SUSE Linux Enterprise High Availability Extension

11 www.novell.com

2009 年7 月31 日 High Availability ガイド

High Availability ガイド

目次

このガイドについて

SUSE® Linux Enterprise High Availability Extensionはオープンソースクラスタリングテクノロジの統合スイートで、高可用性を備えた物理および仮想Linuxクラスタを実装できます。構成と管理をすばやく効率的に行うため、High Availability Extensionにはグラフィカルユーザインタフェース(GUI)とコマンドラインインタフェース(CLI)の両方が備わっています。

このガイドは、High Availability(HA)クラスタのセットアップ、構成、保守を行う必要がある管理者向けに作成されています。両方のアプローチ(GUIとCLI)について詳細に記述し、管理者が主要タスクの実行に必要な、適切なツールを選択できるよう支援します。

ガイドは、次の各部に分かれています。

インストールおよび管理
: クラスタのインストールおよび構成を開始する前に、クラスタの基礎およびアーキテクチャを理解し、主要機能とその利点の概要、および最新リリース以降の変更点を知る必要があります。必要なハードウェアおよびソフトウェア要件、次の手順を実行する前に必要な準備作業について説明します。YaSTを使用してHAクラスタのインストールおよび基本セットアップを実行してください。

設定および管理
: GUIまたは`crm`コマンドラインインタフェースを使用して、リソースを追加、構成、管理します。負荷分散およびフェンシングの使用方法を説明します。独自のリソースエージェントの作成、または既存のエージェントの変更を検討している場合、別の種類のリソースエージェントを作成する方法について背景情報を取得できます。

ストレージおよびデータレプリケーション
: SUSE Linux Enterprise High Availability Extensionには、クラスタを認識するファイルシステム(Oracle Cluster File System: OCFS2)とボリュームマネージャ(clustered Logical Volume Manager: cLVM)が付属しています。データのレプリケーションについては、High Availability Extensionにひあ DRBD(Distributed Replicated Block Device)も付属しており、高可用性サービスのデータをクラスタのアクティブノードからスタンバイノードへのミラーリングに使用できます。

トラブルシューティングと参照情報
独自のクラスタの管理には、一定量のトラブルシューティングを実行する必要があります。よくある問題とその解決方法について説明します。独自のクラスタを管理するための、High Availability Extensionのコマンドラインツールの総合的なリファレンスを用意しています。また、クラスタリソースとリソースエージェントについての最も重要な事実と図を取り上げています。

このマニュアル中の多くの章に、他の資料やリソースへのリンクが記載されています。これらの資料の中には、システムから参照できるものもあれば、インターネット上に公開されているものもあります。

ご使用製品の利用可能なマニュアルと最新のドキュメントアップデートの概要については、`http://www.novell.com/documentation`を参照してください。

1 フィードバック

次のフィードバックチャネルがあります:

- 製品コンポーネントのバグの報告や、改善強化要求の提出には、`https://bugzilla.novell.com/`を使用してください。Bugzillaを初めてご使用になる場合は、 Novell Bugzilla ホームページの*Bug Writing FAQs*が参考になることがあります。
- 本マニュアルおよびこの製品に含まれているその他のマニュアルについて、皆様のご意見やご要望をお寄せください。オンラインドキュメントの各ページの下部にあるユーザコメント機能を使用して、コメントを入力してください。

2 マニュアルの表記規則

本書では、次の書体を使用しています:

- `/etc/passwd`:ディレクトリ名とファイル名
- *`placeholder`*:*`placeholder`*は、実際の値で置き換えられます

- `PATH`:環境変数PATH
- `ls`、`--help`:コマンド、オプション、およびパラメータ
- `user`:ユーザまたはグループ
- Alt、Alt + F1:キー:押すためのキーまたはキーの組み合わせ、キーはキーボードと同様に、大文字で表示されます
- *ファイル*、*ファイル* > *名前を付けて保存*:メニュー項目、ボタン
- この項は、指定されたアーキテクチャにのみ関連しています。矢印は、テキストブロックの先頭と終わりを示します。

  この項は、指定されたアーキテクチャにのみ関連しています。矢印は、テキストブロックの先頭と終わりを示します。
- *Dancing Penguins* (*Penguins*章、↑他のマニュアル):他のマニュアルの章への参照です。.

パート I. インストールおよび管理

1 概念の概要

SUSE® Linux Enterprise High Availability Extensionはオープンソースクラスタ化技術の統合スイートで、可用性の高い物理的および仮想Linuxクラスタを実装し、SPOF (単一障害点)をなくします。データ、アプリケーション、サービスなどの重要なネットワークリソースの高可用性と管理のしやすさを実現します。その結果、ミッションクリティカルなLinuxワークロードに対してビジネスの継続性維持、データ整合性の保護、予期せぬダウンタイムの削減を行います。

基本的な監視、メッセージング、およびクラスタリソース管理機能が標準装備されており、個別に管理されるクラスタリソースのフェールオーバー、フェールバック、およびマイグレーション(負荷分散)をサポートしています。High Availability ExtensionはSUSE Linux Enterprise Server 11へのアドオンとして提供されています。

1.1 製品の機能

SUSE® Linux Enterprise High Availability Extensionを使用することで、ネットワークリソースの可用性を確保し、維持することができます。次のリストは主要な機能を示したものです。

各種のクラスタリングシナリオをサポート
: アクティブ/アクティブおよびアクティブ/パッシブ(N+1、N+M、N対1、N対M)シナリオと、物理的および仮想クラスタのハイブリッド(仮想サーバを物理クラスタとクラスタ化して、サービスの可用性とリソースの利用率を向上)にも対応しています。

最大16のLinuxサーバを含むマルチノードアクティブクラスタ。クラスタ内のどのサーバも、クラスタ内の障害が発生したサーバのリソース(アプリケーション、サービス、IPアドレス、およびファイルシステム)を再起動することができます。

柔軟なソリューション
: High Availability ExtensionにはOpenAISメッセージングとメンバーシップ層、およびPacemakerクラスタリソースマネージャが搭載されています。Pacemakerを使用することで管理者は継続的にリソースのヘルスを監視し、依存性を管理し、柔軟に設定できるルールやポリシーに基づいてサービスの自動開始と停止が行えます。High Availability Extensionではユーザの組織に合わせて特定のアプリケーションおよびハードウェアインフラストラクチャに応じたクラスタのカスタマイズが可能です。時間依存設定を使用して、サービスを特定の時刻に修復済みのノードに自動的にフェールバックさせることができます。

ストレージとデータレプリケーション
: High Availability Extensionでは必要に応じてサーバストレージを自動的に割り当て、再割り当てすることができます。ファイバチャネルまたはiSCSIストレージエリアネットワーク(SAN)をサポートしています。共有ディスクもサポートされていますが、必要要件ではありません。SUSE Linux Enterprise High Availability Extensionにはクラスタ対応ファイルシステム(Oracle Cluster File System、OCFS2)とボリュームマネージャ(クラスタ化された論理ボリュームマネージャ、cLVM)も含まれています。データのレプリケーションについては、High Availability ExtensionにひあDRBD(Distributed Replicated Block Device)も付属しており、高可用性サービスのデータをクラスタのアクティブノードからスタンバイノードへのミラーリングに使用できます。

仮想化環境のサポート
: SUSE Linux Enterprise High Availability Extensionは物理的および仮想Linuxサーバの両方が混在したクラスタリングをサポートしています。SUSE Linux Enterprise Server 11にはオープンソース仮想化hypervisorであるXenが搭載されています。High Availability Extension内のクラスタリソースマネージャは、Xenで作成された仮想サーバ内のサービスと、物理サーバで実行中のサービスを認識、監視、および管理することができます。ゲストシステムは、クラスタにサービスとして管理されます。

リソースエージェント
: SUSE Linux Enterprise High Availability ExtensionにはApache、IPv4、IPv6、その他多数のリソースを管理するための膨大な数のリソースエージェントが含まれています。またIBM WebSphere Application Serverなどの一般的なサードパーティアプリケーション用のリソースエージェントも含まれています。ご利用の製品に含まれているOpen Cluster Framework (OCF)リソースエージェントのリストは、第18章 *HA OCF*エージェント (209 ページ)を参照してください。また、最新のリストは`www.novell.com/products/highavailability`にオンラインで提供されています。

ユーザフレンドリな管理
: 簡単に設定や管理ができるよう、High Availability Extensionにはグラフィカルユーザインタフェース(YaSTやLinux HA Management Clientなど)と強力な統合コマンドラインインタフェースの両方が搭載されています。いずれのアプローチでも、クラスタを一元管理して効果的に監視および管理することができます。次の章でその方法を説明します。

1.2 製品のメリット

High Availability Extensionでは最大16台のLinuxサーバを可用性の高いクラスタ(HAクラスタ)に構成し、クラスタ内の任意のサーバにリソースをダイナミックに切り替えたり、移動することができます。サーバ障害発生時のリソースの自動マイグレーションの設定ができます。また、ハードウェアのトラブルシューティングやワークロードのバランスをとるために、リソースを手動で移動することもできます。

High Availability Extensionは一般的なコンポーネントを使用して、高度な可用性を実現します。アプリケーションと操作をクラスタに統合することによって、運用コストを削減できます。またHigh Availability Extensionではクラスタ全体を一元管理し、変化するワークロード要件に応じてリソースを調整することもできます(手動でのクラスタの「負荷分散」)。3ノード以上でクラスタを構成すると、複数のノードが「ホットスペア」を共用できて無駄がありません。

その他にも重要な利点として、予測できないサービス停止を削減したり、ソフトウェアおよびハードウェアの保守やアップグレードのための計画的なサービス停止を削減できる点が挙げられます。

次に、クラスタによるメリットについて説明します。

- 可用性の向上
- パフォーマンスの改善
- 運用コストの低減
- スケーラビリティ
- 障害回復
- データの保護
- サーバの集約
- ストレージの集約

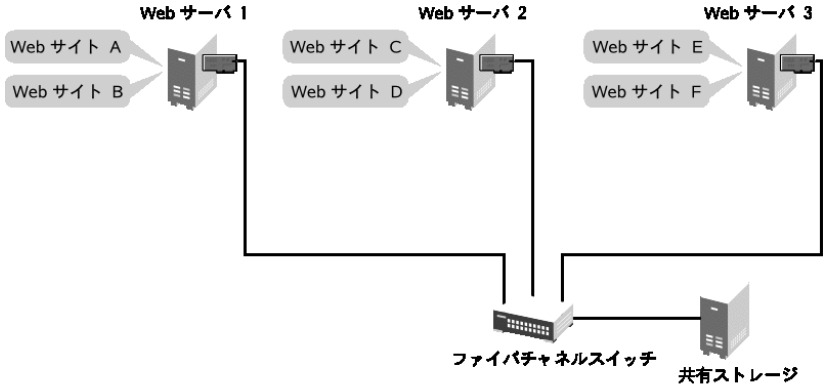
共有ディスクサブシステムにRAIDを導入することによって、共有ディスクの耐障害性を強化できます。

次のシナリオは、High Availability Extensionの利点を紹介するものです。

クラスタシナリオ例

サーバ3台でクラスタが構成され、それぞれのサーバにWebサーバをインストールしたと仮定します。クラスタ内の各サーバが、2つのWebサイトをホストしています。各Webサイトのすべてのデータ、グラフィックス、Webページコンテンツは、クラスタ内の各サーバに接続された、共有ディスクサブシステムに保存されています。次の図は、このクラスタの構成を示しています。

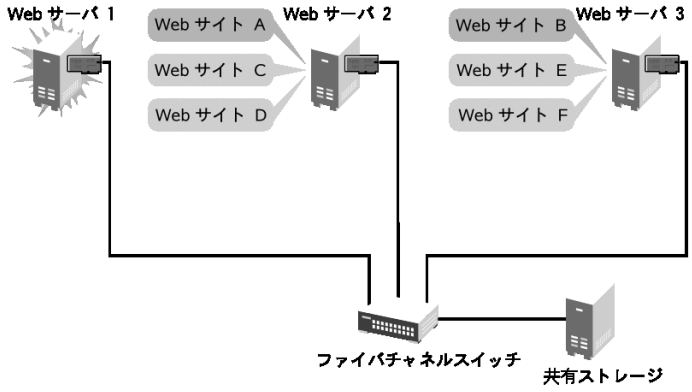
図 1.1　3サーバクラスタ



通常のクラスタ操作では、クラスタ内の各サーバが他のサーバと常に交信し、すべての登録済みリソースを定期的にポーリングして、障害を検出します。

Webサーバ1でハードウェアまたはソフトウェアの障害が発生したため、このサーバを利用してインターネットアクセス、電子メール、および情報収集を行っているユーザの接続が切断されたとします。次の図は、Webサーバ1で障害が発生した場合のリソースの移動を表したものです。

図 1.2　1台のサーバに障害が発生した後の3サーバクラスタ



WebサイトAがWebサーバ2に、WebサイトBがWebサーバ3に移動します。IPアドレスと証明書もWebサーバ2とWebサーバ3に移動します。

クラスタを設定するときに、それぞれのWebサーバがホストしているWebサイトについて、障害発生時の移動先を指定します。先に説明した例では、WebサイトAの移動先としてWebサーバ2が、WebサイトBの移動先としてWebサーバ3が指定されています。このようにして、Webサーバ1によって処理されていたワークロードが、残りのクラスタメンバーに均等に分散され、可用性を維持できます。

Webサーバ1に障害が発生すると、High Availability Extensionソフトウェアは次のような処理を行います。

- 障害を検出し、Webサーバ1が本当に機能しなくなっていることをSTONITHを使用して検証
- Webサーバ1にマウントされていた共有データディレクトリを、Webサーバ2およびWebサーバ3に再マウント
- Webサーバ1で動作していたアプリケーションを、Webサーバ2およびWebサーバ3で再起動
- IPアドレスをWebサーバ2およびWebサーバ3に移動

この例では、フェールオーバープロセスが迅速に実行され、ユーザはWebサイトの情報へのアクセスを数秒程度で回復できます。多くの場合、再度ログインする必要はありません。

ここで、Webサーバ1で発生した問題が解決し、通常に操作できる状態に戻ったと仮定します。WebサイトAおよびWebサイトBは、Webサーバ1に自動的にフェールバック(復帰)することも、そのままの状態を維持することもできます。これは、リソースの設定方法によって決まります。Webサーバ1へのマイグレーションは多少のダウンタイムを伴うため、High Availability Extensionではサービス中断がほとんど、またはまったく発生しないタイミングまでマイグレーションを延期することもできます。いずれの場合でも利点と欠点があります。

High Availability Extensionはリソースマイグレーション機能も提供しています。アプリケーション、Webサイトなどをシステム管理の必要性に応じて、クラスタ内の他のサーバに移動することができます。

たとえば、WebサイトAまたはWebサイトBをWebサーバ1からクラスタ内の他のサーバに手動で移動することができます。これは、Webサーバ1のアップグ

レードや定期メンテナンスを実施する場合、また、Webサイトのパフォーマンスやアクセスを向上させる場合に有効な機能です。

1.3 クラスタ構成

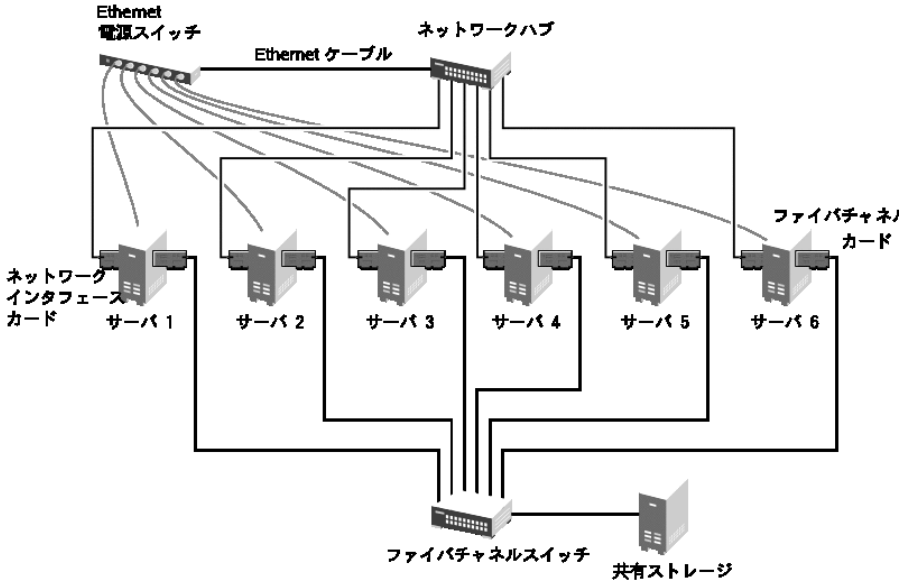
High Availability Extensionでのクラスタ構成には、共有ディスクサブシステムが含まれる場合と含まれない場合があります。共有ディスクサブシステムの接続には、高速ファイバチャネルカード、ケーブル、およびスイッチを使用でき、また構成にはiSCSIを使用することができます。サーバの障害発生時には、障害が発生したサーバにマウントされていた共有ディスクのディレクトリが、クラスタ内で指定された他のサーバ上に自動的にマウントされます。この機能によって、ネットワークユーザは、共有ディスクサブシステム上のディレクトリに対するアクセスを中断することなく実行できます。

重要項目: cLVMを伴う共有ディスクサブシステム

共有ディスクサブシステムをcLVMと使用する場合、クラスタ内の、アクセスが必要なすべてのサーバにそのサブシステムを接続する必要があります。

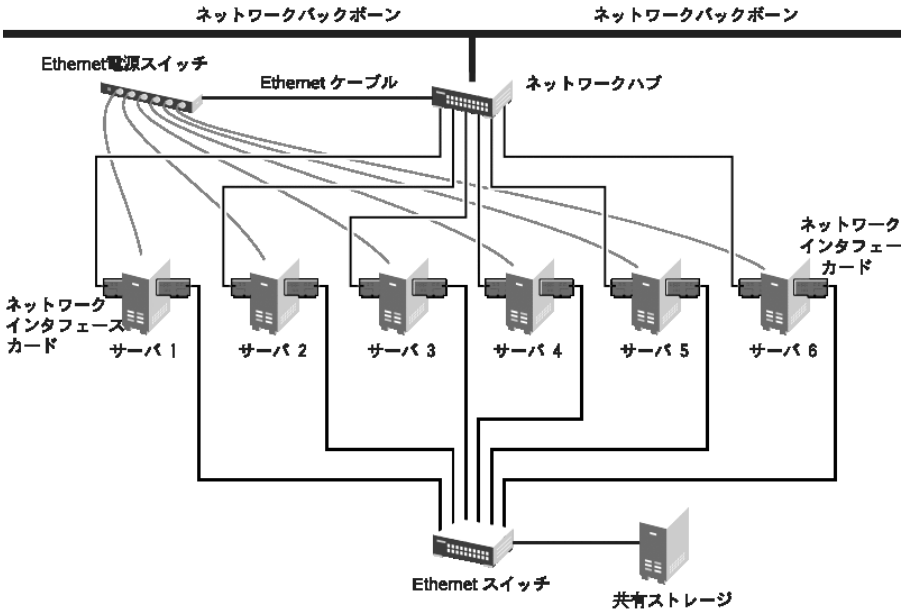
一般的なリソースの例としては、データ、アプリケーション、およびサービスなどがあります。次の図は、一般的なファイバチャネルクラスタの構成を表したものです。

図 1.3　一般的なファイバチャネルクラスタの構成



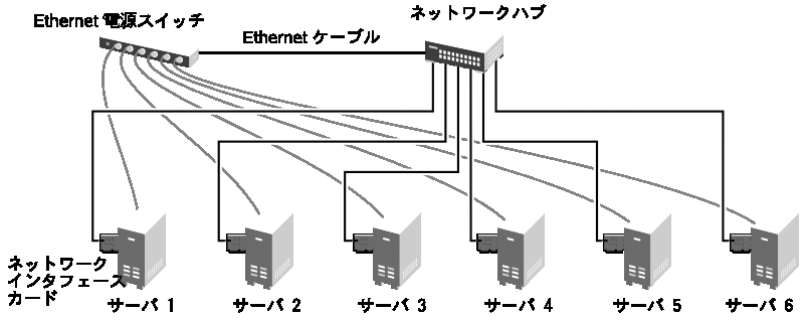
ファイバチャネルは最も高いパフォーマンスを提供しますが、iSCSI.を利用するようにクラスタを設定することもできます。iSCSIは低コストなストレージエリアネットワーク(SAN)を作成するための方法として、ファイバチャネルの代わりに使用できます。次の図は、一般的なiSCSIクラスタの構成を表したものです。

図 1.4　一般的なiSCSIクラスタの構成



ほとんどのクラスタには共有ディスクサブシステムが含まれていますが、共有ディスクサブシステムなしのクラスタを作成することもできます。次の図は、共有ディスクサブシステムなしのクラスタを表したものです。

図 1.5　共有ストレージなしの一般的なクラスタ構成



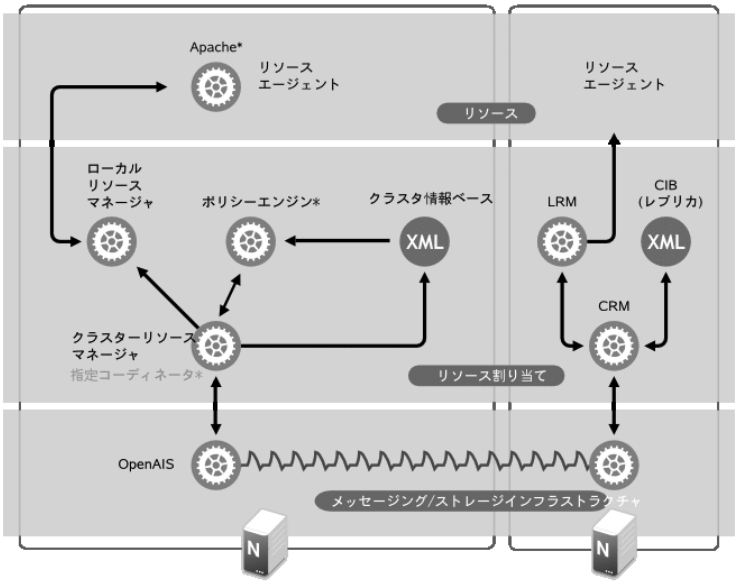
1.4 アーキテクチャ

このセクションではHigh Availability Extensionアーキテクチャについて簡単に説明します。アーキテクチャコンポーネントと、その相互運用方法について説明します。

1.4.1 アーキテクチャ層

High Availability Extensionは階層化されたアーキテクチャになっています。図 1.6. 「アーキテクチャ」 (12 ページ)に異なる層と関連するコンポーネントを示します。

図 1.6 アーキテクチャ



メッセージングおよびインフラストラクチャ層

プライマリ、または最初の層は、メッセージングおよびインフラストラクチャ層で、OpenAIS層とも呼ばれています。この層には、「I'm alive」信号やその他の情報を含むメッセージを送信するコポーネントが含まれます。High Availability Extensionのプログラムはメッセージングおよびインフラストラクチャ層に常駐しています。

リソース割り当て層

次の層はリソース割り当て層です。この層は最も複雑で、次のコンポーネントから構成されていいます。

CRM (クラスターリソースマネージャ)
: リソース割り当て層のすべてのアクションは、クラスターリソースマネージャを通過します。リソース割り当て層の他のコンポーネント(または上位層のコンポーネント)による通信の必要性が発生した場合は、ローカルCRM経由で行います。

  CRMは各ノードにCIB (クラスタ情報ベース) (13 ページ)を持っており、ここにはすべてのクラスタオプション、ノード、リソース、関係の定義や現在の状態が含まれています。クラスタ内の1つのCRMがDC(指定コーディネータ)として選択され、マスタCIBがそこに保存されます。クラスタ内のその他すべてのCIBはマスタCIBのレプリカです。CIBに対する通常の読み書き操作は、マスタCIBによってシリアルに処理されます。DCは、ノードのフェンシングやリソースの移動など、クラスタ全体に及ぶ変更が必要かどうかを判断できる、クラスタ内で唯一のエンティティです。

CIB (クラスタ情報ベース)
: クラスタ情報ベースは、メモリ内でクラスタ全体の設定や現在の状態をXML形式で表すものです。すべてのクラスタオプション、ノード、リソース、制約、相互関係の定義が含まれます。CIBはすべてのクラスタノードへの更新の同期化も行います。DCが管理するマスタCIBがクラスタ内に1つあります。その他のノードにはCIBのレプリカが含まれます。

PE (ポリシーエンジン)
: 指定コーディネータがクラスタ全体に及ぶ変更(新しいCIBへの反映)が必要になった場合、ポリシーエンジンが現在の状態と設定に基づき、クラスタの次の状態を計算します。PEは(リソース)アクションのリストと、次の

クラスタ状態に移るために必要な依存性を含む遷移グラフも作成します。PEはDCのフェールオーバー速度を上げるため、各ノードで実行されます。

LRM(ローカルリソースマネージャ)
LRMはCRMに代わってローカルリソースエージェントを呼び出します(リソース層項 (14 ページ)を参照)。そのため、操作の開始、停止、監視を行い、結果をCRMに報告します。リソースエージェントに対してサポートされているスクリプト標準規格(OCF、LSB、Heartbeat Version 1)間の違いも非表示にします。LRMはそのローカルノード上にある全リソースに関連する情報の信頼されたソースです。

リソース層

最も上位の層はリソース層です。リソース層には1つ以上のリソースエージェント(RA)が含まれます。リソースエージェントは通常シェルスクリプトなど、特定の種類のサービス(リソース)を開始、停止、監視するためのプログラムです。リソースエージェントの呼び出しはLRMだけが行います。サードパーティはファイルシステム内の定義された場所に独自のエージェントを配置して、自社ソフトウェア用に、すぐに使えるクラスタ統合機能を提供することができます。

1.4.2 プロセスフロー

SUSE Linux Enterprise High Availability ExtensionはPacemakerをCRMとして使用します。CRMは各クラスタノード上にインスタンスを持つデーモン(`crmd`)として実装されます。Pacemakerは、マスタとして動作する1つのcrmdインスタンスを選択し、クラスタのすべての意思決定を一元化します。選択したcrmdプロセス(またはその下のノード)で障害が発生したら、新しいcrmdプロセスが確立されます。

クラスタの構成とクラスタ内のすべてのリソースの現在の状態を反映したCIBが、各ノードに保存されます。CIBのコンテンツはクラスタ全体で自動的に同期化されます。

クラスタ内で実行するアクションの多くは、クラスタ全体に及ぶ変更を伴います。これらのアクションにはクラスタリソースの追加や削除、リソース制約の変更などがあります。このようなアクションを実行すると、クラスタ内でどのような変化が発生するのかを理解することが重要です。

たとえば、クラスタIPアドレスリソースを追加するとします。そのためには、コマンドラインツールかGUIを使用してCIBを変更できます。DC上でアクションを実行する必要はなく、クラスタ内の任意のノードでいずれかのツールを使用すれば、DCに反映されます。そして、DCがすべてのクラスタノードにCIBの変更を複製します。

CIBの情報に基づき、PEがクラスタの理想的な状態と実行方法を計算し、指示リストをDCに送ります。DCはメッセージングおよびインフラストラクチャ層を経由してコマンドを送信し、他のノードのcrmdピアがこれらを受信します。各crmdはLRM(lrmdとして実装)を使用してリソースを変更します。lrmdはクラスタに対応しておらず、リソースエージェント(スクリプト)と直接通信します。

すべてのピアノードは操作結果をDCに返送します。DCが、すべての必要な操作がクラスタ内で成功したことを確認すると、クラスタはアイドル状態に戻り、次のイベントを待機します。予定通り実行されなかった操作があれば、CIBに記録された新しい情報を元に、PEを再度呼び出します。

場合によっては、共有データの保護や完全なリソース復旧のためにノードの電源を切らなければならないことがあります。このPacemakerにはフェンシングサブシステムとしてstonithdが内蔵されています。STONITHは「Shoot The Other Node In The Head」(他のノードの即時強制終了)の略語で、通常はリモート電源スイッチを使用して実装されます。Pacemaker、STONITHデバイスはリソースとしてモデル化され(CIB内で環境設定)、障害を簡単に監視できるようにします。ただし、stonithdはクライアントがフェンシングの必要なノードを指定するだけで、残りの作業を処理できるような、STONITHトポロジを把握します。

1.5 新機能

SUSE Linux Enterprise Server 11では、クラスタスタックがHeartbeatからOpenAISに変更しました。OpenAISは業界標準のAPIとしてService Availability Forumが発行しているApplication Interface Specification (AIS)を実装しています。SUSE Linux Enterprise Server 10のクラスタリソースマネージャも残っていますが、大幅に機能が強化され、OpenAISに移植され、現在はPacemakerと呼ばれています。

SUSE® Linux Enterprise Server 10 SP2からSUSE Linux Enterprise High Availability Extension 11へのHigh Availabilityコンポーネントにおける変更点の詳細は、次のセクションを参照してください。

1.5.1 新しく追加された機能

マイグレーションのしきい値と失敗タイムアウト
: High Availability Extensionにはマイグレーションのしきい値と失敗タイムアウトというコンセプトが新たに追加されました。新しいノードへのマイグレートを行う基準となるリソースの失敗回数を定義できます。デフォルトでは、管理者がリソースの失敗回数を手動でリセットするまで、ノードは失敗したリソースを実行できなくなります。ただし、リソースの`failure-timeout`オプションを設定することで、リソースの失敗回数を失効させることができます。

リソースと操作のデフォルト
: リソースオプションと操作にグローバルなデフォルトを設定できるようになりました。

オフラインの設定変更のサポート
: 設定を個別に更新する前に、一連の変更による効果をプレビューで確認したほうがよい場合があります。構成の「シャドー」コピーを作成して、実行前にコマンドラインインタフェースで編集し、アクティブなクラスタ構成を個別に変更できるようになりました。

ルール、オプション、操作セットの再利用
: ルール、インスタンス属性、メタ属性、および操作セットは1度定義しておけば、複数の場所で参照できます。

CIB内の特定操作に対するXPath式の使用
: CIBでXPathベースの`create`、`modify`、`delete`操作が使用できるようになりました。詳細は、`cibadmin`のヘルプテキストを参照してください。

多次元のコロケーションと順序の制約
: 一連のコロケーションリソースを作成する場合、これまではリソースグループを定義するか(設計を必ずしも正確に表現していない場合があった)、個別の制約として各関係を定義するかのいずれかが可能でした。その結果、リソースや組み合わせの数が増加するにつれて、制約も膨大なものに

なることもありました。今回、resource_setsの定義によって別な形式でコロケーションの制約を指定できるようになりました。

クラスタ化されていないマシンからのCIBへの接続
: Pacemakerがマシンにインストールされていれば、マシン自体がクラスタに属していない場合でもクラスタに接続できます。

反復アクションを既知の回数トリガ
: デフォルトでは、リソースの開始時刻に対して相対的に反復アクションがスケジュールされますが、これが適切ではない場合がありました。操作を相対的に実施する日付/時刻を指定するため、操作の間隔開始時刻を設定します。クラスタはこの時刻を使用して、開始時刻+(間隔*N)で操作を開始するように、適切な開始遅延を計算します。

1.5.2 変更された機能

リソースとクラスタオプションに関する命名規則
: すべてのリソースとクラスタオプションには、アンダースコア(_）の代わりにダッシュ(-)を使用するようになりました。たとえばmaster_maxメタオプションは、master-maxという名前に変更されました。

master_slaveリソースの名前変更
: master_slaveリソースは、masterという名前に変更されました。マスタリソースは、2つのモードのいずれかで実行可能な特殊なクローンタイプです。

属性のコンテナタグ
: attributesコンテナタグは削除されました。

前提条件の操作フィールド
: pre-req操作フィールドは、requiresという名前に変更されました。

操作間隔
: すべての操作に間隔を指定する必要があります。開始および停止操作の場合、間隔は0(ゼロ)に設定する必要があります。

コロケーション属性と順序の制約
: コロケーション属性および順序の制約の名前をわかりやすく変更しました。

障害によるマイグレーションのためのクラスタオプション
: `resource-failure-stickiness`クラスタオプションは、`migration-threshold`クラスタオプションに替わりました。マイグレーションのしきい値と失敗タイムアウト (16 ページ)も参照してください。

コマンドラインツールの引数
: コマンドラインツールの引数が一定になりました。リソースとクラスタオプションに関する命名規則 (17 ページ)も参照してください。

XMLの検証と解析
: クラスタ構成はXMLで作成されます。従来のDTD(文書型定義)の代わりに、より強力なRELAX NGスキーマを使用して、構造とコンテンツのパターンを定義するようになりました。`libxml2`はパーサとして使用します。

`id`フィールド
: `id`フィールドは次の制限を持つXML IDになりました。

  - IDにコロンを含めることはできません。
  - IDは数字から開始できません。
  - IDはグローバルに固有なものでなければなりません(そのタグで固有なだけではなく)。

他のオブジェクトの参照
: フィールドの中にはIDREFフィールドがあります(リソースを参照する制約のフィールドなど)。したがって、有効な構成にするために既存のリソースまたはオブジェクトを参照しなければなりません。そのため、別な場所で参照されているオブジェクトの削除は失敗します。

1.5.3 削除された機能

リソースメタオプションの設定
: リソースメタオプションを最上位の属性として設定できなくなりました。代わりにメタ属性を使用してください。

グローバルデフォルトの設定
: リソースと操作デフォルトは`crm_config`から読み込まれなくなりました。

2 はじめに

以降では、システム要件とHigh Availability Extensionをインストールする前の準備について説明します。クラスタのインストールとセットアップのための基本手順の概要を説明します。

2.1 ハードウェア要件

次のリストは、SUSE® Linux Enterprise High Availability Extensionに基づくクラスタのハードウェア要件を示します。これらの要件は、最低のハードウェア構成を表しています。クラスタの使用方法によっては、ハードウェアを追加しなければならないこともあります。

- 2.2項 「ソフトウェアの必要条件」 (20 ページ)に指定されたソフトウェアを搭載した1～16台のLinuxサーバ。サーバのハードウェア構成(メモリ、ディスクスペースなど)は、同一である必要はありません
- 少なくとも2つのTCP/IP通信メディア。クラスタノードは通信にマルチキャストを使用するため、ネットワーク装置でマルチキャスティングをサポートする必要があります。通信メディアは100Mbit/s以上のデータレートをサポートする必要があります。可能ならば、Ethernetチャネルをボンドします。
- オプション:クラスタ内の、アクセスが必要なすべてのサーバに接続された、共有ディスクサブシステム。

- STONITHメカニズム。STONITHは「Shoot the other node in the head」の略です。STONITHデバイスは、クラスタが停止または誤動作していると考えられるノードをリセットするために使用する、電源スイッチです。ハートビートを実行していないノードのリセットは、ハングして停止したようにしか見えないノードによるデータ破損を防ぐ、唯一の信頼できる方法です。

  詳細については、第8章 *フェンシングと*STONITH (83 ページ)を参照してください。

2.2 ソフトウェアの必要条件

次のソフトウェア要件を満たしていることを確認してください。

- クラスタの一部となるすべてのノードに、使用できるすべてのオンライン更新がインストールされた、SUSE® Linux Enterprise Server 11。
- クラスタの一部となるすべてのノードに、使用できるすべてのオンライン更新がインストールされた、SUSE Linux Enterprise High Availability Extension 11。

2.3 共有ディスクのシステム要件

クラスタで、データの可用性を高めたい場合は、共有ディスクシステム(SAN:Storage Area Network)の利用をお勧めします。共有ディスクシステムを使用する場合は、次の要件を満たしていることを確認してください。

- メーカーの指示のに従い、共有ディスクシステムが適切に設定され、正しく動作していることを確認します。
- 共有ディスクシステム中のディスクを、ミラーリングまたはRAIDを使用して耐障害性が高められるように設定してください。ハードウェアベースのRAIDをお勧めします。ホストベースのソフトウェアRAIDはどの構成でもサポートされていません。
- 共有ディスクシステムのアクセスにiSCSIを使用している場合、iSCSIイニシエータとターゲットを正しく設定していることを確認します。

- DRBDを使用してデータを2台のマシンに分配するミラーリングRAIDシステムを実装する際は、複製されたデバイスのみにアクセスしてください。クラスタの残りが提供される冗長性を利用する、同じ(ボンドされた)NICを使用します。

2.4 準備作業

インストール前に、次の準備作業を実施してください。

- ホスト名解決を設定し、クラスタ内の各サーバの`/etc/hosts`ファイルを編集して静的ホスト情報を使用します。詳細は、`http://www.novell.com/documentation`にある『*SUSE Linux Enterprise Server Administration Guide*』の「*Basic Networking*」の章の「*Configuring Hostname and DNS*」の項を参照してください。

  クラスタのメンバーが名前で互いを見つけられることが重要です。名前を使用できない場合、内部クラスタ通信は失敗します。

- クラスタノードをクラスタ外部のタイムサーバと同期させ、時刻同期を構成します。詳細は、`http://www.novell.com/documentation`にある『*SUSE Linux Enterprise Server Administration Guide*』の「*Time Synchronization with NTP*」の章を参照してください。

  クラスタノードは、タイムサーバを時刻同期ソースとして使用します。

2.5 概要:クラスタのインストールとセットアップ

準備が完了したら、クラスタをSUSE® Linux Enterprise High Availability Extensionでインストールしてセットアップする次の手順が必要です。

1. SUSE® Linux Enterprise Server 11とSUSE® Linux Enterprise High Availability Extension 11をSUSE Linux Enterprise Serverにアドオンとしてインストールします。詳細については、3.1項 「High Availability Extensionのインストール」 (23 ページ)を参照してください。

2. OpenAISを構成します。詳細については、3.2項 「クラスタの初期セットアップ」 (24 ページ)を参照してください。

3. OpenAISを起動して、クラスタの状態を監視します。詳細については、3.3項 「クラスタをオンラインにする」 (27 ページ)を参照してください。

4. クラスタリソースを、グラフィカルユーザインタフェース(GUI)またはコマンドラインで追加し、構成します。詳細については、第4章 *GUIによるクラスタリソースの設定* (31 ページ)または第5章 *クラスタリソースのコマンドラインからの設定* (61 ページ)を参照してください。

   データがフェンシングとSTONITHによって破損することを防ぐため、STONITHデバイスをリソースとして構成します。詳細については、第8章 *フェンシングとSTONITH* (83 ページ)を参照してください。

   まだない場合は、ファイルシステムを共有ディスク(SAN)上で作成する必要もあります。また、必要に応じてこれらのファイルシステムをクラスタリソースとして構成します。

