luci フィールド	cluster.conf 属性	説明
Name	**name**	ログ記録と他の目的用のサービス名を指定します。
Config File	**config_file**	設定ファイルへの絶対パスを指定します。デフォルト値は **/etc/openldap/slapd.conf** です。
URL List	**url_list**	デフォルト値は **ldap:///** です。
slapd Options	**slapd_options**	**slapd** 用の他のコマンドラインオプションです。
Shutdown Wait (seconds)	**shutdown_wait**	サービスシャットダウンの正確な終了までの待機秒数を指定します。

表B.14 Oracle 10g/11g フェイルオーバーインスタンス (oracledb リソース)

luci フィールド	cluster.conf 属性	説明
Instance Name (SID) of Oracle Instance	**name**	インスタンスの名前です。
Oracle Listener Instance Name	**listener_name**	Oracle リスナーのインスタンス名です。複数の Oracle インスタンスを稼働している場合は、同一マシン上で異なる名前の複数のリスナーが必要になる可能性があります。
Oracle User Name	**user**	これは、Oracle AS インスタンスが Oracle ユーザーとして実行するユーザー名です。
Oracle Application Home Directory	**home**	これは Oracle (ユーザーではなく、アプリケーション) のホームディレクトリです。Oracle のインストール時に設定します。
Oracle Installation Type	**type**	Oracle インストールタイプです。 • デフォルト: **10g** • **base**: データベースインスタンスおよびリスナーのみ • **base-11g**: Oracle11g データベースインスタンスおよびリスナーのみ • **base-em** (または **10g**): データベース、リスナー、Enterprise Manager、および iSQL*Plus • **base-em-11g**: データベース、リスナー、Enterprise Manager dbconsole • **ias** (または **10g-ias**): Internet Application Server (Infrastructure)
Virtual Hostname (optional)	**vhost**	Oracle 10g のインストールホスト名にマッチする仮想ホスト名です。oracledb リソースの開始/停止の間に、使用ホスト名は一時的にこのホスト名へ変更されることに注意して下さい。そのため、oracledb リソースを専用サービスの一部としてのみ設定する必要があります。
TNS_ADMIN (optional)	**tns_admin**	特定のリスナー設定ファイルへのパス。

表B.15 Oracle 10g/11g フェイルオーバーインスタンス (orainstance リスナー)

luci フィールド	cluster.conf 属性	説明
Instance name (SID) of Oracle instance	**name**	インスタンスの名前です。
Oracle User Name	**user**	これは、Oracle インスタンスが Oracle ユーザーとして実行するユーザー名です。
Oracle Application Home Directory	**home**	これは Oracle (ユーザーではなく、アプリケーション) のホームディレクトリです。Oracle のインストール時に設定します。
List of Oracle Listeners (オプションです。空白で区切ります)	**listeners**	データベースインスタンスで開始する Oracle リスナーの一覧です。リスナー名は空白で区切ります。デフォルトは空白で、リスナーを無効にします。
Path to Lock File (optional)	**lockfile**	Oracle が実行しているかどうかを確認するために使用される lockfile の場所です。デフォルトの場所は **/tmp** 下です。
TNS_ADMIN (optional)	**tns_admin**	特定のリスナー設定ファイルへのパス。

表B.16 Oracle 10g/11g リスナー (oralistener リソース)

luci フィールド	cluster.conf 属性	説明
Listener Name	**name**	リスナーの名前です。
Oracle User Name	**user**	これは、Oracle インスタンスが Oracle ユーザーとして実行するユーザー名です。
Oracle Application Home Directory	**home**	これは Oracle (ユーザーではなく、アプリケーション) のホームディレクトリです。Oracle のインストール時に設定します。
TNS_ADMIN (optional)	**tns_admin**	特定のリスナー設定ファイルへのパス。

表B.17 PostgreSQL 8 (postgrest-8 リソース)

luci フィールド	cluster.conf 属性	説明
Name	**name**	ログ記録と他の目的用のサービス名を指定します。
Config File	**config_file**	設定ファイルまでの絶対パスを定義します。デフォルト値は **/var/lib/pgsql/data/postgresql.conf** です。
Postmaster User	**postmaster_user**	root ではデータベースサーバーを実行できないため、それを実行するユーザーです。デフォルト値は postgres です。
Postmaster Options	**postmaster_options**	postmaster の他のコマンドラインオプションです。
Shutdown Wait (seconds)	**shutdown_wait**	サービスシャットダウンの正確な終了までの待機秒数を指定します。

表B.18 SAP データベース (SAPDatabase リソース)

luci フィールド	cluster.conf 属性	説明
SAP Database Name	**SID**	一意の SAP システム識別子を指定します。例えば、P01 です。
SAP Executable Directory	**DIR_EXECUTABLE**	**sapstartsrv** と **sapcontrol** への完全修飾パスを指定します。
Database Type	**DBTYPE**	Oracle、DB6、又は ADA のデータベースタイプのうち 1 つを指定します。
Oracle Listener Name	**NETSERVICENAME**	Oracle TNS リスナー名を指定します。
ABAP Stack is Not Installed, Only Java Stack is Installed	**DBJ2EE_ONLY**	ABAP スタックが SAP データベース内にインストールされていない場合は、このパラメーターを有効にしてください。
Application Level Monitoring	**STRICT_MONITORING**	アプリケーションレベルモニタリングをアクティベートします。
Automatic Startup Recovery	**AUTOMATIC_RECOVER**	スタートアップの自動修復を有効/無効にします。

luci フィールド	cluster.conf 属性	説明
Path to Java SDK	**JAVE_HOME**	Java SDK へのパスです。
File Name of the JDBC Driver	**DB_JARS**	JDBC ドライバーのファイル名です。
Path to a Pre-Start Script	**PRE_START_USEREXIT**	pre-start スクリプトへのパスです。
Path to a Post-Start Script	**POST_START_USEREXIT**	post-start スクリプトへのパスです。
Path to a Pre-Stop Script	**PRE_STOP_USEREXIT**	pre-stop スクリプトへのパスです。
Path to a Post-Stop Script	**POST_STOP_USEREXIT**	post-stop スクリプトへのパスです。
J2EE Instance Bootstrap Directory	**DIR_BOOTSTRAP**	J2EE インスタンスブートストラップディレクトリへの完全修飾パスです。例えば、**/usr/sap/P01/J00/j2ee/cluster/bootstrap** です。
J2EE Security Store Path	**DIR_SECSTORE**	J2EE セキュリティストアディレクトリへの完全修飾パスです。例えば、**/usr/sap/P01/SYS/global/security/lib/tools** です。

表B.19 SAP インスタンス (SAPInstance リソース)

luci フィールド	cluster.conf 属性	説明
SAP Instance Name	**InstanceName**	完全修飾 SAP インスタンス名です。例えば、P01_DVEBMGS00_sapp01ci です。
SAP Executable Directory	**DIR_EXECUTABLE**	**sapstartsrv** と **sapcontrol** への完全修飾パスです。
Directory Containing the SAP START Profile	**DIR_PROFILE**	SAP START プロファイルへの完全修飾パスです。

luci フィールド	cluster.conf 属性	説明
Name of the SAP START Profile	**START_PROFILE**	SAP START プロファイルの名前を指定します。
Number of Seconds to Wait Before Checking Startup Status	**START_WAITTIME**	スタートアップステータスを確認するまでの待機秒数を指定します (J2EE-Addin を待機しません)。
Enable Automatic Startup Recovery	**AUTOMATIC_RECOVER**	スタートアップの自動修復を有効/無効にします。
Path to a Pre-Start Script	**PRE_START_USEREXIT**	pre-start スクリプトへのパスです。
Path to a Post-Start Script	**POST_START_USEREXIT**	post-start スクリプトへのパスです。
Path to a Pre-Stop Script	**PRE_STOP_USEREXIT**	pre-stop スクリプトへのパスです。
Path to a Post-Stop Script	**POST_STOP_USEREXIT**	post-stop スクリプトへのパスです。

注記

表B.20「Samba サーバー (**samba** リソース)」 に関して、クラスターサービスの作成/編集時には、Samba-service リソースをサービス内のリソース **ではなく**、直接サービスに接続してください。