   クラスタを認識(OCFS 2)または認識しないファイルシステムをHigh Availability Extensionで構成できます。必要な場合は、DRBDでデータレプリケーションを使用することもできます。詳細については、パート III. 「ストレージおよびデータレプリケーション」 (109 ページ)を参照してください。

YaSTのインストールと基本セットアップ

3

High Availabilityクラスタに必要なソフトウェアのインストール方法は複数あります。コマンドラインからzypperを使用する方法や、YaSTのグラフィカルユーザインタフェースを使用する方法です。クラスタに含まれるすべてのノードにソフトウェアをインストールした後、次の手順はクラスタを初期設定し、ノードが相互に通信できるようにします。これは手動(設定ファイルの編集)でも、YaSTクラスタモジュールによっても実行できます。

注意: ソフトウェアパッケージのインストール

High Availabilityクラスタに必要なソフトウェアパッケージはクラスタノードに自動的にコピーされません。SUSE® Linux Enterprise Server 11およびSUSE® Linux Enterprise High Availability Extension 11をクラスタの一部となるすべてのノードにインストールします。

3.1 High Availability Extensionのインストール

High Availability Extensionによるクラスタの構成と管理に必要なパッケージは、高可用性インストールパターンに含まれています。このパターンは、SUSE® Linux Enterprise High Availability Extensionがアドオンとしてインストールされた後にのみ利用できます。アドオン製品のインストール方法については、『SUSE Linux Enterprise 11 *Deployment Guide*』の「*Installing Add-On Products*」を参照してください。

1 YaSTを起動して*ソフトウェア*> *ソフトウェア管理*を選択して、YaSTパッケージマネージャを開きます。

2 *フィルタ*リストで、*パターン*を選択して、パターンリストで*高可用性*をアクティブにします。

3 *同意する*をクリックして、パッケージのインストールを開始します。

3.2 クラスタの初期セットアップ

HAパッケージをインストールした後、YaSTでクラスタの初期セットアップを設定できます。これには、ノード間の通信チャネル、暗号化通信の使用などのセキュリティ面、OpenAISのサービスとしての起動などがあります。

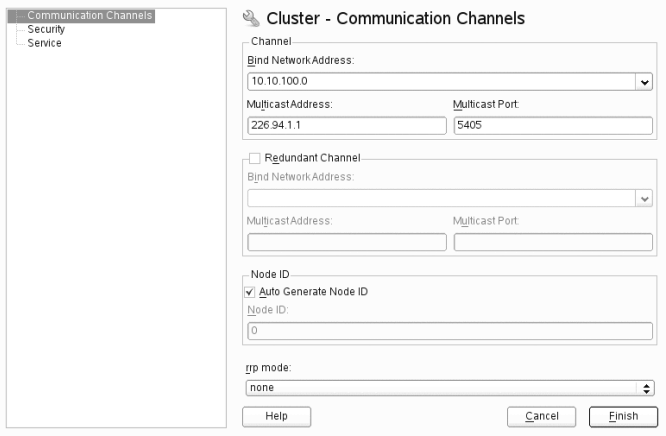
通信チャネルについては、バインドネットワークアドレス(`bindnetaddr`)、マルチキャストアドレス(`mcastaddr`)、マルチキャストポート(`mcastport`)を定義する必要があります。`bindnetaddr`はバインド先のネットワークアドレスです。クラスタ間の設定ファイルの共有を軽減するため、OpenAISはネットワークインタフェースネットマスクを使用して、ネットワークのルーティングに使用されるアドレスビットのみをマスクします。`mcastaddr`はIPv4またはIPv6マルチキャストアドレスです。`mcastport`は`mcastaddr`に指定されたUDPポートです。

クラスタ内のノードは同じマルチキャストアドレスと同じポート番号を使用していることで、互いに認識されます。別のクラスタは、別のマルチキャストアドレスを使用します。

手順 3.1 *クラスタの構成*

1 YaSTを起動して*その他*> *クラスタ*を選択するか、コマンドラインで`yast2 cluster`を実行して、クラスタ初期設定ダイアログを起動します。

2 *Communication Channel(通信チャネル)*カテゴリで、クラスタノード間の通信に使用されるチャネルを設定します。この情報は`/etc/ais/openais.conf`環境設定ファイルに書き込まれます。

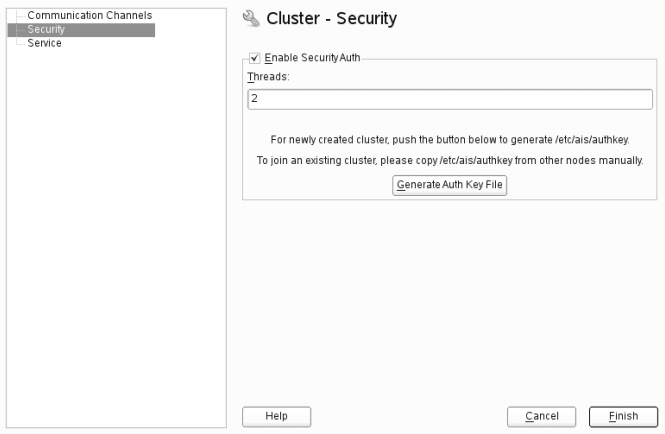


すべてのクラスタノードに使用する、*Bind Network Address(バインドネットワークアドレス)*、*Multicast Address(マルチキャストアドレス)*、*Multicast Port(マルチキャストポート)*を定義します。

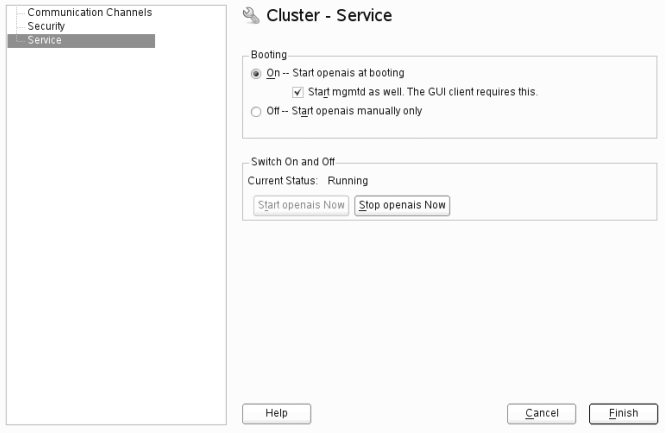
3 各クラスタノードに一意の*Node ID（ノードID）*を指定します。1から開始することを推奨します。

4 セキュリティカテゴリで、クラスタの認証設定を定義します。*Enable Security Authentication(セキュリティ認証を有効にする)*が有効な場合、HMAC/SHA1認証がクラスタノード間の通信に使用されます。

この認証方法では共有秘密が必要で、メッセージの保護と認証に使用されます。指定した認証キー(パスワード)が、クラスタ中のすべてのノードで使用されます。新規作成したクラスタで、*Generate Auth Key File(認証キーファイルの生成)*をクリックして/etc/ais/authkeyに書き込まれる認証キーを作成します。



5 サービスカテゴリで、このクラスタサーバで起動するたびにOpenAISを起動するかどうかを選択します。



オフを選択した場合は、クラスタサーバのブート時に手動でOpenAISを起動する必要があります。OpenAISを手動で起動するには、`rcopenais start`コマンドを使用します。

OpenAISを即時起動するには、*Start OpenAIS Now(今すぐOpenAISを開始)*をクリックします。

6 すべてのオプションが希望どおりに設定されたら、*完了*をクリックします。YaSTはファイアウォール設定も自動的に調整し、マルチキャストに使用されるUDPポートを開きます。

7 初期設定が完了したら、設定をクラスタ内のその他のノードに転送する必要があります。これを実行する簡単な方法は、`/etc/ais/openais.conf`ファイルをクラスタ内のその他のノードにコピーすることです。各ノードには一意のノードIDが必要なため、ファイルをコピーした後にそれぞれのノードIDを調整してください。

8 暗号化通信を使用する場合、`/etc/ais/authkey`もクラスタ内のその他のノードにコピーします。

3.3 クラスタをオンラインにする

基本設定の後、スタックをオンラインにして状態を確認します。

1 クラスタの各ノードについて次のコマンドを実行して、OpenAISを起動します。

```
rcopenais start
```

2 ノードの1つで、次のコマンドを使用してクラスタの状態を確認します。

```
crm_mon
```

すべてのノードがオンラインの場合、出力は次のようになります。

```
============
Last updated: Thu Feb  5 18:30:33 2009
Current DC: d42 (d42)
Version: 1.0.1-node: b7ffe2729e3003ac8ff740bebc003cf237dfa854
3 Nodes configured.
0 Resources configured.
============

Node: d230 (d230): online
Node: d42 (d42): online
Node: e246 (e246): online
```

基本設定が完了してノードがオンラインになったら、`crm`コマンドラインツールまたはグラフィカルユーザインタフェースを使用して、クラスタリソースの設定を開始できます。詳細は、第4章 *GUIによるクラスタリソースの設定*

(31ページ)または第5章 クラスタリソースのコマンドラインからの設定(61ページ)を参照してください。

パート II. 設定および管理

GUIによるクラスタリソースの設定

4

HAクラスタの主な目的はユーザサービスの管理です。ユーザサービスの一般的な例として、Apache Webサーバまたはデータベースがあります。サービスとは、ユーザの観点からすると、指示に基づいて特別な何かを行うことを意味していまが、クラスタにとっては開始や停止できるリソースにすぎません。サービスの性質はクラスタには無関係なのです。

クラスタの管理者は、クラスタ内のサーバ上の各リソースや、サーバ上で実行する各アプリケーションに対してクラスタリソースを作成する必要があります。クラスタリソースには、Webサイト、電子メールサーバ、データベース、ファイルシステム、仮想マシン、およびユーザが常時使用できるその他のサーバベースのアプリケーションまたはサービスなどが含まれます。

クラスタリソースを作成するには、グラフィカルユーザインタフェース(Linux HA Management Client)または`crm`コマンドラインユーティリティのいずれかを使用します。コマンドラインを使用した方法については、第5章 *クラスタリソースのコマンドラインからの設定* (61 ページ)を参照してください。

この章ではLinux HA Management Clientについて説明します。また、リソースの作成、制約の設定、フェールオーバーノードとフェールバックノードの指定、リソース監視の設定、リソースの開始と削除、リソースグループやクローンリソースの構成、手動でのリソースのマイグレーションなど、クラスタ構成時に必要ないくつかの情報についても説明します。

クラスタリソースを設定するためのグラフィカルユーザインタフェースは、`pacemaker-pygui`パッケージに含まれています。

4.1 Linux HA Management Client

Linux HA Management Clientを開始するには、クラスタに接続する必要があります。

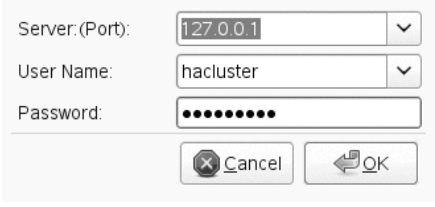
注意: haclusterユーザのパスワード

インストールによってhaclusterという名前のLinuxユーザが作成されます。Linux HA Management Clientを使用する前に、haclusterユーザのパスワードを設定する必要があります。そのためにはrootになり、コマンドラインにpasswd haclusterと入力し、haclusterユーザのパスワードを入力します。

Linux HA Management Clientを使用して接続先の各ノードについてこの作業を行います。

Linux HA Management Clientを開始するには、コマンドラインにcrm_guiと入力します。クラスタに接続するには、*接続*> *ログイン*の順に選択します。デフォルトでは、サーバフィールドにローカルホストのIPアドレスとhaclusterがユーザ名として表示されています。ユーザのパスワードを入力して続行します。

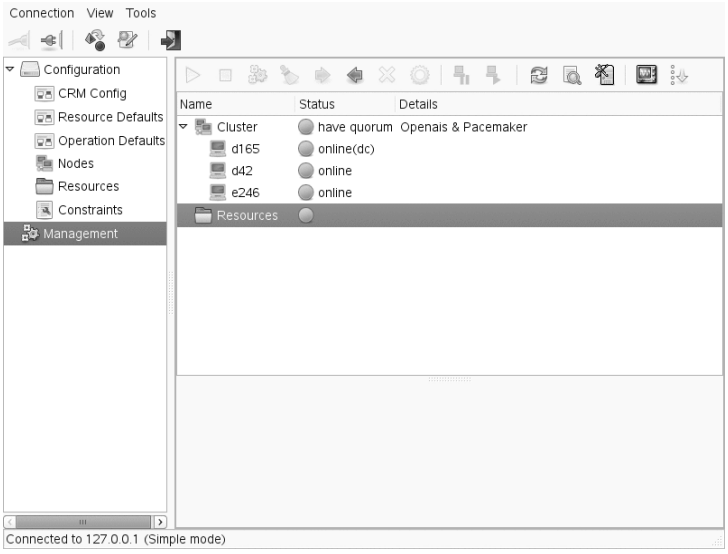
図 4.1 *クラスタへの接続*



Linux HA Management Clientをリモートに実行している場合は、クラスタノードのIPアドレスをサーバとして入力します。ユーザ名として入力すると、haclientグループに属している他のユーザ使用して、クラスタに接続することができます。

接続後、メインウィンドウが開きます。

図 4.2　Linux HA Management Client - メインウィンドウ



Linux HA Management Clientではリソース、制約、設定などの追加と変更が行えます。またリソースの開始、停止、マイグレーション、クリーンアップ、ノードのstandby設定など、クラスタコンポーネントを管理するための機能も提供されています。さらに、*環境設定*サブ項目のいずれかを選択して、*表示*>*XMLモード*を選択することで、CIBのXML構造の表示、編集、インポート、エクスポートが簡単に行えます。

次に紹介するいくつかの例では、Linux HA Management Clientを使用したクラスタリソースの作成および管理方法を示します。

4.2 クラスタリソースの作成

次のタイプのリソースを作成できます。

プリミティブ
: プリミティブリソースはリソースの中で最も基本的なタイプです。

グループ
: グループには、一緒に配置する必要があり、連続して開始し、その逆の順序で停止する必要があるリソースセットが含まれます。詳細については、4.10項 「クラスタリソースグループの設定」 (50 ページ)を参照してください。

クローン
: クローンは、複数のホスト上でアクティブにできるリソースです。対応するリソースエージェントがサポートしていれば、どのようなリソースもクローン化できます。詳細については、4.11項「クローンリソースの構成」 (56 ページ)を参照してください。

マスタ
: マスタは特殊なクローンリソースのタイプで、複数のモードがあります。マスタには単一のグループ、または単一の正規リソースだけを含む必要があります。

手順 4.1 *プリミティブリソースの追加*

1. 4.1項 「Linux HA Management Client」 (32 ページ)で説明したように、Linux HA Management Clientを起動してクラスタにログインします。

2. 左側のペインで、*リソース*を選択し、［*追加*> *プリミティブ*］の順にクリックします。

3. 次のダイアログで、リソースに次のようなパラメータを設定します。

   3a リソースに固有のIDを入力します。

   3b クラスリストから、そのリソースに使用するリソースエージェントクラスを選択します。*heartbeat*、*lsb*、*ocf*、または*stonith*を選択できます。詳細については、17.1項 「サポートされるリソースエージェントクラス」 (201 ページ)を参照してください。

   3c *ocf*をクラスとして選択した場合、OCFリソースエージェントの*プロバイダ*も指定します。OCFの指定によって、複数のベンダが同じリソースエージェントを提供できるようになります。

   3d タイプリストから、使用するリソースエージェントを選択します(たとえば*IPaddr*または*Filesystem*)。このリソースエージェントの簡単な説明を次に表示します。

タイプリストに表示される選択肢は、選択したクラスに(OCFリソースの場合は、*プロバイダ*にも)よって異なります。

3e *オプション*の下で、*Initial state of resource(リソースの当初の状態)*を設定します。

3f リソースのヘルスが維持されているかどうかをクラスタに監視させる場合は、*Add monitor operation(モニタ操作の追加)*を有効にします。

4 *進む*をクリックします。次のウィンドウには、そのリソースに定義したパラメータの概要が表示されます。そのリソースに必要なすべての*Instance Attributes (インスタンス属性)*が一覧が表示されます。適切な値に設定するには、編集する必要があります。展開や設定によっては、属性の追加が必要な場合もあります。詳細についてはメタ属性とインスタンス属性の追加または変更 (36 ページ)を参照してください。

5 すべてのパラメータが希望どおりに設定されたら、*適用*をクリックして、そのリソースの設定を完了します。環境設定ダイアログが閉じて、メインウィンドウに新しく追加されたリソースが表示されます。

プリミティブリソースの次のパラメータをいつでも追加または変更できます。

メタ属性
: メタ属性はリソースに追加できるオプションです。CRMに対して特定のリソースの処理方法を伝えます。使用可能なメタ属性の概要、値とデフォルトについては、17.3項 「リソースオプション」 (205 ページ)を参照してください。

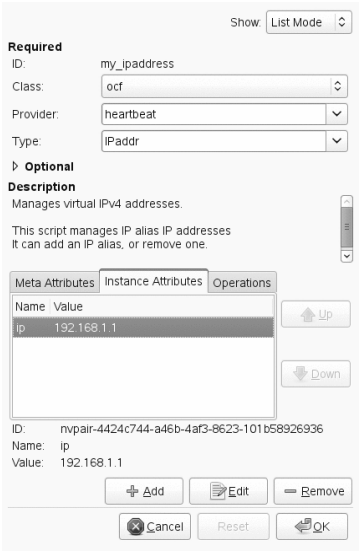
インスタンス属性
: インスタンス属性は動作方法や、制御するサービスインスタンスを決定する特定のリソースクラスのパラメータです。詳細については、17.5項「インスタンス属性」 (208 ページ)を参照してください。

操作
: リソースに追加される監視操作。クラスタに対して、リソースのヘルス状態を維持するように指示します。監視操作は、すべてのリソースエージェントクラスに追加できます。`start`または`stop`操作に対する`Timeout`など、特定のパラメータも設定できます。詳細については、4.7項 「リソース監視の設定」 (47 ページ)を参照してください。

手順 4.2 *メタ属性とインスタンス属性の追加または変更*

1. Linux HA Management Clientのメインウィンドウで、左側のペインの*リソース*をクリックして、そのクラスタ用に設定されているリソースを表示します。

2. 右側のペインで、変更するリソースを選択し、*編集*をクリックします(またはリソースをダブルクリックします)。次のウィンドウには、そのリソースに定義された基本的なリソースパラメータと*メタ属性*、*インスタンス属性*、または*操作*が表示されます。



3 新しいメタ属性またはインスタンス属性を追加するには、該当するタブを選択して*追加*をクリックします。

4 追加する属性の*名前*を選択します。短い*説明*が表示されます。

5 必要に応じて、属性の*値*を指定します。指定しなければ、その属性のデフォルト値が使用されます。

6 *OK*をクリックして変更を確認します。新しく追加または変更された属性がタブに表示されます。

7 すべてのパラメータが希望どおりに設定されたら、*OK*をクリックして、そのリソースの設定を完了します。環境設定ダイアログが閉じ、メインウィンドウに変更済みのリソースが表示されます。

ティップ: XMLソースコード

特定のリソースや、すべてのリソースに対して定義したパラメータから生成されたXMLを、Linux HA Management Clientで表示することができます。リソースの環境設定ダイアログ、またはメインウィンドウのリソースビューで、*表示* > *XMLモード*］を選択します。

XMLコードを表示しているエディタで、XML要素をインポートまたはエクスポート、あるいは手動でXMLコードを編集することができます。

4.3 STONITHリソースの作成

フェンシングを構成するには、複数のSTONITHリソースを構成する必要があります。

手順 4.3　STONITHリソースの追加

1. 4.1項 「Linux HA Management Client」 (32 ページ)で説明したように、Linux HA Management Clientを起動してクラスタにログインします。
2. 左側のペインで、*リソース*を選択し、［*追加*> *プリミティブ*］の順にクリックします。
3. 次のダイアログで、リソースに次のようなパラメータを設定します。
   - **3a** リソースに固有のIDを入力します。
   - **3b** クラスリストで、リソースエージェントクラスとして*stonith*を選択します。
   - **3c** タイプリストから、使用しているSTONITHデバイスのSTONITHプラグインを選択します。このプラグインの簡単な説明が下に表示されます。
   - **3d** *オプション*の下で、*Initial state of resource (リソースの当初の状態)*を設定します。
   - **3e** クラスタにフェンシングデバイスの監視を行わせたい場合は、*Add monitor operation(監視操作の追加)*を起動します。詳細については、8.4項 「フェンシングデバイスの監視」 (91 ページ)を参照してください。
4. *進む*をクリックします。次のウィンドウには、そのリソースに定義したパラメータの概要が表示されます。選択したSTONITHプラグインに必要なすべての*インスタンス属性*が一覧に表示されます。適切な値に設定

するには、編集する必要があります。展開と設定によっては、監視操作のための属性を追加しなければならない場合もあります。詳細はメタ属性とインスタンス属性の追加または変更 (36 ページ)および4.7項 「リソース監視の設定」 (47 ページ)を参照してください。

5 すべてのパラメータが希望どおりに設定されたら、*適用*をクリックして、そのリソースの設定を完了します。環境設定ダイアログが閉じて、メインウィンドウに新しく追加されたリソースが表示されます。

フェンシングを設定するには、制約の追加とクローンの使用のいずれか、またはその両方を行います。詳細については、第8章 *フェンシングと*STONITH (83 ページ)を参照してください。

4.4 リソース制約の設定

すべてのリソースを構成することは、ジョブのほんの一部分です。クラスタが必要なすべてのリソースを認識しても、正しく処理できるとは限りません。リソースの制約を指定して、リソースを実行可能なクラスタノード、リソースのロード順序、特定のリソースが依存している他のリソースを指定することができます。

使用可能な制約には3種類あります。

情報の取得先
: 場所の制約はリソースを実行できるノード、できないノード、または実行に適したノードを定義するものです。

リソースコロケーション
: コロケーションの制約は、ノード上で一緒に実行可能な、または一緒に実行することが禁止されているリソースをクラスタに伝えます。

Resource Order (リソース順序)
: アクションの順序を定義する、順序の制約。

制約を定義する際は、スコアも扱う必要があります。あらゆる種類のスコアはクラスタの動作方法と密接に関連しています。スコアの操作によって、リソースのマイグレーションから、速度が低下したクラスタで停止するリソースの決定まで、あらゆる作業を実行できます。スコアはリソースごとに計算され、リソースに対して負のスコアが付けられているノードは、そのリソー

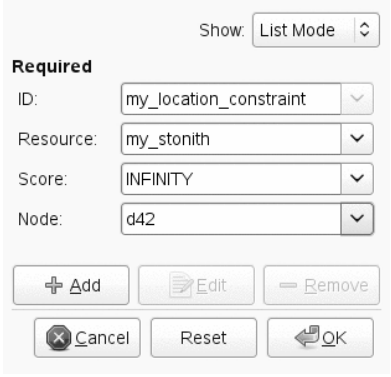
スを実行できません。リソースのスコアを計算後、クラスタはスコアが最も高いノードを選択します。INFINITYは現在1,000,000と定義されています。この値に対する増減は次の3つの基本的なルールに従って行います。

- 任意の値+ INFINITY = INFINITY
- 任意の値- INFINITY = -INFINITY
- INFINITY - INFINITY = -INFINITY

リソース制約を定義する際、各制約のスコアも指定します。スコアはこのリソース制約に割り当てる値を示します。スコアの高い制約は、それよりもスコアが低い制約より先に適用されます。特定のリソースに対して異なるスコアで追加の場所の制約を作成することで、リソースのフェールオーバー先のノードの順序を指定できます。

手順 4.4 *場所の制約の追加または変更*

1. 4.1項 「Linux HA Management Client」 (32 ページ)で説明したように、Linux HA Management Clientを起動してクラスタにログインします。
2. Linux HA Management Clientのメインウィンドウの左側のペインで、*制約*をクリックしてそのクラスタに設定済みの制約を表示します。
3. 左側のペインで*制約*を選択し、*追加*をクリックします。
4. *Resource Location (リソース位置)*を選択し、*OK*をクリックします。
5. 制約に固有の*ID*を入力します。既存の制約を変更する場合、IDはすでに定義されているため、環境設定ダイアログに表示されます。
6. 制約を設定するリソースを選択します。リストには、そのクラスタに設定されているすべてのリソースのIDが表示されます。
7. 制約の*Score(スコア)*を設定します。正の値は、下で指定したノードでリソースを実行できることを示します。負の値は、このノードでリソースを実行できないことを示します。+/- INFINITYの値は、「can」をmustに変更します。
8. 制約を設定するノードを選択します。



9 *ノードおよびScore (スコア)フィールドを空のままにしておくと、追加>ルールの順にクリックしてルールを追加することもできます。有効期間を追加するには、追加> 有効期間の順にクリックします。*

10 すべてのパラメータが希望どおり設定されたら、*OK*をクリックして、制約の設定を完了します。環境設定ダイアログが閉じて、メインウィンドウに新しく追加または変更された制約が表示されます。

手順 4.5 *コロケーションの制約の追加または変更*

1 4.1項 「Linux HA Management Client」 (32 ページ)で説明したように、Linux HA Management Clientを起動してクラスタにログインします。

2 Linux HA Management Clientのメインウィンドウの左側のペインで、*制約*をクリックしてそのクラスタに設定済みの制約を表示します。

3 左側のペインで*制約*を選択し、*追加*をクリックします。

4 *Resource Collocation (リソースコロケーション)*を選択し、*OK*をクリックします。

5 制約に固有の*ID*を入力します。既存の制約を変更する場合、IDはすでに定義されているため、環境設定ダイアログに表示されます。

6 コロケーションソースとなるリソースを選択します。リストには、そのクラスタに設定されているすべてのリソースのIDが表示されます。

制約が満たされないと、クラスタはリソースがまったく実行しないようにすることがあります。

7 リソースと*With Resource (対象リソース)*フィールドを両方とも空のままにしておくと、*追加*> リソースセットの順にクリックしてリソースを追加することもできます。有効期間を追加するには、*追加*> *有効期間*の順にクリックします。

8 *With Resource(対象リソース)*には、コロケーション先を定義します。クラスタはこのリソースの配置先を最初に決定し、次にリソースフィールドのリソースを配置する場所を決定します。

9 *Score(スコア)*を定義して、両方のリソース間の位置関係を決定します。正の値は、リソースを同じノードで実行しなければならないことを示します。負の値は、リソースを同じノードで実行してはならないことを示します。`+/- INFINITY`の値は`should`を`must`に変更します。スコアを他の要因と組み合わせて、ノードの配置先を決定します。

10 必要に応じて、*Resource Role(リソース役割)*などの追加のパラメータも指定します。

選択したパラメータとオプションに応じて、短い*説明*が表示され、設定しているコロケーションの制約の効果を確認できます。

11 すべてのパラメータが希望どおり設定されたら、*OK*をクリックして、制約の設定を完了します。環境設定ダイアログが閉じて、メインウィンドウに新しく追加または変更された制約が表示されます。

手順 4.6 *順序の制約の追加または変更*

1 4.1項 「Linux HA Management Client」 (32 ページ)で説明したように、Linux HA Management Clientを起動してクラスタにログインします。

2 Linux HA Management Clientのメインウィンドウの左側のペインで、*制約*をクリックして、そのクラスタに設定されている制約を表示します。

3 左側のペインで*制約*を選択し、*追加*をクリックします。

4 *Resource Order(リソース順序)*を選択し、*OK*をクリックします。

5 制約に固有の*ID*を入力します。既存の制約を変更する場合、IDはすでに定義されているため、環境設定ダイアログに表示されます。

6 *First(最初)*では、*Then(次に)*のリソースより前に開始するリソースを定義します。

7 *Then(次に)*では、*First(最初)*のリソースより後に開始するリソースを定義します。

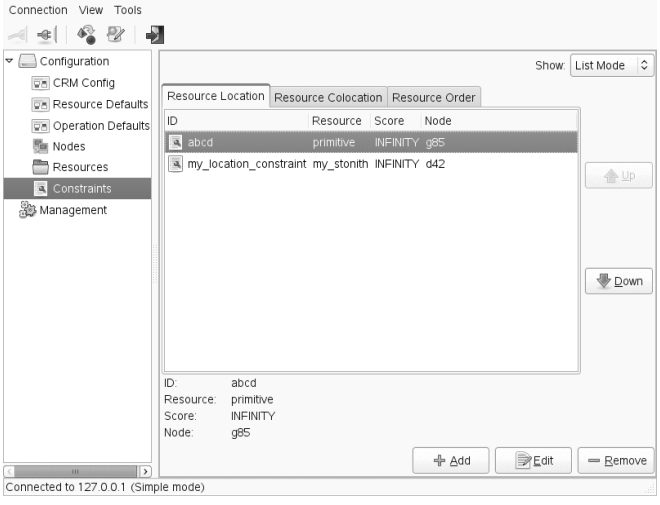
8 必要に応じて、*Score(スコア)*(ゼロより大きい場合は制約は必須、それ以外は単なる推奨)や、*Symmetrical(対称)*(「`true`」の場合は逆順にリソースを停止する)などの追加のパラメータを定義します。

選択したパラメータとオプションに応じて、短い*説明*が表示され、設定している順序の制約の効果を確認できます。

9 すべてのパラメータが希望どおり設定されたら、*OK*をクリックして、制約の設定を完了します。環境設定ダイアログが閉じて、メインウィンドウに新しく追加または変更された制約が表示されます。

Linux HA Management Clientの*制約*ビューで設定したすべての制約にアクセスして変更することができます。

図 4.3 Linux HA Management Client - 制約



制約の設定の詳細や、オーダーおよびコロケーションの基本的な概念について詳細なバックグラウンド情報は、http://clusterlabs.org/wiki/Documentationに提供されている次のドキュメントを参照してください。

- 『*Configuration 1.0の概要*』の「*リソース制約*」の章
- 『*コロケーションの概要*』
- 『*オーダーの概要*』

4.5 リソースフェールオーバーノードの指定

リソースに障害が発生すると、自動的に再起動されます。現在のノードで再起動できない場合、または現在のノードでN回失敗した場合は、別なノードへのフェールオーバーを試みます。新しいノードへのマイグレートを行う基準(migration-threshold)となるリソースの失敗数を定義できます。クラスタ内に3つ以上ノードがある場合、特定のリソースのフェールオーバー先のノードはHigh Availabilityソフトウェアが選択します。

リソースのフェールオーバー先のノードを指定する場合は、次の手順に従います。

1. 場所の制約の追加または変更 (40 ページ)に記載の手順に従って、そのリソースの場所の制約を設定します。
2. メタ属性とインスタンス属性の追加または変更 (36 ページ)に記載の手順に従って、migration-thresholdメタ属性をそのリソースに追加し、マイグレーションしきい値の*値*を入力します。INFINITY未満の正の値を指定する必要があります。
3. リソースの失敗回数を自動的に失効させる場合は、メタ属性とインスタンス属性の追加または変更 (36 ページ)に記載の手順に従ってfailure-timeoutメタ属性をそのリソースに追加し、失敗タイムアウトの*値*を入力します。
4. リソースの優先的な実行先として、追加のフェールオーバーノードを指定する場合は、追加の場所の制約を作成します。

たとえば、リソース「r1」の場所の制約を設定し、このリソースを「node1」で優先的に実行するように指定したと仮定します。そのノードで実行できなかった場合は、「migration-threshold」を確認して失敗回数と比較します。失敗回数>=マイグレーションしきい値の場合は、リソースは次の優先実行先として指定されているノードにマイグレートされます。

デフォルトでは、しきい値に達すると、管理者がリソースの失敗回数を手動でリセットするまで(失敗原因を修正してから)、実行できなかったリソースをそのノードで実行することはできません。

ただし、リソースの失敗タイムアウトオプションを設定することで、失敗回数を失効させることができます。したがって、「migration-threshold=2」と「failure-timeout=60s」を設定すると、2回の失敗の後に新しいノードにマイグレートし、1分後に復帰させられる可能性があります(固着性と制約スコアによる)。

マイグレーションしきい値には2つ例外があり、リソースが開始または停止できない場合がこれに相当します。開始時の失敗では失敗回数がINFINITYに設定され、すぐにマイグレーションが行われます。停止時の失敗ではフェンシングが発生します(［stonith-enabled］がデフォルトである「true」に設定されている場合)。STONITHリソースが定義されていない場合は(または［stonith-enabled］が「false」に設定されている場合)、リソースのマイグレーションはまったく行われません。

Linux HA Management Clientで失敗回数をクリーンアップするには、左側のペインで*管理*を選択し、右側のペインで該当するリソースを選択してツールバー内の*Cleanup Resource (リソースのクリーンアップ)*をクリックします。これによって指定したノード上の指定したリソースに対して、コマンド crm_resource -Cおよびcrm_failcount -Dが実行されます。詳細についてはcrm_resource(8) (172 ページ)とcrm_failcount(8) (162 ページ)も参照してください。

4.6 リソースフェールバックノードの指定(リソースの固着性)

ノードがオンライン状態に戻り、クラスタ内にある場合は、リソースが元のノードにフェールバックすることがあります。フェールオーバー前に実行していたノードへのリソースのフェールバックを防止する場合、またはリソースのフェールバック先に別なノードを指定する場合は、リソースの固着性の値を変更する必要があります。リソースの作成時、または作成後にリソースの固着性を指定できます。

リソースの固着性の値を指定する場合は次の点に注意してください。

0の値:
: デフォルトです。リソースはシステム内で最適な場所に配置されます。現在よりも「状態のよい」、または負荷の少ないノードが使用可能になると、移動することを意味しています。このオプションは自動フェールバックとほとんど同じですが、以前アクティブだったノード以外でもリソースをフェールバックできるという点が異なります。

0より大きい値:
: リソースは現在の場所に留まることを望んでいますが、状態がよいノードが使用可能になると移動される可能性があります。値が大きくなるほど、リソースが現在の場所に留まることを強く望んでいることを示します。

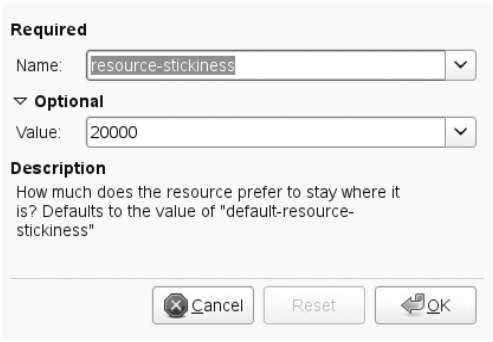
0より小さい値:
: リソースは現在の場所から別な場所に移動することを望んでいます。絶対値が大きくなるほど、リソースが移動を強く望んでいることを示します。

INFINITYの値:
: ノードがリソースの実行権利がなくなったために強制終了される場合(ノードのシャットダウン、ノードのスタンバイ、migration-thresholdに到達、または設定変更)以外は、リソースは常に現在の場所に留まります。このオプションは自動フェールバックを完全に無効にする場合とほとんど同じです。

-INFINITYの値:
: リソースは現在の場所から常に移動されます。

手順 4.7　リソースの固着性の指定

1 メタ属性とインスタンス属性の追加または変更 (36 ページ)に従って、`resource-stickiness`メタ属性をリソースに追加します。



2 resource-stickinessの*値*として、`-INFINITY`から`INFINITY`の範囲の値を指定します。

4.7 リソース監視の設定

High Availability Extensionはノード障害を検出できますが、ノード上の個々のリソースで障害が発生した場合にも検出することができます。リソースが実行中であることを確認するには、そのリソースのリソース監視を設定しておく必要があります。リソース監視は、タイムアウト、開始遅延のいずれか、または両方と、間隔を指定することで設定できます。間隔の指定によって、CRMにリソースステータスの確認頻度を指示します。

手順 4.8　監視操作の追加または変更

1 4.1項 「Linux HA Management Client」 (32 ページ)で説明したように、Linux HA Management Clientを起動してクラスタにログインします。

2 Linux HA Management Clientのメインウィンドウで、左側のペインの*リソース*をクリックして、そのクラスタ用に設定されているリソースを表示します。

3 右側のペインで、変更するリソースを選択して*編集*をクリックします。次のウィンドウには、そのリソースに定義された基本的なリソースパラメータとメタ属性、インスタンス属性、および操作が表示されます。

4 新しい監視操作を追加するには、該当するタブを選択して*追加*をクリックします。

既存の操作を変更するには、該当するエントリを選択して*編集*をクリックします。

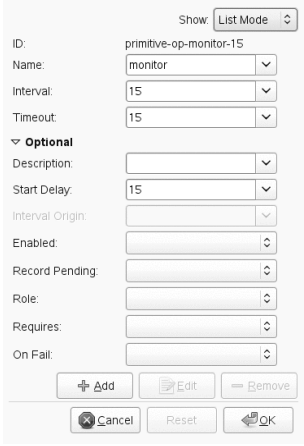
5 監視操作に固有の*ID*を入力します。既存の監視操作を変更する場合、IDはすでに定義されているため、環境設定ダイアログに表示されます。

6 *名前*で、`monitor`、`start`、*stop*など、実行するアクションを選択します。

7 *間隔*フィールドに、値を秒単位で入力します。

8 タイムアウトフィールドに、値を秒単位で入力します。指定したタイムアウトを過ぎると、操作は`failed`と見なされます。PEは何を行うか、あるいは監視操作の*On Fail (障害発生時の動作)*フィールドで指定した内容を実行するかどうかを判断します。

9 必要に応じて、オプションのパラメータを設定します。たとえば*On Fail (障害発生時の動作)*(アクション失敗時の動作) や*必要*(このアクションを発声する前に満たす必要がある条件)を設定します。



10 すべてのパラメータが希望どおりに設定されたら、*OK*をクリックして、そのリソースの設定を完了します。構成ダイアログが閉じて、メインウィンドウに変更されたリソースが表示されます。

リソースの監視を設定しなかった場合、開始成功後のリソース障害は通知されず、クラスタは常にリソース状態を良好として表示してしまいます。

リソースモニタが障害を検出すると、次のアクションが行われます。

- `/etc/ais/openais.conf` (by default, written to syslog, usually `/var/log/messages`)の`logging`セクションの設定に従って、ログファイルメッセージが生成されます。

- 障害はLinux HA Management Client、`crm_mon`ツール、そしてCIBのステータスセクションにも反映されます。Linux HA Management Clientで表示するには、左側のペインで*管理*をクリックし、詳細を表示するリソースを右側のペインで選択します。

- クラスタはリソースを停止して障害状態を修復させ、別なノードでローカルにリソースを再起動するなど、顕著な復旧アクションを開始します。設定やクラスタの状態によっては、リソースがまったく再起動されないこともあります。

4.8 新しいクラスタリソースの開始

注意: リソースの開始

High Availability Extensionでリソースを設定する場合、同じリソースを手動で開始または停止するべきではありません(クラスタの外)。High Availability Extensionソフトウェアが、すべてのサービスの開始または停止アクションを実行します。

リソース作成時の最初の状態が`stopped`に設定されている場合(`target-role`メタ属性の値が`stopped`)、作成後は自動的に開始されません。Linux HA Management Clientで新しいクラスタリソースを開始するには、左側のペインで*管理*を選択します。右側のペインでリソースを右クリックして、*開始*を選択します(またはツールバーから開始)。

4.9 クラスタリソースの削除

Linux HA Management Clientでクラスタリソースを削除するには、左側のペインで*リソース*ビューに切り替え、該当するリソースを選択して*削除*をクリックします。

注意: 参照されているリソースの削除

任意の制約によってIDが参照されているクラスタリソースは削除できません。リソースを削除できない場合は、リソースIDが参照されている場所を確認し、最初に制約からリソースを削除します。

4.10 クラスタリソースグループの設定

クラスタリソースの中には、他のコンポーネントやリソースに依存しているものがあり、各コンポーネントやリソースを特定の順番で開始したり、同じサーバ上で一緒に実行しなければならないものもあります。この構成を簡単にするため、グループのコンセプトをサポートしています。

グループには次のプロパティがあります。

リソースの開始と停止
: リソースは表示された順番で開始し、逆の順番で停止します。

依存関係
: グループ内のリソースがどこかで開始できない場合は、グループ内のその後の全リソースは実行することができません。

グループの内容
: グループにはプリミティブクラスタリソースしか含むことができません。グループリソースの子を参照するには、グループではなく子のIDを使用します。

制約
: 制約でグループの子を参照することはできますが、通常はグループ名を使用することをお勧めします。

固着性
: 固着性はグループ内で統合可能なプロパティです。グループ内のアクティブな各メンバーは、グループの合計値に対して固着性を追加します。したがって`resource-stickiness`が`100`で、グループに7つのメンバーがあり、そのうち5つがアクティブな場合、グループ全体として現在の場所への固着性が`500`になります。

リソース監視
: グループのリソース監視を有効にするには、グループ内で監視の必要な各リソースに対して監視を設定する必要があります。

注意: 空のグループ

グループには1つ以上のリソースを含む必要があります。空の場合は設定は無効になります。

手順 4.9 *リソースグループの追加*

1. 4.1項 「Linux HA Management Client」 (32 ページ)で説明したように、Linux HA Management Clientを起動してクラスタにログインします。
2. 左側のペインで*リソース*を選択し、*追加*> *グループ*の順にクリックします。

3 グループに固有のIDを入力します。

4 *オプション*の下で、*Initial state of resource (リソースの当初の状態)*を設定し、*転送*をクリックします。

5 次のステップでは、グループのサブリソースとしてプリミティブを追加できます。プリミティブはプリミティブリソースの追加 (34 ページ)と類似の方法で作成します。

6 すべてのパラメータが希望どおりに設定されたら、*適用*をクリックして、プリミティブの設定を完了します。

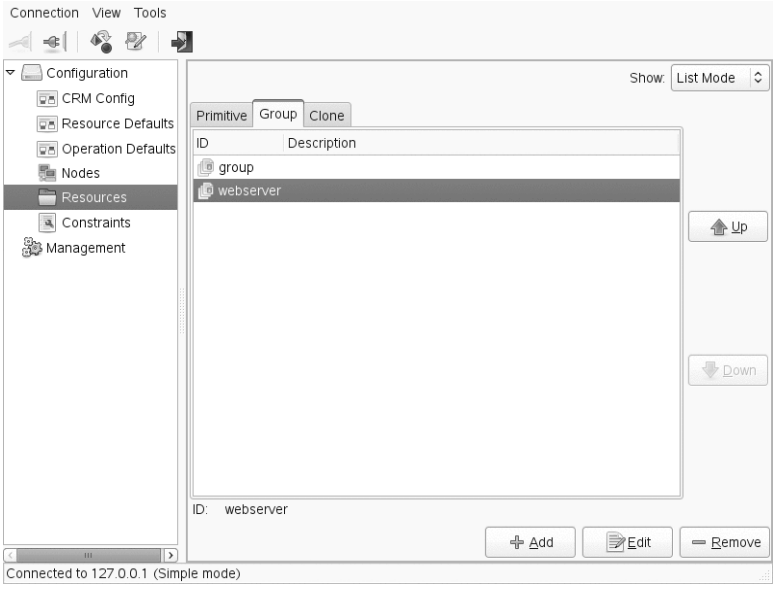
7 次のウィンドウでは、再度プリミティブを選択し、*OK*をクリックすることで、グループのサブリソースの追加を継続できます。

グループに追加するプリミティブがなくなったら、代わりにキャンセルをクリックします。次のウィンドウには、そのグループに定義したパラメータの概要が表示されます。グループの*Meta Attributes(メタ属性)*およびプリミティブの一覧が表示されます。プリミティブタブのリソースの場所は、クラスタ内でのリソース開始順序を示します。

8 グループ内のリソース順序は重要なので、*Up(上に移動)*および*Down(下に移動)*ボタンを使用して、グループ内でプリミティブのソートまたはソート変更を行います。

9 すべてのパラメータが希望どおりに設定されたら、*OK*をクリックして、そのグループの設定を完了します。環境設定ダイアログが閉じ、メインウィンドウに新しく作成された、または変更されたグループが表示されます。

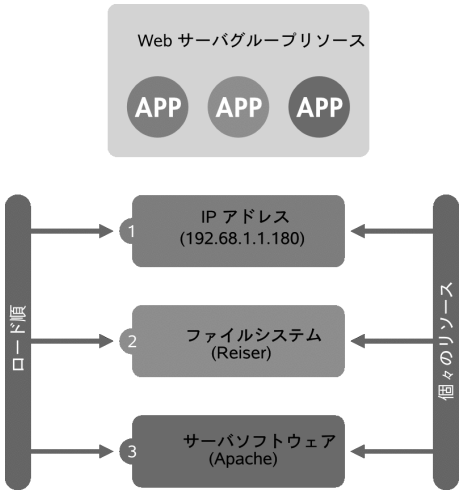
図 4.4　Linux HA Management Client - グループ



例 4.1　Webサーバのリソースグループ

リソースグループの1例として、IPアドレスとファイルシステムが必要なWebサーバが挙げられます。この場合、各コンポーネントはクラスタリソースグループに結合された個別のクラスタリソースです。リソースグループは1つ以上のサーバで実行し、ソフトウェアやハードウェアの障害発生時には、個々のクラスタリソースと同様に、クラスタ内の別なサーバにフェールオーバーします。

図 4.5　グループリソース



リソースグループの追加 (51 ページ)では、リソースグループの作成方法を説明しました。上記で説明したようなリソースグループを作成済みであると仮定します。既存のグループへのリソースの追加 (54 ページ)は例 4.1.「Webサーバのリソースグループ」 (53 ページ)と一致するグループの変更方法を示したものです。

手順 4.10　*既存のグループへのリソースの追加*

1. 4.1項 「Linux HA Management Client」 (32 ページ)で説明したように、Linux HA Management Clientを起動してクラスタにログインします。

2. 左側のペインでリソースビューに切り替え、右側のペインで、変更するグループを選択して*編集*をクリックします。次のウィンドウには、そのリソースに定義された基本的なグループパラメータとメタ属性とプリミティブが表示されます。

3. *プリミティブ*タブをクリックして、*追加*をクリックします。

4. 次のダイアログで、次のパラメータを設定してIPアドレスをグループのサブリソースとして追加します。

   4a 固有なIDとして、`my_ipaddress`などを入力します。

4b クラスリストで、リソースエージェントクラスとして*ocf*を選択します。

4c OCFリソースエージェントの*プロバイダ*として、*heartbeat*を選択します。

4d *タイプ*リストで、リソースエージェントとして*IPaddr*を選択します。

4e *進む*をクリックします。

4f *Instance Attribute(インスタンス属性)*タブで、*IP*エントリを選択して*編集*をクリックします(または*IP*エントリをダブルクリックします)。

4g *値*として、目的のIPアドレスを入力します。たとえば`192.168.1.1`と入力します。

4h *OK*、*適用*の順にクリックします。グループ設定ダイアログには、新しく追加去れたプリミティブが表示されます。

5 再度*追加*をクリックして、次のサブリソース(ファイルシステムとWebサーバ)を追加します。

6 「ステップ 4a (54 ページ)」から「ステップ 4h (55 ページ)」のような手順に従って、グループの全サブリソースの設定を終了するまで、各サブリソースの該当するパラメータを設定します。

クラスタ内で開始する順序でサブリソースを設定したので、*プリミティブ*タブ内の順序はすでに正しいものになっています。

7 グループのリソースグループの順序を変更する必要がある場合は、*Up (上に移動)*および*Down (下に移動)*ボタンを使用して*プリミティブ*タブ内のリソースをソートしなおします。

8 グループからリソースを削除するには、*プリミティブ*タブのリソースを選択し、*削除*をクリックします。

9 *OK*をクリックしてそのグループの設定を完了します。環境設定ダイアログが閉じて、メインウィンドウに変更されたグループが表示されます。

4.11 クローンリソースの構成

クラスタ内の複数のノードで特定のリソースを同時に実行することができます。このためには、リソースをクローンとして設定する必要があります。クローンとして設定するリソースの1例として、STONITHや、OCFS2などのクラスタファイルシステムが挙げられます。リソースのリソースエージェントがサポートしていれば、任意のリソースをクローン化することができます。クローンリソースは、ホスティングされているノードによって異なる設定をすることもできます。

リソースクローンには次の3つのタイプがあります。

匿名クローン
: 最も簡単なクローンタイプです。実行場所にかかわらず、同じ動作をします。このため、マシンごとにアクティブな匿名クローンのインスタンスは1つだけ存在できます。

グローバルに固有なクローン
: このリソースは独自のエントリです。1つのノードで実行しているクローンのインスタンスは、別なノードの別なインスタンスとは異なり、同じノードの2つのインスタンスが同一になることもありません。

ステートフルなクローン
: このリソースのアクティブインスタンスは、アクティブとパッシブという2つの状態に分けられます。プライマリとセカンダリ、またはマスタとスレーブと呼ばれることもあります。ステートフルなクローンが、匿名またはグローバルに固有の場合もあります。

手順 4.11 *クローンの追加または変更*

1. 4.1項 「Linux HA Management Client」 (32 ページ)で説明したように、Linux HA Management Clientを起動してクラスタにログインします。

2. 左側のペインで*リソース*を選択し、*追加*> *クローン*の順にクリックします。

3. クローンに固有のIDを入力します。

4. *オプション*の下で、*Initial state of resource (リソースの当初の状態)*を設定します。

5. クローンに設定するオプションを起動し、*転送*をクリックします。

6. 次のステップでは、*プリミティブ*または*グループ*をクローンのサブリソースとして追加することができます。プリミティブリソースの追加 (34 ページ)またはリソースグループの追加 (51 ページ)で説明している方法のように作成します。

7. クローン設定ダイアログ内のすべてのパラメータが希望どおりに設定されたら、*適用*をクリックして、クローンの設定を完了します。

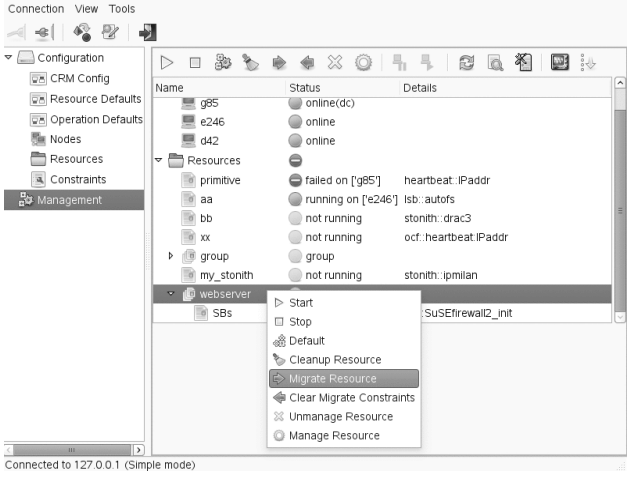
4.12 クラスタリソースのマイグレーション

「4.5項 「リソースフェールオーバーノードの指定」 (44 ページ)」で説明したように、ソフトウェアまたはハードウェアの障害時には、クラスタは定義可能な特定のパラメータ(たとえばマイグレーションしきい値やリソースの固着性など)に従って、リソースを自動的にフェールオーバー(マイグレート)させます。それ以外に、クラスタリソース内の別なノードにリソースを手動でマイグレートさせることもできます。

手順 4.12 *手動によるリソースのマイグレーション*

1. 4.1項 「Linux HA Management Client」 (32 ページ)で説明したように、Linux HA Management Clientを起動してクラスタにログインします。

2. 左側のペインの*管理*ビューに切り替え、次に右側のペインの該当するリソースを右クリックして*Migrate Resource (リソースのマイグレート)*を選択します。



3. 新規ウィンドウで、*To Node(マイグレート先ノード)*で、リソースの移動先ノードを選択します。これによって移動先ノードに対して`INFINITY`スコアの場所の制約が作成されます。

4 リソースを一時的にマイグレートするには、*Duration(期間)*をアクティブにしてリソースが新規ノードにマイグレートされる時間を入力します。指定した時間を経過したら、リソースは元の場所に戻ることができます。あるいは、現在の場所に残ることもできます(リソースへの固着性による)。

5 リソースをマイグレートできない場合は(リソースの固着性と制約スコアの合計が現在のノードでINFINITYを超えている場合)、*強制*オプションを有効にします。これによって現在の場所に対するルールと-INFINITYのスコアを作成して、リソースを強制的に移動させます。

注意

その結果、*Clear Migrate Constraints (マイグレート制約の消去)*によって制約が削除されるまで、または指定期間を経過するまでこのノードで実行を継続することを防止します。

---

6 *OK*をクリックして、マイグレーションを確認します。

リソースを再度元に戻すには、*管理*に切り替え、リソースビューを右クリックして*Clear Migrate Constraints (マイグレート制約の消去)*を選択します。これによってcrm_resource -Uコマンドが使用されます。リソースは元の場所に戻ることができます。あるいは現在の場所に残ることもできます(リソースの固着性によって)。詳細は、crm_resource(8) (172 ページ)またはhttp://clusterlabs.org/wiki/Documentationに提供されている『*Configuration 1.0の概要*』の「*リソースのマイグレーション*」のセクションを参照してください。

4.13 詳細情報

http://clusterlabs.org/
: High Availability Extensionに含まれているクラスタリソースマネージャであるPacemakerのホームページ。

http://linux-ha.org
: The High Availability Linuxプロジェクトのホームページ。

http://clusterlabs.org/wiki/Documentation
『*CRM*コマンドラインインタフェース』: crmコマンドラインツールの紹介。

http://clusterlabs.org/wiki/Documentation
『*Configuration 1.0*の概要』: Pacemakerの設定に必要なコンセプトを説明します。包括的で非常に詳細な参照用情報です。

クラスタリソースのコマンドラインからの設定

5

第4章 (31 ページ)と同様に、クラスタリソースは、クラスタ内のサーバ上で動作するすべてのリソースまたはアプリケーションに対して作成する必要があります。クラスタリソースには、Webサイト、電子メールサーバ、データベース、ファイルシステム、仮想マシン、およびユーザが常時使用できるその他のサーバベースのアプリケーションまたはサービスなどが含まれます。

グラフィカルHA管理クライアントユーティリティ、またはcrmコマンドラインユーティリティを使用してリソースを作成できます。この章では、いくつかのcrmユーティリティを紹介します。

5.1 コマンドラインツール

インストール後、クラスタを管理するいくつかのツールを利用できます。通常は、crmコマンドのみが必要です。このコマンドには、いくつかのサブコマンドがあります。crm helpを実行すると、使用できるすべてのコマンドの概要が表示されます。例が組み込まれた、完全なヘルプシステムが用意されています。

crmツールには管理機能(サブコマンドresourcesおよびnode)があり、構成に使用されます(cib、configure)。管理サブコマンドは*即座*に反映されますが、構成には最終的なコミットが必要です。

5.2 構成の変更のデバッグ

変更をクラスタにロードする前に、変更をptestで確認することを推奨します。ptestはコミットする変更によって生じる操作の図を表示します。図を表示するには、praphizパッケージが必要です。次の例は監視操作を追加するスクリプトです。

```
# crm
crm(live)# configure
crm(live)configure# show fence-node2
primitive fence-node2 stonith:apcsmart \
        params hostlist="node2"
crm(live)configure# monitor fence-node2 120m:60s
crm(live)configure# show changed
primitive fence-node2 stonith:apcsmart \
        params hostlist="node2" \
        op monitor interval="120m" timeout="60s"
crm(live)configure# ptest
crm(live)configure# commit
```

5.3 クラスタリソースの作成

クラスタで使用できるRA(リソースエージェント)には3種類あります。最初は、従来のHeartbeat 1スクリプトです。高可用性ではLSB初期化スクリプトを使用できます。最後に、クラスタには固有のOCF(Open Cluster Framework)エージェントのセットがあります。このマニュアルでは、LSBスクリプトとOCFエージェントについて説明します。

クラスタリソースを作成するには、crmツールを使用します。新しいリソースをクラスタに追加するには、一般的には次のような手順になります。

1. シェルを開いてrootになります。
2. crmを実行して、crmのインストールシェルを開きます。プロンプトがcrm(live)#に変化します。
3. プリミティブIPアドレスを設定ます。

```
crm(live)# configure
crm(live)configure# primitive myIP ocf:heartbeat:IPaddr \
     params ip=127.0.0.99 op monitor intervall=60s
```

前のコマンドは「プリミティブ」に名前myIPを設定します。クラス(ここではocf)、プロバイダ(heartbeat)、およびタイプ(IPaddr)が必要です。さらに、このプリミティブではIPアドレスのようなパラメータを予期します。アドレスをセットアップに合わせて変更する必要があります。

4 行った変更を表示して確認します。

```
crm(live)configure# show
```

XML構造を表示するには、次を使用します。

```
crm(live)configure# show xml
```

5 変更をコミットして反映させます。

```
crm(live)configure# commit
```

5.3.1 LSB初期化スクリプト

すべてのLSBスクリプトは一般にディレクトリ/etc/init.dにあります。いくつかのアクションが実装され、少なくともstart、stop、restart、reload、force-reload、statusが実装されている必要があります(http://www.linux-foundation.org/spec/refspecs/LSB_1.3.0/gLSB/gLSB/iniscrptact.htmlで説明)。

これらのサービスの構成は標準化されていません。LSBをHigh Availabilityとともに使用する場合、それぞれのスクリプトの構成方法を理解してください。これについてのマニュアルは、/usr/share/doc/packages/*PACKAGENAME*の対応するパッケージのマニュアルにあることがあります。

注意: 高可用性で使用されるサービスは変更しないでください

High Availabilityとともに使用する場合、サービスはどのような方法でも変更してはいけません。ブート、再起動、または手動でも開始または停止もしてはいけないということです。ただし、サービスが適切に構成されているか確認したい場合は手動で開始しますが、High Availabilityが起動する前に停止してください。

LSBリソースを使用する前に、このリソースの構成が存在し、すべてのクラスタノードで同一であることを確認してください。構成はHigh Availabilityによって管理されません。自分自身で管理する必要があります。

5.3.2 OCFリソースエージェント

すべてのOCFエージェントは/usr/lib/ocf/resource.d/heartbeat/にあります。これらはLSBスクリプトと同様の機能を持つ小規模なプログラムです。ただし、構成は常に環境変数で実行されます。すべてのOCFリソースエージェントは少なくともstart、stop、status、monitor、meta-dataのアクションを持つ必要があります。meta-dataアクションは、エージェントの構成方法についての情報を取得します。たとえば、IPaddrエージェントについて詳細を知りたい場合は、次のコマンドを使用します。

```
OCF_ROOT=/usr/lib/ocf /usr/lib/ocf/resource.d/heartbeat/IPaddr meta-data
```

シンプルなXMLフォーマットに多くの情報が出力されます。出力はra-api-1.dtdDTDで検証できます。基本的に、このXMLフォーマットには3つのセクションがあります。最初ははいくつかの共通の説明、2番目は使用できるすべてのパラメータ、最後はこのエージェントの使用できるアクションです。

この出力は機械が読み取れる形式で、理解は困難です。このため、crmツールにはraコマンドがあり、リソースエージェントに関する別の情報を取得します。

```
# crm
crm(live)# ra
crm(live)ra#
```

コマンドclassesは、すべてのクラスとプロバイダのリストを返します。

```
crm(live)ra# classes
stonith
lsb
ocf / lvm2 ocfs2 heartbeat pacemaker
heartbeat
```

クラス(およびプロバイダ)に使用できるすべてのリソースエージェントの概要を取得するには、listを使用します。

```
crm(live)ra# list ocf
AudibleAlarm       ClusterMon         Delay              Dummy
Filesystem         ICP                IPaddr             IPaddr2
IPsrcaddr          IPv6addr           LVM                LinuxSCSI
```

```
MailTo              ManageRAID          ManageVE              Pure-FTPd
Raid1               Route               SAPDatabase           SAPInstance
SendArp             ServeRAID           SphinxSearchDaemon    Squid
...
```

リソースエージェントの詳細情報は、metaで表示できます。

```
crm(live)ra# meta Filesystem ocf heartbeat
Filesystem resource agent (ocf:heartbeat:Filesystem)

Resource script for Filesystem. It manages a Filesystem on a shared storage
medium.

Parameters (* denotes required, [] the default):
...
```

ビューアはQを押して終了できます。構成の例は、第6章 シンプルなテスティングリソースのセットアップ(77 ページ)を参照してください。

5.3.3 NFSサーバの構成例

NFSサーバをセットアップするには、ファイルシステムリソース、drbdリソース、NFSサーバのグループとIPアドレスの3つのリソースが必要です。次のサブセクションでは、セットアップ方法を示します。

ファイルシステムリソースのセットアップ

filesystemリソースはOCFプリミティブリソースとして構成されます。開始および停止の要求時に、デバイスをディレクトリにマウントおよびアンマウントするタスクを担当しています。この場合、デバイスは/dev/drbd0でマウントポイントとして使用するディレクトリは/srv/failoverです。使用されるファイルシステムはxfsです。

次のコマンドをcrmシェルで使用して、ファイルシステムリソースを構成します。

```
crm(live)# configure
crm(live)configure# primitive filesystem_resource \
    ocf:heartbeat:Filesystem \
    params device=/dev/drbd0 directory=/srv/failover fstype=xfs
```

drbdの構成

drbd High Availability構成を開始する前に、drbdデバイスを手動でセットアップします。基本的には、これは/etc/drbd.confでdrbdを構成し、同期させています。drbd構成の正確な手順は、*Storage Administration Guide*に記載されています。ここでは、両方のクラスタノードの/dev/drbd0デバイスでアクセスされる、リソースr0を構成したと仮定します。

drbdリソースはOCFマスタスレーブリソースです。これはdrbd RAのメタデータの説明にあります。ただし、より重要なことは、メタデータのactionsセクションにはpromoteとdemoteアクションがあることです。これらはマスタスレーブリソースに必須で、その他のリソースでは一般に利用できません。

High Availabilityについては、マスタスレーブリソースは別のノードに複数のマスタを持つことがあります。マスタトスレーブを同じノードに持つこともできます。このため、このリソースをただ1つのマスタとスレーブを持ち、それぞれが別のノードで実行するように構成します。masterリソースのmeta属性でこれを実行します。マスタスレーブリソースはHigh Availabilityでのクローンリソースの特殊な種類です。マスタとスレーブはそれぞれクローンとしてカウントされます。

次のコマンドをcrmシェルで使用して、マスタスレーブリソースを構成します。

```
crm(live)# configure
crm(live)configure# primitive drbd_r0 ocf:heartbeat:drbd params
crm(live)configure# ms drbd_resource drbd_r0 \
   meta clone_max=2 clone_node_max=1 master_max=1 master_node_max=1 notify=true
crm(live)configure# commit
```

NFSサーバとIPアドレス

NFSサーバを常に同じIPアドレスで使用できるようにするため、マシンが通常操作に使用するアドレスのほかに追加のIPアドレスを使用します。このIPアドレスは、システムのIPアドレスに加えてアクティブなNFSサーバに割り当てられます。

NFSサーバとNFSサーバのIPアドレスは常に同じマシン上でアクティブにする必要があります。この場合、開始順序はそれ程重要ではありません。同時に開始してかまいません。これはグループリソースの代表的な要件です。

High Availability RA構成を開始する前に、NFSサーバをYaSTで構成します。システムでNFSサーバを起動させないでください。環境設定ファイルをセットアップするだけにします。この作業を手動で行うには、マニュアルページのexports(5) (`man 5 exports`)を参照してください。環境設定ファイルは`/etc/exports`です。NFSサーバはLSBリソースとして構成されます。

IPアドレスをHigh Availability RA構成で完全に構成します。システムではこれ以外の変更は不要です。IPアドレスRAはOCF RAです。

```
crm(live)# configure
crm(live)configure# primitive nfs_resource lsb:nfsserver
crm(live)configure# primitive ip_resource ocf:heartbeat:IPaddr \
      params ip=10.10.0.1
crm(live)configure# group nfs_group nfs_resource ip_resource
crm(live)configure# commit
crm(live)configure# end
crm(live)# quit
```

5.4 STONITHリソースの作成

`crm`からは、STONITHデバイスは単なる1つのリソースと認識されます。STONITHリソースを作成するには、次の手順に従います。

1. `crm`コマンドをシステム管理者として実行します。プロンプトが`crm(live)`に変化します。
2. 次のコマンドで、すべてのSTONITHタイプのリストを取得します。

```
crm(live)# ra list stonith
apcmaster               apcsmart                baytech
cyclades                drac3                   external/drac5
external/hmchttp        external/ibmrsa         external/ibmrsa-telnet
external/ipmi           external/kdumpcheck     external/rackpdu
external/riloe          external/sbd            external/ssh
external/vmware         external/xen0           external/xen0-ha
ibmhmc                  ipmilan                 meatware
null                    nw_rpc100s              rcd_serial
rps10                   ssh                     suicide
```

3. 上記のリストからSTONITHタイプを選択し、利用できるオプションのリストを表示します。次のコマンドを使用します(ビューアを閉じるにはQを押します)。

```
crm(live)# ra meta external/ipmi stonith
IPMI STONITH external device (stonith:external/ipmi)

IPMI-based host reset

Parameters (* denotes required, [] the default):
...
```

4 次のように、クラスstonith、ステップ3で選択したタイプ、必要に応じてそれぞれのパラメータでSTONITHリソースを作成します。

```
crm(live)# configure
crm(live)configure# primitive my-stonith stonith:external/ipmi \
   meta target-role=Stopped \
   operations my_stonith-operations \
     op monitor start-delay=15 timeout=15 hostlist='' \
                pduip='' community=''
```

5.5 リソース制約の設定

すべてのリソースを構成することは、ジョブのほんの一部分です。クラスタが必要なすべてのリソースを認識しても、正しく処理できるとは限りません。たとえば、ファイルシステムをdrbdのスレーブノードにマウントしようとしても意味がありません(実際、drbdでは失敗します)。クラスタにこのようなことを通知するには、制約を定義します。

High Availabilityでは、使用できる制約には3種類あります。

- リソースを実行するノードを定義する、場所の制約(crmシェルで、locationコマンドで定義)。
- クラスタにノード上で一緒に実行できる、または実行できないリソースを通知する、コロケーションの制約(colocation)。
- アクションの順序を定義する、順序の制約(order)。

5.5.1 場所の制約

この種類の制約は、各リソースに複数追加できます。すべてのrsc_location制約は、所定のリソースに対して評価されます。IDfs1-locのリソースを名

前earthのノードで実行する確率を100に増加する簡単な例を、次に示します。

```
crm(live)configure# location fs1-loc fs1 100: earth
```

5.5.2 コロケーションの制約

colocationコマンドは、同じホストまたは別のホストで実行するリソースを定義するために使用します。通常、次の順序を使用します。

```
crm(live)configure# order rsc1 rsc2
crm(live)configure# colocation rsc2 rsc1
```

+INFINITYまたは-INFINITYのスコアのみが設定でき、同じノードで常に実行されるか、または常に実行されないリソースを定義します。たとえば、IDがfilesystem_resourceとnfs_groupの2つのリソースを常に同じホストで実行するには、次の制約を使用します。

```
crm(live)configure# colocation nfs_on_filesystem inf: nfs_group
filesystem_resource
```

マスタスレーブ構成では、現在のノートがマスタかどうかと、リソースをローカルに実行しているかどうかを把握することが必要です。これは追加のto_roleまたはfrom_role属性からチェックできます。

5.5.3 順序の制約

サービスを開始させる順序の指定が必要となる場合があります。たとえば、デバイスがシステムで利用できるようになるまで、ファイルシステムはマウントできません。順序の制約を使用して、開始、停止、マスタへの昇格など、別のリソースが特殊な条件を満たす直前または直後に、サービスを開始または停止できます。次のコマンドをcrmシェルで使用して、順序の制約を設定します。

```
crm(live)configure# order nfs_after_filesystem mandatory: group_nfs
filesystem_resource
```

5.5.4 サンプル構成の制約

この章で使用される例は、制約を追加しないと予期したように動作しません。すべてのリソースが同じマシン上でdrbdリソースのマスタとして実行することが不可欠です。もう1つの重要なことは、その他のリソースが開始する前にdrbdリソースがマスタになることが必要であることです。drbdがマスタでないときにdrbdデバイスをマウントしようとしても、失敗します。満たす必要がある制約は、次のようになります。

- ファイルシステムは常に同じノード上にdrbdリソースのマスタとして存在する必要があります。

```
crm(live)configure# colocation filesystem_on_master inf: \
    filesystem_resource drbd_resource:Master
```

- NFSサーバとIPアドレスは、ファイルシステムとして同じノード上に存在する必要があります。

```
crm(live)configure# colocation nfs_with_fs inf: \
   nfs_group filesystem_resource
```

- NFSサーバとIPアドレスは、ファイルシステムがマウントされた後に開始されます。

```
crm(live)configure# order nfs_second mandatory: \
   filesystem_resource nfs_group
```

- ファイルシステムは、drbdリソースがこのノードのマスタに昇格した後にマウントされる必要があります。

```
crm(live)configure# order drbd_first inf: \
    drbd_resource:promote filesystem_resource
```

5.6 リソースフェールオーバーノードの指定

リソースフェールオーバーを判定するには、メタ属性migration-thresholdを使用します。次に例を示します。

```
crm(live)configure# location r1-node1 r1 100: node1
```

通常、r1はノード1で実行されます。そこで失敗すると、migration-thresholdがチェックされ、失敗回数と比較されます。失敗回数がmigration-threshold以上の場合、次の候補のノードにマイグレートします。

start-failure-is-fatalオプションに応じて、Startの失敗によって失敗回数はINFINITYに設定されます。stopの失敗により、フェンシングが発生します。STONITHが定義されていない場合、リソースはまったくマイグレートされません。

5.7 リソースフェールバックノードの指定(リソースの固着性)

rscは、失敗数のためにマイグレートされた後、管理者が失敗回数をリセットするか、失敗が期限切れになった場合のみ、フェールバックします(failure-timeoutメタ属性を参照)。

```
crm resource failcount RSC delete NODE
```

5.8 リソース監視の設定

リソースを監視するには、opキーワードで監視処理を定義するか、monitorコマンドを使用するか、2つの方法があります。次の例では、Apacheリソースを構成して、opキーワードで30分ごとに監視します。

```
crm(live)configure# primitive apache apache \
  params ... \
  op monitor interval=60s timeout=30s
```

同じことを次のようにしても実行できます。

```
crm(live)configure# primitive apache apache \
   params ...
crm(live)configure# monitor apache 60s:30s
```

5.9 新しいクラスタリソースの開始

新しいクラスタリソースを開始するには、そのIDが必要です。次の手順に従います。

1. crmコマンドをシステム管理者として実行します。プロンプトがcrm(live)に変化します。
2. コマンドstatusでそれぞれのリソースを検索します。
3. 次のようにしてリソースを開始します。

```
crm(live)# resource start ID
```

5.10 クラスタリソースの削除

クラスタリソースを削除するには、そのIDが必要です。次の手順に従います。

1. crmコマンドをシステム管理者として実行します。プロンプトがcrm(live)に変化します。
2. 次のコマンドを実行して、リソースのリストを取得します。

```
crm(live)# resource status
```

たとえば、出力はこのようになります(ここでmyIPはリソースの該当するID)。

```
myIP    (ocf::IPaddr:heartbeat) ...
```

3. 該当するIDのリソースを削除します(commitも意味します)。

```
crm(live)# configure delete YOUR_ID
```

4. 変更をコミットします。

```
crm(live)# configure commit
```

5.11 クラスタリソースグループの構成

クラスタの共通要素の1つは、一緒に配置する必要があるリソースセットで、連続して開始し、その逆の順序で停止する必要があるものです。この構成を簡単にするため、グループのコンセプトをサポートしています。次の例では、2つのプリミティブを作成します(IPアドレスと電子メールリソース)。

1 crmコマンドをシステム管理者として実行します。プロンプトが crm(live)に変化します。

2 プリミティブを構成します。

```
crm(live)# configure
crm(live)configure# primitive Public-IP ocf:IPaddr:heartbeat \
   params ip=1.2.3.4
crm(live)configure# primitive Email lsb:exim
```

3 それぞれのIDでプリミティブを正しい順序でグループ化します。

```
crm(live)configure# group shortcut Public-IP Email
```

5.12 クローンリソースの構成

クローンは当初、IPアドレスのN個のインスタンスを開始し、負荷分散のためにクラスタ上に分散させる便利な方法と考えられていました。DLMとの統合、フェンシングサブシステム、OCFS2など、多数の目的にも非常に有効であることがわかってきました。リソースエージェントがサポートしていれば、どのようなリソースもクローン化できます。

次のような種類のクローン化されたリソースがあります。

匿名リソース
: 匿名クローンは、最もシンプルな種類です。これらのリソースは実行するどこでも完全に同一の振る舞いをします。このため、マシンごとにアクティブな匿名クローンのコピーは1つだけ存在できます。

マルチステートリソース
: マルチステートリソースは、クローンが得意とするところです。インスタンスを2つの操作モードのうちの1つにします。これらのモードは「マス

タ」と「スレーブ」と呼ばれますが、深い意味はありません。唯一の制限は、インスタンスの起動時の状態がスレーブでなければならないということです。

5.12.1 匿名クローンリソースの作成

匿名クローンリソースを作成するには、まずプリミティブリソースを作成して、それをcloneコマンドで指定することです。次の操作を実行します。

1. crmコマンドをシステム管理者として実行します。プロンプトがcrm(live)に変化します。
2. 次のように、プリミティブを構成します。

```
crm(live)# configure
crm(live)configure# primitive Apache lsb:apache
```

3. プリミティブをクローンします。

```
crm(live)configure# clone apache-clone Apache \
   meta globally-unique=false
```

5.12.2 ステートフル/マルチステートクローンリソースの作成

ステートフルクローンリソースを作成するには、まずプリミティブリソースを作成してから、マスタスレーブリソースを作成します。

1. crmコマンドをシステム管理者として実行します。プロンプトがcrm(live)に変化します。
2. プリミティブを作成します。必要に応じて間隔を変更します。

```
crm(live)# configure
crm(live)configure# primitive myRSC ocf:myCorp:myAppl \
  operations foo \
    op monitor interval=60 \
    op monitor interval=61 role=Master
```

3. マスタスレーブリソースを作成します。

```
crm(live)configure# clone apache-clone Apache \
   meta globally-unique=false
```

5.13 クラスタリソースのマイグレーション

リソースは、ハードウェアまたはソフトウェアに障害が発生した場合、クラスタ内の他のノードに自動的にフェールオーバー(マイグレーション)するよう設定されていますが、Linux HA Management Clientまたはコマンドラインを使用して、手動でリソースをクラスタ内の別のノードにマイグレートすることもできます。

1. crmコマンドをシステム管理者として実行します。プロンプトが crm(live)に変化します。
2. ipaddress1という名前のリソースをnode2という名前にクラスタノードにマイグレートするには、次のように入力します。

```
crm(live)# resource
crm(live)resource# migrate ipaddress1 node2
```

5.14 シャドー構成のテスト

注意: 熟練した管理者のみ

コンセプトは平易ですが、シャドー構成は、どうしても必要でHigh Availabilityについて経験があるときに限り使用することを推奨します。

シャドー構成は、異なる構成シナリオのテストに使用されます。複数のシャドー構成を作成した場合、1つずつテストして変更による影響を確認できます。

通常の処理は次のようになります。

1. crmツールを起動します。

2. configureサブコマンドに切り替えます。

```
crm(live)# configure
crm(live)configure#
```

3. 変更を行えるようになります。ただし、危険性があると考えられる場合、または後で適用したい場合は、新しいシャドー構成に保存できます。

```
crm(live)configure# cib new myNewConfig
INFO: myNewConfig shadow CIB created
crm(myNewConfig)configure# commit
```

4. シャドー構成を作成したあと、変更を行えます。

5. ライブクラスタ構成に戻り、このコマンドを使用します。

```
crm(myNewConfig)configure# cib use
crm(live)configure#
```

5.15 詳細情報

http://linux-ha.org
: High Availability Linuxのホームページ

http://www.clusterlabs.org/mediawiki/images/8/8d/Crm_cli.pdf
: CRM CLIツールの概要を説明しています。

http://www.clusterlabs.org/mediawiki/images/f/fb/Configuration_Explained.pdf
: Pacemaker構成を説明しています。

シンプルなテスティングリソースのセットアップ

6

第3章 *YaSTのインストールと基本セットアップ* (23 ページ)で説明したようにクラスタをインストールしてセットアップし、GUIまたはコマンドラインでリソースを構成する方法を理解した後、この章ではシンプルなリソース、IPアドレスの構成の基本的な例を示します。Linux HA Management Clientまたは`crm`コマンドラインツールをのいずれかを使用した、両方の方法を紹介します。

次の例では、クラスタには少なくとも2つのノードがあると仮定します。

6.1 GUIによるリソースの構成

サンプルのクラスタリソースを作成して別のサーバにマイグレートすると、クラスタが正常に機能していることの確認に役立ちます。構成とマイグレートのシンプルなリソースは、IPアドレスです。

手順 6.1 *IPアドレスクラスタリソースの作成*

1. 4.1項 「Linux HA Management Client」 (32 ページ)で説明したように、Linux HA Management Clientを起動してクラスタにログインします。
2. 左側のペインでリソースビューに切り替え、右側のペインで、変更するグループを選択して*編集*をクリックします。次のウィンドウには、そのリソースに定義された基本的なグループパラメータとメタ属性とプリミティブが表示されます。
3. *プリミティブ*をクリックして、*追加*をクリックします。