表B.20 Samba サーバー (samba リソース)

luci フィールド	cluster.conf 属性	説明
Name	**name**	Samba サーバーの名前を指定します。
Config File	**config_file**	Samba の設定ファイルです。

luci フィールド	cluster.conf 属性	説明
Other Command-Line Options for smbd	**smbd_options**	smbd 用の他のコマンドラインオプションです。
Other Command-Line Options for nmbd	**nmbd_options**	nmbd 用の他のコマンドラインオプションです。
Shutdown Wait (seconds)	**shutdown_wait**	サービスシャットダウンの正確な終了までの待機秒数を指定します。

表B.21 スクリプト (script リソース)

luci フィールド	cluster.conf 属性	説明
Name	**name**	カスタムユーザースクリプトの名前を指定します。スクリプトリソースにより、標準の LSB 準拠の init スクリプトはクラスター化サービスを起動するのに使用できます。
Full Path to Script File	**file**	このカスタムスクリプトが配置してある場所へのパスを入力します (例えば、 **/etc/init.d/*userscript***)。

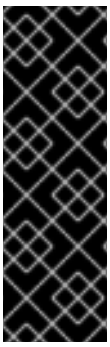
表B.22 Sybase ASE フェイルオーバーインスタンス (ASEHAagent リソース)

luci フィールド	cluster.conf 属性	説明
Instance Name	**name**	Sybase ASE リソースのインスタンス名を指定します。
ASE Server Name	**server_name**	HA サービス用に設定してある ASE サーバーの名前です。
SYBASE Home directory	**sybase_home**	Sybase 製品のホームディレクトリです。
Login File	**login_file**	ログイン/パスワードペアを含むログインファイルのフルパスです。
Interfaces File	**interfaces_file**	ASE サーバーの開始/アクセスに使用されるインターフェースファイルのフルパスです。
SYBASE_ASE Directory Name	**sybase_ase**	ASE 製品がインストールされている sybase_home 下のディレクトリ名です。

luci フィールド	cluster.conf 属性	説明
SYBASE_OCS Directory Name	**sybase_ocs**	OCS 製品がインストールされている sybase_home 下のディレクトリ名です。例えば、ASE-15_0 です。
Sybase User	**sybase_user**	ASE サーバーを実行するユーザーです。
Start Timeout (seconds)	**start_timeout**	起動のタイムアウト値です。
Shutdown Timeout (seconds)	**shutdown_timeout**	シャットダウンのタイムアウト値です。
Deep Probe Timeout	**deep_probe_timeout**	詳細検出（deep probe）を実行している間にサーバーが反応しないことを判定するまで ASE サーバーの対応を待つ期間の最大秒数です。

表B.23 Tomcat 6 (tomcat-6 リソース)

luci フィールド	cluster.conf 属性	説明
Name	**name**	ログ記録と他の目的用のサービス名を指定します。
Config File	**config_file**	設定ファイルへの絶対パスを指定します。デフォルト値は **/etc/tomcat6/tomcat6.conf** です。
Shutdown Wait (seconds)	**shutdown_wait**	サービスシャットダウンの正しい終了までの待機期間の秒数を指定します。デフォルトは 30 です。



重要

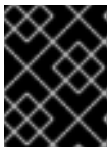
表B.24「仮想マシン (**vm** リソースResource)」 に関して、仮想マシンリソースを使用してご使用のクラスターを設定する場合、**rgmanager** ツールを使用して、仮想マシンの起動/停止を行うことをお勧めします。**virsh** を使用してマシンを起動すると、仮想マシンが複数の場所で実行することにつながり、仮想マシン内のデータが破損する恐れがあります。クラスター及び非クラスターツールの両方を使用して管理者が仮想マシンを誤って「ダブル起動」するリスクを減らすためのシステム設定については、「クラスター化環境での仮想マシンの設定」 を参照してください。

注記

仮想マシンリソースの設定は、他のクラスターリソースと異なります。**luci** を使用して仮想マシンリソースを設定するには、サービスグループをクラスターに追加した後、リソースをサービスに追加します。**Virtual Machine** をリソースタイプとして選択し、仮想マシンリソースパラメーターを入力します。**ccs** を使用した仮想マシンの設定については、「仮想マシンのリソース」 を参照してください。

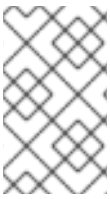
表B.24 仮想マシン (vm リソースResource)

luci フィールド	cluster.conf 属性	説明
Service Name	**name**	仮想マシンの名前を指定します。**luci** インターフェースを使用している時には、これをサービス名として指定します。
Automatically Start This Service	**autostart**	有効になっている場合、仮想マシンはクラスターが定員を構成した後に自動的に開始されます。このパラメーターが **無効** の場合は、クラスターが定員を構成した後、自動的に開始 **されません。** 仮想マシンは ***disabled***（無効）状態になります。
Run Exclusive	**exclusive**	有効の場合、この仮想マシンは別のノードのみで実行する、すなわち他の仮想マシンが稼働していないノード上で実行するように再配置されるだけです。仮想マシンが単独で実行できるノードがない場合は、仮想マシンは障害が発生すると再起動しません。さらに、他の仮想マシンは ***Run exclusive*** としてこの仮想マシンを実行しているノードに自動的に再配置されることはありません。このオプションを上書きするには、手動で起動/再配置の操作を行います。
Failover Domain	**domain**	仮想マシンが失敗した場合に試行するクラスターメンバーの一覧を定義します。
Recovery Policy	**recovery**	***Recovery policy*** は以下のオプションを提供します： • ***Disable*** — 仮想マシンが失敗した場合、それを無効にします。 • ***Relocate*** — 仮想マシンを別のノード上で再起動する試行をします。すなわち、現行のノードでは再起動を試行しません。 • ***Restart*** — 仮想マシンを別のノードに再配置する (デフォルト) 試行の前に、仮想マシンをローカル (現行のノード) で再起動を試行します。 • ***Restart-Disable*** — サービスが失敗した場合、再起動します。しかし、サービスの再起動が失敗すると、クラスター内の別のホストに移動される代わりにサービスは無効になります。
Restart Options	**max_restarts**, **restart_expire_time**	サービスの復元ポリシーとして **Restart（再開始）** 又は **Restart-Disable（再開始と無効化）** を選択すると、サービスの移動/無効化までに再開始が失敗する最大回数と、再開始を破棄するまでの時間を秒単位で指定することができます。
Migration Type	**migrate**	移行タイプである ***live*** 又は ***pause*** を指定します。デフォルトセッティングは ***live*** です。

luci フィールド	cluster.conf 属性	説明
Migration Mapping	**migration_mapping**	移行用に代替のインターフェースを指定します。これを指定するのは、ノード上で仮想マシンの移行用に使用されるネットワークアドレスが、クラスター通信に使用されるノードのアドレスと異なる場合などです。 以下を指定すると、仮想マシンを **member** から **member2** に 移行する時、実際には **target2** に移行することを示します。同様にして **member2** から **member** に移行する時は **target** を使用して移行することになります。 **member:target,member2:target2**
Status Program	**status_program**	仮想マシンの存在を知るための標準チェックに加えて実行するステータスプログラムです。 指定されていると、ステータスプログラムは 1 分毎に 1 度実行されます。これにより、仮想マシン 内の重要なサービスのステータスを確認できるようになります。例えば、仮想マシンがウェブサーバーを実行すると ステータスプログラムはウェブサーバーが立ち上がって稼働しているかを見るためにチェックできます。 このステータスチェックが失敗すると（ゼロ以外の値が戻って来る）仮想マシンは復元されます。 仮想マシンの開始後、仮想マシンのリソースエージェントは定期的にステータスプログラムを呼び出して、成功の戻りコード（ゼロ）を待ちます。これは 5 分後にタイムアウトとなります。
Path to xmlfile Used to Create the VM	**xmlfile**	**libvirt** ドメイン定義を含む **libvirt** XML ファイルへのフルパス
VM Configuration File Path	**path**	仮想マシンリソースエージェント (**vm.sh**) が仮想マシン設定ファイルを検出するためのコロンで区切ったパス指定です。例えば、**/mnt/guests/config:/etc/libvirt/qemu** です。  **重要** パスは**決して**仮想マシンの設定ファイルへ直接的にポイントすべきではありません。
Path to the VM Snapshot Directory	**snapshot**	仮想マシンイメージが保存されるスナップショットディレクトリへのパスです。
Hypervisor URI	**hypervisor_uri**	ハイパバイサーの URI (通常自動) です。
Migration URI	**migration_uri**	移行の URI (通常自動) です。
Tunnel data over ssh during migration	**tunnelled**	移行中の ssh によるトンネルデータです。