4 次のダイアログで、次のパラメータを設定してIPアドレスをグループのサブリソースとして追加します。

4a 固有なIDとして、myIPなどを入力します。

4b クラスリストで、リソースエージェントクラスとして*ocf*を選択します。

4c OCFリソースエージェントの*プロバイダ*として、*heartbeat*を選択します。

4d *タイプ*リストから、リソースエージェントとして*IPaddr*を選択します。

4e *進む*をクリックします。

4f *Instance Attribute(インスタンス属性)*タブで、*IP*エントリを選択して*編集*をクリックします(または*IP*エントリをダブルクリックします)。

4g *値*として、目的のIPアドレスを入力します。たとえば、10.10.0.1と入力して、*OK*をクリックします。

4h 新規インスタンス属性を*追加*して、nicを*名前*に、eth0を*値*に指定して*OK*をクリックします。

名前と値は、ハードウェア構成、およびHigh Availability Extensionソフトウェアのインストール中に選択したメディア構成とは独立しています。

5 すべてのパラメータが希望どおりに設定されたら、*OK*をクリックして、そのリソースの設定を完了します。構成ダイアログが閉じて、メインウィンドウに変更されたリソースが表示されます。

リソースをLinux HA Management Clientで起動するには、左側のペインの*管理*を選択します。右側のペインで、リソースを右クリックして*開始*を選択します(またはツールバーから開始します)。

IPアドレスリソースを別のノード(satum)にマイグレートするには、次のようにします。

手順 6.2　リソースの他のノードへのマイグレート

1. 左側のペインの*管理*ビューに切り替え、次に右側のペインのIPアドレスリソースを右クリックして*Migrate Resource(リソースのマイグレート)*を選択します。
2. 新規ウィンドウで、*To Node(マイグレート先ノード)*ドロップダウンリストで`satum`を選択し、選択したリソースをノード`satum`に移動します。
3. リソースを一時的にマイグレートするには、*Duration(期間)*をアクティブにしてリソースが新規ノードにマイグレートされる時間を入力します。
4. *OK*をクリックして、マイグレーションを確認します。

6.2 リソースの手動構成

リソースは、コンピュータが提供するあらゆる種類のサービスです。リソースはRA(リソースエージェント)によって管理されている場合はHigh Availabilityに認識され、これにはLSBスクリプト、OCFスクリプト、従来のHeartbeat 1リソースがあります。すべてのリソースは`crm`コマンドで、またはXMLとしてCIB(Cluster Information Base)の`resources`セクションで構成されます。使用できるリソースの概要は、第18章 *HA OCFエージェント* (209 ページ)を参照してください。

IPアドレス`10.10.0.1`をリソースとして現在の構成に追加するには、`crm`コマンドを使用します。

手順 6.3　IPアドレスクラスタリソースの作成

1. `crm`コマンドをシステム管理者として実行します。プロンプトが`crm(live)`に変化します。
2. `configure`サブコマンドに切り替えます。

```
crm(live)# configure
```

3. IPアドレスリソースを作成します。

```
crm(live)configure# resource
primitive myIP ocf:heartbeat:IPaddr params ip=10.10.0.1
```

注意

リソースをHigh Availabilityで構成する場合、同じリソースをinitで初期化できません。高可用性はすべてのサービスのstartまたはstopアクションを実施します。

構成が正常に終了した場合、新規リソースはクラスタのランダムノードで開始されたcrm_monに表示されます。

リソースを別のノードにマイグレートするには、次のようにします。

手順 6.4 *リソースの他のノードへのマイグレート*

1. シェルを起動してrootになります。
2. リソースmyipをノードsaturnにマイグレートします。

```
crm resource migrate myIP saturn
```

リソースエージェントの追加または変更

7

クラスタで管理が必要なすべてのタスクはリソースとして使用できなければなりません。区別しなければならない主なグループとして、リソースエージェントとSTONITHエージェントの2つがあります。両方のカテゴリで、エージェントの追加や所有が可能で、クラスタ機能を各自のニーズに合わせて拡張することができます。

7.1 STONITHエージェント

クラスタではノードの1つの誤動作が検出され、それを除外する必要があることがあります。これをフェンシングと呼び、一般にSTONITHリソースで実行されます。すべてのSTONITHリソースは各ノードの`/usr/lib/stonith/plugins`にあります。

警告: SSHおよびSTONITHはサポートされていません。

SSHが他のシステムの問題にどのような反応するのかを知る方法はありません。そのため、SSHとSTONITHエージェントは本番環境用にはサポートされていません。

現在使用可能なすべてのSTONITHデバイス(ソフトウェア側から)のリストを入手するには、コマンド`stonith -L`を使用します。

残念ながら、STONITHエージェントの作成方法についてのドキュメントはまだありません。新しいSTONITHエージェントを作成する場合は、

heartbeat-commonパッケージのソースに提供されている例を参照してください。

7.2 OCFリソースエージェントの作成

すべてのOCF RAが/usr/lib/ocf/resource.d/に提供されています。詳細は17.1項「サポートされるリソースエージェントクラス」 (201 ページ)を参照してください。名前の重複を避けるため、新しいリソースエージェントを作成するときには別にサブディレクトリを作成してください。たとえばリソースグループkitchenにリソースcoffee_machineがある場合、このリソースを/usr/lib/ocf/resource.d/kitchen/ディレクトリに追加します。このRAにアクセスするには、コマンドcrmを実行します。

```
configure
primitive coffee_1 ocf:coffee_machine:kitchen ...
```

独自のOCF RAを実装する場合は、このエージェントに複数のアクションを提供します。OCFリソースエージェントの作成についての詳細は、http://www.linux-ha.org/OCFResourceAgentを参照してください。High Availability 2のコンセプトについての詳細は、第1章 *概念の概要* (3 ページ)を参照してください。

8 フェンシングとSTONITH

フェンシングはHA(High Availability)向けコンピュータクラスタにおいて、非常に重要なコンセプトです。クラスタではノードの1つの誤動作が検出され、それを除外する必要があることがあります。これをフェンシングと呼び、一般にSTONITHリソースで実行されます。フェンシングは、HAクラスタを既知の状態にするための方法として定義できます。

クラスタのリソースにはそれぞれの状態が関連付けられており、たとえば、「リソースr1はnode1で起動された」などです。HAクラスタでは、このような状態は「リソースr1はnode1以外のすべてのノードで停止している」ことを示します。HAクラスタは各リソースが1つのノードでのみ起動されるようにするためです。各ノードはリソースに生じた変更を報告する必要があります。つまり、クラスタの状態は、リソースの状態とノードの状態の集まりです。

どのような理由であれ、一部のノードまたはリソースの状態を正確に確立できない場合、フェンシングが行われます。クラスタが一部のノードで発生していることを認識していなくても、フェンシングによりこのノードで重要なリソースが実行されないように保証できます。

8.1 フェンシングのクラス

フェンシングには、リソースレベルとノードレベルのフェンシングという、2つのクラスがあります。後者について、この章で主に説明します。

リソースレベルのフェンシング
: リソースレベルのフェンシングを使用して、クラスタはノードが1つ以上のリソースにアクセスできないようにさせます。代表的な一例はSANで、フェンシング操作によってSANスイッチのルールを変更し、ノードからのアクセスを拒否します。

  リソースレベルのフェンシングは、保護対象のリソースが依存している通常のリソースを使用して実行できます。このようなリソースは、このノードでの起動を拒否するため、それに依存するリソースは同じノード上で実行されません。

ノードレベルのフェンシング
: ノードレベルのフェンシングは、ノードがどのリソースも実行しないようにします。これは通常シンプルですが乱暴な方法で実行され、ノードは電源スイッチを使用してリセットされます。ノードはまったく反応しなくなるため、最終的な方法となります。

8.2 ノードレベルのフェンシング

SUSE® Linux Enterprise High Availability Extensionでは、フェンシングの実装はSTONITH(Shoot The Other Node in the Head)です。これにより、ノードレベルのフェンシングが実行されます。High Availability Extensionには`stonith`コマンドラインツールが付属し、これはクラスタ上のノードの電源をリモートでオフにする拡張インタフェースです。使用できるオプションの概要については、`stonith --help`を実行するか、または`stonith`のマニュアルページで詳細を参照してください。

8.2.1 STONITHデバイス

ノードレベルのフェンシングを使用するには、まずフェンシングデバイスを用意する必要があります。High Availability Extensionでサポートされている STONITHデバイスのリストを取得するには、次のコマンドを`root`として任意のノード上で実行します。

```
stonith -L
```

STONITHデバイスは次のカテゴリに分類できます。

電源分配装置(PDU)
: 電源分配装置は、重要なネットワーク、サーバ、データセンター装置の電力と機能を管理する、重要な要素です。接続した装置のリモートロード監視と、個々のコンセントでリモート電源オン/オフのための電力制御を実行できます。

無停電電源装置(UPS)
: 無停電電源装置は、通常の電力が使用できない場合に別の電源から電力を供給し、接続した装置へ非常電力を供給します。

ブレード電源制御デバイス
: クラスタを一連のブレード上で実行している場合、ブレードエンクロージャの電源制御デバイスがフェンシングの唯一の候補となります。当然、このデバイスは1台のブレードコンピュータを管理できる必要があります。

ライトアウトデバイス
: ライトアウトデバイス(IBM RSA、HP iLO、Dell DRAC)は急速に広まっており、今後は既成コンピュータの標準装備となると思われます。ただし、電源をホスト(クラスタノード)と共有するため、これらはUPSデバイスに内蔵されています。ノードに電力が供給されないままでは、それを制御するデバイスも役に立ちません。この場合、CRMはノードをフェンスしようとして失敗し、これが繰り返されます。その他すべてのリソース操作がフェンシング/STONITH操作の成功を待機するからです。

テスティングデバイス
: テスティングデバイスは、テスト専用に使用されます。通常、ハードウェアにあまり負担をかけないようになっています。クラスタが運用に使用される際には、実際のフェンシングデバイスに交換されます。

STONITHデバイスは、予算と使用するハードウェアの種類に応じて選択します。

8.2.2 STONITHの実装

SUSE® Linux Enterprise High Availability Extension のSTONITH実装には、2つのコンポーネントがあります。

stonithd
: stonithdは、ローカルプロセスまたはネットワーク経由でアクセスできるデーモンです。フェンシング操作に対応する、rest、power-off、power-on

コマンドを受け付けます。フェンシングデバイスの状態チェックも行います。

stonithdデーモンはCRM HAクラスタの各ノードで実行されます。DCノードで実行されるstonithdインスタンスは、CRMからフェンシング要求を受け取ります。目的のフェンシング操作を実行するのは、このインスタンスとその他のstonithdプログラムです。

STONITHプラグイン
: サポートするフェンシングデバイスそれぞれについて、このデバイスを制御するSTONITHプログラムがあります。STONITHプラグインはフェンシングデバイスへのインタフェースです。すべてのSTONITHプラグインは各ノードの`/usr/lib/stonith/plugins`にあります。すべてのSTONITHプラグインはstonithdからは同一のものと認識されますが、フェンシングデバイスの性質を反映しているため、大きな違いがあります。

  一部のプラグインは、複数のデバイスをサポートします。代表的な例は`ipmilan`(または`external/ipmi`)で、IPMIプロトコルを実装し、このプロトコルをサポートする任意のデバイスを制御できます。

8.3 STONITHの構成

フェンシングを構成するには、複数のSTONITHリソースを構成する必要があります。stonithdデーモンでは構成は不要です。すべての構成はCIBに保存されます。STONITHリソースはクラス`stonith`のリソースです(17.1項「サポートされるリソースエージェントクラス」 (201 ページ)を参照)。STONITHリソースはSTONITHプラグインのCIBでの表現です。フェンシング操作の他、STONITHリソースはその他のリソースと同様、開始、停止、監視できます。STONITHリソースの開始と停止とは、この場合STONITHの有効化と無効化を意味します。開始と停止は管理上の操作であるため、フェンシングデバイス自体での操作にはなりません。ただし、監視はデバイス状態に反映されます。

STONITHリソースはその他のリソースと同様にして構成できます。リソースの構成については、4.3項 「STONITHリソースの作成」 (38 ページ)または5.4項 「STONITHリソースの作成」 (67 ページ)を参照してください。

パラメータ(属性)のリストは、それぞれのSTONITHの種類に依存します。特定のデバイスのパラメータ一覧を表示するには、stonithコマンドを実行します。

```
stonith -t stonith-device-type -n
```

たとえば、ibmhmcデバイスタイプのパラメータを表示するには、次のように入力します。

```
stonith -t ibmhmc -n
```

デバイスの簡易ヘルプテキストを表示するには、-hオプションを使用します。

```
stonith -t stonith-device-type -h
```

8.3.1 STONITHリソースの構成例

以降では、crmコマンドラインツールの構文で作成された構成例を紹介します。これを適用するには、サンプルをテキストファイルに格納して(sample.txtなど)、実行します。

```
crm < sample.txt
```

crmコマンドラインツールでのリソースの構成については、第5章 クラスタリソースのコマンドラインからの設定 (61 ページ)を参照してください。

警告: テスティングの構成

次の例の一部は、説明およびテストのみを目的としています。テスティングの構成例を実際のクラスタシナリオで使用しないでください。

例 8.1 *テスティングの構成*

```
configure
primitive st-null stonith:null \
params hostlist="node1 node2"
clone fencing st-null
commit
```

*例 **8.2*** *テスティングの構成*

別の構成:

```
configure
 primitive st-node1 stonith:null \
 params hostlist="node1"
 primitive st-node2 stonith:null \
 params hostlist="node2"
 location l-st-node1 st-node1 -inf: node1
 location l-st-node2 st-node2 -inf: node2
 commit
```

この構成例は、クラスタソフトウェアに関してはまったく問題ありません。実際の構成との違いは、フェンシング操作が行われないことだけです。

*例 **8.3*** *テスティングの構成*

より現実的な例として、次のexternal/ssh構成を挙げます。これもテスト目的のみです。

```
configure
 primitive st-ssh stonith:external/ssh \
 params hostlist="node1 node2"
 clone fencing st-ssh
 commit
```

これもノードをリセットできます。この構成は、null STONITHデバイスを利用する最初の例と非常によく似ています。この例では、クローンが使用されています。これはCRM/Pacemakerの機能です。クローンは基本的にショートカットで、n個の同一リソースに別の名前を付けて定義しなくても、1つのクローンされたリソースで足ります。クローンの最もよく使われる方法は、STONITHデバイスがすべてのノードからアクセスできる場合、STONITHリソースとともに使用することです。

*例 **8.4**　IBM RSAライトアウトデバイスの構成*

実際のデバイス構成とはそれほど違いはありませんが、一部のデバイスにはより多くの属性が必要となります。IBM RSAライトアウトデバイスは、次のようにして構成できます。

```
configure
primitive st-ibmrsa-1 stonith:external/ibmrsa-telnet \
params nodename=node1 ipaddr=192.168.0.101 \
userid=USERID passwd=PASSW0RD
primitive st-ibmrsa-2 stonith:external/ibmrsa-telnet \
params nodename=node2 ipaddr=192.168.0.102 \
userid=USERID passwd=PASSW0RD
location l-st-node1 st-ibmrsa-1 -inf: node1
location l-st-node2 st-ibmrsa-2 -inf: node2
commit
```

この例では、STONITH操作は一定の確率で失敗するため、location制約が使用されています。したがって、実行側でもあるノード上のSTONITH操作は信頼できません。ノードがリセットされていない場合、フェンシング操作結果について通知を送信できません。これを実行する方法は、操作が成功すると仮定して事前に通知を送信するほかありません。しかし操作が失敗すると、問題が発生します。このため、stonithdは通例、ホストの終了を拒否します。

*例 **8.5** UPSフェンシングデバイスの構成*

UPSタイプのフェンシングデバイスの構成は、上記の例と同様で、詳細は読者への演習とします。UPSデバイスはすべてフェンシングに同一のメカニズムを採用していますが、デバイス自身へのアクセス方法が異なります。旧式のUPSデバイス(プロフェッショナル向けとされていたもの)は通常、シリアルポートが1つだけで、特殊なシリアルケーブルを使用して1200ボーで接続されています。新型の多くにはまだシリアルポートがありますが、USBインタフェースまたはEthernetインタフェースも備えています。使用できる接続の種類は、プラグインが何をサポートしているかによります。

たとえば、`apcmaster`を`apcsmart`デバイスと、`stonith -t` *`stonith-device-type`* `-n`コマンドを使用して比較します。

```
stonith -t apcmaster -h
```

次の情報が返されます。

```
STONITH Device: apcmaster - APC MasterSwitch (via telnet)
NOTE: The APC MasterSwitch accepts only one (telnet)
connection/session a time. When one session is active,
subsequent attempts to connect to the MasterSwitch will fail.
For more information see http://www.apc.com/
List of valid parameter names for apcmaster STONITH device:
ipaddr
login
 password
```

今度は次のコマンドを使用します。

```
stonith -t apcsmart -h
```

次の結果が得られます。

```
STONITH Device: apcsmart - APC Smart UPS
(via serial port - NOT USB!).
Works with higher-end APC UPSes, like
Back-UPS Pro, Smart-UPS, Matrix-UPS, etc.
(Smart-UPS may have to be >= Smart-UPS 700?).
See http://www.networkupstools.org/protocols/apcsmart.html
for protocol compatibility details.
For more information see http://www.apc.com/
List of valid parameter names for apcsmart STONITH device:
ttydev
hostlist
```

最初のプラグインは、ネットワークポートとtelnetプロトコルを持つAPC UPSをサポートします。2番目のプラグインはAPC SMARTプロトコルをシリアル

回線で使用します。これはその他多数のAPC UPS製品ラインでサポートされているものです。

8.3.2 制約とクローン

8.3.1項 「STONITHリソースの構成例」 (87 ページ)で、STONITHリソースを制約、クローン、またはその両方を使用して構成する方法をいくつか説明しました。構成にどちらの構成を使用するかは、いくつかの要因(フェンシングデバイスの性質、デバイスで管理されるホスト数、クラスタノード数)によって決まり、また個人の好みにも左右されます。

まとめると、クローンを構成で安心して使用でき、構成が縮小される場合は、クローンされたSTONITHリソースを使用します。

8.4 フェンシングデバイスの監視

その他のリソースと同様、STONITHクラスエージェントは状態のチェックに使用される監視操作もサポートします。

注意: STONITHリソースの監視

STONITHリソースの監視を強く推奨します。定期的に、間隔を置いて監視します。

フェンシングデバイスはHAクラスタの欠かせない要素ですが、使用する必要が少ないようが好都合です。電源管理装置は通信側では脆弱であることが知られています。一部のデバイスは、回線上のブロードキャストトラフィックが多すぎると処理を放棄してしまいます。1分間に10本程度の接続しか処理できないものもあります。2つのクライアントが同時に接続しようとすると、混乱したり性能が低下したりするものもあります。大半は、同時に複数のセッションを処理できません。

したがって、フェンシングシステムは数時間おきに確認するだけで多くの場合は十分です。この数時間のうちにフェンシング操作が必要になり、電源スイッチで障害が発生する確率は通常低いものです。

監視操作の構成方法の詳細は、GUIアプローチについてはメタ属性とインスタンス属性の追加または変更 (36 ページ)、コマンドラインアプローチについては5.8項 「リソース監視の設定」 (71 ページ)を参照してください。

8.5 特殊なフェンシングデバイス

実際のデバイスを処理するプラグインとは違い、STONITHプラグインは多少の違いがあり、特別な注意が必要です。

`external/kdumpcheck`
: カーネルコアダンプの取得が重要な場合があります。このプラグインは、ダンプが進行中かどうかを確認するために使用されます。この場合、ノードがフェンスされた場合と同様、trueを返します。これは実際、その時点でリソースを実行できないときにtrueになります。`kdumpcheck`は通常、別の実際のフェンシングデバイスと一緒に使用されます。詳細は`/usr/share/doc/packages/heartbeat/stonith/README_kdumpcheck.txt`を参照してください。

`external/sbd`
: これは自己フェンシングデバイスです。共有ディスクに挿入されることがある、いわゆる「ポイズンピル」に反応します。共有ストレージの接続が失われた場合、ノードを停止させます。詳しくは`http://www.linux-ha.org/SBD_Fencing`を参照してください。

`meatware`
: `meatware`ではユーザが操作を支援する必要があります。起動すると、`meatware`はードのコンソールに表示されるCRIT重大度メッセージを記録します。オペレータはノードがダウンしていることを確認して、`meatclient(8)`コマンドを発行する必要があります。これにより、`meatware`にノードがダウンしていると思われることをクラスタに通知できることを認識させます。詳細は`/usr/share/doc/packages/heartbeat/stonith/README.meatware`を参照してください。

`null`
: これはさまざまなテスティングデバイスで使用される仮想デバイスです。常にノードを停止したように動作しますが、何もしません。処理内容を理解している場合を除き、使用しないでください。

`suicide`
: これはソフトウェアのみのデバイスで、`reboot`コマンドを使用して実行しているノードを再起動できます。これにはノードのオペレーティングシステムによる操作が必要で、特定の状況では失敗することがあります。このため、できる限りこのデバイスは使用しないでください(ただし、1ノードのクラスタでは使用できます)。

`suicide`と`null`は、「自分のホストを停止させない」というルールへの唯一の例外です。

8.6 詳細情報

`/usr/share/doc/packages/heartbeat/stonith/`
: インストールしたシステムで、このディレクトリには多数のSTONITHプラグインおよびデバイスのREADMEファイルが格納されています。

`http://linux-ha.org/STONITH`
: The High Availability Linuxプロジェクトのホームページの STONITHについての情報。

`http://linux-ha.org/fencing`
: The High Availability Linuxプロジェクトのホームページのフェンシングについての情報。

`http://linux-ha.org/ConfiguringStonithPlugins`
: The High Availability Linuxプロジェクトのホームページの STONITHプラグインについての情報。

`http://linux-ha.org/CIB/Idioms`
: The High Availability Linuxプロジェクトのホームページの STONITHについての情報。

`http://clusterlabs.org/wiki/Documentation`, *Configuration 1.0 Explained*
: Pacemakerの構成に使用されるコンセプトの説明。包括的で非常に詳細な参照用情報です。

http://techthoughts.typepad.com/managing_computers/2007/10/split-brain-quo.html
HAクラスタでのスプリットブレイン、クォーラム、フェンシングのコンセプトを説明する記事。

Linux Virtual Serverによる負荷分散

9

LVS(Linux Virtual Server)は、複数のサーバにネットワーク接続を振り分けてワークロードを共有させる基本フレームワークの提供を目的としています。Linux Virtual Serverは、1つ以上のロードバランサとサービス実行用の数台の実際のサーバから成るサーバクラスタですが、外部のクライアントには1つの高速な大型サーバのように見えます。この単一サーバのように見えるサーバは、*仮想サーバ*と呼ばれます。Linux Virtual Serverは、高度な負荷分散ソリューションであり、高度にスケーラブルで可用性の高いネットワークサービス(Web、キャッシュ、メール、FTP、メディア、VoIPなど)の構築に使用できます。

実際のサーバとロードバランサは、高速LANまたは地理的に分散されたWANのいずれでも、相互に接続できます。ロードバランサは、様々なサーバに要求をディスパッチし、クラスタのパラレルサービスを1つのIPアドレスからの仮想サービスであるかのように実行させることができます。要求のディスパッチでは、IP負荷分散技術またはアプリケーションレベル負荷分散技術を使用できます。クラスタ内のノードのトランスペアレントな追加または削除によって、システムのスケーラビリティが達成されます。ノードまたはデーモンの障害の検出とシステムの適宜な再設定によって、高度な可用性が提供されます。

9.1 概念の概要

LVSは、2つの主要コンポーネントで構成されます。

カーネルコード: ip_vs (またはIPVS)

IPVSコードでパッチされたLinuxカーネルを実行するノードは、ディレクタと呼ばれます。ディレクタで実行されるIPVSコードは、LVSの必須機能です。

クライアントはディレクタに接続し、ディレクタは実際のサーバにパケットを転送します。ディレクタは、レイヤ4ルータとして、LVS用に修正されたルーティングルールのセット(たとえば、接続はディレクタで開始/終了しない、ディレクタは受信確認を送らないなど)を保持します。ディレクタと実際のサーバは、クライアントには1つのマシンであるかのように見える仮想サーバを構成します。

様々な転送方法があり、ディレクタによってクライアントから実際のサーバへパケットが送信される方法を決定します。クライアントから要求された新しい接続に使用する実際のサーバの決定には、各種のアルゴリズムが使用されます。これらのアルゴリズムは、モジュールとして使用できるので、特定のニーズに適合させることができます。ディレクタは、クライアントから接続要求を受信すると、スケジュールに基づいて実際のサーバをクライアントに割り当てます。スケジューラは、IPVSカーネルコードの一部として、次の新しい接続を取得する実際のサーバを決定します。

ユーザスペースコントローラ: ipvsadm

ipvsadmは、`ipvsadm`パッケージで提供されるユーザインタフェースであり、Linux Virtual Serverの管理に使用できます。たとえば、処理対象のサービスにルールを設定したり、フェールオーバーを処理したり、スケジューラタイプを設定できます。

次の設定には、ipvsadmをコマンドライン(またはrcファイル)から使用します。

- ディレクタが振り分けるサービスまたはサーバ(たとえば、HTTPは実際のサーバすべてにリダイレクトし、FTPは1つの実際のサーバにだけリダイレクトするなど)。
- 実際のサーバに与える重み付け(あるサーバが他のサーバより高速な場合に有用)
- スケジューリングアルゴリズム

ipvsadmは、次のタスクにも使用できます。

- サービスの追加
- サービスのシャットダウン
- サービスの削除

9.2 High Availability

可用性の高いLinux Virtual Serverクラススタを構築するには、このソフトウェアの内蔵機能を使用できます。一般に、ロードバランサでサービスモニタデーモンが実行され、定期的にサーバヘルスをチェックします。特定時間内に、サーバからサービスアクセス要求またはICMP ECHO_REQUESTに対する応答がない場合、サービスモニタは、サーバがダウンしたとみなし、ロードバランサにある使用可能なサーバのリストから、そのサーバを削除します。したがって、この機能しないサーバには新しい要求が送信されなくなります。サービスモニタは、ダウンしていたサーバが回復し、再度機能していることを検出すると、そのサーバを使用可能サーバリストに戻します。したがって、ロードバランサは、自動的に、サービスデーモンまたはサーバの障害をマスクできます。

さらに、管理者は、システムサービス全体を停止することなく、システムツールを使用して、システムスループットの増大のために新しいサーバを追加したり、システム保守のためにサーバを削除したりできます。

ロードバランサをシステム全体のシングルポイント障害にしないためには、ロードバランサのバックアップ(場合によっては複数のバックアップ)をセットアップする必要があります。そこで、2つのハートビートデーモンがプライマリとバックアップで実行されます。それらは、定期的に、シリアル回線および/またはネットワークインタフェースを介して、「I'm alive」メッセージをハートビートとして交換します。バックアップのハートビートデーモンは、特定時間内にプライマリからの ハートビートメッセージを聞くことができないと、負荷分散サービスを提供するため、仮想ＩＰアドレスを引き継ぎます。

障害の発生したロードバランサが回復した場合は、その結果として、2つの可能性があります。つまり、障害から回復したロードバランサが自動的にバックアップロードバランサになるか、アクティブなロードバランサがＶＩＰアドレスを解放し、回復したロードバランサがそのＶＩＰアドレスを引き継いで再びプライマリロードバランサになるかです。プライマリロードバランサは、接続の状態を保持しています。つまり、プライマリは、接続の転送先サー

バを知っています。その接続情報なしで、バックアップのロードバランサが処理を引き継ぐと、クライアントは、サービスにアクセスするため、要求を再度送信しなければなりません。クライアントアプリケーションに対してロードバランサのフェールオーバーをトランスペアレントにするため、IPVSでは接続の同期を行います。つまり、プライマリのIPVSロードバランサが、UDPマルチキャストを介して、バックアップのロードバランサと接続情報を同期します。プライマリロードバランサに障害発生後、バックアップロードバランサが処理を引き継いだ場合、バックアップロードバランサには大半の接続の状態が保持されています。したがって、ほとんどすべての接続が、バックアップロードバランサを介して、引き続きサービスにアクセスできます。

9.3 詳細情報

Linux Virtual Serverの詳細については、プロジェクトのホームページ(`http://www.linuxvirtualserver.org/`)を参照してください。

10 ネットワークデバイスボンディング

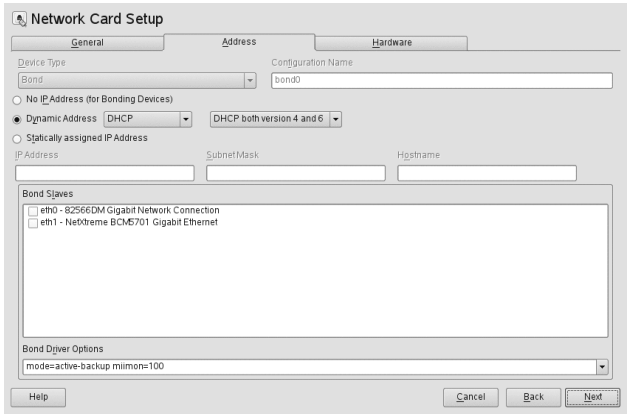
多くのシステムで、通常のEthernetデバイスの標準データセキュリティ/可用性の要件を超えるネットワーク接続の実装が望ましいことがあります。その場合、数台のEthernetデバイスを集めて1つのボンディングデバイスを構成できます。

ボンディングデバイスの構成には、ボンディングモジュールオプションを使用します。ボンディングデバイスの動作は、ボンディングデバイスのモードによって決定されます。デフォルトの動作は、`mode=active-backup`であり、アクティブなスレーブに障害が発生すると、別のスレーブデバイスがアクティブになります。

OpenAISの使用時は、クラスタソフトウェアでボンディングデバイスが管理されることはありません。したがって、ボンディングデバイスにアクセスする可能性のあるクラスタノード ごとに、ボンディングデバイスを設定する必要があります。

ボンディングデバイスを設定するには、次の手順に従います。

1. *YaST* > *ネットワークデバイス* > *ネットワーク設定*の順に選択します。
2. *追加*を使用し、*デバイスの型*を*ボンド*に変更します。*次へ*で続行します。



3 IPアドレスをボンディングデバイスに割り当てる方法を選択します。3つの方法から選択できます。

- IPアドレスなし
- 可変IPアドレス(DHCPまたは Zeroconf)
- 固定IPアドレス

ご使用の環境に適合する方法を使用します。OpenAISが仮想IPアドレスを管理する場合は、*固定IPアドレス*を選択し、インタフェースに基本IPアドレスを割り当てます。

4 *ボンドスレーブ*の該当するオプションのチェックボックスをオンにして、ボンドに含めるEthernetデバイスを選択します。

5 *ボンドドライバオプション*を編集します。次のモードを使用できます。

balance-rr
: 負荷分散と耐障害性を提供します。

active-backup
: 耐障害性を提供します。

balance-xor
: 負荷分散と耐障害性を提供します。

ブロードキャスト
耐障害性を提供します。

802.3ad
接続されるスイッチでサポートされる場合は、ダイナミックリンク集合を提供します。

balance-tlb
発信トラフィックの負荷分散を提供します。

balance-alb
使用中にハードウェアアドレスの変更が可能なネットワークデバイスを使用する場合は、着信トラフィックと発信トラフィックの負荷分散を提供します。

6 パラメータ`miimon=100`が*ボンドドライバオプション*に追加されていることを確認します。このパラメータがないと、データの整合性が定期的にチェックされません。

7 *次へ*をクリックし、*OK*でYaSTを終了してデバイスを作成します。

すべてのモードと他の多数のオプションの詳細は、「*Linux Ethernet Bonding Driver HOWTO*」に記載されています。このドキュメントは、`kernel-source`パッケージをインストールすれば、`/usr/src/linux/Documentation/networking/bonding.txt`で読むことができます。

SUSE Linux Enterprise 11へのクラスタの更新

11

SUSE® Linux Enterprise Server 10 SP2をベースとする既存クラスタがある場合は、そのクラスタをSUSE® Linux Enterprise Server 11上でHigh Availability Extensionを使用するように更新できます。マイグレーションのためには、すべてのクラスタノードをオフラインにして、クラスタを全体としてマイグレートする必要があります。SUSE Linux Enterprise Server 10クラスタとSUSE Linux Enterprise Server 11クラスタの混在はサポートされていません。

便宜のため、SUSE® Linux Enterprise High Availability Extensionには、`hb2openais.sh`スクリプトが含まれており、このスクリプトを使用すると、HeartbeatからOpenAISクラススタスタックへの移動時にデータを変換できます。スクリプトは、`/etc/ha.d/ha.cf`に保存されている環境設定を解析し、OpenAISクラスタスタック用の新しい環境設定ファイルを生成します。さらに、CIBを調整してOpenAIS表記規則と一致させ、OCFS2ファイルシステムを変換し、EVMSをcLVMで置き換えます。

クラスタをSUSE Linux Enterprise Server 10 SP2からSUSE Linux Enterprise Server 11へ正常にマイグレートするには、次の手順を実行する必要があります。

1. SUSE Linux Enterprise Server 10 SP2クラスタの準備 (104 ページ)
2. SUSE Linux Enterprise 11への更新 (105 ページ)
3. 変換のテスト (106 ページ)
4. データの変換 (106 ページ)

変換が正常に完了したら、更新したクラスタを再度オンラインにすることができます。

注意: 更新を元に戻すには

SUSE Linux Enterprise Server 11への更新後にSUSE Linux Enterprise Server 10へ戻すプロセスは、*サポートされていません*。

11.1 準備とバックアップ

クラスタを最新製品バージョンへ更新し、適宜、データを変換したら、現在のクラスタを準備する必要があります。

手順 11.1 *SUSE Linux Enterprise Server 10 SP2クラスタの準備*

1. クラスタにログインします。
2. Heartbeat環境設定ファイル`/etc/ha.d/ha.cf`をレビューし、すべての通信メディアがマルチキャストをサポートしているかどうかチェックします。
3. 次のファイルがすべてのノードで等しいことを確認します: `/etc/ha.d/ha.cf`および`/var/lib/heartbeat/crm/cib.xml`
4. 各ノードで`rcheartbeat stop`を実行して、すべてのノードをオフラインにします。
5. 最新バージョンへの更新前の一般システムバックアップ(推奨)に加え、次のファイルのバックアップを実行します。これらのファイルは、SUSE Linux Enterprise Server 11への更新後に変換スクリプトを実行する際に必要となります。
   - `/var/lib/heartbeat/crm/cib.xml`
   - `/var/lib/heartbeat/hostcache`
   - `/etc/ha.d/ha.cf`
   - `/etc/logd.cf`

11.2 更新/インストール

クラスタを準備し、ファイルをバックアップしたら、クライアントノードを最新製品バージョンへ更新できます。更新を実行する代わりに、クラスタノードでSUSE Linux Enterprise 11の新規インストールを実行することもできます。

手順 11.2 *SUSE Linux Enterprise 11への更新*

1. すべてのクラスタノードで、SUSE Linux Enterprise Server 10 SP2からSUSE Linux Enterprise Server 11への更新を実行します。ご使用製品の更新方法については、『SUSE Linux Enterprise Server 11 *Deployment Guide*』の章「*Updating SUSE Linux Enterprise*」を参照してください。

   または、すべてのクラスタノードで、SUSE Linux Enterprise Server 11を新規でインストールすることもできます。

2. すべてクラスタノードで、SUSE Linux Enterprise Serverのアドオンとして、SUSE Linux Enterprise High Availability Extension 11をインストールします。詳細については、3.1項 「High Availability Extensionのインストール」 (23 ページ)を参照してください。

11.3 データの変換

SUSE Linux Enterprise Server 11およびHigh Availability Extensionをインストールしたら、データ変換を開始できます。High Availability Extensionとともに出荷される変換スクリプトは、注意深く設定されていますが、完全な自動モードですべての設定を行うことはできません。このスクリプトでは、実効する変更について管理者に警告し、対話と管理者側での決定を必要とします。管理者は、クラスタの詳細を知っている必要があり、変更の妥当性を確認する責任があります。変換スクリプトは、`/usr/lib/heartbeat`(64ビットマシンの場合は、`/usr/lib64/heartbeat`)に格納されています。

注意: テストランの実行

変換プロセスをよく知るために、まず、変換をテストすること(変更なしで)を強くお勧めします。同じテストディレクトリを使用すると、ファイルを1回コピーするだけで、テストランを繰り返すことができます。

手順 11.3 *変換のテスト*

1. ノードの1つで、テストディレクトリを作成し、そのテストディレクトリにバックアップファイルをコピーします。

```
$ mkdir /tmp/hb2openais-testdir
$ cp /etc/ha.d/ha.cf /tmp/hb2openais-testdir
$ cp /var/lib/heartbeat/hostcache /tmp/hb2openais-testdir
$ cp /etc/logd.cf /tmp/hb2openais-testdir
$ sudo cp /var/lib/heartbeat/crm/cib.xml /tmp/hb2openais-testdir
```

2. 次のコマンドで、テストランを開始します。

```
$ /usr/lib/heartbeat/hb2openais.sh -T /tmp/hb2openais-testdir -U
```

   64ビットシステムを使用する場合は、次のコマンドを使用します。

```
$ /usr/lib64/heartbeat/hb2openais.sh -T /tmp/hb2openais-testdir -U
```

3. 結果として生成された`openais.conf`ファイルと`cib-out.xml`ファイルを読んで検証します。

```
$ cd  /tmp/hb2openais-testdir
$ less openais.conf
$ crm_verify -V -x cib-out.xml
```

変換段階の詳細については、インストールしたHigh Availability Extensionの`/usr/share/doc/packages/pacemaker/README.hb2openais`を参照してください。

手順 11.4 *データの変換*

テストランを実行し、出力をチェックしたら、データ変換を開始できます。変換は、*1*つのノードで実行するだけで済みます。メインクラスタ構成(CIB)が自動的にその他のノードにレプリケートされます。レプリケートの必要がある他のすべてのファイルは、変換スクリプトによって自動的にコピーされます。

1. 変換スクリプトで他のクラスタノードにファイルを正常にコピーするため、`root`に許可されたアクセスでsshdがすべてのノードで実行されることを確認します。

2. High Availability Extensionは、デフォルトのOpenAIS環境設定ファイルとともに出荷されています。以降の手順で、デフォルトの環境設定を上書

きしたくない場合は、/etc/ais/openais.conf環境設定ファイルのコピーを作成します。

3 変換スクリプトをrootとして起動します。sudoを使用する場合は、-uオプションで特権ユーザを指定します。

```
$ /usr/lib/heartbeat/hb2openais.sh -u root
```

/etc/ha.d/ha.cfに保存されている環境設定に基づいて、スクリプトは、OpenAISクラスタスタック用の新しい環境設定ファイル/etc/ais/openais.confを生成します。スクリプトは、CIBの設定を分析し、HeartbeatからOpenAISへの変更に伴いクラスタ設定の変更が必要かどうか通知してきます。すべてのファイル処理は、変換が実行されるノードで行われ、他のノードにレプリケートされます。

4 画面の指示に従います。

変換が正常に完了したら、新しいクラスタスタックを「3.3項「クラスタをオンラインにする」 (27 ページ)」の説明に従って起動します。

アップグレードプロセスの後で、SUSE Linux Enterprise Server 10に戻すことはできません。

11.4 詳細情報

変換スクリプトおよび変換の各段階の詳細については、インストールしたHigh Availability Extensionの/usr/share/doc/packages/pacemaker/README.hb2openaisを参照してください。

パート III. ストレージおよびデータレプリケーション

12 Oracle Cluster File System 2

Oracle Cluster File System 2は、Linux 2.6以降のカーネルと完全に統合された、汎用のジャーナルファイルシステムです。Oracle Cluster File System 2を利用すれば、アプリケーションバイナリファイル、データファイル、およびデータベースを、共有ストレージ中のデバイスに保管することができます。このファイルシステムには、クラスタ中のすべてのノードが同時に読み書きすることができます。ユーザスペース管理デーモンはクローンリソースによって管理され、特にOpenAISおよびDLMにおいて、HAスタックとの統合を実現します。

12.1 特長と利点

SUSE Linux Enterprise Server 10以降では、次のストレージソリューションでOCFS2を使用することができます。

- 一般アプリケーションと負荷
- クラスタ中のXENイメージ

  サーバ間でXEN仮想マシンの素早く簡単な移植性を活用するために、XEN仮想マシンおよび仮想サーバを、クラスタサーバによりマウントされたOracle Cluster File System 2ボリュームに保管することができます。

- LAMP(Linux、Apache、MySQL、およびPHP | PERL | Python)スタック

また、OpenAISと完全に統合されています。

Oracle Cluster File System 2は、高性能なパラレルクラスタファイルシステムで、次の機能をサポートしています。

- クラスタ中のすべてのノードが、アプリケーションのファイルを利用することができます。ユーザは、クラスタ中のOracle Cluster File System 2ボリュームに1回インストールするだけで構いません。
- 標準のファイルシステムインタフェースを通じて、すべてのノードが並行してストレージに読み書きできるため、クラスタにまたがって稼働するアプリケーションの管理が容易になります。
- ファイルアクセスは、分散ロックマネージャ(DLM:Distributed Lock Manager)により、管理、調整されます。

  ほとんどの場合、DLMによる制御は適切に機能しますが、DLMとファイルアクセスを競合するようなアプリケーションなど、アプリケーションの設計によっては、スケーラビリティが制限される可能性もあります。
- すべてのバックエンドストレージで、ストレージのバックアップ機能を利用することができます。共有アプリケーションファイルのイメージを簡単に作成することができるため、災害発生時でも素早くデータを復元することができます。

Oracle Cluster File System 2には、次の機能も用意されています。

- メタデータのキャッシュ処理.
- メタデータのジャーナル処理.
- ノード間にまたがるファイルデータの整合性.
- 最大16TBまでのボリュームで、最高4KBまでの複数ブロックサイズをサポート(各ボリュームで異なるブロックサイズを使用可能)
- 16台までのクラスタノードをサポート.
- データベースのパフォーマンスを向上する非同期、直接I/Oのサポート.