付録C HA リソースの動作

本付録は HA リソースの一般的な動作を説明しています。ここでは、HA サービスの設定に役立つ補足的な情報を提供することが目的です。パラメーターを設定するには、**luci** を使用するか、**/etc/cluster/cluster.conf** を編集します。HA リソースパラメーターの詳細については、付録B *HA リソースパラメーター* を参照してください。リソースエージェントをさらに詳しく理解するには、任意のクラスターノードの **/usr/share/cluster** にあるリソースエージェントを参照してください。



注記

本付録内の情報を完全に把握するには、リソースエージェントとクラスター設定ファイル、**/etc/cluster/cluster.conf** を詳しく理解していただく必要が場合があります。

HA サービスは一貫したエンティティに設定されているクラスターリソースのグループであり、クライアントに特殊化したサービスを提供します。HA サービスは、(各クラスターノード内の) **/etc/cluster/cluster.conf** クラスター設定ファイルでリソースツリーとして表示されます。クラスター設定ファイルでは、各リソースツリーは各リソース、その属性、及びリソースツリー内の他のリソースとの関係 (親、子、兄弟の関係) を指定する XML 表現です。

注記

HA サービスは階層ツリー型に編成されているリソースで構成されるため、サービスは *リソースツリー* 又は *リソースグループ* と呼ばれることがあります。この両方の呼称とも **HA サービス** と同義語です。

各リソースツリーの根元には、リソースの特殊タイプ — *サービスリソース* があります。残りのサービスは、他のタイプのリソースにより構成され、その特性を決定します。HA サービスの設定は、サービスリソースの作成、従属クラスターリソースの作成、及びサービスの階層的制約に従う一貫したエンティティへそれらを組織することによって行われます。

本付録は以下のセクションで構成されます：

- 「リソース間の親、子、兄弟の関係」
- 「兄弟開始の順序とリソースの子の順序」
- 「継承、<resources>ブロック、リソースの再利用」
- 「障害からの回復と独立したサブツリー」
- 「サービスとリソース順序へのデバッグとテスト」

注記

クラスター設定ファイル、**/etc/cluster/cluster.conf** の例の後に表示されているセクションは解説のみを目的として使用されています。

C.1. リソース間の親、子、兄弟の関係

クラスターサービスとは **rgmanager** の制御下で実行する統合されたエンティティです。サービス内の全てのリソースは同じノード上で稼働します。**rgmanager** の視点からは、クラスターサービスは開

始、停止、又は再配置できるエンティティです。ただし、クラスターサービス内では、リソースの階層により各リソースが開始、停止する順番が決定されます。この階層レベルは親、子、兄弟で構成されています。

例C.1「foo サービスのリソース階層」 では、**foo** サービスのリソースツリーのサンプルを示しています。この例では、リソース間の関係は以下のようになります：

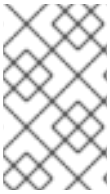
- **`fs:myfs`** (<fs name="myfs" ...>) と **`ip:10.1.1.2`** (<ip address="10.1.1.2 .../>) は兄弟です。
- **`fs:myfs`** (<fs name="myfs" ...>) は **`script:script_child`** (<script name="script_child"/>) の親です。
- **`script:script_child`** (<script name="script_child"/>) は **`fs:myfs`** (<fs name="myfs" ...>) の子です。

例C.1 foo サービスのリソース階層

```
<service name="foo" ...>
    <fs name="myfs" ...>
        <script name="script_child"/>
    </fs>
    <ip address="10.1.1.2" .../>
</service>
```

リソースツリーでは以下のルールが、親/子 の関係に適用されます：

- 親は子の前に開始される。
- 親が停止する前に、全ての子がクリーンに停止しなければならない。
- リソースが健全な状態にあると見なされるには、全ての子が健全である必要がある。



注記

フローティング IP アドレスリソースを含むクラスターサービスの依存関係ツリー設定時には、IP リソースを別のリソースの子としてではなく、最初のエントリーとして設定する必要があります。

C.2. 兄弟開始の順序とリソースの子の順序

サービスリソースは、以下のように子リソースに対して子のタイプ属性を指定するかどうかに応じて子リソースの開始順と停止順を決定します。

- 子のタイプ属性を指定する (*型指定*された子リソース) — サービスリソースが子リソースに対して子のタイプ属性を指定すると、子リソースは **型指定** します。子のタイプ属性は明示的に子リソースの開始順と停止順を決定します。
- 子のタイプ属性を **指定しない** (*型指定されていない* 子リソース) — サービスリソースが子リソースに対して子のタイプ属性を **指定しない** と、子リソースは **型指定されません**。サービスリソースは型指定されていない子リソースの開始順と停止順を明示的に制御しません。しかし、型指定されていない子リソースは **`/etc/cluster/cluster.conf`** 内の順番に従って開始、停止します。また、型指定されていない子リソースは、全ての型指定された子リソースが開始した後に開始し、いずれかの型指定された子リソースが停止する前に停止します。

注記

定義された **子リソースタイプ** 順を実装する唯一のリソースはサービスリソースです。

型指定された子リソースの開始と停止の順序に関する詳細情報は、「型指定された子リソースの開始と停止の順序」 を参照してください。型指定されていない子リソースの開始と停止の順序に関する詳細情報は、「型指定されていない子リソースの開始と停止の順序」 を参照してください。

C.2.1. 型指定された子リソースの開始と停止の順序

型指定された子リソースでは、子リソースのタイプ属性は 1 から 100 までの番号を使って各リソースタイプの開始順と停止順を定義します。開始及び停止用の値はそれぞれ 1 つです。数字が低いほど、リソースタイプは早く開始/停止します。例えば、表C.1「子リソースタイプの開始と停止の順序」 は各リソースタイプの開始値と停止値を示しています。例C.2「リソースの開始と停止の値： サービスリソースエージェント **service.sh** からの抜粋」 は、開始値と停止値がサービスリソースエージェントである **service.sh** に表示されるとおり示しています。サービスリソースについては、全ての LVM 子群が最初に開始します。次に、全てのファイルシステム子群、その後に全てのスクリプト子群と、開始していきます。

表C.1 子リソースタイプの開始と停止の順序

リソース	子タイプ	開始順の値	停止順の値
LVM	lvm	1	9
ファイルシステム	fs	2	8
GFS2 ファイルシステム	clusterfs	3	7
NFS マウント	netfs	4	6
NFS エクスポート	nfsexport	5	5
NFS クライアント	nfsclient	6	4
IP Address	ip	7	2
Samba	smb	8	3
スクリプト	script	9	1

例C.2 リソースの開始と停止の値： サービスリソースエージェント service.sh からの抜粋

```
<special tag="rgmanager">
    <attributes root="1" maxinstances="1"/>
    <child type="lvm" start="1" stop="9"/>
    <child type="fs" start="2" stop="8"/>
    <child type="clusterfs" start="3" stop="7"/>
    <child type="netfs" start="4" stop="6"/>
    <child type="nfsexport" start="5" stop="5"/>
```

```
    <child type="nfsclient" start="6" stop="4"/>
    <child type="ip" start="7" stop="2"/>
    <child type="smb" start="8" stop="3"/>
    <child type="script" start="9" stop="1"/>
</special>
```

リソースタイプ内での順番はクラスター設定ファイル、**/etc/cluster/cluster.conf** に存在する通りに保存されてます。 例えば、例C.3「リソースタイプ内の順序」 内の型指定された子リソースの開始順と停止順を考えてみましょう。

例C.3 リソースタイプ内の順序

```
<service name="foo">
  <script name="1" .../>
  <lvm name="1" .../>
  <ip address="10.1.1.1" .../>
  <fs name="1" .../>
  <lvm name="2" .../>
</service>
```

型指定された子リソースの開始順

例C.3「リソースタイプ内の順序」 では、リソースは以下の順序で開始します：

1. **lvm:1** — これは LVM のリソースです。全ての LVM リソースが最初に開始します。**lvm:1** (**<lvm name="1" .../>**) は、**/etc/cluster/cluster.conf** の **foo** サービス部分でリストされている最初の LVM リソースであるため、LVM リソースの中では最初に開始します。

2. **lvm:2** — これは LVM リソースです。全ての LVM リソースが最初に開始します。**lvm:2** (**<lvm name="2" .../>**) は、**/etc/cluster/cluster.conf** の **foo** サービス部分の中で **lvm:1** の後にリストされているため、**lvm:1** の次に開始します。

3. **fs:1** — これはファイルシステムリソースです。**foo** サービス内に他のファイルシステムリソースがある場合は、**/etc/cluster/cluster.conf** の **foo** サービス部分にリストされている順序で開始します。

4. **ip:10.1.1.1** — これは IP アドレスリソースです。**foo** サービス内に他の IP アドレスのリソースがある場合は、**/etc/cluster/cluster.conf** の **foo** サービス部分にリストされている順序で開始します。

5. **script:1** — これはスクリプトリソースです。**foo** サービス内に他のスクリプトリソースがある場合は、**/etc/cluster/cluster.conf** の **foo** サービス部分にリストされている順序で開始します。

型指定された子リソースの停止順

例C.3「リソースタイプ内の順序」 では、リソースは以下の順序で停止します：

1. **script:1** — これはスクリプトリソースです。**foo** サービス内に他のスクリプトリソースがある場合は、**/etc/cluster/cluster.conf** の **foo** サービス部分にリストされている逆の順番で停止します。

2. **ip:10.1.1.1** — これは IP アドレスリソースです。**foo** サービス内に他の IP アドレスのリソースがある場合は、**/etc/cluster/cluster.conf** の **foo** サービス部分にリストされている逆の順番で停止します。

3. **fs:1** — これはファイルシステムリソースです。**foo** サービス内に他のファイルシステムリソースがある場合は、**/etc/cluster/cluster.conf** の **foo** サービス部分にリストされている逆の順番で停止します。

4. **lvm:2** — これは LVM リソースです。全ての LVM リソースは最後に停止します。**lvm:2** (**<lvm name="2" .../>**) は **lvm:1** より先に停止します。リソースタイプのグループ内のリソース群は **/etc/cluster/cluster.conf** の **foo** サービス部分にリストされている逆の順番で停止します。

5. **lvm:1** — これは LVM リソースです。全ての LVM リソースは最後に停止します。**lvm:1** (**<lvm name="1" .../>**) は **lvm:2** の後に停止します。リソースタイプのグループ内のリソース群は **/etc/cluster/cluster.conf** の **foo** サービス部分にリストされている逆の順番で停止します。

C.2.2. 型指定されていない子リソースの開始と停止の順序

型指定されていない子リソースには、さらなる配慮が必要です。型指定されていない子リソースについては、サービスリソースによる、開始順、停止順の明示指定がありません。代わりに **/etc/cluster/cluster.conf** の子リソースの順番に従い、開始、停止の順番が決定されます。また、型指定されていない子リソースは、型指定されたすべての子リソースの後に開始され、型指定された子リソースより先に停止されます。

例えば、例C.4「サービス内にある型指定されていないリソースと型指定されたリソース」 にある型指定されていない子リソースの開始順と停止順を考えてみましょう。

例C.4 サービス内にある型指定されていないリソースと型指定されたリソース

```
<service name="foo">
  <script name="1" .../>
  <nontypedresource name="foo"/>
  <lvm name="1" .../>
  <nontypedresourcetwo name="bar"/>
  <ip address="10.1.1.1" .../>
  <fs name="1" .../>
  <lvm name="2" .../>
</service>
```

型指定されていない子リソースの開始順

例C.4「サービス内にある型指定されていないリソースと型指定されたリソース」 では、子リソースは以下の順序で開始します：

1. **lvm:1** — これは LVM のリソースです。全ての LVM リソースが最初に開始します。**lvm:1** (**<lvm name="1" .../>**) は、**/etc/cluster/cluster.conf** の **foo** サービス部分でリストされている最初の LVM リソースであるため、LVM リソースの中では最初に開始します。

2. **lvm:2** — これは LVM リソースです。全ての LVM リソースが最初に開始します。**lvm:2** (**<lvm name="2" .../>**) は、**/etc/cluster/cluster.conf** の **foo** サービス部分の中で **lvm:1** の後にリストされているため、**lvm:1** の次に開始します。

3. **fs:1** — これはファイルシステムリソースです。**foo** サービス内に他のファイルシステムリソースがある場合は、**/etc/cluster/cluster.conf** の **foo** サービス部分にリストされている順序で開始します。

4. **ip:10.1.1.1** — これは IP アドレスリソースです。**foo** サービス内に他の IP アドレスのリソースがある場合は、**/etc/cluster/cluster.conf** の **foo** サービス部分にリストされている順序で開始します。

5. **script:1** — これはスクリプトリソースです。**foo** サービス内に他のスクリプトリソースがある場合は、**/etc/cluster/cluster.conf** の **foo** サービス部分にリストされている順序で開始します。

6. **nontypedresource:foo** — これは型指定されていないリソースです。型指定されていないリソースであるため、型指定されたリソースが開始した後に開始します。また、サービスリソース内での順序は、他の型指定されていないリソース **nontypedresourcetwo:bar** より前であるため、**nontypedresourcetwo:bar** の前に開始します (型指定されていないリソースはサービスリソース内に表れる順序で開始します)。

7. **nontypedresourcetwo:bar** — これは型指定されていないリソースです。型指定されていないリソースであるため、型指定されたリソースが開始した後に開始します。また、サービスリソース内での順序は、他の型指定されていないリソース **nontypedresource:foo** の後であるため、**nontypedresource:foo** の後に開始します (型指定されていないリソースはサービスリソース内に表れる順序で開始します)。

型指定されていない子リソースの停止順

例C.4「サービス内にある型指定されていないリソースと型指定されたリソース」 では、子リソースは以下の順序で停止します：

1. **nontypedresourcetwo:bar** — これは型指定されていないリソースです。型指定されていないリソースであるため、型指定されたリソースが停止する前に停止します。また、サービスリソース内での順序は、他の型指定されていないリソース **nontypedresource:foo** の後であるため、**nontypedresource:foo** の前に停止します (型指定されていないリソースはサービスリソース内に表れる逆の順序で停止します)。

2. **nontypedresource:foo** — これは型指定されていないリソースです。型指定されていないリソースであるため、型指定されたリソースが停止する前に停止します。また、サービスリソース内での順序は、他の型指定されていないリソース **nontypedresourcetwo:bar** より前であるため、**nontypedresourcetwo:bar** の後に停止します (型指定されていないリソースはサービスリソース内に表れる逆の順序で停止します)。

3. **script:1** — これはスクリプトリソースです。**foo** サービス内に他のスクリプトリソースがある場合は、**/etc/cluster/cluster.conf** の **foo** サービス部分にリストされている逆の順番で停止します。

4. **ip:10.1.1.1** — これは IP アドレスリソースです。**foo** サービス内に他の IP アドレスのリソースがある場合は、**/etc/cluster/cluster.conf** の **foo** サービス部分にリストされている逆の順番で停止します。

5. **fs:1** — これはファイルシステムリソースです。**foo** サービス内に他のファイルシステムリソースがある場合は、**/etc/cluster/cluster.conf** の **foo** サービス部分にリストされている逆の順番で停止します。

6. **lvm:2** — これは LVM リソースです。全ての LVM リソースは最後に停止します。**lvm:2** (**<lvm name="2" .../>**) は **lvm:1** より先に停止します。リソースタイプのグループ内のリ

ソース群は **/etc/cluster/cluster.conf** の **foo** サービス部分にリストされている逆の順番で停止します。

7. **lvm:1** — これは LVM リソースです。全ての LVM リソースは最後に停止します。**lvm:1** (**<lvm name="1" .../>**) は **lvm:2** の後に停止します。リソースタイプのグループ内のリソース群は **/etc/cluster/cluster.conf** の **foo** サービス部分にリストされている逆の順番で停止します。

C.3. 継承、<RESOURCES>ブロック、リソースの再利用

一部のリソースは、親リソースから値を継承することにメリットがあります。NFS サービスではこれが一般的に当てはまります。例C.5「リソース再利用と継承の為の NFS サービスセットアップ」 は、リソースの再利用と継承のためにセットアップされた標準的な NFS サービス設定を示しています。