12.2 管理ユーティリティとコマンド

OCFS2ユーティリティを次の表に示します。これらのコマンドの指定形式については、マニュアルページを参照してください。

表 12.1 OCFS2ユーティリティ

OCFS2ユーティリティ	説明
debugfs.ocfs2	デバッグの目的で、Oracle Cluster File System 2のファイルシステムの状態を調査します。
fsck.ocfs2	ファイルシステムにエラーがないかをチェックし、必要に応じてエラーを修復します。
mkfs.ocfs2	デバイス上にOCFS2ファイルシステムを作成します。通常は、共有物理/論理ディスク上のパーティションに作成します。
mounted.ocfs2	クラスタシステム上のすべてのOCFS2ボリュームを検出、表示します。OCFS2デバイスをマウントしているシステム上のすべてのノードを検出、表示するか、またはすべてのOCFS2デバイスを表示します。
tunefs.ocfs2	ボリュームラベル、ノードスロット数、すべてのノードスロットのジャーナルサイズ、およびボリュームサイズなど、OCFS2ファイルのシステムパラメータを変更します。

12.3 OCFS2のパッケージ

SUSE Linux Enterprise Server 11 HAE以降では、OCFS2カーネルモジュール(`ocfs2`)は自動的にインストールされます。OCFS2を使用するには、YaST(またはコマンドライン)を使って`ocfs2-tools`と一致する`ocfs2-kmp-*`パッケージを、クラスタ中の各ノードにインストールします。

1 `root`、または同等のユーザとしてログインし、YaSTコントロールセンタを起動します。

2 *ソフトウェア*> *アドオン製品*を選択します。

3 *追加*を選択して、SUSE Linux Enterprise High Availability Extensionを有効にします。

4 *ソフトウェアマネージャの実行*を選択して、*フィルタ*>*パターン*を選択します。

5 インストールする*高可用性*パターンを選択します。

6 *受諾*をクリックして、画面上の指示に従います。

12.4 OCFS2ボリュームの作成

ここでは、OCFS2ボリュームの作成方法、およびOCFS2を使用するためのシステムの設定方法について説明していきます。

12.4.1 前提条件

まず、次の作業を行ってください。

- OCFS2ボリュームに使用する予定のブロックデバイスを準備します。デバイスには、空き領域を残してください。

  アプリケーションファイルとデータファイルは、異なるOCFS2ボリュームに保管することをお勧めします。アプリケーション用ボリュームとデータ用ボリュームでマウントの要件が異なる場合は、必ずそうしてください。

- `ocfs2-tools`パッケージがインストールされていることを確認します。これらのパッケージがインストールされていない場合は、YaSTまたはコマンドラインを使ってインストールします。YaSTを使ったインストール方法については、12.3項 「OCFS2のパッケージ」 (113 ページ)を参照してください。

12.4.2 OCFS2サービスの設定

OCFS2ボリュームを作成する前に、OCFS2サービスを設定する必要があります。

クラスタ中の1台のノードに対して、次の作業を行います。

1. ターミナルウィンドウを開いて、`root`または同等のユーザとしてログインします。
2. 分散ロックマネージャ構成を追加します。
   - **2a** crmシェルを起動して、新しいスクラッチ構成を作成します。

     ```
     crm
     cib new stack-glue
     ```

   - **2b** DLMサービスを作成して、クラスタ内のすべてのマシン上で実行します。

     ```
     configure
     primitive dlm ocf:pacemaker:controld op monitor interval=120s
     clone dlm-clone dlm meta globally-unique=false interleave=true
     end
     ```

   - **2c** クラスタに行った変更を確認してからコミットします。

     ```
     cib diff
     configure verify
     ```

   - **2d** 構成をクラスタにアップロードして、シェルを終了します。

     ```
     cib commit stack-glue
     quit
     ```

3. O2CB構成を*crm*で追加します。
   - **3a** crmシェルを起動して、新しいスクラッチ構成を作成します。

```
crm
cib new oracle-glue
```

3b Pacemakerを構成してo2cbサービスをクラスタ内の各ノードで起動します。

```
configure
primitive o2cb ocf:ocfs2:o2cb op monitor interval=120s
clone o2cb-clone o2cb meta globally-unique=false interleave=true
```

3c Pacemakerがp2cbサービスのみをdlmサービスのコピーが既に実行されているノード上で起動することを確認してください。

```
colocation o2cb-with-dlm INFINITY: o2cb-clone dlm-clone
order start-o2cb-after-dlm mandatory: dlm-clone o2cb-clone
end
```

3d 構成をクラスタにアップロードして、シェルを終了します。

```
cib commit stack-glue
quit
```

12.4.3 OCFS2ボリュームの作成

OCFS2ファイルシステムを作成してクラスタに新しいノードを追加する作業は、クラスタ中の1台のノードに対してのみ行います。

1. ターミナルウィンドウを開いて、rootまたは同等のユーザとしてログインします。
2. クラスタがオンラインであることをcrm_monで確認します。
3. 次のいずれかの方法を使って、ボリュームを作成、フォーマットします。
   - mkfs.ocfs2ユーティリティを使用します。このコマンドの指定形式については、mkfs.ocfs2マニュアルページを参照してください。

最大16クラスタノードをサポートする新しいOCFS2ファイルシステムを/dev/sdb1上に作成するには、次のコマンドを使用します。

```
mkfs.ocfs2 -N 16 /dev/sdb1
```

推奨する設定については、次の表を参照してください。

OCFS2パラメータ	説明と推奨設定
Volume label	異なるノードへのマウント時に、正しく識別できるように、一意のわかりやすいボリューム名を指定します。 ラベルを変更するには、tunefs.ocfs2ユーティリティを使用します。
Cluster size	クラスタサイズは、ファイルに割り当てられる、データ保管領域の最小単位です。 4、8、16、32、64、128、256、512、および1024KBを指定することができます。ボリュームのフォーマット後にクラスタサイズを変更することはできません。 Oracleでは、データベースボリュームのクラスタサイズに128KB以上を指定することを推奨しています。また、Oracle Homeの場合は32KBまたは64KBを指定することも推奨しています。
Number of node slots	同時にボリュームをマウントできる最大ノード数を指定します。各ノードについて、OCFS2はジャーナルなどの個別のシステムファイルを作成します。ボリュームにアクセスするノードに、リトルエンディアン形式のノード(x86、x86-64、およびia64など)とビッグエンディアン形式のノード(ppc64やs390xなど)が混在しても構いません。 ノード固有のファイルは、ローカルファイルとして参照されます。ローカルファイルには、ノードスロット番号が付加されます。たとえば、「journal:0000」は、スロット番号0ががが割り当てられているノードに所属します。

OCFS2パラメータ	説明と推奨設定
	各ボリュームの作成時に、ボリュームに同時にマウントする予定のノード数に応じて、そのボリュームの最大ノードスロット数を指定します。必要に応じて、tunefs.ocfs2ユーティリティを使って、ノードスロット数を増やすことができます。ただし、ノードスロット数を減らすことはできません。
Block size	ファイルシステムがアドレス可能な領域の最小単位を指定します。ブロックサイズは、ボリュームの作成時に指定します。 512Byte (推奨されません)、1KB、2KB、または4KB (ほぼすべてのボリュームに最適)を選択することができます。ブロックサイズは、ボリュームのフォーマット後に変更することはできません。

12.5 OCFS2ボリュームのマウント

1 ターミナルウィンドウを開いて、rootまたは同等のユーザとしてログインします。

2 クラスタがオンラインであることをcrm_monで確認します。

3 次のいずれかの方法を使って、ボリュームをマウントします。

警告: 手動でマウントされたOCFS2デバイス

ocfs2ファイルシステムをテスト目的で手動でマウントした場合、OpenAISでそのシステムを使用する前に、ファイルシステムをアンマウントする必要があります。

- ocfs2consoleで、［Available Devices］リストからデバイスを選択して*Mount*をクリックします。次に、ディレクトリのマウントポ

イントと、必要に応じてマウントオプションを指定し、*OK*をクリックします。

- コマンドラインから、mountコマンドを使ってボリュームをマウントします。
- クラスタマネージャを使用してファイルシステムをマウントします。このタスクには、ocfリソースのFilesystemを使用できます。詳細については、クラスタマネージャによるファイルシステムのマウント (119 ページ)を参照してください。

マウントが正常に完了すると、ocfs2consoleのデバイスリストに、そのマウントポイントとデバイスが表示されます。

オプション	説明
datavolume	Oracleプロセスが、o_directフラグでファイルを開きます。
nointr	割り込みなし。I/Oが信号により割り込まれないようにします。

ファイルシステムリソースを構成するには、次の手順に従います。

手順 12.1 *クラスタマネージャによるファイルシステムのマウント*

1. crmシェルを起動して、新しいスクラッチ構成を作成します。

```
crm
cib new filesystem
```

2. Pacemakerを構成して、ファイルシステムをクラスタ内の各ノードにマウントします。

```
configure
primitive fs ocf:heartbeat:Filesystem \
   params device="/dev/sdb1" directory="/mnt/shared" fstype="ocfs2" \
   op monitor interval=120s
clone fs-clone fs meta interleave="true" ordered="true"
```

3 Pacemakerがo2cbリソースが既に実行されているノード上でo2cbクローンリソースのみを起動することを確認してください。

```
colocation fs-with-o2cb INFINITY: fs-clone o2cb-clone
order start-fs-after-o2cb mandatory: o2cb-clone fs-clone
end
```

4 構成をクラスタにアップロードして、シェルを終了します。

```
cib commit filesystem
quit
```

12.6 追加情報

OCFS2の使用の詳細については、 *OCFS2 project at Oracle [http://oss.oracle.com/projects/ocfs2/documentation/]*にある『OCFS2 User Guide [http://oss.oracle.com/projects/ocfs2/]』を参照してください。

13 クラスタLVM

クラスタ上の共有ストレージを管理する場合、ストレージサブシステムに行った変更を各ノードに伝える必要があります。Linux Volume Manager 2 (LVM2)はローカルストレージの管理に多用されており、クラスタ全体のボリュームグループのトランスペアレントな管理をサポートするために拡張されています。クラスタ化されたボリュームグループを、ローカルストレージと同じコマンドで管理できます。

cLVMを実装するには、Fire Channel、FCoE、SCSI、またはiSCSI SANやDRBDで提供されている共有ストレージセットアップが使用できなければなりません。

cLVMは内部でクラスタスタックのDistributed Lock Manager (DLM)コンポーネントを使用して、LVM2メタデータへのアクセスを調整します。DLMはフェンシングなどのスタック内の他のコンポーネントに統合されるので、共有ストレージの整合性は常に保護されます。

cLVMは共有データへのアクセスを自身では調整しません。調整するためには、OCFS2または他のクラスタ2対応アプリケーションをcLVMの管理対象ストレージ上に設定する必要があります。

13.1 cLVMの環境設定

クラスタ対応のボリュームグループをセットアップするには、次のタスクを正しく完了する必要があります。

1 LVM2のロックタイプをクラスタ対応になるように変更します。

ファイル/etc/lvm/lvm.confを編集し、次の行を探します。

```
locking_type = 1
```

ロックタイプを3に変更し、環境設定をディスクに書き込みます。この環境設定をすべてのノードにコピーします。

2 clvmdリソースをベースメーカーの環境設定でクローンとして保存し、DLMクローンリソースに依存させます。これはcrm環境設定シェルの典型的なsnippetです。

```
primitive dlm ocf:pacemaker:controld
primitive clvm ocf:lvm2:clvmd \
        params daemon_timeout="30"
clone dlm-clone dlm \
        meta target-role="Started" interleave="true" ordered="true"
clone clvm-clone clvm \
        meta target-role="Started" interleave="true" ordered="true"
colocation colo-clvm inf: clvm-clone dlm-clone
order order-clvm inf: dlm-clone clvm-clone
...
```

次に進む前に、クラスタ内でこれらのリソースが正しく開始したことを確認してください。crm_monまたはGUIを使用して、実行中のサービスを確認することができます。

3 次のコマンドでLVMの物理ボリュームを準備します。

```
pvcreate <physical volume path>
```

4 クラスタ対応ボリュームグループを作成します。

```
vgcreate --clustered y <volume group name> <physical volume path>
```

5 たとえば次のように論理ボリュームを適宜作成します。

```
lvcreate --name testlv -L 4G <volume group name>
```

6 ボリュームグループがクラスタ全体でアクティブ化されるようにするには、LVMリソースを次のように設定します。

```
primitive vg1 ocf:heartbeat:LVM \
```

```
        params volgrpname="<volume group name>"
clone vg1-clone vg1 \
        meta interleave="true" ordered="true"
colocation colo-vg1 inf: vg1-clone clvm-clone
order order-vg1 inf: clvm-clone vg1-clone
```

7 ボリュームグループを1ノードだけで排他的にアクティブ化したい場合は、次の例を使用します。この場合、クラスタ化されていないアプリケーション保護の追加対策として、cLVMはVG内のすべての論理ボリュームが複数ノードでアクティブ化されないように保護します。

```
primitive vg1 ocf:heartbeat:LVM \
        params volgrpname="<volume group name>" exclusive="yes"
colocation colo-vg1 inf: vg1 clvm-clone
order order-vg1 inf: clvm-clone vg1
```

8 VG内の論理ボリュームは、ファイルシステムのマウントまたはraw用として使用できるようになりました。論理ボリュームを使用しているサービスにコロケーションのための正しい依存性があることを確認し、VGをアクティブ化したら論理ボリュームの順序付けを行います。

このような設定手順を終了すると、LVM2の環境設定は他のスタンドアロンワークステーションと同様に行えます。

13.2 有効なLVM2デバイスの明示的な設定

複数のデバイスが同じ物理ボリュームの署名を共有していると思われる場合(マルチパスデバイスやdrbdなどのように)、LVM2がPVを走査するデバイスを明示的に設定しておくことをお勧めします。

たとえばコマンドvgcreateがミラーブロックデバイスの代わりに物理デバイスを使用すると、DRBDは混乱してしまい、DRBDのスプリットブレイン状態が発生する場合があります。

LVM2用の単一のデバイスを非アクティブ化するには、次の手順に従います。

1 ファイル/etc/lvm/lvm.confを編集し、filterから始まる行を検索します。

2 そこに記載されているパターンは正規表現として処理されます。冒頭の「a」は走査にデバイスパターンを受け入れることを、冒頭の「r」はそのデバイスパターンのデバイスを拒否することを意味します。

3 /dev/sdb1という名前のデバイスを削除するには、次の表現をフィルタルールに追加します。

```
"r|^/dev/sdb1$|"
```

完全なフィルタ行は次のようになります。

```
filter = [ "r|^/dev/sdb1$|", "r|/dev/.*/by-path/.*|",
"r|/dev/.*/by-id/.*|", "a/.*/" ]
```

DRBDとMPIOデバイスは受け入れ、その他のすべてのデバイスは拒否するフィルタ行は次のようになります。

```
filter = [ "a|/dev/drbd.*|", "a|/dev/.*/by-id/dm-uuid-mpath-.*|",
"r/.*/" ]
```

4 環境設定ファイルを書き込み、すべてのクラスタノードにコピーします。

13.3 詳細情報

http://www.clusterlabs.org/wiki/Help:Contentsに記載のペースメーカーのメーリングリストに詳しい情報が提供されています。

cLVMのFAQのオフィシャルサイトはhttp://sources.redhat.com/cluster/wiki/FAQ/CLVMです。

Distributed Replicated Block Device (DRBD)

14

DRBDを使用すると、IPネットワーク内の2つの異なるサイトに位置する2つのブロックデバイスのミラーを作成できます。OpenAISと共に使用すると、DRBDは分散高可用性Linuxクラスタをサポートします。

重要項目

ミラー間のデータトラフィックは暗号化されません。安全なデータ交換を実現するには、接続に関してVPN(仮想私設網)ソリューションを導入する必要があります。

プライマリデバイス上のデータがセカンダリデバイスに複製され、両方のデータのコピーが常に同一に保たれます。ocfs2などのクラスタを認識するファイルシステムを使用する場合、両方のノードをプライマリデバイスとして実行することもできます。

DRBDでは、デフォルトで、DRBDノード間の通信にTCPポート7788を使用します。ファイアウォールでこのポートの通信が許可されていることを確認してください。

まず、DRBDデバイスを設定してから、その上にファイルシステムを作成する必要があります。ユーザデータに関することはすべて、rawデバイス上ではなく、`/dev/drbd<n>`デバイスを介してだけ実行される必要があります。これは、DRBDが、メタデータ用にrawデバイスの最後の128MBを使用するからです。ファイルシステムは、rawデバイス上ではなく、必ず、`/dev/drbd<n>`デバイスにのみ作成するようにしてください。

たとえば、rawデバイスのサイズが1024MBの場合、DRBDデバイスは、896MBしかデータ用に使用できません。128MBは隠され、メタデータ用に予約されています。896MB～1024MBのスペースへのアクセスは、そのスペースがユーザデータ用でないので、すべて失敗します。

14.1 DRBDサービスのインストール

drbdに必要なパッケージをインストールするには、パートI.「インストールおよび管理」 (1 ページ)に説明するように、High Availability Extensionアドオン製品をネットワーククラスタの両方のSUSE Linux Enterprise Serverマシンにインストールします。High Availability Extensionをインストールすると、drbdプログラムファイルもインストールされます。

クラスタスタック全体は不要で、drbdを使用したいだけの場合は、High Availability Extensionアドオン製品を追加して、drbdのインストールに進むことができます。これにより、必要なカーネルモジュールもインストールされます。

14.2 DRBDサービスの設定

注意

次のプロシージャでは、サーバ名としてnode 1とnode 2、およびクラスタリソース名r0を使用します。node 1は、プライマリノード として設定します。必ず、手順を変更して、ご使用のノード名とファイルの名前を使用してください。

1. YaSTを起動して、*その他* > *drbd*で構成モジュールを選択します。
2. *起動設定*> ブートで、*有効*を選択してブート時に常にdrbdを起動させます。
3. 複数の複製リソースを構成する必要がある場合は、*グローバルな設定*を選択します。入力フィールドマイナーカウントで、コンピュータを再起動せずに構成するdrbdリソースの数を選択します。

4 リソースの実際の構成は、*ソース設定*で実行されます。*追加*をクリックして、新規リソースを作成します。次のパラメータを設定する必要があります。

リソース名	リソース名。`r0`と呼ばれます。
名前	それぞれのノードのホスト名。
アドレス:ポート	それぞれのノードのIPアドレスとポート番号。
デバイス	それぞれのノードで複製されたデータを保持するデバイス。このデバイスを使用して、ファイルシステムとマウント処理を作成します。
ディスク	両方のノード間で複製されるデバイス。
メタディスク	メタディスクは、値`internal`に設定されるか、またはインデックスで拡張された、drbdで必要なメタデータを保持する明示的なデバイスを指定します。 `internal`を使用する場合、複製されたデバイスの最後の128MBはメタデータの保存に使用されます。 複数のdrbdリソースに実際のデバイスを使用することもできます。たとえば、最初のリソースに対してメタディスクが`/dev/sda6[0]`の場合、`/dev/sda6[1]`を2番目のリソースに使用できます。ただし、このディスク上で各リソースについて少なくとも128MBのスペースが必要です。

これらのオプションはすべて、`/usr/share/doc/packages/drbd/drbd.conf`ファイルの例と`drbd.conf(5)`のマニュアルページで説明されています。

5 `/etc/drbd.conf`ファイルを、セカンダリサーバ(node 2)上の`/etc/drbd.conf`にコピーします。

```
scp /etc/drbd.conf <node 2>:/etc
```

6 各ノードで、次のように入力して、DRBDサービスを両方のシステム上で初期化し、起動します。

```
drbdadm create-md r0
rcdrbd start
```

7 node1上で次のコマンドの入力により、node1をプライマリノード として設定します。

```
drbdsetup /dev/drbd0 primary --overwrite-data-of-peer
```

8 各ノードで、次のコマンドを入力して、DRBDサービスのステータスをチェックします。

```
rcdrbd status
```

続行する前に、両方のノード のブロックデバイスが完全に同期されるまで待機します。rcdrbd statusコマンドを繰り返して、同期の進捗を追跡します。

9 両方のノードのブロックデバイスが完全に同期されたら、reiserfsなどのファイルシステムで、プライマリノード上のDRBDデバイスをフォーマットします。任意のLinuxファイルシステムを使用できます。例えば、次のように入力して、すべてのフォントについてアンチエイリアスを無効にします。

```
mkfs.reiserfs -f /dev/drbd0
```

重要項目

コマンドでは、常に、/dev/drbd<n>名を使用し、実際の/dev/diskデバイス名は使用しないでください。

14.3 DRBDサービスのテスト

インストールと設定のプロシージャが予期どおりの結果となった場合は、DRBD機能の基本的なテストを実行できます。このテストは、DRDBソフトウェアの機能を理解する上でも役立ちます。

1 node 1でのDRBDサービスのテスト

1a 端末コンソールを開いて、rootユーザまたは同等の権限でログインします。

1b 次のコマンドで、node 1にマウントポイント(/srv/r0mountなど)を作成します。

```
mkdir -p /srv/r0mount
```

1c 次のコマンドで、drbdデバイスをマウントします。

```
mount -o rw /dev/drbd0 /srv/r0mount
```

1d 次のコマンドで、プライマリノードからファイルを作成します。

```
touch /srv/r0mount/from_node1
```

2 node 2でのDRBDサービスのテスト

2a 端末コンソールを開いて、rootユーザまたは同等の権限でログインします。

2b node 1で、次のコマンドを入力して、node 1のディスクをディスマウントします。

```
umount /srv/r0mount
```

2c node 1で、次のコマンドを入力して、node 1のDRBDサービスをダウングレードします。

```
drbdadm secondary r0
```

2d node 2で、次のコマンドで、DRBDサービスをプライマリにプロモートします。

```
drbdadm primary r0
```

2e node 2で、次のコマンドを入力して、node 2がプライマリにあるかどうかをチェックします。

```
rcdrbd status
```

2f node 2で、次のコマンドを入力して、/srv/r0mountなどのマウントポイントを作成します。

```
mkdir /srv/r0mount
```

2g node 2で、次のコマンドを入力して、DRBDデバイスをマウントします。

```
mount -o rw /dev/drbd0 /srv/r0mount
```

2h 次のコマンドを入力して、node 1で作成したファイルを表示できることを確認します。

```
ls /srv/r0mount
```

/srv/r0mount/from_node1ファイルがリストされるはずです。

3 サービスが両方のノードで稼動していれば、DRBDの設定は完了です。

4 再度、node 1をプライマリとして設定します。

4a node 2で、次のコマンドを入力して、node 2のディスクをディスマウントします。

```
umount /srv/r0mount
```

4b node 2で、次のコマンドを入力して、node 2のDRBDサービスをダウングレードします。

```
drbdadm secondary r0
```

4c node 1で、次のコマンドを入力して、DRBDサービスをプライマリにプロモートします。

```
drbdadm primary r0
```

4d node 1で、次のコマンドを入力して、node 1がプライマリになっているかどうかチェックします。

```
rcdrbd status
```

5 サービスを自動的に起動させ、サーバに問題が発生した場合はフェールオーバーさせるためには、OpenAISでDRBDを高可用性サービスとして設定できます。OpenAIS for SUSE Linux Enterprise 11のインストールと構成の詳細は、パート II. 「設定および管理」 (29 ページ)を参照してください。

14.4 DRBDのトラブルシュート

drbdセットアップには、多数の異なるコンポーネントが使用され、別のソースから問題が発生することがあります。以降のセクションでは、いくつかの一般的な問題を取り上げ、問題を解決するためのヒントを説明します。

14.4.1 設定

初期のdrbdセットアップが予期どおりに機能しない場合は、おそらく、環境設定に問題があります。

環境設定の情報を取得するには:

1 端末コンソールを開き、rootユーザとしてログインします。

2 drbdadmに-dオプションを指定して、環境設定ファイルをテストします。<Enter>

```
drbdadm -d adjust r0
```

adjustオプションのドライランでは、drbdadmは、DRBDリソースの実際の設定をご使用のDRBD環境設定ファイルと比較しますが、コール

は実行しません。出力をレビューして、エラーのソースおよび原因を確認してください。

3 drbd.confファイルにエラーがある場合は、それらのエラーを修正してから続行します。

4 パーティションと設定が正しい場合は、drbdadmを-dオプションなしで、再度実行します。<Enter>

```
drbdadm adjust r0
```

このコマンドは、環境設定ファイルをDRBDリソースに適用します。

14.4.2 ホスト名

DRBDの場合、ホスト名の指定では大文字小文字を区別します。したがって、Node0は、node0とは異なるホストです。

複数のネットワークデバイスがあり、専用ネットワークデバイスを使用する場合、ホスト名は使用されたIPアドレスに解決されません。この場合、パラメータdisable-ip-verificationを使用して、DRBDでこれを無視します。

14.4.3 TCPポート7788

システムがピアに接続できない場合は、ローカルファイアウォールに問題のある可能性があります。DRBDは、デフォルトでは、TCPポート7788を使用して、もう一方のノードにアクセスします。このポートを両方のノードからアクセスできるかどうか確認してください。

14.4.4 DRBDデバイスが再起動後に破損した

DRBDサブシステムが実際のどのデバイスが最新データを保持しているか認識していない場合、スプリットブレイン受験に変更されます。この場合、それぞれのDRBDサブシステムがセカンダリとして機動され、互いに接続しません。この場合、次のメッセージが/var/log/messagesに書き込まれます。

```
Split-Brain detected, dropping connection!
```

この状況を解決するには、修正が破棄されるノードを1つ選択する必要があります。このノードにログインして、次のコマンドを実行します。

```
drbdadm secondary r0
drbdadm -- --discard-my-data connect r0
```

別のノードで、次のコマンドを実行します。

```
drbdadm connect r0
```

14.5 追加情報

DRBDについては、次のオープンソースリソースを利用できます。

- 配布パッケージには、次のDRBDのマニュアルページが含まれています。

  ```
  drbd(8)
  drbddisk(8)
  drbdsetup(8)
  drbdadm(8)
  drbd.conf(5)
  ```

- コメント付きのDRBD構成例が、`/usr/share/doc/packages/drbd/drbd.conf`にあります。

- プロジェクトホームページ`http://www.drbd.org`。

- `http://clusterlabs.org/wiki/DRBD_HowTo_1.0`(Linux Pacemaker Cluster Stack Projectによる)。

パート IV. トラブルシューティングと参照情報

15 トラブルシューティング

Heartbeatを使い始めたときは特に、理解が困難な奇妙な問題が発生することがあります。ただし、Heartbeat内部プロセスを詳細に調査するために使用できる、いくつかのユーティリティがあります。この章を使用して、問題を解決するヒントを得られます。

15.1 インストールの問題

パッケージのインストールまたはクラスタのオンライン化に問題がある場合、次のリストに従って作業を進めます。

HAパッケージはインストールされているか
: クラスタの構成を管理に必要なパッケージは、High Availability Extensionで使用できる高可用性インストールパターンに付属しています。

  High Availability Extensionが各クラスタノードにアドオンとしてSUSE Linux Enterprise Server 11にインストールされているか、*高可用性*パターンが3.1項 「High Availability Extensionのインストール」 (23 ページ)で説明するように各マシンにインストールされているか、確認します。

初期構成がすべてのクラスタノードについて同一か
: 相互に通信するため、3.2項 「クラスタの初期セットアップ」 (24 ページ)で説明するように、同じクラスタに属するすべてのノードは同じ`bindnetaddr`、`mcastaddr`、`mcastport`を使用する必要があります。

/etc/ais/openais.confで設定されている通信チャネルとオプションがすべてのクラスタノードについて同一であることを確認します。

暗号化通信を使用する場合は、/etc/ais/authkeyファイルがすべてのクラスタノードで使用できるかどうかを確認します。

ファイアウォールでmcastportによる通信が許可されているか
クラスタノード間の通信に使用されるmcastportがファイアウォールでブロックされている場合、ノードは相互に認識できません。3.1項 「High Availability Extensionのインストール」 (23 ページ)で示すように、YaSTで初期セットアップを構成しているときに、ファイアウォール設定は通常、自動的に調整されます。

mcastportがファイアウォールでブロックされないようにするには、各ノードの/etc/sysconfig/SuSEfirewall2の設定を確認します。または、各クラスタノードのYaSTファイアウォールモジュールを起動します。*許可されるサービス*>*詳細*をクリックして、mcastportを許可された*UDP*ポートのリストに追加し、変更を確定します。

OpenAISが各クラスタノードで起動されているか
各クラスタノードのOpenAISの状態を/etc/init.d/openais statusで確認します。OpenAISが実行されていない場合、/etc/init.d/openais startを実行して起動します。

15.2 HAクラスタの「デバッグ」

クラスタで何かがうまく機能しない場合、まず次のことを試してください。リソース操作の履歴(オプション-o)と非アクティブなリソース(-r)が表示されます。

```
crm_mon -o -r
```

表示は10秒ごとに更新されます(Ctrl + Cでキャンセルできます)。次に例を示します。

*例 **15.1*** *停止されたリソース*

```
Refresh in 10s...

============
Last updated: Mon Jan 19 08:56:14 2009
Current DC: d42 (d42)
3 Nodes configured.
3 Resources configured.
============

Online: [ d230 d42 ]
OFFLINE: [ clusternode-1 ]

Full list of resources:

Clone Set: o2cb-clone
         Stopped: [  o2cb:0 o2cb:1o2cb:2 ]
Clone Set: dlm-clone
         Stopped [ dlm:0 dlm:1 dlm:2 ]
mySecondIP      (ocf::heartbeat:IPaddr):        Stopped

Operations:
* Node d230:
   aa: migration-threshold=1000000
    + (5) probe: rc=0 (ok)
    + (37) stop: rc=0 (ok)
    + (38) start: rc=0 (ok)
    + (39) monitor: interval=15000ms rc=0 (ok)
* Node d42:
   aa: migration-threshold=1000000
    + (3) probe: rc=0 (ok)
    + (12) stop: rc=0 (ok)
```

まず、ノードをオンラインにします(15.3項 (140 ページ)を参照)。その後、リソースと操作を確認します。

`http://clusterlabs.org/wiki/Documentation`にある*Configuration Explained* PDFの「*How Does the Cluster Interpret the OCF Return Codes?(クラスタが**OCF**戻りコードを解釈する方法)*」のセクションでは、3種類の回復を説明しています。

15.3 FAQ

クラスタの状態とは何ですか。
クラスタの現在の状態を確認するには、プログラムcrm_monを使用します。これにより、現在のDCと、現在のノードが認識しているノードとリソースのすべてが表示されます。

クラスタのノードの一部が互いに認識しません。
これにはいくつかの理由が考えられます。

- まず環境設定ファイル/etc/ais/openais.confを見て、マルチキャストアドレスがクラスタ内のすべてのノードについて同一であるか確認します(interfaceセクションをキーmcastaddrで参照します)。
- ファイアウォール設定を確認します。
- スイッチがマルチキャストアドレスをサポートしているか確認します。
- もう1つの理由として、ノード間の接続が破損していることが考えられます。これはファイアウォールの構成が正しくないことが大半の原因です。また、これはスプリットブレインの理由にもなり、クラスタがパーティション化されます。

現在吉のリソースを一覧表示したい。
コマンドcrm_resource -Lを使用して、現在のリソースの情報を取得できます。

リソースを構成しましたが、いつも失敗します。
リソースエージェントを手動で実行してみます。LSBで、*scriptname* startと*scriptname* stopを実行します。OCFスクリプトを確認するには、必要な環境変数を先に設定します。たとえば、IPaddr OCFスクリプトをテストする場合、変数ipの値を、変数の名前にOCF_RESKEY_をプレフィックスに付けた環境変数を設定して設定します。たとえば、次のコマンドを実行します。

```
export OCF_RESKEY_ip=<your_ip_address>
/usr/lib/ocf/resource.d/heartbeat/IPaddr validate-all
/usr/lib/ocf/resource.d/heartbeat/IPaddr start
/usr/lib/ocf/resource.d/heartbeat/IPaddr stop
```

これが失敗する場合、必須変数が不足しているか、またはパラメータを誤ってタイプしたことが考えられます。

エラーメッセージを受け取りました。詳細情報を取得できますか。
コマンドにはいつでも-Vパラメータを追加できます。これを複数回行うと、デバッグ出力の情報量が増加します。

リソースはどのようにクリーンアップしますか。
リソースのIDがわかっている場合は(crm_resource -Lで取得できます)、特定のリソースをcrm_resource -C -r *resource id* -H *HOST*で削除します。

ocfs2デバイスをマウントできません。
/var/log/messageに次の行があるか、確認します。

```
Jan 12 09:58:55 clusternode2 lrmd: [3487]: info: RA output:
(o2cb:1:start:stderr) 2009/01/12_09:58:55
  ERROR: Could not load ocfs2_stackglue
Jan 12 16:04:22 clusternode2 modprobe: FATAL: Module ocfs2_stackglue not
 found.
```

この場合、カーネルモジュールocfs2_stackglue.koがありません。インストールされたカーネルに応じて、パッケージocfs2-kmp-default、ocfs2-kmp-pae、またはocfs2-kmp-xenをインストールします。

15.4 その他の情報

LinuxおよびHeartbeatの、クラスタリソースの設定、およびHeartbeatクラスタの管理とカスタマイズなど、高可用性に関するその他の情報については、『http://clusterlabs.org/wiki/Documentation』を参照してください。

16 クラスタ管理ツール

High Availability Extensionにはクラスタをコマンドラインから管理する際に役立つ、包括的なツールセットが付属しています。この章では、CIBおよびクラスタリソースでのクラスタ構成を管理するために必要なツールを紹介します。リソースエージェントを管理するためのその他のコマンドラインツール、またはセットアップのデバッグおよびトラブルシューティングに使用するツールについては、第15章 *トラブルシューティング* (137 ページ)で説明しています。

次のリストは、クラスタ管理に関連するいくつかのタスクを示しており、これらのタスクを実行するために使用するツールを簡単に説明しています。

クラスタの状態の監視
: `crm_mon`コマンドでは、クラスタの状態と構成を監視できます。出力には、ノード数、uname、uuid、状態、クラスタで構成されたリソース、それぞれの現在の状態が含まれます。`crm_mon`の出力は、コンソールに表示したり、HTMLファイルに出力したりできます。statusセクションのないクラスタ設定ファイルが指定された場合、`crm_mon`はファイルに指定されたノードとリソースの概要を作成します。このツールの使用方法とコマンド構文の詳細については、crm_mon(8) (168 ページ)を参照してください。

CIBの管理
: `cibadmin`コマンドは、Heartbeat CIBを操作するための低レベル管理コマンドです。CIBのすべてまたは一部のダンプ、CIBのすべてまたは一部の更新、すべてまたは一部の変更、CIB全体の削除、その他のCIB管理操作

に使用できます。このツールの使用方法とコマンド構文の詳細については、cibadmin(8) (146 ページ)を参照してください。

設定の変更の管理

crm_diffコマンドは、XMLパッチの作成と適用をサポートします。クラスタ設定の2つのバージョン間の差異を表示する、または後でcibadmin(8) (146ページ)を使用して適用できるように変更を保存する場合に有効です。このツールの使用方法とコマンド構文の詳細については、crm_diff(8) (159 ページ)を参照してください。