例C.5 リソース再利用と継承の為の NFS サービスセットアップ

```
    <resources>
        <nfsclient name="bob" target="bob.example.com"
options="rw,no_root_squash"/>
        <nfsclient name="jim" target="jim.example.com"
options="rw,no_root_squash"/>
        <nfsexport name="exports"/>
    </resources>
    <service name="foo">
        <fs name="1" mountpoint="/mnt/foo" device="/dev/sdb1"
fsid="12344">
            <nfsexport ref="exports">  <!-- nfsexport's path and fsid
attributes
                                           are inherited from the
mountpoint &
                                           fsid attribute of the
parent fs
                                           resource -->
                <nfsclient ref="bob"/> <!-- nfsclient's path is
inherited from the
                                           mountpoint and the fsid
is added to the
                                           options string during
export -->
                <nfsclient ref="jim"/>
            </nfsexport>
        </fs>
        <fs name="2" mountpoint="/mnt/bar" device="/dev/sdb2"
fsid="12345">
            <nfsexport ref="exports">
                <nfsclient ref="bob"/> <!-- Because all of the critical
data for this
                                           resource is either
defined in the
                                           resources block or
inherited, we can
                                           reference it again! -->
                <nfsclient ref="jim"/>
            </nfsexport>
```

```
        </fs>
        <ip address="10.2.13.20"/>
    </service>
```

サービスが平坦（すなわち、親/子 関係が無い状態）である場合、以下のように設定される必要があります：

- サービスは 4 つの nfsclient リソースを必要とします — ファイルシステム毎に 1 つ (ファイルシステム群で計 2 つ)、そしてターゲットマシン毎に 1 つ (ターゲットマシン群で計 2 つ)
- サービスは各 nfsclient にエクスポートパスとファイルシステム ID を指定する必要があります。これにより、設定にエラーが生じる可能性があります。

一方、例C.5「リソース再利用と継承の為の NFS サービスセットアップ」 では NFS クライアントリソース **nfsclient:bob** と **nfsclient:jim** は、一度だけ定義されています。同じように NFS エキスポートリソース **nfsexport:exports** も一度だけ定義されています。リソースで必要とされる全ての属性は親リソースから継承されます。この継承属性は動的である (かつ相互に競合しない) ため、それらのリソースを再利用することは可能です — これがリソース群がリソースブロック内で定義される理由です。いくつかのリソースを複数の場所で設定することは実用的ではない場合があります。例えば、複数の場所でファイルシステムリソースを設定すると、1 つのファイルシステムを 2 つのノードにマウントすることになる場合があり、問題になります。

C.4. 障害からの回復と独立したサブツリー

ほとんどのエンタープライズ環境において、いずれかのサービスコンポーネントに障害があった場合にサービス障害から回復するには、通常サービス全体を再起動します。例えば、例C.6「サービス **foo** 障害からの通常の回復」 では、このサービス内に定義されたスクリプトのいずれかが失敗した場合、通常の手順としてはサービスの再起動 (又は、サービス回復ポリシーに応じて再配置や無効化) を行います。しかし、一部のケースではサービスの特定の部分が非クリティカルであると見なされる場合があるため、通常の回復を試行する前にサービスの適切な部分だけを再起動する必要がある場合があります。これを行うには、 ***__independent_subtree*** 属性を使用します。例えば、例C.7「***__independent_subtree*** 属性を使用した **foo** サービス障害からの回復」 では、***__independent_subtree*** 属性を使用して以下の動作を実行します：

- script:script_one が失敗すると、script:script_one、script:script_two、及び script:script_three を再起動します。
- script:script_two が失敗すると、script:script_two のみを再起動します。
- script:script_three が失敗すると、restart script:script_one、script:script_two、及び script:script_three を再起動します。
- script:script_four が失敗すると、全サービスを再起動します。

例C.6 サービス foo 障害からの通常の回復

```
<service name="foo">
      <script name="script_one" ...>
          <script name="script_two" .../>
      </script>
      <script name="script_three" .../>
</service>
```

例C.7 *__independent_subtree* 属性を使用した foo サービス障害からの回復

```
<service name="foo">
      <script name="script_one" __independent_subtree="1" ...>
          <script name="script_two" __independent_subtree="1" .../>
          <script name="script_three" .../>
      </script>
      <script name="script_four" .../>
</service>
```

一部の状況では、サービスの 1 つのコンポーネントに障害があった場合、サービス全体ではなく、そのコンポーネントのみを無効にすることをお勧めします。これは、そのサービスの他のコンポーネントを使用する他のサービスに影響を与えないようにするためです。Red Hat Enterprise Linux 6.1 リリース以降でこれを行うには、独立したサブツリーを非クリティカルとして指定する ***__independent_subtree="2"*** 属性を使用します。

注記

単一参照のリソースにのみ非クリティカルのフラグを使用できます。この非クリティカルフラグは、リソースツリーの全レベルにある全リソースで機能しますが、サービスや仮想マシンの定義時はトップレベルでは使用しないことをお勧めします。

Red Hat Enterprise Linux 6.1 リリース以降、独立したサブツリー用にリソースツリー内でノード毎に最大再開始数と再開始期限をセットすることができます。これらの閾値をセットするには、以下の属性を使用します:

- **__max_restarts** は、中断するまでに許容される再起動の最大回数を設定します。
- **__restart_expire_time** で設定する秒数が経過すると、再起動は試行されなくなります。

C.5. サービスとリソース順序へのデバッグとテスト

サービスとリソースの順序のデバッグとテストには、**rg_test** ユーティリティを使用できます。**rg_test** はコマンドラインユーティリティであり、シェル又はターミナルから実行する **rgmanager** パッケージにより提供されます（**Conga** にはありません）。表C.2「**rg_test** ユーティリティの要約」 は **rg_test** ユーティリティの動作と構文をまとめています。

表C.2 rg_test ユーティリティの要約

アクション	構文
rg_test が理解するリソースルールを表示する	**rg_test rules**

アクション	構文
エラー、又は余剰リソースエージェントの為に設定 (及び /usr/share/cluster) をテストする	**rg_test test /etc/cluster/cluster.conf**
サービスの開始/停止の順序を表示する	開始順を表示する： **rg_test noop /etc/cluster/cluster.conf start service *servicename*** 停止順を表示する： **rg_test noop /etc/cluster/cluster.conf stop service *servicename***
サービスを明示的に開始/停止する	 **重要** これを実行する場合は、1 つのノード上でのみ行ってください。また、常に rgmanager 内のサービスを最初に無効にしてください。 サービスを開始する： **rg_test test /etc/cluster/cluster.conf start service *servicename*** サービスを停止する： **rg_test test /etc/cluster/cluster.conf stop service *servicename***
２つの cluster.conf ファイル間でリソースのツリーデルタを計算して表示します。	**rg_test delta *cluster.conf file 1* *cluster.conf file 2*** 例えば： **rg_test delta /etc/cluster/cluster.conf.bak /etc/cluster/cluster.conf**

付録D クラスターサービスリソースチェックとフェイルオーバーのタイムアウト

本付録は、**rgmanager** がクラスターリソースのステータスをモニターする方法、ステータスチェック間隔を修正する方法について説明しています。また、動作がタイムアウトすることでサービスが失敗することを示す **__enforce_timeouts** サービスパラメーターについても説明しています。

注記

本付録の内容を十分に理解するためには、リソースエージェント及びクラスター設定ファイルである **/etc/cluster/cluster.conf** を詳しく理解していただく必要があります。**cluster.conf** の要素と属性の総括的な一覧と説明は、**/usr/share/cluster/cluster.rng** にあるクラススキーマと **/usr/share/doc/cman-X.Y.ZZ/cluster_conf.html** (例えば、**/usr/share/doc/cman-3.0.12/cluster_conf.html**) にある注釈付きスキーマを参照してください。

D.1. リソースステータスチェック間隔の修正

rgmanager は、全サービスではなく、個別のリソースのステータスを確認します。10 秒ごとに、rgmanager はリソースツリーをスキャンし、「ステータスチェック」間隔が経過したリソースを探します。

各リソースエージェントは、定期的に行われるステータスチェック間の期間を指定します。各リソースは、特別な **<action>** タグを使用して **cluster.conf** ファイルで明示的に上書きされていない限りは、これらのタイムアウト値を利用します。

```
<action name="status" depth="*" interval="10" />
```

このタグは **cluster.conf** ファイル内のリソースの特別な子です。例えば、ステータスチェックの間隔を上書きしたいファイルシステムリソースがある場合、**cluster.conf** ファイルのファイルシステムリソースを以下のように指定できます。

```
<fs name="test" device="/dev/sdb3">
  <action name="status" depth="*" interval="10" />
  <nfsexport...>
  </nfsexport>
</fs>
```