CIB属性の操作

crm_attributeコマンドで、CIBで使用されているノード属性およびクラスタ構成オプションを問い合わせて操作できます。このツールの使用方法とコマンド構文の詳細については、crm_attribute(8) (156 ページ)を参照してください。

クラスタ構成の検証

crm_verifyコマンドは、構成データベース(CIB)の整合性およびその他の問題を確認します。構成を含むファイルを確認したり、実行中のクラスタに接続したりできます。2種類の問題を報告します。エラーは、解決しないとHeartbeatが正常に機能できず、警告の解決は管理者が担当します。cm_verifyは新規または変更された構成の作成を支援します。実行中のクラスタのCIBのローカルコピーを作成し、編集し、crm_verifyを使用して検証し、新規構成をcibadminを使用して適用できます。このツールの使用方法とコマンド構文の詳細については、crm_verify(8) (197 ページ)を参照してください。

リソース構成の管理

crm_resourceコマンドは、クラスタ上でリソース関連のさまざまなアクションを実行します。構成されたリソースの定義の変更、リソースの開始と停止、リソースの削除およびノード間での移行を実行できます。このツールの使用方法とコマンド構文の詳細については、crm_resource(8) (172 ページ)を参照してください。

リソースの失敗回数の管理

crm_failcountコマンドは、所定のノードのリソースごとの失敗回数を問い合わせます。このツールは、失敗回数のリセットにも使用でき、リソースが頻繁に失敗したノード上で再度実行できるようにします。このツールの使用方法とコマンド構文の詳細については、crm_failcount(8) (162 ページ)を参照してください。

ノードのスタンバイ状態の管理

`crm_standby`コマンドは、ノードのスタンバイ属性を操作します。スタンバイモードのノードはすべて、リソースをホストすることができず、そのノードにあるリソースは削除する必要があります。スタンバイモードはカーネルアップデートなどの保守タスクに有効です。ノードからスタンバイ属性を削除すると、クラスタの完全なアクティブメンバーに戻ります。このツールの使用方法とコマンド構文の詳細については、crm_standby(8) (194 ページ)を参照してください。

cibadmin (8)

cibadmin — クラスタの環境設定に直接アクセスできるようにします

書式

環境設定、またはその一部のクエリ、変更、置換、削除を行えるようにします。

```
cibadmin (--query|-Q) -[Vrwlsmfbp] [-i xml-object-id|-o
  xml-object-type] [-t t-flag-whatever] [-h hostname]

cibadmin (--create|-C) -[Vrwlsmfbp] [-X xml-string]
  [-x xml- filename] [-t t-flag-whatever] [-h hostname]

cibadmin (--replace-R)  -[Vrwlsmfbp]  [-i xml-object-id|
  -o xml-object-type]  [-X  xml-string] [-x xml-filename] [-t
  t-flag- whatever] [-h hostname]

cibadmin (--update|-U) -[Vrwlsmfbp]  [-i xml-object-id|
  -o xml-object-type]  [-X  xml-string] [-x xml-filename] [-t
  t-flag- whatever] [-h hostname]

cibadmin (--modify|-M) -[Vrwlsmfbp]  [-i xml-object-id|
           -o xml-object-type]  [-X  xml-string] [-x xml-filename] [-t
           t-flag- whatever] [-h hostname]

cibadmin (--delete|-D) -[Vrwlsmfbp]  [-i xml-object-id|
           -o xml-object-type] [-t t-flag-whatever] [-h hostname]

cibadmin (--delete_alt|-d) -[Vrwlsmfbp] -o
           xml-object-type [-X xml-string|-x xml-filename]
           [-t t-flag-whatever]  [-h hostname]

cibadmin --erase (-E)

cibadmin --bump (-B)

cibadmin --ismaster (-m)

cibadmin --master (-w)

cibadmin --slave (-r)

cibadmin --sync (-S)

cibadmin --help (-?)
```

説明

cibadminコマンドは、Heartbeat CIBを操作するための低レベル管理コマンドです。CIBのすべて、または一部のダンプ、更新、変更、CIB全体の削除、その他各種のCIB管理操作を実行するために使用します。

cibadminはCIBのXMLツリーに対して機能します。ほとんどの場合、実行する更新やクエリの意味は把握していません。つまり、XMLツリー要素の意味を理解している人には自然に思えるショートカットも、cibadminでは利用できません。これにはあいまいな点がまったくない状態でなければなりません。そのため、入出力の両方で有効なXMLサブツリー(タグおよび要素)のみ扱うことができます。

注意

cibadminは、手動によるcib.xmlファイルの編集に常に優先して使用する必要があります。特にクラスタがアクティブな場合はその必要があります。クラスタは手動編集の検出とその回避にあらゆる手段を駆使してユーザのデータを紛失、または破壊しないようにしています。

オプション

`--obj_type` *`object-type`*, `-o` *`object-type`*
操作実行対象のオブジェクトタイプを指定します。有効な値はnodes、resources、constraints、crm_status、およびstatusです。

`--verbose`, `-V`
デバッグモードを有効にします。-Vオプションを追加すると、詳細な出力が得られます。

`--help`, `-?`
cibadminからヘルプメッセージを入手します。

`--xpath` *`PATHSPEC`*, `-A` *`PATHSPEC`*
obj_typeの代わりに使用する有効なXPathを提供します。

コマンド

--bump, -B
CIBのepochバージョンカウンタを増やします。通常この値は、新しいリーダーを選出するとクラスタが自動的に増分します。古い環境設定(使用していないクラスタノードのものなど)を廃止する場合には、手動で増やす方法は便利です。

--create, -C
引数のXMLコンテンツから新しいCIBを作成します。

--delete, -D
たとえば<op id="rsc1_op1" name="monitor"/>のように、指定した基準と一致する最初のオブジェクトを削除します。要素を削除するには、タグ名とすべての属性が一致しなければなりません。

--erase, -E
CIB全体のコンテンツを削除します。

--ismaster, -m
CIBソフトウェアのローカルインスタンスがマスタインスタンスかどうかを示すメッセージを印刷します。マスタインスタンスであればリターンコード0を、それ以外の場合は35を返して終了します。

--modify, -M
CIBのXMLツリーでオブジェクトを検索して更新します。

--query, -Q
CIBの一部をクエリします。

--replace, -R
CIB内のXMLオブジェクトを反復置換します。

--sync, -S
特定ホスト(-h使用時)、またはDC(-h未使用時)で、CIBのすべてのノードの再同期化を強制的に実行します。

XMLデータ

`--xml-text` *`string`*, `-X` *`string`*
crmadminを実行するXMLタグまたはXML断片を指定します。完全なタグまたはXML断片でなければなりません。

`--xml-file` *`filename`*, `-x` *`filename`*
cibadminを実行するファイルからXMLを指定します。完全なタグまたはXML断片でなければなりません。

`--xml_pipe`, `-p`
`cibadmin`を実行するXMLを、標準入力からのものとして指定します。完全なタグまたはXML断片でなければなりません。

詳細オプション

`--host` *`hostname`*, `-h` *`hostname`*
指定されたホストにコマンドを送信します。`query`および`sync`コマンドにのみ適用されます。

`--local`, `-l`
コマンドをローカルに適用します(ほとんど使用されない高度なオプション)。

`--no-bcast`, `-b`
CIBを変更した場合でも、コマンドはブロードキャストされません。

重要項目

別なクラスタを発生しないように、このオプションは注意して使用してください。

`--sync-call`, `-s`
呼び出しが完了するまで待って復帰します。

例

stdoutに配信されたアクティブなCIB(ステータスセクションなどを含む)全体のコピーを入手するには、このコマンドを実行します。

```
cibadmin -Q
```

IPaddr2リソースを*resources*セクションに追加するには、最初に、次のコンテンツを含んだfooファイルを作成します。

```
   <primitive id="R_10.10.10.101" class="ocf" type="IPaddr2"
    provider="heartbeat">
    <instance_attributes id="RA_R_10.10.10.101">
     <attributes>
      <nvpair id="R_ip_P_ip" name="ip" value="10.10.10.101"/>
      <nvpair id="R_ip_P_nic" name="nic" value="eth0"/>
     </attributes>
    </instance_attributes>
   </primitive>
```

そして次のコマンドを実行します。

```
cibadmin --obj_type resources -U -x foo
```

すでに追加されているIPaddr2リソースのIPアドレスを変更するには、次のコマンドラインを実行します。

```
cibadmin -M -X '<nvpair id="R_ip_P_ip" name="ip" value="10.10.10.102"/>'
```

注意

これは、リソース名を新しいIPアドレスに変更するものではありません。追加するには、一旦削除して、新しいIDタグで再度リソースを追加してください。

過去に追加されたIPアドレスリソースを削除せずに停止(無効に)するには、次のコンテンツを含むbarと呼ばれるファイルを作成します。

```
<primitive id="R_10.10.10.101">
 <instance_attributes id="RA_R_10.10.10.101">
  <attributes>
   <nvpair id="stop_R_10.10.10.101" name="target-role" value="Stopped"/>
  </attributes>
 </instance_attributes>
</primitive>
```

そして次のコマンドを実行します。

```
cibadmin --obj_type resources -U -x bar
```

前の手順で停止したIPアドレスリソースを再開するには、次のコマンドを実行します。

```
cibadmin -D -X '<nvpair id="stop_R_10.10.10.101">'
```

CIBからIPアドレスリソースを完全に削除するには、次のコマンドを実行します。

```
cibadmin -D -X '<primitive id="R_10.10.10.101"/>'
```

CIBを手動で編集した新しいCIBに入れ替えるには、次のコマンドを実行します。

```
cibadmin -R -x $HOME/cib.xml
```

ファイル

`/var/lib/heartbeat/crm/cib.xml`—ディスク上のCIB(ステータスセクションを除く)。

参照

crm_resource(8) (172 ページ)、crmadmin(8) (153 ページ)、lrmadmin(8)、heartbeat(8)

著者

`cibadmin`はAndrew Beekhofが作成しました。

このマニュアルページは本来Alan Robertsonが作成したものです。

注意事項

ローカルディスク上に自動保存されているCIBのコピーを使って操作することは避けてください。クラスタに変更があるたびにCIBが更新されます。した

がってCIBの古いバックアップコピーを使用して設定変更を伝播すると、クラスタの整合性が失われることがあります。

crmadmin (8)

crmadmin — Clusterリソースマネージャを制御します

書式

```
crmadmin [-V|-q] [-i|-d|-K|-S|-E] node
crmadmin [-V|-q] -N -B
crmadmin [-V|-q] -D
crmadmin -v
crmadmin -?
```

説明

crmadminは当初CRMデーモンの大半の走査を制御するために設計されたものです。ただし、crm_attributeやcrm_resourceなどの機能の多くは他のツールでは使用されなくなりました。残っている機能の多くはテストやcrmdプロセスのステータスに関するものです。

警告

crmadminオプションの中にはテスト用のものがあり、使用方法を間違うと問題を発生してしまうことがあります。特に--killまたは--electionオプションは、内容に精通していない限り使用しないでください。

オプション

--help, -?
: ヘルプテキストを印刷します。

--version, -v
: HA、CRM、およびCIB機能セットのバージョン詳細を印刷します。

--verbose, -V
: コマンドのデバッグ情報を有効にします。

注意

追加のインスタンスを提供すると、詳細になります。

`--quiet`, `-q`
デバッグ情報はまったく提供せずに、出力を最小限にします。

`--bash-export`, `-B`
`export uname=uuid`の形式のbashエクスポートエントリを作成します。これは`crmadmin -N node`コマンドにのみ適用されます。

注意

`-B`機能はほとんど使用されないので、将来のバージョンでは削除される可能性があります。

コマンド

`--debug_inc node`, `-i node`
指定したノードのCRMデーモンのデバッグレベルを上げます。USR1信号をcrmdプロセスに送信することで同じ結果が得られます。

`--debug_dec node`, `-d node`
指定したノードのCRMデーモンのデバッグレベルを下げます。USR2信号をcrmdプロセスに送信することで同じ結果が得られます。

`--kill node`, `-K node`
指定したノードのCRMデーモンをシャットダウンします。

警告

非常に慎重に使用してください。このアクションは通常Heartbeatだけが実行するもので、二次的な影響が発生します。

`--status node`, `-S node`
指定したノードのCRMデーモンのステータスをクエリします。

出力には一般的な状態を示すインジケータと、crmdプロセスのFSM状態が含まれています。クラスタの作業内容を確認するには有効な方法です。

`--election` *node*, `-E` *node*
指定したノードの選択を開始します。

警告

非常に慎重に使用してください。このアクションは通常内部で開始され、二次的な影響が発生します。

`--dc_lookup`, `-D`
現在のDCのunameをクエリします。

DCの場所は内部でcrmdにとってのみ重要なもので、ログを確認するノードを決定する場合以外で管理者が使用することはまれです。

`--nodes`, `-N`
すべてのメンバノードのunameをクエリします。このクエリの結果には`offline`モードのノードも含まれます。

注意

`-i`、`-d`、`-K`、および`-E`オプションが使用されることはまれで、将来のバージョンでは削除される可能性があります。

参照

crm_attribute(8) (156 ページ)、 crm_resource(8) (172 ページ)

著者

`crmadmin`はAndrew Beekhofによって作成されました。

crm_attribute (8)

crm_attribute — ノード属性とクラスタオプションのクエリ、変更、削除が行えるようにします。

書式

```
crm_attribute [options]
```

説明

crm_attributeコマンドはCIBで使用されているノード属性とクラス他環境設定のクエリと操作を行います。

オプション

--help, -?
ヘルプメッセージを印刷します。

--verbose, -V
デバッグ情報を有効にします。

> **注意**
>
> 追加のインスタンスを提供すると、詳細になります。

--quiet, -Q
-Gを使用して属性クエリを行う場合、値だけをstdoutに印刷します。このオプションは-Gと一緒に使用します。

--get-value, -G
初期設定を設定するのではなく、取得します。

--delete-attr, -D
属性を設定するのではなく、削除します。

--attr-id *string*, -i *string*
上級ユーザ専用です。id属性を識別します。

--attr-value *string*, -v *string*
設定する値。-Gと一緒に使用すると無視されます。

--node *node_name*, -N *node_name*
変更するノードのuname。

--set-name *string*, -s *string*
属性を読み書きする属性セットを指定します。

--attr-name *string*, -n *string*
設定またはクエリする属性を指定します。

--type *string*, -t *type*
属性を設定するCIBのセクションや、クエリを行う属性が属しているCIBのセクションを決定します。nodes、status、またはcrm_configの値を指定できます。

例

nodesセクションで、CIB内のホスト*myhost*のlocation属性値のクエリを行います。

```
crm_attribute -G -t nodes -U myhost -n location
```

CIB内のcrm_configセクションで、cluster-delay属性値のクエリを行います。

```
crm_attribute -G -t crm_config -n cluster-delay
```

CIB内のcrm_configセクションで、cluster-delay属性値のクエリを行います。値だけを印刷します。

```
crm_attribute -G -Q -t crm_config -n cluster-delay
```

CIB内のnodesセクションから、ホスト*myhost*のlocation属性を削除します。

```
crm_attribute -D -t nodes -U myhost -n location
```

officeという値を持つlocationという名前の属性をCIBのnodesセクションの*set*サブセクションに追加します(設定はホスト*myhost*に適用されます)。

```
crm_attribute -t nodes -U myhost -s set -n location -v office
```

*myhost*ホストのnodesセクションのlocation属性値を変更します。

```
crm_attribute -t nodes -U myhost -n location -v backoffice
```

ファイル

/var/lib/heartbeat/crm/cib.xml—ディスク上のCIB(ステータスセクションを除く)。このファイルを直接編集しないでください。

参照

cibadmin(8) (146 ページ)

著者

crm_attributeはAndrew Beekhofによって作成されました。

crm_diff (8)

crm_diff — クラスタ環境設定への変更を識別し、環境設定ファイルにパッチを適用します。

書式

```
crm_diff [-?|-V] [-o filename] [-O string] [-p filename] [-n filename] [-N string]
```

説明

crm_diffコマンドはXMLパッチの作成と適用を支援します。クラスタの環境設定の2つのバージョンの違いを視覚的に確認する場合や、変更を保存しておき、後でcibadminを使用して適用する場合には便利です。

オプション

--help, -?
　ヘルプメッセージを印刷します。

--original *filename*, -o *filename*
　差分をとる、またはパッチを適用する元のファイルを指定します。

--new *filename*, -n *filename*
　新しいファイルの名前を指定します。

--original-string *string*, -O *string*
　差分をとる、またはパッチを適用する元の文字列を指定します。

--new-string *string*, -N *string*
　新しい文字列を指定します。

--patch *filename*, -p *filename*
　元のXMLにパッチを適用します。必ず-oと一緒に使用してください。

--cib, -c
入力をCIBとして比較、またはパッチを適用します。-oと一緒に元のバージョンを指定し、パッチファイルを-pと一緒に、新しいバージョンを-nと一緒に指定します。

--stdin, -s
stdinから入力を読み込みます。

例

crm_diffを使用して各種のCIB環境設定ファイルの比較や、パッチの作成を行います。パッチを使用することで、1つずつcibadminコマンドを使用しなくても環境設定項目を簡単に再利用できます。

1. 比較する2つのクラスタセットアップに対してcibadminを実行して、2つの異なる環境設定ファイルを取得します。

```
cibadmin -Q > cib1.xml
cibadmin -Q > cib2.xml
```

2. お互いにファイル全体を比較するのか、環境設定の一部だけを比較するのかを決定します。

3. ファイル間の差分をstdoutに印刷するには、次のコマンドを使用します。

```
crm_diff -o cib1.xml -n cib2.xml
```

4. ファイル間の差分をファイルに印刷してパッチを作成するには、次のコマンドを使用します。

```
crm_diff -o cib1.xml -n cib2.xml > patch.xml
```

5. パッチを元のファイルに適用します。

```
crm_diff -o cib1.xml -p patch.xml
```

ファイル

/var/lib/heartbeat/crm/cib.xml—ディスク上のCIB(ステータスセクションを除く)。このファイルを直接編集しないでください。

参照

cibadmin(8) (146 ページ)

著者

`crm_diff`はAndrew Beekhofによって作成されました。

crm_failcount (8)

crm_failcount — 各リソースの失敗を記録するカウンタを管理します。

書式

```
crm_failcount [-?|-V] -D -u|-U node -r resource
crm_failcount [-?|-V] -G -u|-U node -r resource
crm_failcount [-?|-V] -v string -u|-U node -r resource
```

説明

Heartbeatはリソースが現在のノードで失敗した場合に、別なノードへのリソースのフェールオーバーを計算して強制的に行うための高度な手段です。リソースにはresource-stickiness属性があり、特定のノードでの実行にどの程度こだわっているのかを決定します。また別なノードへのリソースのフェールオーバーを決定するしきい値を決定するmigration-threshold属性もあります。

failcount属性がリソースに追加され、リソース監視が失敗した場合に増分されます。failcountの値にmigration-thresholdを乗算したものが、このリソースのフェールオーバースコアになります。この値がこのリソースに設定されている初期設定値を超えると、リソースが別なノードに移動され、failure countがリセットされるまで元のノードでは実行されなくなります。

crm_failcountコマンドは、特定のノードのリソースごとの失敗数のクエリを行います。このツールはfailcountのリセットにも使用でき、リソースが頻繁に失敗していたノードで再度実行できるようにします。

オプション

--help, -?
　ヘルプメッセージを印刷します。

--verbose, -V
　デバッグ情報を有効にします。

注意

追加のインスタンスを提供すると、詳細になります。

--quiet, -Q
-Gを使用して属性クエリを行う場合、値だけをstdoutに印刷します。このオプションは-Gと一緒に使用します。

初期設定を設定するのではなく、取得します。

--delete-attr, -D
削除する属性を指定します。

--attr-id *string*, -i *string*
上級ユーザ専用です。id属性を識別します。

--attr-value *string*, -v *string*
使用する値を指定します。このオプションは-Gと一緒に使用すると無視されます。

--node *node_uname*, -U *node_uname*
変更するノードのunameを指定します。

--resource-id *resource name*, -r *resource name*
操作するリソースの名前を指定します。

例

ノードnode1上のリソースmyrscの失敗回数をリセットします。

```
crm_failcount -D -U node1 -r my_rsc
```

ノードnode1上のリソースmyrscの現在の失敗回数のクエリを行います。

```
crm_failcount -G -U node1 -r my_rsc
```

ファイル

/var/lib/heartbeat/crm/cib.xml—ディスク上のCIB(ステータスセクションを除く)。このファイルを直接編集しないでください。

参照

crm_attribute(8) (156 ページ)、cibadmin(8) (146 ページ)、およびLinuxの高可用性に関するFAQのWebサイト [http://www.linux-ha.org/v2/faq/forced_failover]を参照してください。

著者

crm_failcountはAndrew Beekhofによって作成されました。

crm_master (8)

crm_master—特定のノードで昇格するマスタ/スレーブリソースの初期設定を管理します。

書式

```
crm_master [-V|-Q] -D [-l lifetime]
crm_master [-V|-Q] -G [-l lifetime]
crm_master [-V|-Q] -v string [-l string]
```

説明

crm_masterはリソースエージェントスクリプト内部から呼び出され、マスタモードに昇格するリソースインスタンスを決定します。コマンドラインからは使用しないでください。リソースエージェントの単なるヘルパーユーティリティです。RAはcrm_masterを使用して特定のインスタンスをマスタモードに昇格するか、この初期設定をそのインスタンスから削除します。有効期間を割り当てることで、この設定がノードの再起動後も有効なのか(有効期間をforeverに設定)または再起動後は無効にするのか(有効期間をrebootに設定)を決定します。

リソースエージェントは、どのリソースでcrm_masterを実行するのかを決定する必要があります。これらのクエリはリソースエージェントスクリプト内で処理する必要があります。crm_masterの実際の呼び出しは、crm_attributeコマンドに類似の構文に従います。

オプション

--help, -?
　ヘルプメッセージを印刷します。

--verbose, -V
　デバッグ情報を有効にします。

注意

追加のインスタンスを提供すると、詳細になります。

--quiet, -Q
-Gを使用して属性クエリを行う場合、値だけをstdoutに印刷します。このオプションは-Gと一緒に使用します。

--get-value, -G
昇格する初期設定を設定するのではなく、取得します。

--delete-attr, -D
属性を設定するのではなく、削除します。

--attr-id *string*, -i *string*
上級ユーザ専用です。id属性を識別します。

--attr-value *string*, -v *string*
設定する値。-Gと一緒に使用すると無視されます。

--lifetime *string*, -l *string*
初期設定の有効期限を指定します。rebootまたはforeverの値を指定できます。

環境変数

OCF_RESOURCE_INSTANCE—リソースインスタンス名

ファイル

/var/lib/heartbeat/crm/cib.xml—ディスク上のCIB(ステータスセクションを除く)。

参照

cibadmin(8) (146 ページ)、crm_attribute(8) (156 ページ)

著者

crm_masterはAndrew Beekhofによって作成されました。

crm_mon (8)

crm_mon — クラスタのステータスを監視します

書式

```
crm_mon [-V] -d -pfilename -h filename
crm_mon [-V] [-1|-n|-r] -h filename
crm_mon [-V] [-n|-r] -X filename
crm_mon [-V] [-n|-r] -c|-1
crm_mon [-V] -i interval
crm_mon -?
```

説明

crm_monコマンドによって、クラスタのステータスと環境設定を監視できます。ノード数、uname、uuid、ステータス、クラスタに設定されたリソース、それぞれの現在のステータスが出力に含まれます。crm_monの出力はコンソールに表示することも、HTMLファイルに印刷することもできます。ステータスセクションのないクラスタの環境設定ファイルが提供された場合、crm_monはファイル内で指定されたノードとリソースの概要を作成します。

オプション

--help, -?
: ヘルプを表示する.

--verbose, -V
: 詳細なデバッグ出力を行います。

--interval *seconds*, -i *seconds*
: 更新頻度を決定します。-iが指定されていない場合、15秒のデフォルト値が採用されます。

`--group-by-node`, `-n`
ノードごとにリソースをグループ化します。

`--inactive`, `-r`
非アクティブなリソースを表示します。

`--simple-status`, `-s`
簡単な一行の出力としてクラスタステータスを1度表示します(nagiosには適している)。

`--one-shot`, `-1`
コンソールにクラスタステータスを1度表示して、終了します(ncursesは使用しない)。

`--as-html` *filename*, `-h` *filename*
クラスタステータスを指定のファイルに書き込みます。

`--web-cgi`, `-w`
CGIに適した出力のWebモード。

`--daemonize`, `-d`
バックグラウンドでデーモンとして実行します。

`--pid-file` *filename*, `-p` *filename*
デーモンのpidファイルを指定します。

例

クラスタのステータスを表示し、15秒ごとに更新済みのリストを取得します。

```
crm_mon
```

クラスタのステータスを表示し、`-i`で指定した間隔で更新済みのリストを取得します。`-i`を指定しない場合、デフォルトのリフレッシュ間隔として15秒が採用されます。

```
crm_mon -i interval[s]
```

クラスタのステータスをコンソールに表示します。

```
crm_mon -c
```

クラスタのステータスをコンソールに1度だけ表示して、終了します。

```
crm_mon -1
```

クラスタのステータスとノードごとのグループリソースを表示します。

```
crm_mon -n
```

クラスタのステータス、ノードごとのグループリソースを表示し、リストに非アクティブなリソースを含みます。

```
crm_mon -n -r
```

クラスタのステータスをHTMLファイルに書き込みます。

```
crm_mon -h filename
```

crm_monをバックグランドでデーモンとして実行し、デーモンのpidファイルを指定してデーモンプロセスの制御を簡単に行えるようにし、HTML出力を作成します。このオプションは他の監視用アプリケーションで簡単に処理できるHTML出力を継続的に作成できるようにします。

```
crm_mon -d -p filename -h filename
```

既存のクラスタ環境設定ファイル(*filename*)のクラスタの環境設定を表示し、ノードごとにリソースをグループ化して、非アクティブなリソースをリストに含みます。このコマンドは、ライブクラスタへの導入前に、クラスタの環境設定をテストするために使用できます。

```
crm_mon -r -n -X filename
```

ファイル

/var/lib/heartbeat/crm/cib.xml—ディスク上のCIB(ステータスセクションを除く)。このファイルを直接編集しないでください。

著者

crm_monはAndrew Beekhofによって作成されました。

crm_node (8)

crm_node — クラスタのメンバーを一覧表示します。

書式

```
crm_node [-V] [-p|-e|-q]
```

説明

クラスタのメンバーを一覧表示します。

オプション

`-V`
　詳細な出力を行います

`--partition`, `-p`
　このパーティションのメンバーを印刷します。

`--epoch`, `-e`
　このノードがパーティションに参加したエポックを印刷します。

`--quorum`, `-q`
　パーティションに定数がある場合は1を印刷します。

crm_resource (8)

crm_resource — クラスタリソースに関連するタスクを実行します。

書式

```
crm_resource  [-?|-V|-S] -L|-Q|-W|-D|-C|-P|-p [options]
```

説明

crm_resourceコマンドは、クラスタに対して各種のリソース関連アクションを実行します。設定済みのリソースの定義変更、リソースの開始や停止、リソースの削除、ノード間でのリソースのマイグレーションが行えます。

--help, -?
ヘルプメッセージを印刷します。

--verbose, -V
デバッグ情報を有効にします。

注意

追加のインスタンスを提供すると、詳細になります。

--quiet, -Q
stdoutに値だけを印刷します(-Wで使用)。

コマンド

--list, -L
すべてのリソースを一覧表示します。

--query-xml, -x
リソースのクエリを行います。

-rが必要です。

--locate, -W
リソースの場所を探します。

必須: -r

--migrate, -M
現在の場所からリソースをマイグレートします。-Nを使用してマイグレート先を指定します。

-Nを指定しなかった場合、現在の場所に対するルールと-INFINITYのスコアを作成することで、リソースを強制的に移動することができます。

注意

これによって-Uで制約が解除されるまで、このノードでリソースが実行されないようにします。

必須: -r、オプション: -N, -f

--un-migrate, -U
-Mで作成されたすべての制約を削除します。

必須: -r

--delete, -D
CIBからリソースを削除します。

必須: -r, -t

--cleanup, -C
LRMからリソースを削除します。

必須: -rオプション: -H

--reprobe, -P
CRM以外から起動されたリソースを再確認します。

オプション: -H

--refresh, -R
LRMからCIBを更新します。

オプション: -H

--set-parameter *string*, -p *string*
指定されたパラメータをリソースに設定します。

必須: -r, -v. オプション: -i, -s, --meta

--get-parameter *string*, -g *string*
リソース用に指定されたパラメータ取得します。

必須: -rオプション: -i, -s, --meta

--delete-parameter *string*, -d *string*
リソース用に指定されたパラメータを削除します。

必須: -rオプション: -i, and --meta

--list-operations *string* , -O *string*
アクティブなリソース操作を一覧表示します。リソース、ノードのいずれか、または両方によってオプションでフィルタします。オプション: -N, -r

--list-all-operations *string* , -o *string*
すべてのリソース操作を一覧表示します。リソース、ノードのいずれか、または両方によってオプションでフィルタします。オプション: -N, -r

オプション

--resource *string*, -r *string*
リソースIDを指定します。

--resource-type *string*, -t *string*
リソースタイプ(primitive、clone、groupなど)を指定します。

--property-value *string*, -v *string*
プロパティ値を指定します。

--node *string*, -N *string*
ホスト名を指定します。

--meta
リソースの環境設定オプションを、リソースエージェントスクリプトに渡されたもの以外に変更します。-p、-g、および-dと一緒に使用します。

--lifetime *string*, -u *string*
マイグレーション制約の有効期限。

--force, -f
現在の場所のルールと-INFINITYのスコアを作成することで、リソースを強制的に移動します。

リソースの固着性と制約スコアの合計がINFINITY(現在は100,000)を超える場合に使用します。

注意

これによって -Uで制約が解除されるまで、このノードでリソースが実行されないようにします。

-s *string*
(上級ユーザのみ)変更するinstance_attributesオブジェクトのIDを指定します。

-i *string*
(上級ユーザのみ)変更または削除するnvpairオブジェクトのIDを指定します。

例

すべてのリソースの一覧表示:

```
crm_resource -L
```

リソースの実行場所を確認(実行しているかどうかも含め):

```
crm_resource -W -r my_first_ip
```

my_first_ipリソースが実行中の場合は、このコマンドの出力によって実行中のノードが明らかになります。実行していない場合は、出力で判明します。

リソースの開始または停止:

```
crm_resource -r my_first_ip -p target_role -v started
crm_resource -r my_first_ip -p target_role -v stopped
```

リソース定義のクエリ:

```
crm_resource -Q -r my_first_ip
```

リソースを現在の場所から離れたところにマイグレートします。

```
crm_resource -M -r my_first_ip
```

リソースを特定の場所にマイグレートします。

```
crm_resource -M -r my_first_ip -H c001n02
```

リソースが通常の場所に復帰できるようにします。

```
crm_resource -U -r my_first_ip
```

注意

resource_stickinessとdefault_resource_stickinessの値は、元には戻らないことを意味します。その場合は、-Mを使用してこのコマンドの実行前に復帰させます。

CRMからリソースを削除します。

```
crm_resource -D -r my_first_ip -t primitive
```

CRMからリソースグループを削除します。

```
crm_resource -D -r my_first_group -t group
```

CRM内のリソースのリソース管理を無効にします。

```
crm_resource -p is-managed -r my_first_ip -t primitive -v off
```

CRM内のリソースのリソース管理を有効にします。

```
crm_resource -p is-managed -r my_first_ip -t primitive -v on
```

手動でのクリーンアップ後、失敗したリソースをリセットします。

```
crm_resource -C -H c001n02 -r my_first_ip
```

CRM以外から開始されたリソースの全ノードを確認します。

```
crm_resource -P
```

CRM以外から開始されたリソースの1ノードを確認します。

```
crm_resource -P -H c001n02
```

ファイル

/var/lib/heartbeat/crm/cib.xml—ディスク上のCIB(ステータスセクションを除く)。このファイルを直接編集しないでください。

参照

cibadmin(8) (146 ページ)、crmadmin(8) (153 ページ)、lrmadmin(8)、heartbeat(8)

著者

crm_resourceはAndrew Beekhofによって作成されました。

crm_shadow (8)

crm_shadow — ライブクラスタの更新前にSandboxの環境設定変更を実行します

書式

```
crm_shadow [-V] [-p|-e|-q]
```

説明

環境設定ツール(cibadmin、crm_resourceなど)がライブクラスタではなく、オフラインで機能し、二次的な効果をプレビュー、及びテストできるような環境をセットアップします。

オプション

--verbose, -V
　デバッグ情報を有効にします。インスタンスを追加すると詳細な情報が得られます

--which, -w
　アクティブなシャドーコピーを示します

--display, -p
　シャドーコピーの内容を表示します

--diff, -d
　シャドーコピーの変更を表示します

--create-empty, -e*NAME*
　空のクラスタ構成で指定された名前のシャドーコピーを作成します

--create, -c*NAME*
　アクティブクラスタ構成で指定された名前のシャドーコピーを作成します

`--reset`, `-r`*NAME*
　アクティブクラスタ構成から指定された名前のシャドーコピーを作成しなおします

`--commit`, `-c`*NAME*
　指定された名前のクラスタのシャドーコピーの内容をアップロードします

`--delete`, `-d`*NAME*
　指定された名前のシャドーコピーの内容を削除します

`--edit`, `-e`*NAME*
　指定された名前のシャドーコピーを、好みのエディタで編集します

`--batch`, `-b`
　新しいシェルは生成しません

`--force`, `-f`
　新しいシェルは生成しません

`--switch`, `-s`
　指定された名前のシャドーコピーに切り替えます

内部コマンド

シャドー構成を扱うには、最初にシャドー構成を作成する必要があります。

```
crm_shadow --create-empty YOUR_NAME
```

それによってcrmツールを使用した場合と同様な内部シェルが提供されます。helpを使用してすべての内部コマンドの概要を入手するか、help *subcommand*を使用して特定のコマンドの概要を確認します。

表 16.1　*内部コマンドの概要*

コマンド	構文/説明
alias	`alias [-p] [name[=value] ... ]` aliasを引数なし、または-pオプションを指定して実行すると、標準出力にalias NAME=VALUEの別名リストが印刷されま

コマンド	構文/説明
	す。それ以外の場合、VALUEが指定された各NAMEの別名が定義されます。別名の展開時に、VALUEの後続スペースによって、その次の用語に対して別名の置換が確認されます。NAMEに対して別名が定義されていなければaliasはtrueを返します。
bg	`bg [JOB_SPEC ...]` &で開始した場合のように、各JOB_SPECをバックグラウンドに配置します。JOB_SPECがない場合、シェルが現在のジョブに対して抱いている概念が使用されます。

bind	`bind [-lpvsPVS] [-m keymap] [-f filename]`
	`[-q name] [-u name] [-r keyseq]`
	`[-x keyseq:shell-command]`
	`[keyseq:readline-function or readline-command]`
	主要なキーシーケンスをReadline関数またはマクロにバインドするか、Readline変数を設定します。オプションなしの引数構文は`~/.inputrc`と同じですが、たとえば`"\C-x\C-r":`
	`re-read-init-file`のバインドなど、単一引数として渡す必要があります。

break	`break [N]` `for`、`while`、または`until`ループから抜けます。Nを指定した場合、Nレベルを打破します。
builtin	`builtin [shell-builtin [arg ...]]` シェルビルトインとして実行します。シェルビルトインを関数に名前を変更したい場合で、関数内にビルトイン機能が必要な場合には便利です。
caller	`caller [EXPR]` 現在のサブルーチンコールのコンテキストを返します。*EXPR*を指定しない場合、`$line $filename`を返します。*EXPR*を指定すると、`$line $subroutine $filename`を返します。このような追加情報を使用してスタックトレースを行えます。

コマンド	構文/説明
case	`case WORD in [PATTERN [| PATTERN] [COMMANDS;;] ... esac` *PATTERN*に一致する*WORD*に基づき、*COMMANDS*を選択的に実行します。「|」を使用して、複数のパターンを区切ります。
cd	`cd [-L|-P] [dir]` カレントディレクトリをDIRに変更します。

command	`command [-pVv]`
	`command [arg ...]`
	
	*COMMAND*を*ARGS*付きで実行し、シェル関数を無視します。「ls」という名前のシェル関数があり、コマンド名を「ls」にする場合は、「command ls」と指定することができます。-pオプションを指定すると、PATHのデフォルト値が使用され、すべての標準ユーティリティを確実に検索することができます。-Vまたは-vオプションを指定すると、COMMANDを説明する文字列が印刷されます。-Vオプションのほうが詳細な説明が得られます。

compgen	`compgen [-abcdefgjksuv] [-o option] [-A action]`
	`  [-G globpat] [-W wordlist] [-P prefix]`
	`  [-S suffix] [-X filterpat] [-F function]`
	`  [-C command] [WORD]`
	
	オプションごとに予想される結果を表示します。シェル関数内で、予想されれる結果を生成するために使用します。オプションの*WORD*引数を提供すると、*WORD*との一致が生成されます。

complete	`complete [-abcdefgjksuv] [-pr] [-o option]`
	`  [-A action] [-G globpat] [-W wordlist] [-P prefix]`
	`  [-S suffix] [-X filterpat] [-F function] [-C command]`
	`  [name ...]`
	
	各*NAME*に対して、引数の完了方法を指定します。-pオプションを提供すると、またはオプションを提供した場合、既存の完了方法が入力として再利用できる方法で印刷されます。-rオプションは各*NAME*の完了方法を削除します。*NAME*が指定されていない場合は、すべての完了方法が削除されます。

コマンド	構文/説明
continue	continue [N] 含まれているFOR、WHILE、またはUNTILの次回のループ処理を再開します。*N*が指定されている場合は、含まれている*N*番目のループから再開します。
declare	declare [-afFirtx] [-p] [name[=value] ...] 変数の宣言を行うか、属性を指定する、あるいはその両方を行います。*NAME*が指定されていない場合は、代わりに変数値を表示します。-pオプションは各*NAME*の属性と値を表示します。
dirs	dirs [-clpv] [+N] [-N] 現在記憶しているディレクトリのリストを表示します。pushdコマンドによってディレクトリはリストへのパスを検出します。popdコマンドを使用すると、リストのバックアップが入手できます。
disown	disown [-h] [-ar] [JOBSPEC ...] デフォルトでは、アクティブなジョブのテーブルから各*JOBSPEC*を削除します。-hオプションを指定すると、ジョブをテーブルから削除せずに、マーキングを行い、シェルがSIGHUPを受信するとSIGHUPがジョブに送信されないようにします。-aオプションを指定すると、*JOBSPEC*を指定しなかった場合に、ジョブテーブルからすべてのジョブを削除します。-rオプションは、実行中のジョブだけを削除します。
echo	echo [-neE] [arg ...] ARGを出力します。-nを指定すると、後続の改行は抑制されます。-eオプションを指定すると、バックスラッシュでエスケープ処理した次の文字が有効になります。 \a (アラート、ベル)

コマンド	構文/説明
	\b (バックスペース) \c (後続の改行を抑制) \E (エスケープ文字) \f (用紙のフィード) \n (改行) \r (復帰改行) \t (水平タブ) \v (垂直タブ) \\ (バックスラッシュ) \0nnn (ASCIIコードがNNN (8進数)の文字で、 NNNには0から3の8進数を指定できます)。 -Eオプションを使用すると、 上記の文字の解釈を明示的に無効化できます。
enable	enable [-pnds] [-a] [-f filename] [name...] ビルトインシェルコマンドを有効または無効にします。これによって、フルパス名を指定しなくても、シェルビルトインと同じ名前のディスクコマンドを使用できます。-nを使用すると、*NAME*は無効になります。それ以外の場合は*NAME*が有効です。たとえばビルトインバージョンではなく、$PATH内のtestを使用する場合、enable -n testと入力します。ダイナミックロードをサポートしているシステムでは、-fオプションを使用して共有オブジェクト*FILENAME*から新しいビルトインをロードすることができます。-dオプションは、-fでロードされたビルトインを削除します。非オプション名が指定されていない場合、または-pオプションが提供されている場合、ビルトインのリストが印刷されます。-aオプションは、各ビルトインと、それが有効になっているかどうかを印刷します。-sオプションは、POSIX.2の「特殊」ビルトインへの出力を禁止します。-nオプションは、無効になっているすべてのビルトインのリストを表示します。
eval	eval [ARG ...]