一部のエージェントは、複数の「深さ」でチェックを行います。例えば、通常のファイルシステムステータスチェック (深さ 0) は、ファイルシステムが正しい場所にマウントされているかを確認します。これより強度なチェックは深さ 10 であり、ファイルシステムからのファイルの読み取りが可能かどうかを確認します。深さ 20 のステータスチェックは、ファイルシステムへの書き込みが可能かどうかを確認します。ここの例では、**depth** は ***** に設定されています。これは、これらの値がすべての深さに使用されることを示しています。結果として、10 秒ごとにリソースエージェントにより最高に定義された深さ (この場合は 20) で **test** ファイルシステムを確認します。

D.2. リソースタイムアウトの強制

リソースの開始、停止、フェイルオーバーにタイムアウトはありません。一部のリソースは開始/停止に

長時間かかります。残念なことに、停止のエラー (タイムアウトを含む) が発生すると、サービスは動作不可能 (失敗した状態) になります。必要に応じて、個別にサービスの各リソースでタイムアウトを強制的に有効にすることもできます。そのためには、**__enforce_timeouts="1"** を **cluster.conf** ファイルの参照に追加します。

以下の例は、**__enforce_timeouts** 属性を **netfs** リソースに設定したクラスターサービスを示しています。この属性が設定された状態で、回復プロセス時に NFS ファイルシステムをアンマウントするのに 30 秒超かかる場合は、動作はタイムアウトになり、サービスは失敗します。

```
</screen>
<rm>
  <failoverdomains/>
  <resources>
    <netfs export="/nfstest" force_unmount="1" fstype="nfs"
host="10.65.48.65"
           mountpoint="/data/nfstest" name="nfstest_data"
options="rw,sync,soft"/>
  </resources>
  <service autostart="1" exclusive="0" name="nfs_client_test"
recovery="relocate">
    <netfs ref="nfstest_data" __enforce_timeouts="1"/>
  </service>
</rm>
```

付録E コマンドラインツールの要約

表E.1「コマンドラインツールの概要」 は、High Availability アドオンの設定と管理を行う場合に推奨されるコマンドラインツールを要約しています。コマンドと変数についての詳細情報は、それぞれのコマンドラインツールの man ページを参照してください。

表E.1 コマンドラインツールの概要

コマンドラインツール	使用対象	目的
ccs_config_dump — クラスター設定のダンプツール	クラスターインフラストラクチャー	**ccs_config_dump** は、実行している設定の XML 出力を生成します。一部のサブシステムはデフォルトの情報を設定に格納/セットすることがあるため、実行している設定がファイル上に保存されている設定と異なることがあります。これらの値は、通常設定のディスク上バージョンには存在しませんが、クラスターがランタイム時に正常に機能するために必要です。このツールに関する詳細は ccs_config_dump(8) の man ページを参照してください。
ccs_config_validate — クラスター設定の妥当性検証ツール	クラスターインフラストラクチャー	**ccs_config_validate** は、各ノード上の **/usr/share/cluster/cluster.rng** にあるスキーマ **cluster.rng** に対して **cluster.conf** の妥当性を検証します。このツールに関する詳細は ccs_config_validate(8) の man ページを参照してください。
clustat — クラスターステータスのユーティリティ	高可用性サービス管理のコンポーネント	**clustat** コマンドはクラスターのステータスを表示します。メンバーシップの情報、定足数表示、設定済みの全ユーザーサービスの状態を示します。このツールの詳細については clustat(8) の man ページを参照してください。
clusvcadm — クラスターユーザーサービス管理のユーティリティ	高可用性サービス管理のコンポーネント	**clusvcadm** コマンドを使用すると、クラスター内の高可用性サービスの有効、無効、再配置、再起動を行うことができます。このツールの詳細については clusvcadm(8) の man ページを参照してください。
cman_tool — クラスター管理ツール	クラスターインフラストラクチャー	**cman_tool** は CMAN クラスターマネージャーを管理するプログラムです。これにより、クラスターへの参加と離脱、ノードの kill、クラスター内でノードの定足数投票の期待値の変更が可能になります。このツールの詳細については cman_tool(8) の man ページを参照してください。

コマンドラインツール	使用対象	目的
fence_tool — フェンスツール	クラスターインフラストラクチャー	**fence_tool** はフェンスドメインに対する参加と離脱に使用するプログラムです。このツールの詳細については、ence_tool(8) の man ページを参照してください。

付録F HIGH AVAILABILITY LVM (HA-LVM)

Red Hat High Availability アドオンはフェイルオーバー設定の LVM ボリューム (HA-LVM) に対応しています。これは、Clustered Logical Volume Manager (CLVM) により有効になるアクティブ/アクティブ構成とは異なります。CLVM とは、コンピューターのクラスターが共有ストレージを管理できるようにする LVM のクラスタリング拡張機能セットです。

CLVM 又は HA-LVM は、デプロイされるアプリケーションやサービスの必要性に基づき使用することをお勧めします。

- アプリケーションがクラスター対応で、かつ一度に複数のマシンで同時に実行するようチューニングされている場合は、CLVM の使用をお勧めします。具体的には、使用しているクラスターの複数のノードにストレージへのアクセスが必要で、そのストレージがアクティブなノード間で共有される場合は、CLVM を使用する必要があります。論理ボリュームが設定されている間に物理ストレージへのアクセスをロックすることにより、ユーザーは CLVM を使用して共有ストレージ上で論理ボリュームを設定できます。また、CLVM はクラスター化されたロッキングサービスを使用して共有ストレージを管理します。CLVM 及び LVM 設定の全般情報は、『論理ボリュームマネージャ管理』 を参照してください。

- ストレージにアクセスする単一ノードのみがいつでもアクティブなアクティブ/パッシブ (フェイルオーバー) 構成でアプリケーションが最適に実行する場合は、High Availability Logical Volume Management (HA-LVM) エージェントの使用をお勧めします。

大半のアプリケーションは、他のインスタンスと同時に実行するよう設計/最適化されていないためアクティブ/パッシブ構成で適切に実行します。クラスター化された論理ボリュームでクラスター対応していないアプリケーションを実行することを選択した場合、論理ボリュームがミラーされるとパフォーマンスは低下します。こうしたインスタンスでは、論理ボリュームにクラスター通信オーバーヘッドがあるためです。クラスター対応のアプリケーションは、クラスターファイルシステム及びクラスター対応論理ボリュームにより生じるパフォーマンス低下を超えた高いパフォーマンスを実現できるはずです。これは、一部のアプリケーションとワークロードにとっては、他より簡単に実現できます。クラスターの要件、及びアクティブ/アクティブクラスターを最適化するための別途作業が役立つかどうかは、2 つの LVM バリアント間で選択する方法によって決まります。大半のユーザーは、HA-LVM を使用することで最適な HA 結果を得ることができます。

HA-LVM と CLVM が類似しているのは、両方とも LVM メタデータとその論理ボリュームの破損を防ぐ点です。複数のマシンが重複する変更を実行できるようになっていると、この破損が発生します。HA-LVM は、論理ボリュームは排他的にのみアクティベートできる、という制限を課します。つまり、一度に 1 つのマシン上でのみアクティブであるということです。これは、ローカルで (非クラスター化) 実装されたストレージドライバーのみが使用されることを意味します。この方法で調整されたクラスターのオーバーヘッドを回避することで、パフォーマンスが向上します。CLVM はこうした制限を課しません。つまり、ユーザーはクラスター内の全マシンの論理ボリュームを自由にアクティベートできます。これにより、クラスター対応のストレージドライバーを強制的に使用することになるため、クラスター対応のファイルシステムおよびアプリケーションが上部に置かれます。

HA-LVM を設定するには、単独の論理ボリュームをアクティブ化する命令を行う 2 つのメソッドのうち 1 つを使用します。

- 推奨されるメソッドは CLVM の使用ですが、これは論理ボリュームを単独でアクティブ化するだけです。これによるメリットは、簡単に設定できること、管理上のミス (使用中の論理ボリュームの削除など) の回避が向上することです。CLVM を使用するには、**clvmd** デーモンを含む High Availability アドオン及び Resilient Storage アドオンソフトウェアが実行されている必要があります。

  このメソッドを使用して HA-LVM を設定する手順は、「CLVM を使用した HA-LVM フェイルオーバーの設定 (推奨)」 に記載されています。

- 2 番目のメソッドは、ローカルマシンのロックと LVM 「タグ」の使用です。このメソッドのメリットは、LVM クラスターパッケージが必要でない点です。一方で、設定する際に必要な手順は他にもあり、管理者がアクティブでないクラスター内のノードから論理ボリュームを誤って削除することを防ぐことはできません。このメソッドを使用して HA-LVM を設定する手順は、「タグ付けによる HA-LVM フェイルオーバーの設定」 に記載されています。

F.1. CLVM を使用した HA-LVM フェイルオーバーの設定 (推奨)

(推奨される CLVM バリアントを使用して) HA-LVM フェイルオーバーを設定するには、以下の手順を実行します。

1. ご使用のシステムが CLVM に対応するよう設定されていることを確認してください。以下が要件です。
   - CLVM 論理ボリュームがミラーされる場合は **cmirror** パッケージも含め、High Availability アドオン及び Resilient Storage アドオンがインストールされている。
   - **/etc/lvm/lvm.conf** ファイル内のグローバルセクションの **locking_type** パラメーターが「3」の値に設定されている。
   - **clvmd** デーモンを含め、High Availability アドオン及び Resilient Storage アドオンソフトウェアが実行している必要がある。CLVM ミラーリングの場合は、**cmirrord** サービスも起動している必要があります。
2. 以下の例のとおり、標準 LVM 及びファイルシステムのコマンドを使用して論理ボリュームとファイルシステムを作成します。

```
# pvcreate /dev/sd[cde]1

# vgcreate -cy shared_vg /dev/sd[cde]1

# lvcreate -L 10G -n ha_lv shared_vg

# mkfs.ext4 /dev/shared_vg/ha_lv

# lvchange -an shared_vg/ha_lv
```

   LVM 論理ボリュームの作成についての詳細は、『論理ボリュームマネージャの管理』 を参照してください。

3. **/etc/cluster/cluster.conf** ファイルを編集して、新規作成された論理ボリュームを使用しているサービスの 1 つにリソースとして含めます。別の方法として、**Conga** 又は **ccs** コマンドを使用して、LVM 及びファイルシステムリソースをクラスター用に設定することも可能です。以下の例は、クラスターリソースとして CLVM 論理ボリュームを設定する **/etc/cluster/cluster.conf** ファイルのリソースマネージャーのセクションです。

```
<rm>
   <failoverdomains>
       <failoverdomain name="FD" ordered="1" restricted="0">
          <failoverdomainnode name="neo-01" priority="1"/>
          <failoverdomainnode name="neo-02" priority="2"/>
       </failoverdomain>
   </failoverdomains>
   <resources>
```

```
        <lvm name="lvm" vg_name="shared_vg" lv_name="ha-lv"/>
        <fs name="FS" device="/dev/shared_vg/ha-lv" force_fsck="0"
force_unmount="1" fsid="64050" fstype="ext4" mountpoint="/mnt"
options="" self_fence="0"/>
   </resources>
   <service autostart="1" domain="FD" name="serv"
recovery="relocate">
        <lvm ref="lvm"/>
        <fs ref="FS"/>
   </service>
</rm>
```

F.2. タグ付けによる HA-LVM フェイルオーバーの設定

/etc/lvm/lvm.conf ファイルのタグを使用して HA-LVM フェイルオーバーを設定するには、以下の手順を実行します。

1. **/etc/lvm/lvm.conf** ファイル内のグローバルセクションの **locking_type** パラメーターが「1」の値に設定されているようにしてください。

2. 以下の例のとおり、標準 LVM 及びファイルシステムのコマンドを使用して論理ボリュームとファイルシステムを作成します。

   ```
   # pvcreate /dev/sd[cde]1

   # vgcreate shared_vg /dev/sd[cde]1

   # lvcreate -L 10G -n ha_lv shared_vg

   # mkfs.ext4 /dev/shared_vg/ha_lv
   ```

   LVM 論理ボリュームの作成についての詳細は、『論理ボリュームマネージャの管理』 を参照してください。

3. **/etc/cluster/cluster.conf** ファイルを編集して、新規作成された論理ボリュームを使用しているサービスの 1 つにリソースとして含めます。別の方法として、**Conga** 又は **ccs** コマンドを使用して、LVM 及びファイルシステムリソースをクラスター用に設定することも可能です。以下の例は、クラスターリソースとして CLVM 論理ボリュームを設定する **/etc/cluster/cluster.conf** ファイルのリソースマネージャーのセクションです。

   ```
   <rm>
      <failoverdomains>
          <failoverdomain name="FD" ordered="1" restricted="0">
             <failoverdomainnode name="neo-01" priority="1"/>
             <failoverdomainnode name="neo-02" priority="2"/>
          </failoverdomain>
      </failoverdomains>
      <resources>
          <lvm name="lvm" vg_name="shared_vg" lv_name="ha_lv"/>
          <fs name="FS" device="/dev/shared_vg/ha_lv" force_fsck="0"
   force_unmount="1" fsid="64050" fstype="ext4" mountpoint="/mnt"
   options="" self_fence="0"/>
      </resources>
   ```

```
   <service autostart="1" domain="FD" name="serv"
recovery="relocate">
       <lvm ref="lvm"/>
       <fs ref="FS"/>
   </service>
</rm>
```

注記

ボリュームグループに論理ボリュームが複数ある場合は、**lvm** リソースの論理ボリューム名 (**lv_name**) は、空欄か指定しないようにしてください。また、HA-LVM 設定では、ボリュームグループは単一のサービスによってのみ使用されることがある点に注意してください。

4. **/etc/lvm/lvm.conf** ファイルの **volume_list** フィールドを編集します。**/etc/cluster/cluster.conf** ファイルに一覧表示されているとおり、ご使用の root ボリュームグループ名とホスト名は @ を先頭に付けて入力してください。ここに含めるホスト名は、編集している **lvm.conf** ファイルがあるマシンであり、リモートのホスト名ではありません。なお、この文字列は、**cluster.conf** ファイル内のノード名とマッチ **しなければなりません**。以下は、**/etc/lvm/lvm.conf** ファイルのサンプルエントリです。

```
volume_list = [ "VolGroup00", "@neo-01" ]
```

このタグは、共有 VG 又は LV をアクティブ化するために使用します。HA-LVM を使用して共有されるボリュームグループ名は **含めないようにしてください**。

5. 使用しているクラスターノード上の **initrd** デバイスを更新します。

```
# dracut -H -f /boot/initramfs-$(uname -r).img $(uname -r)
```

6. 全ノードを再起動して、正しい **initrd** デバイスが使用中であることを確認してください。

付録G 改訂履歴

改訂 6.0-21.3 **Mon Jul 21 2014** **Chester Cheng**
Fix typo https://bugzilla.redhat.com/show_bug.cgi?id=1072699

改訂 6.0-21.2 **Mon Apr 7 2014** **Red Hat Localization Services**
翻訳完成

改訂 6.0-21.1 **Mon Apr 7 2014** **Red Hat Localization Services**
翻訳ファイルを XML ソースバージョン 6.0-21 と同期

改訂 6.0-21 **Wed Nov 13 2013** **Steven Levine**
6.5 GA リリース向けのバージョン

改訂 6.0-20 **Wed Nov 6 2013** **Steven Levine**
#986462 を解決
oracledb リソース表を更新

改訂 6.0-16 **Tue Oct 29 2013** **Steven Levine**
#1021045 を解決
iptables ルールの例を修正

改訂 6.0-15 **Fri Sep 27 2013** **Steven Levine**
6.5 ベータリリース向けのバージョン

改訂 6.0-12 **Thu Sep 26 2013** **Steven Levine**
#884758、#893575、#969525、#969139、#987151、#987623 を解決
フェンスデバイスパラメーター表のマイナー更新

#901637、#983739、#986462 を更新
リソースパラメーター表のマイナー更新

#633495 を更新
`nfsexport` および `nfsserver` リソースの設定を文書化

#852966、#975512、#977194、#991297、#874211、#908328、#919600、#955405、#972521、#986474、#987135、#698454、#967986 を解決
本ドキュメント全体にわたり、マイナーな修正および明確化

改訂 6.0-3 **Thu Sep 05 2013** **Steven Levine**
すべての 6.5 BZ で主要レビュー改訂

改訂 6.0-2 **Fri Jun 14 2013** **Steven Levine**
nfsserver および nfsexport 設定セクションを追加