コマンド	構文/説明
	*ARG*をシェルへの入力として読み込み、結果として生成されるコマンドを実行します。
exec	`exec [-cl] [-a name] file [redirection ...]` *FILE*を実行し、このシェルを指定したプログラムに置き換えます。*FILE*を指定しなかった場合、このシェル内でリダイレクションが行われます。最初の引数が-lの場合、ログインと同様に、*FILE*に渡された0番目の引数にダッシュを配置します。-cオプションを提供すると、FILEはnull環境で実行されます。-aオプションは、実行したプロセスのargv[0]を*NAME*に設定します。ファイルを実行できず、シェルがインタラクティブでない場合、シェルオプションexecfailが設定されていない限り、シェルは終了します。
exit	`exit [N]` *N*のステータスでシェルを終了します。*N*を省略すると、最後に実行したコマンドの終了ステータスになります。

export	`export [-nf] [NAME[=value] ...]`
	`export -p`
	*NAME*は、続けて実行したコマンドの環境に自動的にエクスポートするためのマーキングです。-fオプションが指定されている場合、*NAME*は関数を参照します。*NAME*が指定されていない場合、または-pが指定されている場合、このシェルでエクスポートされたすべての名前のリストが印刷されます。-nの引数は、以降の*NAME*からexportプロパティを削除します。--の引数は以降のオプション処理を無効にします。

false	`false` 結果として失敗を返します。
fc	`fc [-e ename] [-nlr] [FIRST] [LAST]` `fc -s [pat=rep] [cmd]`

コマンド	構文/説明
	fcは履歴リストのコマンドを表示、または編集し、再度実行します。*FIRST*および*LAST*には範囲を示す数字を指定することができます。または、*FIRST*に文字列を指定すると、その文字列で始まる最後のコマンドを示します。
fg	`fg [JOB_SPEC]` フォアグラウンドに*JOB_SPEC*を配置し、現在のジョブにします。*JOB_SPEC*がない場合、シェルが現在のジョブに対して抱いている概念が使用されます。
for	`for NAME [in WORDS ... ;] do COMMANDS; done` forループは、項目リスト内の各メンバーに対して一連のコマンドを実行します。`in WORDS ...;`がない場合、`in "$@"`が想定されます。*WORDS*内の各要素に対して、*NAME*が設定され、*COMMANDS*が実行されます。

function	`function NAME { COMMANDS ; }`
	`function NAME () { COMMANDS ; }`
	*COMMANDS*を実行する*NAME*によって呼び出される簡単なコマンドを作成します。コマンドラインの引数が*NAME*と一緒に`$0`として関数に渡されます。`$n`.

getopts	`getopts OPTSTRING NAME [arg]` シェルプロシージャはgetoptsisを使用して、位置パラメータを解析します。
hash	`hash [-lr] [-p PATHNAME] [-dt] [NAME...]` 各*NAME*について、コマンドのフルパス名が決定され、記憶されます。`-p`オプションを提供すると、*PATHNAME*を*NAME*のフルパス名として使用し、解析は行いません。`-r`オプションは、シェルに記憶しているすべての場所を忘れさせます。`-d`オプションは、シェルに記憶している各*NAME*の場所を忘れさせま

コマンド	構文/説明
	す。-tオプションを提供すると、各*NAME*が対応しているフルパス名が印刷されます。複数の*NAME*引数が-tと一緒に指定されている場合は、*NAME*はハッシュされたフルパス名より前に印刷されます。-lオプションは、入力として再利用可能なフォーマットで出力を表示します。引数を指定しなかった場合、記憶しているコマンドの情報が表示されます。

history	`history [-c] [-d OFFSET] [n]`
	`history -ps arg [arg...]`
	`history -awrm [filename]`
	
	行番号付きで履歴リストを表示します。*付きの行は変更があったことを示しています。*N*の引数は、最後の*N*行だけを一覧表示します。-cオプションは、すべてのエントリを削除して、履歴リストを消去します。-dオプションは、*OFFSET*のオフセットで履歴エントリを削除します。-wオプションは現在の履歴を履歴ファイルに書き込みます。-rはファイルを読み込み、内容を履歴リストに追加します。-aはこのセッションの履歴行を、履歴ファイルに追加します。引数-nは履歴ファイルから読み込んでいないすべての履歴行を読み込み、履歴リストに追加します。

jobs	`jobs [-lnprs] [JOBSPEC ...]`
	`job -x COMMAND [ARGS]`
	
	アクティブジョブを一覧表示します。-lオプションは通常の情報以外にプロセスidも一覧表示します。-pオプションはプロセスidだけを一覧表示します。-nを指定すると、前回の通知以降にステータスを変更したプロセスだけが印刷されます。*JOBSPEC*はそのジョブだけに出力を制限します。-rおよび-sオプションは、実行中および停止したジョブだけにそれぞれ出力を制限します。オプションを指定しなかった場合、すべてのアクティブなジョブのステータスが印刷されます。-xを指定すると、*ARGS*に記載のすべてのジョブ仕様を、そのジョブのプロセスグループリーダーのプロセスIDに置き換えた後で*COMMAND*が実行されます。

コマンド	構文/説明

kill	`kill [-s sigspec | -n signum | -sigspec] pid | JOBSPEC ...`
	`kill -l [sigspec]`
	
	PIDで指定されたプロセス(または*JOBSPEC*)にSIGSPEC信号を送信します。SIGSPECがない場合は、SIGTERMが想定されます。`-l`の引数は信号名を一覧表示します。`-l`の後に引数が続く場合は、名前を一覧表示する信号番号と見なされます。killは次の2つの理由でシェルビルトインとして扱われます。プロセスIDの代わりにジョブIDを使用すること、そして作成可能なプロセス数に達したら、プロセスを強制終了するためのプロセスは開始する必要がないという点です。

let	`let ARG [ARG ...]` 各*ARG*は評価する数値式です。評価はオーバーフローのチェックなしで、固定幅の整数で行われますが、0での除算はトラップされ、エラーのフラグが立てられます。次の演算子リストは、優先度が同じ演算子ごとのレベルにグループ分けされています。レベルは降順に一覧表示します。
local	`local NAME[=VALUE] ...` *NAME*と呼ばれるローカル変数を作成し、*VALUE*を与えます。`local`は関数内だけで使用できます。その関数とその子だけが変数*NAME*を見ることができるようにします。
logout	`logout` ログインシェルからログアウトします。
popd	`popd [+N | -N] [-n]` ディレクトリスタックからエントリを削除します。引数がなければ、そのスタックの最上位ディレクトリを削除し、新しい最上位ディレクトリへのcdを実行します。
printf	`printf [-v var] format [ARGUMENTS]`

コマンド	構文/説明
	printfは*FORMAT*の制御下で、*ARGUMENTS*のフォーマットと印刷を行います。*FORMAT*は標準出力にコピーされるプレーンな文字、変換後、標準出力にコピーされる文字エスケープシーケンス、フォーマット仕様という3つのタイプのオブジェクトを含む文字列で、いずれも後続の引数を印刷します。標準的なprintf(1)フォーマット以外に、%bは該当する引数内のバックスラッシュエスケープシーケンスを拡張し、および%qはシェル入力として再利用できるような方法で引数を引用することができます。-vオプションを提供すると、出力は標準出力ではなく、シェル変数VARの値に配置されます。
pushd	`pushd [dir | +N | -N] [-n]` ディレクトリスタックの最上部にディレクトリを追加するか、スタックを回転し、新しいスタックの最上部を現在の作業ディレクトリにします。引数がなければ、上位2つのディレクトリを交換します。
pwd	`pwd [-LP]` 現在の作業ディレクトリを印刷します。-Pオプションを指定すると、pwdはシンボリックリンクなしで物理的なディレクトリを印刷します。-Lオプションはpwdをシンボリックリンクに従わせます。
read	`read [-ers] [-u fd] [-t timeout] [-p prompt] [-a array] [-n nchars] [-d delim] [NAME ...]` 指定した*NAME*は読み込み専用としてマーキングされ、これらの*NAME*の値が以降の割り当てによって変更されることはありません。-fオプションを指定すると、NAMEに対応する関数にその旨のマーキングが行われます。引数が指定されない場合、または-pを指定した場合は、すべての読み込み専用の名前のリストが印刷されます。-aオプションは、各NAMEを配列変数として扱います。引数--はオプションの以降の処理を無効にします。

コマンド	構文/説明

readonly	`readonly [-af] [NAME[=VALUE] ...]`
	`readonly -p`
	
	指定した*NAME*は読み込み専用としてマーキングされ、これらの*NAME*の値が以降の割り当てによって変更されることはありません。`-f`オプションを指定すると、*NAME*に対応する関数にもその旨がマーキングされます。引数が指定されない場合、または`-p`を指定した場合は、すべての読み込み専用の名前のリストが印刷されます。`-a`オプションは各*NAME*を配列変数として扱います。引数`--`はオプションの以降の処理を無効にします。

return	`return [N]` *N*で指定された戻り値で関数を終了させます。*N*を省略すると、最後のコマンドの戻りステータスが採用されます。
select	`select NAME [in WORDS ... ;] do COMMANDS; done` *WORDS*は展開され、用語リストが生成されます。展開した用語セットは、それぞれの前に番号が挿入され、標準エラーに印刷されます。`in WORDS`がない場合、`in "$@"`が想定されます。その後`PS3`プロンプトが表示され、標準入力から行が読み込まれます。表示されている用語のいずれかと対応する番号から行が構成されている場合、*NAME*がその用語に設定されます。行が空の場合、*WORDS*とプロンプトが再度表示されます。EOFを読み込むと、コマンドが終了します。その他の値を読み込むと、*NAME*をnullに設定します。読み込んだ行が変数*REPLY*に保存されます。breakコマンドが実行されるまで、各選択後に*COMMANDS*が実行されます。
set	`set [--abefhkmnptuvxBCHP] [-o OPTION] [ARG...]` 内部シェルオプションを設定します。
shift	`shift [n]`

コマンド	構文/説明
	$N+1 ...の位置パラメータ名が$1 ...に変わりました。*N*を指定していない場合は、1と想定されます。
shopt	shopt [-pqsu] [-o long-option] OPTNAME [OPTNAME...] オプション動作を制御する変数値をトグルします。-sフラグは各*OPTNAME*を有効に(設定)します。-uフラグは各*OPTNAME*の設定を解除します。-qフラグは出力を抑制します。終了ステータスは各*OPTNAME*が設定されているのか、解除されているのかを示します。-oオプションは、*OPTNAME*をset -oで使用するために定義されたものだけに制限します。オプションが指定されていない場合、または-pオプションを指定した場合、設定可能なすべてのオプションのリストが、そのオプションが設定されているかどうかも含めて表示されます。
source	source FILENAME [ARGS] *FILENAME*からコマンドを読み込んで実行し、結果を返します。$PATH内のパス名を使用して、*FILENAME*を含むディレクトリを検出します。*ARGS*を提供すると、*FILENAME*の実行時に位置パラメータとして機能します。
suspend	suspend [-f] SIGCONT信号を受信するまで、このシェルの実行をサスペンドします。-fを指定すると、これがログインシェルであるとの主張を行わず、サスペンドのみ行います。
test	test [expr] *EXPR*の評価に応じて、0 (true)または1 (false)のステータスで終了します。式は単項または二項のいずれかです。単項式は一般的にファイルのステータスを検査するために使用されます。文字列演算子や数値比較演算子もあります。
time	time [-p] PIPELINE

コマンド	構文/説明
	*PIPELINE*を実行し、*PIPELINE*実行時にリアルタイムの概要、実行に要したユーザCPU時間、システムのCPU時間を印刷します。戻りステータスは、PIPELINEの戻りステータスです。-pオプションは、若干異なるフォーマットでタイミング概要を印刷します。TIMEFORMAT変数値を出力フォーマットとして使用します。
times	`times` シェルからプロセスを実行する際のユーザおよびシステム時間の合計を印刷します。
trap	`trap [-lp] [ARG SIGNAL_SPEC ...]` シェルが*SIGNAL_SPEC*信号を受信すると、コマンド*ARG*を読み込み、実行します。*ARG*がない場合(そして単一の*SIGNAL_SPEC*が提供された場合)、または-の場合、指定された各信号は元の値にリセットされます。*ARG*がnull文字列の場合、各*SIGNAL_SPEC*はシェルや、それが呼びだすコマンドに無視されます。*SIGNAL_SPEC*がEXIT (0)の場合、シェルからの終了時、コマンドARGが実行されます。*SIGNAL_SPEC*がDEBUGの場合、簡単な各コマンドの後で*ARG*が実行されます。-pオプションが提供される場合、各*SIGNAL_SPEC*に関連付けられたtrapコマンドが表示されます。引数が指定されていない場合、または-pが指定されている場合、trapは各信号に関連付けられたコマンドのリストを印刷します。各*SIGNAL_SPEC*は`signal.h`内の信号名、または信号番号です。信号名は大文字、小文字を区別せず、SIGプレフィックスはオプションです。`trap -l`は信号名と対応する番号のリストを印刷します。`kill -signal $$`によって信号をシェルに送信できる点に注意してください。
true	`true` 結果として成功を返します。

コマンド	構文/説明
type	`type [-afptP] NAME [NAME ...]` 現在は使用されていません。詳細はdeclareを参照してください。
typeset	`typeset [-afFirtx] [-p] name[=value]` 現在は使用されていません。詳細はdeclareを参照してください。
ulimit	`ulimit [-SHacdfilmnpqstuvx] [limit` Ulimitはその種の制御が可能なシステムで、シェルによって開始されたプロセスで使用可能なリソースを制御します。
umask	`umask [-p] [-S] [MODE]` ユーザファイル作成マスクは*MODE*に設定されます。*MODE*を省略すると、または-Sを提供すると、マスクの現在値が印刷されます。-Sオプションはシンボルとして出力します。それ以外の場合は、8進数が出力されます。-pを提供し、*MODE*を省略すると、入力として使用可能な形式で出力が行われます。*MODE*が数字から始まる場合は、8進数として解釈されます。それ以外の場合は、chmod(1)で受け入れられるようなシンボル文字列モードになります。
unalias	`unalias [-a] NAME [NAME ...]` 定義済みの別名リストから*NAME*を削除します。-aオプションを指定すると、すべての別名定義が削除されます。
unset	`unset [-f] [-v] [NAME ...]` *NAME*ごとに、対応する変数または関数が削除されます。-vフラグを指定すると、変数だけにunsetが機能します。-fフラグを指定すると、関数だけにunsetが機能します。いずれのフラグも指定されない場合は、unsetは最初に変数に対して実行を試み、失敗すると、関数に対して実行を試みます。変数の中に

コマンド	構文/説明
	はunsetを行えないものがあります。readonlyも参照してください。
until	`until COMMANDS; do COMMANDS; done` until *COMMANDS*の最後のコマンドがゼロ以外の終了ステータスであれば、*COMMANDS*を展開して実行します。
wait	`wait [N]` 指定されたプロセスを待機し、終了ステータスを報告します。*N*を指定しなかった場合、現在アクティブなすべての子プロセスを待機して、ゼロの戻りコードが返されます。*N*にはプロセスIDまたはジョブ仕様を指定できます。ジョブ仕様を指定した場合、ジョブパイプライン内のすべてのプロセスを待機します。
while	`while COMMANDS; do COMMANDS; done` while *COMMANDS*の最後のコマンドがゼロ以外の終了ステータスであれば、*COMMANDS*を展開して実行します。

crm_standby (8)

crm_standby — ノードのスタンバイ属性を操作して、このノードでリソースを実行できるかどうかを判断します

書式

```
crm_standby [-?|-V] -D -u|-U node -r resource

crm_standby  [-?|-V] -G -u|-U node -r resource

crm_standby [-?|-V] -v string -u|-U node -r resource [-l string]
```

説明

crm_standbyコマンドはノードのスタンバイ属性を操作します。スタンバイモードのノードは、リソースのホスティングが行えなくなり、そのノード上にあるリソースは移動が必要になります。スタンバイモードはカーネルの更新などの保守作業を行う場合に便利です。再度、クラスタの完全にアクティブなメンバーになる必要が発生したら、ノードからスタンバイ属性を削除します。

standby属性に有効期間を設定することで、スタンバイ設定がノードの再起動後も有効なのか(有効期間を「forever」に設定)、再起動後はリセットするのか(有効期間を「reboot」に設定)を指定します。あるいは、standby属性を削除して、ノードをスタンバイモードから手動で復帰させることもできます。

オプション

--help, -?
　ヘルプメッセージを印刷します。

--verbose, -V
　デバッグ情報を有効にします。

> **注意**
>
> 追加のインスタンスを提供すると、詳細になります。

`--quiet`, `-Q`
: `-G`を使用して属性クエリを行う場合、値だけをstdoutに印刷します。このオプションは`-G`と一緒に使用します。

`--get-value`, `-G`
: 初期設定を設定するのではなく、取得します。

`--delete-attr`, `-D`
: 削除する属性を指定します。

`--attr-value` *string*, `-v` *string*
: 使用する値を指定します。このオプションは`-G`と一緒に使用すると無視されます。

`--attr-id` *string*, `-i` *string*
: 上級ユーザ専用です。id属性を識別します。

`--node` *node_uname*, `-u` *node_uname*
: 変更するノードのunameを指定します。

`--lifetime` *string*, `-l` *string*
: この初期設定を維持する期間を決定します。`reboot`または`forever`の値を指定できます。

> **注意**
>
> `forever`の値がある場合は、CRMは常に`reboot`の値の代わりに使用します。

例

ローカルノードをスタンバイモードにします。

```
crm_standby -v true
```

ノード(node1)をスタンバイモードにします。

```
crm_standby -v true -U node1
```

ノードのスタンバイステータスのクエリを行います。

```
crm_standby -G -U node1
```

ノードのスタンバイプロパティを削除します。

```
crm_standby -D -U node1
```

ノードを永続的にスタンバイモードにします。

```
crm_standby -v true -l forever -U node1
```

次回このノードを再起動するまで、ノードをスタンバイモードにします。

```
crm_standby -v true -l reboot -U node1
```

ファイル

/var/lib/heartbeat/crm/cib.xml—ディスク上のCIB(ステータスセクションを除く)。このファイルを直接編集しないでください。

参照

cibadmin(8) (146 ページ)、crm_attribute(8) (156 ページ)

著者

crm_standbyはAndrew Beekhofによって作成されました。

crm_verify (8)

crm_verify — CIBの整合性を確認します

書式

```
crm_verify [-V] -x file
crm_verify [-V] -X string
crm_verify [-V] -L|-p
crm_verify [-?]
```

説明

crm_verifyは環境設定データベース(CIB)の整合性や、その他の問題がないか確認します。環境設定が含まれているファイルを確認するために使用したり、実行中のクラスタに接続することもできます。エラーと警報という2つの問題クラスを報告します。エラーは修正しなければHeartbeatが正常に動作しません。ただし、警報のあった点も修正するかどうかは管理者の判断に委ねられます。

crm_verifyは環境設定の新規作成または変更に役立ちます。実行中のクラスタのCIBのローカルコピーを作成して、crm_verifyを使用して検証し、cibadminを使用して新しい環境設定を有効にすることができます。

オプション

--help, -h
: ヘルプメッセージを印刷します。

--verbose, -V
: デバッグ情報を有効にします。

> **注意**
>
> 追加のインスタンスを提供すると、詳細になります。

--live-check, -L
実行中のクラスタに接続してCIBを確認します。

--crm_xml *string*, -X *string*
提供された文字列を使用して環境設定を確認します。完全なCIBだけを渡します。

--xml-file *file*, -x *file*
指定されたファイルの環境設定を確認します。

--xml-pipe, -p
stdinパイプ経由で提供された環境設定を使用します。完全なCIBだけを渡します。

例

実行中のクラスタの環境設定の整合性を確認し、詳細な出力を提供します。

```
crm_verify -VL
```

指定されたファイルの環境設定の整合性を確認し、詳細な出力を提供します。

```
crm_verify -Vx file1
```

crm_verifyに環境設定をパイプ経由で渡し、詳細な出力を提供します。

```
cat file1.xml | crm_verify -Vp
```

ファイル

/var/lib/heartbeat/crm/cib.xml—ディスク上のCIB(ステータスセクションを除く)。このファイルを直接編集しないでください。

参照

cibadmin(8) (146 ページ)

著者

`crm_verify`はAndrew Beekhofによって作成されました。

17 クラスタリソース

この章では、クラスタリソースに関する最も重要な事実と数値について概要を述べます。リソースエージェントクラスHigh Availability Extensionのサポート、OCFリソースエージェントのエラーコードおよびエラーコードに対するクラスタの応答、使用できるリソースオプション、リソース操作とインスタンス属性について述べます。リソースを構成する際に、この概要を参照してください(`crm`ラインツールから手動で、またはLinux HA Management Clientを使用します)。

17.1 サポートされるリソースエージェントクラス

追加したそれぞれのクラスタリソースについて、リソースエージェントが従う標準を定義する必要があります。リソースエージェントは提供するサービスを抽象化し、正確な状態をクラスタに提供し、これによってクラスタは管理するリソースを認識しなくてすみます。クラスタは、start、stop、monitorコマンドを与えられた場合、リソースエージェントによって適切な処理を実施します。

通常、リソースエージェントはシェルスクリプトの形式をとります。High Availability Extensionは次のリソースエージェントクラスをサポートします。

従来のHeartbeat 1リソースエージェント
: Heartbeatバージョン1にいは独自のスタイルのリソースエージェントが付属していました。多くのユーザが独自のエージェントをその表記方法に

従って開発したため、このリソースエージェントはまだサポートされています。ただし、可能な場合は構成をHigh Availability OCF RAにマイグレートすることを推奨します。詳細については、http://wiki.linux-ha.org/HeartbeatResourceAgentを参照してください。

Linux Standards Base (LSB)スクリプト
LSBリソースエージェントは一般にオペレーティングシステム/ディストリビューションによって提供され、/etc/init.dにあります。クラスタで使用するには、LSB仕様に準拠する必要があります。たとえば、いくつかのアクションを実装する必要があり、少なくともstart、stop、restart、reload、force-reload、statusをhttp://www.linuxbase.org/spec/refspecs/LSB_3.0.0/LSB-Core-generic/LSB-Core-generic/iniscrptact.htmlで示すように実装する必要があります。

Open Cluster Framework (OCF)リソースエージェント
OCF RAエージェントはHigh Availabilityとの使用に最適で、特にマスタリソースまたは特殊な監視機能を必要とする場合に適しています。エージェントは一般に/usr/lib/ocf/resource.d/heartbeat/にあります。この機能はLSBスクリプトの機能と同様です。ただし、構成は常に環境変数によって実行され、これによってパラメータを簡単に受け渡して処理できます。OCF仕様はhttp://www.opencf.org/cgi-bin/viewcvs.cgi/specs/ra/resource-agent-api.txt?rev=HEAD 4で参照できます(リソースエージェントに関連するため)。OCF仕様ではアクションによってどの出口コードが返されるか、厳密な定義があります。クラスタはこの仕様に正確に従います。詳細については、http://wiki.linux-ha.org/OCFResourceAgentを参照してください。使用できるすべてのOCF RAの詳細なリストは、第18章 *HA OCFエージェント* (209 ページ)を参照してください。

STONITHリソースエージェント
このクラスは、フェンシング関係のリソース専用に使用されます。詳細については、第8章 フェンシングと*STONITH* (83 ページ)を参照してください。

High Availability Extensionで指定されたエージェントはOCF仕様に従って作成されています。

17.2 OCF戻りコード

OCF仕様によると、アクションが返す必要がある出口コードの厳密な定義があります。クラスタは常に、予期される結果に対する戻りコードを確認します。結果が予期された値と一致しない場合、アクションは失敗したとみなされ、回復処理が開始されます。障害回復には3種類あります。

表 17.1　障害回復の種類

回復の種類	説明	クラスタが行うアクション
soft	一時的なエラーが発生しました。	リソースを再起動するか、新しい場所に移動させます。
hard	一時的ではないエラーが発生しました。エラーは現在のノードに固有の場合があります。	リソースを他の場所に移動して、現在のノードで再試行されないようにします。
致命的	すべてのクラスタノードに共通の、一時的ではないエラーが発生しました。これは不正な構成が指定されたことを示しています。	リソースを停止して、どのクラスタノードでも開始されないようにします。

アクションが失敗したと仮定して、次の表では、エラーコードを受け取った場合の異なるOCF戻りコードとクラスタが開始する回復の種類を概説します。

表 17.2　OCF戻りコード

OCF戻りコード	OCFエイリアス	説明	回復の種類
0	OCF_SUCCESS	成功. コマンドは正常に完了しました。これは、すべてのstart、stop、	soft

OCF戻りコード	OCFエイリアス	説明	回復の種類
		promote、demoteコマンドの予期された結果です。	
1	OCF_ERR_GENERIC	汎用の「問題が発生した」ことを示すエラーコード。	soft
2	OCF_ERR_ARGS	リソースの構成がこのマシンで有効ではありません(たとえば、ノード上に見つからない場所/ツールを参照している場合)。	hard
3	OCF_ERR_UNIMPLEMENTED	要求されたアクションは実行されていません。	hard
4	OCF_ERR_PERM	リソースエージェントにはタスクを完了する十分な権限がありません。	hard
5	OCF_ERR_INSTALLED	リソースが必要とするツールがこのマシンにインストールされていません。	hard
6	OCF_ERR_CONFIGURED	リソースの構成が無効です(たとえば、必要なパラメータが不足している場合)。	致命的
7	OCF_NOT_RUNNING	リソースが実行されていません。クラスタは、どのアクションについてもこれを返すリソースを停止しようとしません。	該当なし

OCF戻りコード	OCFエイリアス	説明	回復の種類
		このOCF戻りコードはリソース回復を必要することも必要としないこともあります。予期されたリソースの状態に依存します。予期されない場合は、soft回復を行います。	
8	OCF_RUNNING_MASTER	リソースはマスタモードで実行しています。	soft
9	OCF_FAILED_MASTER	リソースはマスタモードですが、失敗しました。リソースは降格、停止され、再度開始されます(昇格されます)。	soft
その他	該当なし	カスタムエラーコード。	soft

17.3 リソースオプション

追加した各リソースについて、オプションを定義できます。クラスタはオプションを使用して、リソースの動作方法を決定します。CRMに特定のリソースの処理方法を通知します。リソースオプションは、`crm_resource --meta`コマンドまたはGUIを使用して設定できます。

表 17.3　プリミティブリソースのオプション

オプション	説明
`priority`	一部のリソースをアクティブにできない場合、クラスタは優先度の低いリソースを停止して、

オプション	説明
	優先度の高いリソースをアクティブに維持します。
target-role	クラスタがこのリソースで維持する状態。使用できる値: Stopped、Started
is-managed	クラスタがリソースを開始して停止できるかどうか。使用できる値:true、false
resource-stickiness	リソースが現在の状態をどの程度維持したいか。デフォルトはdefault-resource-stickinessの値。
migration-threshold	ノードがこのリソースをホストできなくなるまで、このリソースについてノード上で発生する失敗の回数。デフォルト: none
multiple-active	複数のノードでリソースがアクティブになっていることが検出された場合のクラスタの動作。使用できる値: block (リソースを管理されていないとマークする)、stop_only、stop_start
failure-timeout	失敗が発生していないように動作する(リソースを失敗したノードに戻す)前に、待機する秒数デフォルト: never

17.4 リソース操作

デフォルトで、クラスタはリソースが良好な状態であることを保証しません。クラスタにこれを指示するには、リソースの定義に監視操作を追加する必要があります。監視操作は、すべてのクラスまたはリソースエージェントに追加できます。

表 17.4　リソース操作

説明	説明
id	アクションに指定する名前。一意にする必要があります。
name	実行するアクション。共通の値:monitor、start、stop
interval	操作を実行する頻度。単位: 秒
timeout	アクションが失敗したと宣言する前に待機する長さ。
requires	このアクションが発生する前に満たす必要がある条件。使用できる値:nothing、quorum、fencingデフォルトは、フェンシングが有効でリソースのクラスがstonithかどうかによります。STONITHリソースの場合、デフォルトはnothingです。
on-fail	このアクションが失敗した場合に実行するアクション。使用できる値: • ignore: リソースが失敗しなかったのように動作します。 • block: リソースにこれ以上の操作を実行しません。 • stop: リソースを停止して、他の場所でも開始しません。 • restart: リソースを停止して再起動します(別のノード上で)。 • fence: リソースが失敗したノードを停止します(STONITH)。 • stanby:リソースが失敗したノードからすべてのリソースを移動させます。

説明	説明
enabled	falseの場合、操作は存在していない場合と同様に処理されます。使用できる値:true、false

17.5 インスタンス属性

すべてのリソースクラスのスクリプトでは、動作方法および管理するサービスのインスタンスを指定するパラメータを提供します。リソースエージェントがパラメータをサポートする場合、このパラメータをcrm_resourceコマンドで追加できます。crmコマンドラインユーティリティで、インスタンス属性はparamsと呼ばれます。OCFスクリプトでサポートされているインスタンス属性のリストは、次のコマンドをrootとして実行すると参照できます。

```
crm ra meta resource_agent class
```

例

```
crm ra meta Ipaddr ocf heartbeat
```

次の結果が得られます。

18 HA OCFエージェント

すべてのOCFエージェントでは、起動時に複数のパラメータを設定しなければなりません。次の概要では、これらのエージェントを手動で操作する方法を説明します。この付録で提供されているデータは、該当するRAのmeta-dataの呼び出しによって直接取得することができます。これらの全エージェントは、/usr/lib/ocf/resource.d/heartbeat/に提供されています。

RAの設定時、パラメータ名のOCF_RESKEY_プレフィックスを省略します。角括弧内のパラメータは、環境設定では省略できます。

ocf:anything_ra (7)

ocf:anything_ra — 任意

書式

OCF_RESKEY_binfile=string [OCF_RESKEY_cmdline_options=string]
[OCF_RESKEY_pidfile=string] [OCF_RESKEY_logfile=string]
[OCF_RESKEY_errlogfile=string] [OCF_RESKEY_user=string]
[OCF_RESKEY_monitor_hook=string] anything_ra [start | stop | monitor |
meta-data | validate-all]

説明

これはほとんどすべてのものを管理する汎用OCF RAです。

サポートされているパラメータ

OCF_RESKEY_binfile=実行するバイナリのフルパス名
　実行するバイナリの完全な名です。これは単に何かを実行して終了するのではなく、同じpidで実行を継続するためのものです。

OCF_RESKEY_cmdline_options=コマンドラインオプション
　バイナリに渡すコマンドラインオプション

OCF_RESKEY_pidfile=STDOUTを書き込むファイル
　PIDの読み書きを行うファイル。

OCF_RESKEY_logfile=STDOUTを書き込むファイル
　STDOUTを書き込むファイル

OCF_RESKEY_errlogfile=STDERRを書き込むファイル
　STDERRを書き込むファイル

OCF_RESKEY_user=コマンドを実行するユーザ
　コマンドを実行するユーザ

OCF_RESKEY_monitor_hook=監視操作で実行するコマンド
　監視操作で実行するコマンド

ocf:apache (7)

ocf:apache — Apache Webサーバ

書式

OCF_RESKEY_configfile=string [OCF_RESKEY_httpd=string]
[OCF_RESKEY_port=integer] [OCF_RESKEY_statusurl=string]
[OCF_RESKEY_testregex=string] [OCF_RESKEY_options=string]
[OCF_RESKEY_envfiles=string] apache [start | stop | status | monitor | meta-data
| validate-all]

説明

これは、Apache Webサーバのリソースエージェントです。このリソースエージェントは、バージョン1.xとバージョン2.xの両方のApacheサーバで稼動します。サーバが起動し、動作していることを確認するために監視が繰り返し呼び出されるループで開始操作は終了します。したがって、監視操作が開始操作タイムアウト制限時間内に成功しなかった場合は、Apacheリソースはエラー状態で終了します。デフォルトでは、監視操作によってサーバステータスページがロードされます。これはmod_statusモジュールと対応する環境設定ファイル(通常は /etc/apache2/mod_status.conf)によって異なります。サーバステータスページが機能し、ローカルホスト(アドレスは127.0.0.1)から*のみ*アクセスが許可されることを確認します。詳細はstatusurlおよびtestregex属性を参照してください。http://httpd.apache.org/も参照してください。

サポートされているパラメータ

OCF_RESKEY_configfile=環境設定ファイルのパス
Apache環境設定ファイルのフルパス名。このファイルは、その他各種のリソースエージェントパラメータのデフォルトを提供するために解析されます。

OCF_RESKEY_httpd=httpdバイナリパス
httpdバイナリのフルパス名(オプション)。

OCF_RESKEY_port=httpdポート
: statusurlを使用してステータス情報を検索できるポート番号。環境設定ファイルに記載されたポート番号がデフォルトになります。環境設定ファイルにポート番号の記載がない場合は80です。

OCF_RESKEY_statusurl=url名
: 監視するURL(デフォルトはapacheサーバステータスページ)。指定しなかった場合は、apacheの環境設定ファイルから採用されます。設定した場合は、ローカルホスト(127.0.0.1)から*のみ*成功することを確認します。もしそうでなければ、複数のノードでリソースがアクティブであるというエラーがクラスタから報告されます。

OCF_RESKEY_testregex=正規表現の監視
: statusurlの出力と一致する正規表現。大文字、小文字が区別されます。

OCF_RESKEY_options=コマンドラインオプション
: apacheの起動時に適用する追加のオプション。httpd(8)を参照してください。

OCF_RESKEY_envfiles=環境設定ファイル
: /etc/apache2/envvarsなどの追加の環境変数を含むファイル(1つ以上)。

ocf:AudibleAlarm (7)

ocf:AudibleAlarm — AudibleAlarmリソースエージェント

書式

[OCF_RESKEY_nodelist=string] AudibleAlarm [start | stop | restart | status | monitor | meta-data | validate-all]

説明

AudibleAlarmのリソーススクリプト。特定の間隔で警告音を出す可聴警報を設定します。

サポートされているパラメータ

OCF_RESKEY_nodelist=ノードリスト
　警報は発行しないノードのリスト。

ocf:ClusterMon (7)

ocf:ClusterMon — ClusterMonリソースエージェント

書式

[OCF_RESKEY_user=string] [OCF_RESKEY_update=integer]
[OCF_RESKEY_extra_options=string] OCF_RESKEY_pidfile=string
OCF_RESKEY_htmlfile=string ClusterMon [start | stop | monitor | meta-data | validate-all]

説明

これはClusterMonリソースエージェントです。現在のクラスタステータスをhtmlに出力します。

サポートされているパラメータ

OCF_RESKEY_user=crm_monを実行するユーザ
crm_monを実行するユーザ

OCF_RESKEY_update=更新間隔
クラスタステータスを更新する頻度

OCF_RESKEY_extra_options=追加のオプション
crm_monに渡す追加のオプション。例-n -r

OCF_RESKEY_pidfile=PIDファイル
1つのインスタンスだけが実行中であることを確認するためのPIDファイルの場所

OCF_RESKEY_htmlfile=HTML出力
HTML出力の書き込み先。

ocf:db2 (7)

ocf:db2 — db2リソースエージェント

書式

[OCF_RESKEY_instance=string] [OCF_RESKEY_admin=string] db2 [start | stop | status | monitor | validate-all | meta-data | methods]

説明

db2のリソーススクリプト。DB2 Universal Databaseインスタンスを HAリソースとして管理します。

サポートされているパラメータ

OCF_RESKEY_instance=インスタンス
　データベースのインスタンス。

OCF_RESKEY_admin=管理者
　インスタンスの管理者ユーザ。

ocf:Delay (7)

ocf:Delay — Delayリソースエージェント。

書式

[OCF_RESKEY_startdelay=integer] [OCF_RESKEY_stopdelay=integer] [OCF_RESKEY_mondelay=integer] Delay [start | stop | status | monitor | meta-data | validate-all]

説明

このスクリプトは遅延を発生させるためのテストリソースです。

サポートされているパラメータ

OCF_RESKEY_startdelay=開始遅延
開始操作で遅延させる秒数。

OCF_RESKEY_stopdelay=停止遅延
停止操作で遅延させる秒数。指定されなければ「startdelay」にデフォルト設定します。

OCF_RESKEY_mondelay=監視遅延
監視操作で遅延させる秒数。指定されなければ「startdelay」にデフォルト設定します。

ocf:drbd (7)

ocf:drbd— このリソースエージェントはマスタ/スレーブリソースとしてDRBD (Distributed Replicated Block Device)オブジェクトを管理します。DRBDはストレージの複製用メカニズムです。セットアップの詳細はドキュメントを参照してください。