改訂 6.0-1 **Thu Jun 13 2013** **Steven Levine**
クラスター更新の手順および qdisk の考慮すべき点を更新

改訂 5.0-25 **Mon Feb 18 2013** **Steven Levine**
6.4 GA リリース向けバージョン

改訂 5.0-23 **Wed Jan 30 2013** **Steven Levine**
バグを修正: 901641
iptable ルールの修正、明確化

改訂 5.0-22 **Tue Jan 29 2013** **Steven Levine**

バグを修正: 788636
ccs コマンドを使った RRP 設定を文書化

バグを修正: 789010
cluster.conf ファイルの RRP 設定を文書化

改訂 5.0-20 **Fri Jan 18 2013** **Steven Levine**
バグを修正: 894097
VLAN タグを使用しないようにするというアドバイスを削除

バグを修正: 845365
ボンディングモード 0 と 2 に対応するようになったことを記載

改訂 5.0-19 **Thu Jan 17 2013** **Steven Levine**
バグを修正: 896234
クラスターノード参照の用語を明確化

改訂 5.0-16 **Mon Nov 26 2012** **Steven Levine**
6.4 ベータリリース向けのバージョン

改訂 5.0-15 **Wed Nov 20 2012** **Steven Levine**

バグを修正: 838988
ファイルシステムのリソースエージェントの nfsrestart 属性を文書化

バグを修正: 843169
IBM iPDU フェンスエージェントを文書化

バグを修正: 846121
Eaton Network Power Controller (SNMP Interface) フェンスエージェントを文書化

バグを修正: 856834
HP Bladesystem フェンスエージェントを文書化

バグを修正: 865313
NFS サーバーのリソースエージェントを文書化

バグを修正: 862281
どの **ccs** コマンドが以前の設定を上書きするかを明確化

バグを修正: 846205
igmp コンポーネントの **iptables** ファイアウォールフィルタリングについて文書化

バグを修正: 857172
luci からユーザーを削除する機能を文書化

バグを修正: 857165
IPMI フェンスエージェントの権限レベルのパラメーターについて文書化

バグを修正: 840912
リソースパラメーターテーブルのフォーマットに関する問題を解決

バグを修正: 849240、870292
インストール手順を明確化

バグを修正: 871165
IP アドレスのリソースエージェントに関する説明の IP アドレスパラメーターの箇所を明確化

バグを修正: 845333、869039、856681
誤植の修正、技術面で若干あいまいな部分を明確化

改訂 5.0-12 **Thu Nov 1 2012** **Steven Levine**
新たに対応されたフェンスエージェントを追加

改訂 5.0-7 **Thu Oct 25 2012** **Steven Levine**
オーバーライドのセマンティクスに関する項を追加

改訂 5.0-6 **Tue Oct 23 2012** **Steven Levine**
Post Join Delay のデフォルト値を修正

改訂 5.0-4 **Tue Oct 16 2012** **Steven Levine**
NFS サーバーリソースの説明を追加

改訂 5.0-2 **Thu Oct 11 2012** **Steven Levine**
Conga の説明を更新

改訂 5.0-1 **Mon Oct 8 2012** **Steven Levine**
ccs セマンティクスを明確化

改訂 4.0-5 **Fri Jun 15 2012** **Steven Levine**

6.3 GA リリース向けバージョン

改訂 4.0-4 **Tue Jun 12 2012** **Steven Levine**

#830148 を解決
luci のサンプルポート番号の整合性を図る

改訂 4.0-3 **Tue May 21 2012** **Steven Levine**

#696897 を解決
cluster.conf パラメーターの情報をフェンスデバイスパラメーター及びリソースパラメーターの表に追加

#811643 を解決
別のマシンで **luci** データベースを復元する手順を追加

改訂 4.0-2 **Wed Apr 25 2012** **Steven Levine**

#815619 を解決
GFS2 ファイルシステムによる UDP Unicast の使用に関する警告を削除

改訂 4.0-1 **Fri Mar 30 2012** **Steven Levine**

#771447、#800069、#800061 を解決
Red Hat Enterprise Linux 6.3 バージョンと一貫性を保つため **luci** のドキュメントを更新

#712393 を解決
RGManager のアプリケーションコアをキャプチャする情報を追加

#800074 を解決
condor リソースエージェントについて文書化

#757904 を解決
luci 設定バックアップと復元について文書化

#772374 を解決
クラスター内での仮想マシンの管理に関するセクションを追加

#712378 を解決
HA-LVM 設定に関するドキュメントを追加

#712400 を解決
デバックオプションについて文書化

#751156 を解決
新しい **fence_ipmilan** パラメーターについて文書化

#721373 を解決
クラスターの再起動が必要となる設定変更を文書化

改訂 3.0-5 **Thu Dec 1 2011** **Steven Levine**

Red Hat Enterprise Linux 6.2 の GA 向けリリース

#755849 を解決
monitor_link パラメーターの例を訂正

改訂 3.0-4 **Mon Nov 7 2011** **Steven Levine**

#749857 を解決
RHEV-M REST API フェンスデバイスの文書を追加

改訂 3.0-3 **Fri Oct 21 2011** **Steven Levine**

#747181、#747182、#747184、#747185、#747186、#747187、#747188、#747189、#747190、#747192 を解決
Red Hat Enterprise Linux 6 の QE ドキュメントレビュー時に見つかった誤字及び曖昧さを修正

改訂 3.0-2 **Fri Oct 7 2011** **Steven Levine**

#743757 を解決
トラブルシューティングのセクションにあるサポートされているボンディングモードへの参照を訂正

改訂 3.0-1 **Wed Sep 28 2011** **Steven Levine**

Red Hat Enterprise Linux 6.2 Beta リリース向けの初回バージョン

#739613 を解決
利用可能なフェンスデバイス及びサービスを表示するための新しい **ccs** オプションに対するサポートを文書化

#707740 を解決
Conga インターフェースの更新及び Conga を管理するためのユーザーパーミッションを設定するサポートについて文書化

#731856 を解決
/etc/sysconfig/luci ファイルを使用して **luci** を設定するサポートについて文書化

#736134 を解決
UDPU トランスポートに対するサポートについて文書化

#736143 を解決
Clustered Samba に対するサポートについて文書化

#617634 を解決
luci が機能する IP アドレスのみを設定する方法を文書化

#713259 を解決
fence_vmware_soap エージェントに対するサポートを文書化

#721009 を解決
サポートエッセンシャルの記事へのリンクを提供

#717006 を解決
iptables ファイアウォールによりマルチキャストトラフィックを可能にする情報を記載

#717008 を解決
クラスターサービスのステータスチェックとフェイルオーバータイムアウトに関する情報を追加

#711868 を解決
autostart の詳細を明確化

#728337 を解決
ccs コマンドを使って、**vm** リソースを追加する手順を文書化

#725315、#733011、#733074、#733689 を解決
若干の誤字/誤植を修正

改訂 2.0-1 **Thu May 19 2011** **Steven Levine**

Red Hat Enterprise Linux 6.1 向けの初回改訂

#671250 を解決
SNMP トラップに対するサポートについて文書化

#659753 を解決
ccs コマンドの文書化

#665055 を解決
更新したディスプレイと機能のサポートを反映するために Conga ドキュメントを更新

#680294 を解決
ricci エージェントへのアクセスにパスワードが必要な旨を文書化

#687871 を解決
トラブルシューティングの章を追加

#673217 を解決
誤字/誤植を修正

#675805 を解決
cluster.conf スキーマへの参照を HA リソースパラメーターの表に追加

#672697 を解決
現在サポートされている全てのフェンシングデバイスを含めるためにフェンスデバイスパラメーターの表を更新

#677994 を解決
fence_ilo フェンスエージェントパラメーターの情報を修正

#629471 を解決
2 ノードクラスターの consensus 値の設定に関する技術的な注記を追加

#579585 を解決
Red Hat High Availability アドオンソフトウェアのアップグレードに関するセクションを更新

#643216 を解決
ドキュメント全体に渡る小さな問題を明確化

#643191 を解決
luci ドキュメントの改善と修正

#704539 を解決
仮想マシンのリソースパラメーターの表を更新

改訂 1.0-1 **Wed Nov 10 2010** **Paul Kennedy**
Red Hat Enterprise Linux 6 用の初回リリース

索引

シンボル

H

I

L

N

Q

R

S

T

V

W