書式

OCF_RESKEY_drbd_resource=string [OCF_RESKEY_drbdconf=string]
[OCF_RESKEY_clone_overrides_hostname=boolean]
[OCF_RESKEY_clone_max=integer] [OCF_RESKEY_clone_node_max=integer]
[OCF_RESKEY_master_max=integer]
[OCF_RESKEY_master_node_max=integer] drbd [start | promote | demote | notify | stop | monitor | monitor | meta-data | validate-all]

説明

DRBDのMaster/Slave OCFリソースエージェント

サポートされているパラメータ

OCF_RESKEY_drbd_resource=drbdリソース名
drbd.confファイルのdrbdリソース名。

OCF_RESKEY_drbdconf=drbd.confへのパス
drbd.confファイルへのフルパス。

OCF_RESKEY_clone_overrides_hostname=drbdホスト名の上書き
クローン番号でホスト名に上書きするかどうかを指定します。これを使用してフローティングピアの環境設定を作成することができます。drbdは実際の名前の代わりにnode_<cloneno>をホスト名として使用するように指示され、その後drbd.confで使用できるようになります。

OCF_RESKEY_clone_max=クローン数
　このdrbdリソースのクローン数。デフォルトは変更しないでください。

OCF_RESKEY_clone_node_max=ノード数
　1ノードあたりのクローン数。デフォルトは変更しないでください。

OCF_RESKEY_master_max=プライマリ数
　アクティブプライマリの最大数。デフォルトは変更しないでください。

OCF_RESKEY_master_node_max=1ノードあたりのプライマリ数
　1ノードあたりのプライマリ数。デフォルトは変更しないでください。

ocf:Dummy (7)

ocf:Dummy — Dummyリソースエージェント

書式

OCF_RESKEY_state=string Dummy [start | stop | monitor | reload | migrate_to | migrate_from | meta-data | validate-all]

説明

これはDummyリソースエージェントです。実行中かどうかを追跡する以外にはまったく何もしません。テストを目的として、RAライタのテンプレートとして機能します。

サポートされているパラメータ

OCF_RESKEY_state=状態ファイル
　リソース状態を保存する場所。

ocf:eDir88 (7)

ocf:eDir88 — eDirectoryリソースエージェント

書式

OCF_RESKEY_eDir_config_file=string
[OCF_RESKEY_eDir_monitor_ldap=boolean]
[OCF_RESKEY_eDir_monitor_idm=boolean]
[OCF_RESKEY_eDir_jvm_initial_heap=integer]
[OCF_RESKEY_eDir_jvm_max_heap=integer]
[OCF_RESKEY_eDir_jvm_options=string] eDir88 [start | stop | monitor | meta-data | validate-all]

説明

eDirectoryインスタンスを管理するためのリソーススクリプト。eDirectoryの単一インスタンスをHAリソースとっして管理します。「複数インスタンス」機能またはeDirectoryはバージョン8.8で追加されました。このスクリプトはバージョン8.8以前のeDirectoryでは機能しません。このRAを使用して同じホストに複数のeDirectoryインスタンスをロードすることができます。eDir環境設定ファイル(eDir_config_fileパラメータで)を各ノードのローカルストレージに配置しておくことを強く推奨します。これは、共有ストレージが使用できなくなった場合に、このRAが状況を処理できるようにしておくために必要です。eDir環境設定ファイルが使用できなくなると、RAは機能しなくなり、ハートビートがリソースの管理を行えなくなります。これによる二次的な影響として、STONITHアクション、リソースの管理が行われなくなるといった状況が発生します。アクションタイムアウト値を非常に高い値に設定しておくことを特に強く推奨します。IDMでのeDirは開始までに10分以上かかることがあります。eDirが正常に開始する前にハートビートがタイムアウトしてしまうと、障害が発生します。LDAPモジュールは開始に非常に時間がかかるようです。つまり、このスクリプトは、IDMやLDAPの監視が有効になっている場合、開始コマンドはIDMとLDAPが使用できるようになるまで待機するため、IDMやLDAPでインストールを開始するのに、さらに時間がかかります。

サポートされているパラメータ

OCF_RESKEY_eDir_config_file=eDir環境設定ファイル
eDirectoryインスタンスの環境設定ファイルへのパス。

OCF_RESKEY_eDir_monitor_ldap=eDir用のldap監視
LDAPがeDirectoryインスタンス用に実行されている場合、監視するかどうかを指定します。

OCF_RESKEY_eDir_monitor_idm=eDir用のIDM監視
IDMがeDirectoryインスタンス用に実行されている場合、監視するかどうかを指定します。

OCF_RESKEY_eDir_jvm_initial_heap=DHOST_INITIAL_HEAP値
DHOST_INITIAL_HEAP java環境変数の値。設定されていない場合は、javaのデフォルトが使用されます。

OCF_RESKEY_eDir_jvm_max_heap=DHOST_MAX_HEAP値
DHOST_MAX_HEAP java環境変数の値。設定されていない場合は、javaのデフォルトが使用されます。

OCF_RESKEY_eDir_jvm_options=DHOST_OPTIONS値
DHOST_OPTIONS java環境変数の値。設定されていない場合は、元の値が使用されます。

ocf:Filesystem (7)

ocf:Filesystem — Filesystemリソースエージェント

書式

[OCF_RESKEY_device=string] [OCF_RESKEY_directory=string]
[OCF_RESKEY_fstype=string] [OCF_RESKEY_options=string] Filesystem
[start | stop | notify | monitor | validate-all | meta-data]

説明

Filesystemのリソーススクリプト。共有ストレージメディア上のFilesystemを管理します。

サポートされているパラメータ

OCF_RESKEY_device=ブロックデバイス
ファイルシステムのブロックデバイス名、またはマウント用の-U、-Lオプション、またはNFSマウント指定。

OCF_RESKEY_directory=マウントポイント
ファイルシステムのマウントポイント。

OCF_RESKEY_fstype=ファイルシステムタイプ
マウントするファイルシステムのオプションタイプ。

OCF_RESKEY_options=オプション
マウントする-oオプションとして渡される追加のオプション。バインドマウントの場合は、ここで「bind」を追加して、fstypeを「none」に設定します。「bind,ro」などのオプションに対して有効です。

ocf:ICP (7)

ocf:ICP — ICPリソースエージェント

書式

[OCF_RESKEY_driveid=string] [OCF_RESKEY_device=string] ICP [start | stop | status | monitor | validate-all | meta-data]

説明

ICPのリソーススクリプト。ICP Vortexクラスタホストドライブを HAリソースとして管理します。

サポートされているパラメータ

OCF_RESKEY_driveid=ICPクラスタドライブID
　ICPクラスタドライブのID。

OCF_RESKEY_device=デバイス
　デバイス名。

ocf:ids (7)

ocf:ids — IDS (Informix Dynamic Server)と呼ばれるIBMデータベースサーバのOCFリソースエージェントです。

書式

[OCF_RESKEY_informixdir=string] [OCF_RESKEY_informixserver=string] [OCF_RESKEY_onconfig=string] [OCF_RESKEY_dbname=string] [OCF_RESKEY_sqltestquery=string] ids [start | stop | status | monitor | validate-all | meta-data | methods | usage]

説明

IBM IDS (Informix Dynamic Server)インスタンスを高可用性リソースとして管理するためのOCFリソースエージェントです。

サポートされているパラメータ

OCF_RESKEY_informixdir= INFORMIXDIR環境設定変数
環境変数INFORMIXDIRが、IDSの一般的なインストール後に持つ値です。すなわち、IDSのインストール先パス(後続の「/」を除く)です。このパラメータが指定されていない場合、スクリプトはシェル環境から値を入手しようとします。

OCF_RESKEY_informixserver= INFORMIXSERVER環境設定変数
環境変数INFORMIXSERVERが、IDSの一般的なインストール後に持つ値です。すなわち、管理対象のIDSサーバインスタンス名です。このパラメータが指定されていない場合、スクリプトはシェル環境から値を入手しようとします。

OCF_RESKEY_onconfig= ONCONFIG環境設定変数
環境変数ONCONFIGが、IDSの一般的なインストール後に持つ値です。すなわち、INFORMIXSERVERで指定したIDSインスタンスの環境設定ファイル名です。指定された環境設定ファイルは、「/etc/」で検索されます。

このパラメータが指定されていない場合、スクリプトはシェル環境から値を入手しようとします。

OCF_RESKEY_dbname=監視に使用するデータベースでデフォルトは「sysmaster」

このパラメータはIDSインスタンスの監視に使用するデータベースを定義します。このパラメータが指定されていない場合、スクリプトはデフォルトで「sysmaster」データベースを使用します。

OCF_RESKEY_sqltestquery=監視に使用するSQLテストクエリで、デフォルトは「SELECT COUNT(*) FROM systables;」

パラメータ「dbname」で指定されたデータベースに対して、IDSインスタンスの監視を行い機能しているかどうかを判断するために実行するSQLテストクエリ。このパラメータが指定されていない場合、スクリプトはデフォルトで「SELECT COUNT(*) FROM systables」を使用します。

ocf:IPaddr2 (7)

ocf:IPaddr2 — 仮想IPv4アドレスを管理します

書式

OCF_RESKEY_ip=string [OCF_RESKEY_nic=string]
[OCF_RESKEY_cidr_netmask=string] [OCF_RESKEY_broadcast=string]
[OCF_RESKEY_iflabel=string] [OCF_RESKEY_lvs_support=boolean]
[OCF_RESKEY_mac=string] [OCF_RESKEY_clusterip_hash=string]
[OCF_RESKEY_unique_clone_address=boolean]
[OCF_RESKEY_arp_interval=integer] [OCF_RESKEY_arp_count=integer]
[OCF_RESKEY_arp_bg=string] [OCF_RESKEY_arp_mac=string] IPaddr2 [start
| stop | status | monitor | meta-data | validate-all]

説明

このLinux特有のリソースは、IPの別名IPアドレスを管理します。IPの別名を追加または削除できます。さらに、クローンリソースとして起動した場合は、Cluster Alias IP機能を実装できます。

サポートされているパラメータ

OCF_RESKEY_ip=IPv4アドレス
: たとえば「192.168.1.1」のようにドットで区切った4つの数字桁で設定したIPv4アドレス。

OCF_RESKEY_nic=ネットワークインタフェース
: IPアドレスをオンラインにする基本のネットワークインタフェース。空の場合、スクリプトはルーティングテーブルからこれを判別しようとします。ここではeth0:1やその他の形式で別名インタフェースを指定せずに、基本インタフェースだけを指定してください。

OCF_RESKEY_cidr_netmask=CIDRネットマスク
CIDR形式(たとえば255.255.255.0ではなく24)のインタフェース用のネットマスク。指定されていない場合、スクリプトはこれもルーティングテーブルから判別しようとします。

OCF_RESKEY_broadcast=ブロードキャストアドレス
IPに関連付けられているブロードキャストアドレス。空の場合、スクリプトはネットマスクからこれを判別します。

OCF_RESKEY_iflabel=インタフェースラベル
ここでIPアドレスの追加ラベルを指定できます。このラベルはインタフェース名に追加されます。ラベルがNIC名で指定されている場合、このパラメータは無効です。

OCF_RESKEY_lvs_support=LVS DRのサポートを有効化
LVSダイレクトルーティング環境設定のサポートを可能にします。IPアドレスが停止した場合、ローカルノードがサービス要求に応えられるようにループバックデバイスへの移動のみを行い、ネットワーク上でのアドバタイズは行わないようにします。

OCF_RESKEY_mac=クラスタIP MACアドレス
インタフェースMACアドレスを明示的に設定します。現在はクラスタIP別名の場合にのみ使用されています。空の場合、自動的に選択します。

OCF_RESKEY_clusterip_hash=クラスタIPハッシュ関数
クラスタIP機能に使用されるハッシュアルゴリズムを指定します。

OCF_RESKEY_unique_clone_address=クローンインスタンスに固有のアドレスを作成
trueの場合は、クローンIDをipに指定された値に追加して、管理対象に固有のアドレスを作成します。

OCF_RESKEY_arp_interval=ARPパケット間隔(ミリ秒)
未承認ARPパケットの間隔をミリ秒で指定します。

OCF_RESKEY_arp_count=ARPパケット数
送信する未承認ARPパケット数。

OCF_RESKEY_arp_bg=バックグラウンドからのARP
バックグラウンドでARPパケットを送信するかどうかを指定します。

OCF_RESKEY_arp_mac=ARP MAC
ARPパケットの送信先MACアドレスです。これは変更してはなりません。

ocf:IPaddr (7)

ocf:IPaddr — 仮想IPv4アドレスを管理します

書式

OCF_RESKEY_ip=string [OCF_RESKEY_nic=string]
[OCF_RESKEY_cidr_netmask=string] [OCF_RESKEY_broadcast=string]
[OCF_RESKEY_iflabel=string] [OCF_RESKEY_lvs_support=boolean]
[OCF_RESKEY_local_stop_script=string]
[OCF_RESKEY_local_start_script=string]
[OCF_RESKEY_ARP_INTERVAL_MS=integer]
[OCF_RESKEY_ARP_REPEAT=integer]
[OCF_RESKEY_ARP_BACKGROUND=boolean]
[OCF_RESKEY_ARP_NETMASK=string] IPaddr [start | stop | monitor | validate-all | meta-data]

説明

このスクリプトはIPの別名IPアドレスを管理します。IP別名を追加または削除できます。

サポートされているパラメータ

OCF_RESKEY_ip=IPv4アドレス
　たとえば「192.168.1.1」のようにドットで区切った4つの数字桁で設定したIPv4アドレス。

OCF_RESKEY_nic=Network interface
　IPアドレスをオンラインにする基本のネットワークインタフェース。空の場合、スクリプトはルーティングテーブルからこれを判別しようとします。ここではeth0:1やその他の形式で別名インタフェースを指定せずに、基本インタフェースだけを指定してください。

OCF_RESKEY_cidr_netmask=ネットマスク
CIDR形式のインタフェースのネットマスクです。(すなわち24)またはドットで区切った255.255.255.0の形式)。指定されていない場合は、スクリプトはこれもルーティングテーブルから判別しようとします。

OCF_RESKEY_broadcast=ブロードキャストアドレス
IPに関連付けられているブロードキャストアドレス。空の場合、スクリプトはネットマスクからこれを判別します。

OCF_RESKEY_iflabel=インタフェースラベル
ここでIPアドレスの追加ラベルを指定できます。

OCF_RESKEY_lvs_support=LVS DRのサポートを有効化
LVSダイレクトルーティング環境設定のサポートを可能にします。IPアドレスが停止した場合、ローカルノードがサービス要求に応えられるようにループバックデバイスへの移動のみを行い、ネットワーク上でのアドバタイズは行わないようにします。

OCF_RESKEY_local_stop_script=IPが解放された時に呼び出されるスクリプト
IPが解放されたときに呼び出されるスクリプト

OCF_RESKEY_local_start_script=IPが追加されたときに呼び出されるスクリプト
IPが追加されたときに呼び出されるスクリプト

OCF_RESKEY_ARP_INTERVAL_MS=Gratuitous ARP間隔(ミリ秒)
ARPの間隔をミリ秒で指定します。

OCF_RESKEY_ARP_REPEAT=反復回数
新しいアドレスを立ち上げる際に送信するGratuitous ARP数

OCF_RESKEY_ARP_BACKGROUND=バックグラウンドでの実行
バックグラウンドで実行(もうこれを行う理由がなくなった)

OCF_RESKEY_ARP_NETMASK=ARP用のネットマスク
ARPのネットマスク - 非標準の16進形式。

ocf:IPsrcaddr (7)

ocf:IPsrcaddr — IPsrcaddrリソースエージェント

書式

[OCF_RESKEY_ipaddress=string] IPsrcaddr [start | stop | stop | monitor | validate-all | meta-data]

説明

IPsrcaddrのリソーススクリプト。優先ソースアドレスの変更を管理します。

サポートされているパラメータ

OCF_RESKEY_ipaddress=IPアドレス
IPアドレス。

ocf:IPv6addr (7)

ocf:IPv6addr — IPv6別名を管理します

書式

[OCF_RESKEY_ipv6addr=string] IPv6addr [start | stop | status | monitor | validate-all | meta-data]

説明

このスクリプトはIPv6の別名IPv6アドレスを管理します。IP6別名を追加または削除できます。

サポートされているパラメータ

OCF_RESKEY_ipv6addr=IPv6アドレス
　このRAで管理するIPv6アドレス

ocf:iscsi (7)

ocf:iscsi — iscsiリソースエージェント

書式

[OCF_RESKEY_portal=string] OCF_RESKEY_target=string
[OCF_RESKEY_discovery_type=string] [OCF_RESKEY_iscsiadm=string]
[OCF_RESKEY_udev=string] iscsi [start | stop | status | monitor | validate-all |
methods | meta-data]

説明

iSCSIのOCFリソースエージェントiSCSIターゲットを追加(開始)または削除(停止)します。

サポートされているパラメータ

OCF_RESKEY_portal=ポータル
{ip_address|hostname}[":"port]の形式のiSCSIポータルアドレス

OCF_RESKEY_target=ターゲット
iSCSIターゲット。

OCF_RESKEY_discovery_type=discovery_type
検出タイプ。現在、open-iscsiではsendtargetsタイプのみサポートされています。

OCF_RESKEY_iscsiadm=iscsiadm
iscsiadmプログラムパス。

OCF_RESKEY_udev=udev
次のリソースがデバイスの作成についてudevに依存している場合、終了するまで待機します。通常の方法でロードされたホストの場合は、すぐに行われるはずですが、そうでない場合もあります。udevを使用していない場

合は、これを「no」に設定します。設定しないと、タイムアウトが発生するまでループに入ってしまいます。

ocf:Ldirectord (7)

ocf:Ldirectord — ldirectordのラッパOCFリソースエージェント

書式

OCF_RESKEY_configfile=string [OCF_RESKEY_ldirectord=string]
Ldirectord [start | stop | monitor | meta-data | validate-all]

説明

ldirectord用の簡単なOCF RAラッパーで、ldirectordインタフェースを使用してOCF準拠のインタフェースを作成します。ldirectordの監視が行えます。ただし、ldirectordステータスの請求には多くのリソースが使用されてしまうため、注意が必要です。

サポートされているパラメータ

OCF_RESKEY_configfile=環境設定ファイルへのパス
ldirectord環境設定ファイルのフルパス名。

OCF_RESKEY_ldirectord=ldirectordバイナリパス
ldirectordのフルパス名。

ocf:LinuxSCSI (7)

ocf:LinuxSCSI — LinuxSCSIリソースエージェント

書式

[OCF_RESKEY_scsi=string] LinuxSCSI [start | stop | methods | status | monitor | meta-data | validate-all]

説明

これはLinuxSCSIのリソースエージェントです。SCSIデバイスの可用性をLinuxカーネルの観点から管理します。デバイスがなくなったようにLinuxに信じ込ませて、再起動させることができます。

サポートされているパラメータ

OCF_RESKEY_scsi=SCSIインスタンス
　管理対象のSCSIインスタンス。

ocf:LVM (7)

ocf:LVM — LVMリソースエージェント

書式

[OCF_RESKEY_volgrpname=string] LVM [start | stop | status | monitor | methods | meta-data | validate-all]

説明

LVMのリソーススクリプト。Linux Volume Manager (LVM) ボリュームをHAリソースとして管理します。

サポートされているパラメータ

OCF_RESKEY_volgrpname=ボリュームグループ名
　ボリュームグループ名。

ocf:MailTo (7)

ocf:MailTo — MailToリソースエージェント

書式

[OCF_RESKEY_email=string] [OCF_RESKEY_subject=string] MailTo [start | stop | status | monitor | meta-data | validate-all]

説明

これはMailToのリソースエージェントです。引き継ぎが発生するたびにsysadminに電子メールを送信します。

サポートされているパラメータ

OCF_RESKEY_email=電子メールアドレス
sysadminの電子メールアドレス。

OCF_RESKEY_subject=件名
電子メールの件名。

ocf:ManageRAID (7)

ocf:ManageRAID — RAIDデバイスを管理します

書式

[OCF_RESKEY_raidname=string] ManageRAID [start | stop | status | monitor | validate-all | meta-data]

説明

/etc/conf.d/HB-ManageRAIDで事前に設定されたRAIDデバイスの開始、停止、監視を管理します。

サポートされているパラメータ

OCF_RESKEY_raidname=RAID名
管理するRAID名(大文字、小文字を区別)。(/etc/conf.d/HB-ManageRAIDで事前設定)

ocf:ManageVE (7)

ocf:ManageVE — OpenVZ VEリソースエージェント

書式

[OCF_RESKEY_veid=integer] ManageVE [start | stop | status | monitor | validate-all | meta-data]

説明

このOCF準拠リソースエージェントはOpenVZ VEを管理するので、最近のvzctl utilを含む正しいOpenVZインストールが必要です。

サポートされているパラメータ

OCF_RESKEY_veid=VEのOpenVZ ID
　仮想環境のOpenVZ ID(全割り当て済みIDはvzlist -aの出力を参照)

ocf:mysql (7)

ocf:mysql — MySQLリソースエージェント

書式

[OCF_RESKEY_binary=string] [OCF_RESKEY_config=string]
[OCF_RESKEY_datadir=string] [OCF_RESKEY_user=string]
[OCF_RESKEY_group=string] [OCF_RESKEY_log=string]
[OCF_RESKEY_pid=string] [OCF_RESKEY_socket=string]
[OCF_RESKEY_test_table=string] [OCF_RESKEY_test_user=string]
[OCF_RESKEY_test_passwd=string]
[OCF_RESKEY_enable_creation=integer]
[OCF_RESKEY_additional_parameters=integer] mysql [start | stop | status | monitor | validate-all | meta-data]

説明

MySQLのリソーススクリプト。MySQLのデータベースインスタンスをHAリソースとして管理します。

サポートされているパラメータ

OCF_RESKEY_binary=MySQLバイナリ
　MySQLバイナリの場所

OCF_RESKEY_config=MySQLの環境設定
　環境設定ファイル

OCF_RESKEY_datadir=MySQLのデータディレクトリ
　データベースが保存されているディレクトリ

OCF_RESKEY_user=MySQLユーザ
　MySQLデーモンを実行しているユーザ

OCF_RESKEY_group=MySQLグループ
MySQLデーモンを実行しているグループ(ログファイルとディレクトリ許可用)

OCF_RESKEY_log=MySQLログファイル
mysqldに使用するログファイル。

OCF_RESKEY_pid=MySQL pidファイル
mysqldに使用するpidfile。

OCF_RESKEY_socket=MySQLソケット
mysqldに使用するソケット。

OCF_RESKEY_test_table=MySQLテストテーブル
監視ステートメントでテストするテーブル(database.table形式)

OCF_RESKEY_test_user=MySQLテストユーザ
MySQLテストユーザ

OCF_RESKEY_test_passwd=MySQLテストユーザのパスワード
MySQLテストユーザのパスワード

OCF_RESKEY_enable_creation=データベースが存在しない場合は作成
MySQLデータベースが存在しない場合は、新しく作成されます。

OCF_RESKEY_additional_parameters=mysqldに渡す追加のパラメータ
起動時にmysqldに渡す追加のパラメータ。(--skip-external-lockingまたは--skip-grant-tablesなど)

ocf:nfsserver (7)

ocf:nfsserver — nfsserver

書式

[OCF_RESKEY_nfs_init_script=string]
[OCF_RESKEY_nfs_notify_cmd=string]
[OCF_RESKEY_nfs_shared_infodir=string] [OCF_RESKEY_nfs_ip=string]
nfsserver [start | stop | monitor | meta-data | validate-all]

説明

Nfsserverは、Linux-HAにおけるフェールオーバー可能なリソースとして、Linux nfsサーバを管理します。Linux特有の詳細なNFS実装に依存するため、他のプラットフォームにはまだ移植できません。

サポートされているパラメータ

OCF_RESKEY_nfs_init_script= nfsserver用のinitスクリプト
: Linux distroと一緒に出荷されるデフォルトのinitスクリプト。nfsserverの開始/停止/監視の手順はLinux distroによって異なるため、nfsserverリソースエージェントは開始/停止/監視作業をinitスクリプトに移行します。

OCF_RESKEY_nfs_notify_cmd= 通知送信元のツール。
: NSM再起動通知を送信するツール。nfsserverのフェールオーバーは、別なマシンへの再起動と見なされます。nfsserverリソースエージェントはこのコマンドを使用して、フェールオーバーの発生をすべてのクライアントに通知します。

OCF_RESKEY_nfs_shared_infodir= nfsサーバに関連する情報を保存するディレクトリ。
: nfsserverリソースエージェントは、nfs関連の情報をこのディレクトリに保存します。nfsserverより先にこのディレクトリのフェールオーバーを行う必要があります。

OCF_RESKEY_nfs_ip= IPアドレス。
　nfsサービスにアクセスするために使用するフローティングIPアドレス

ocf:oracle (7)

ocf:oracle — oracleリソースエージェント

書式

OCF_RESKEY_sid=string [OCF_RESKEY_home=string]
[OCF_RESKEY_user=string] [OCF_RESKEY_ipcrm=string]
[OCF_RESKEY_clear_backupmode=boolean]
[OCF_RESKEY_shutdown_method=string] oracle [start | stop | status | monitor | validate-all | methods | meta-data]

説明

oracleのリソーススクリプト。Oracleのデータベースインスタンスを HAリソースとして管理します。

サポートされているパラメータ

OCF_RESKEY_sid=sid
Oracle SID (ORACLE_SID)。

OCF_RESKEY_home=ホーム
Oracleのホームディレクトリ(ORACLE_HOME)。指定していない場合は、SIDとホームが/etc/oratabに一覧表示されているはずです。

OCF_RESKEY_user=ユーザ
Oracleオーナー(ORACLE_OWNER)。指定していない場合は、$ORACLE_HOME/dbs/*${ORACLE_SID}.oraのオーナーに設定されます。正しく機能しない場合は、明示的に設定します。

OCF_RESKEY_ipcrm=ipcrm
場合によってはOracleインスタンスに属しているIPCオブジェクト(共有メモリセグメントとセマフォ)が残ってしまい、インスタンスの開始の妨げになります。どの共有セグメントがどのインスタンスに属しているのかを

簡単に判断することはできません。特に、同じユーザとして実行しているインスタンス数が多い場合は判断が困難です。ここでは「oradebug」機能とその「ipc」トレースユーティリティを使用します。デバッグ情報の解析が最適とはいえませんが、他にIPC情報を調べる方法がありません。トレースレポートの形式や用語が変わった場合、解析は失敗します。他のユーザの操作を妨げないようにするための注意点として他にもいくつかあります。またインスタンスに属するIPCオブジェクトを印刷できるdumpinstipcオプションもあります。トレースファイルを正しく解析しているかを確認するために使用してください。次の3つの設定が可能です-none: IPCは変更せずに、うまくいくことを祈る(ただし遅かれ早かれ失敗することになるという点に注意すること)-instance:インスタンスに属しているIPC項目を検出し、それだけを削除する(デフォルト設定で安全な方法)-orauser: インスタンスを実行しているユーザに属しているすべてのIPCを削除する(同じユーザとして複数のインスタンスを実行している場合、またはこのユーザとして実行している他のアプリケーションがIPCを使用している場合は使用しないでください)。デフォルト設定である「instance」は使用しても安全ですが、インスタンスが開始する保証はありません。誰かがOracleプロセスを強制終了してしまった場合など、IPCオブジェクトが残ってしまっている場合、削除するIPCオブジェクトを調べる方法はありません。この場合、手動での処理が必要になります。おそらく、同じユーザとして実行しているすべてのインスタンスを停止する必要があります。3番目の設定「orauser」では、確実にIPCオブジェクトが削除されますが、IPCオブジェクトのオーナー情報にのみ基づいて削除するため、各インスタンスが別のユーザとして実行している場合にのみ使用する必要があります。問題があれば報告してください。提案や修正についての情報も歓迎いたします。

`OCF_RESKEY_clear_backupmode`=clear_backupmode
ORACLEのバックアップモードを消去します。

`OCF_RESKEY_shutdown_method`=shutdown_method
Oracleの停止方法は好みの問題のようです。デフォルトの方法(「checkpoint/abort」)はシステムチェックポイントの変更、シャットダウンの中断で、インスタンスを終了する最も高速で安全な方法です。「シャットダウンの中断」を使用したくない場合は、この属性を「immediate」に設定することで、即座にシャットダウンすることができます。もっと良いOracleインスタンスのシャットダウン方法がある場合はお知らせください。

ocf:oralsnr (7)

ocf:oralsnr — oralsnrリソースエージェント

書式

OCF_RESKEY_sid=string [OCF_RESKEY_home=string]
[OCF_RESKEY_user=string] OCF_RESKEY_listener=string oralsnr [start | stop | status | monitor | validate-all | meta-data | methods]

説明

Oracle Listenerのリソーススクリプト。Oracle ListenerインスタンスをHAリソースとして管理します。

サポートされているパラメータ

OCF_RESKEY_sid=sid
Oracle SID (ORACLE_SID)。tnsping SIDの実行などopの監視に必要です。

OCF_RESKEY_home=ホーム
Oracleホームディレクトリ(ORACLE_HOME)。指定していない場合は、SIDが/etc/oratabに一覧表示されているはずです。

OCF_RESKEY_user=ユーザ
リスナをこのユーザとして実行します。

OCF_RESKEY_listener=リスナ
開始するリスナインスタンス(listener.oraで定義済み)。LISTENERにデフォルト設定されます。

ocf:pgsql (7)

ocf:pgsql — pgsqlリソースエージェント

書式

[OCF_RESKEY_pgctl=string] [OCF_RESKEY_start_opt=string]
[OCF_RESKEY_ctl_opt=string] [OCF_RESKEY_psql=string]
[OCF_RESKEY_pgdata=string] [OCF_RESKEY_pgdba=string]
[OCF_RESKEY_pghost=string] [OCF_RESKEY_pgport=string]
[OCF_RESKEY_pgdb=string] [OCF_RESKEY_logfile=string]
[OCF_RESKEY_stop_escalate=string] pgsql [start | stop | status | monitor | meta-data | validate-all | methods]

説明

PostgreSQLのリソーススクリプト。PostgreSQLをHAリソースとして管理します。

サポートされているパラメータ

OCF_RESKEY_pgctl=pgctl
pg_ctlコマンドへのパス。

OCF_RESKEY_start_opt=start_opt
開始オプション(pgi_ctl内の-o start_opt)。たとえば「-i -p 5432」などです。

OCF_RESKEY_ctl_opt=ctl_opt
追加のpg_ctlオプション(-w、-Wなど)。デフォルトは""です。

OCF_RESKEY_psql=psql
psqlコマンドへのパス。

OCF_RESKEY_pgdata=pgdata
PostgreSQLデータディレクトリへのパス。

OCF_RESKEY_pgdba=pgdba
PostgreSQLのオーナーであるユーザ。

OCF_RESKEY_pghost=pghost
PosrgeSQLがリスンしているホスト名とIPアドレス

OCF_RESKEY_pgport=pgport
PosrgeSQLがリスンしているポート

OCF_RESKEY_pgdb=pgdb
監視に使用するデータベース。

OCF_RESKEY_logfile=ログファイル
PostgreSQLサーバログ出力ファイルへのパス。

OCF_RESKEY_stop_escalate=エスカレーションの停止
-m immediateに復元するまでの再試行回数(-m fastを使用)。

ocf:pingd (7)

ocf:pingd — pingdリソースエージェント

書式

[OCF_RESKEY_pidfile=string] [OCF_RESKEY_user=string]
[OCF_RESKEY_dampen=integer] [OCF_RESKEY_set=integer]
[OCF_RESKEY_name=integer] [OCF_RESKEY_section=integer]
[OCF_RESKEY_multiplier=integer] [OCF_RESKEY_host_list=integer]
pingd [start | stop | monitor | meta-data | validate-all]

説明

これはpingdリソースエージェントです。ノードが接続できる現在のpingノード数を記録します(CIB内に)。

サポートされているパラメータ

OCF_RESKEY_pidfile=PIDファイル
PIDファイル

OCF_RESKEY_user=pingdを実行するユーザ
pingdを実行するユーザ

OCF_RESKEY_dampen=ダンパー間隔
追加の変更を待機する時間(ダンパー)

OCF_RESKEY_set=セット名
値を設定するinstance_attributesセット名。指定する必要はほとんどありません。

OCF_RESKEY_name=属性名
設定する属性名。制約で使用する名前です。

OCF_RESKEY_section=セクション名
　値を設定するセクションです。指定する必要はほとんどありません。

OCF_RESKEY_multiplier=値の乗数
　接続されているpingノードに乗算する数。

OCF_RESKEY_host_list=ホストリスト
　カウントするpingノードのリスト。デフォルトでは、設定済みのすべてのpingノードです。指定する必要はほとんどありません。

ocf:portblock (7)

ocf:portblock — portblockリソースエージェント

書式

[OCF_RESKEY_protocol=string] [OCF_RESKEY_portno=integer] [OCF_RESKEY_action=string] portblock [start | stop | status | monitor | meta-data | validate-all]

説明

portblockのリソーススクリプト。iptableを使用して一時的にポートをブロックするために使用します。

サポートされているパラメータ

OCF_RESKEY_protocol=プロトコル
　ブロックまたはブロック解除に使用するプロトコル。

OCF_RESKEY_portno=ポート番号
　ブロックまたはブロック解除に使用するポート番号。

OCF_RESKEY_action=アクション
　protocol::portnoで実行するアクション(ブロックまたはブロック解除)。

ocf:Pure-FTPd (7)

ocf:Pure-FTPd — OCFリソースエージェントに準拠しているFTPスクリプト。

書式

OCF_RESKEY_script=string OCF_RESKEY_conffile=string
OCF_RESKEY_daemon_type=string [OCF_RESKEY_pidfile=string]
Pure-FTPd [start | stop | monitor | validate-all | meta-data]

説明

このスクリプトはActive-Passiveのセットアップで Pure-FTPdを管理します。

サポートされているパラメータ

OCF_RESKEY_script=フルパス付きのスクリプト名
Pure-FTPd起動スクリプトへのフルパス。たとえば「/sbin/pure-config.pl」などです。

OCF_RESKEY_conffile=フルパス付きの環境設定ファイル名
フルパス付きのPure-FTPd環境設定ファイル名。たとえば「/etc/pure-ftpd/pure-ftpd.conf」などです。

OCF_RESKEY_daemon_type=フルパス付きの環境設定ファイル名
pure-ftpd-wrapperで呼びだされるPure-FTPdデーモン。有効なオプションは、pure-ftpdの「」、pure-ftpd-mysqlの「mysql」、pure-ftpd-postgresqlの「postgresql」、そしてpure-ftpd-ldapの「ldap」です。

OCF_RESKEY_pidfile=PIDファイル
PIDファイル

ocf:Raid1 (7)

ocf:Raid1 — RAID1リソースエージェント

書式

[OCF_RESKEY_raidconf=string] [OCF_RESKEY_raiddev=string] [OCF_RESKEY_homehost=string] Raid1 [start | stop | status | monitor | validate-all | meta-data]

説明

RAID1のリソーススクリプト。共有ストレージメディア上のソフトウェアRaid1デバイスを管理します。

サポートされているパラメータ

OCF_RESKEY_raidconf=RAID環境設定ファイル
RAID環境設定ファイル。 例: /etc/raidtabまたは/etc/mdadm.conf

OCF_RESKEY_raiddev=ブロックデバイス
使用するブロックデバイス。

OCF_RESKEY_homehost=mdadmのホームホスト
homehostディレクティブの値。これはRAIDが間違ってアクティブ化されないようにするmdadm機能です。「homehost」を特殊な値に設定したクラスタで管理するRAIDを作成して、所有権のないノードが間違って自動的にアセンブルしてしまわないようにすることをお勧めします。

ocf:Route (7)

ocf:Route — ネットワークルートを管理します

書式

OCF_RESKEY_destination=string OCF_RESKEY_device=string OCF_RESKEY_gateway=string OCF_RESKEY_source=string Route [start | stop | monitor | reload | meta-data | validate-all]

説明

ネットワークルートを有効および無効にします。ホストとネットルート、ゲートウェイアドレス経由のルート、特定のソースアドレスを使用しているルートをサポートします。このリソースエージェントは、ノードのルーティングテーブルをノードの役割割り当てに応じて操作する必要がある場合に便利です。たとえば次の使用例を検討してください。-1つのクラスタノードをIPsecトンネルエンドポイントとして使用する。- 他のすべてのノードがIPsecトンネルを使用して特定のリモートネットワーク内のホストに到達する。続いて、次の方法でRouteリソースエージェントを使用してこのスキームを行うことができます。-ipsec LSBリソースを設定します。- クローンのRoute OCFリソースを設定します。- ipsecがRouteより前に起動されるように、実行順序の制約を作成します。- ipsecとRouteリソース間のコロケーションの制約を作成して、クローンのRouteリソースがトンネルエンドポイント自体で開始されないようにします。

サポートされているパラメータ

OCF_RESKEY_destination=宛先ネットワーク
ルートに設定する宛先ネットワーク(またはホスト)。ネットマスクサフィックスをCIDR形式(「/24」など)で指定します。サフィックスが指定されていない場合は、ホストルートが作成されます。このリソースにシステムデフォルトルートを設定する場合は、「0.0.0.0/0」または「default」を指定します。

OCF_RESKEY_device=送信用ネットワークデバイス
　このルートに使用する送信用ネットワークデバイス。

OCF_RESKEY_gateway=ゲートウェイIPアドレス
　このルートに使用するゲートウェイIPアドレス。

OCF_RESKEY_source=ソースIPアドレス
　ルート用に設定するソースIPアドレス。

ocf:rsyncd (7)

ocf:rsyncd — OCFリソースエージェントに準拠したrsyncデーモンスクリプト。

書式

[OCF_RESKEY_binpath=string] [OCF_RESKEY_conffile=string]
[OCF_RESKEY_bwlimit=string] rsyncd [start | stop | monitor | validate-all | meta-data]

説明

このスクリプトはrsyncデーモンを管理します

サポートされているパラメータ

OCF_RESKEY_binpath=rsyncバイナリのフルパス
rsyncバイナリパス。たとえば「/usr/bin/rsync」などです。

OCF_RESKEY_conffile=フルパス付きの環境設定ファイル名
フルパス付きのrsyncデーモン環境設定ファイル名。たとえば「/etc/rsyncd.conf」などです。

OCF_RESKEY_bwlimit=I/O帯域幅の制限(1秒あたりのキロバイト数)
このオプションによって、1秒あたりのキロバイト数で最大転送速度を指定できます。このオプションは、大規模なファイル(複数メガバイト以上)にrsyncを使用する場合に最も効果的です。rsync転送の特性上、データブロックの送信後、rsyncが転送が速すぎたと判断すれば、次のデータブロックの送信を待機します。その結果、指定した制限値に等しい平均的な転送速度になります。0を指定すると無制限になります。

ocf:SAPDatabase (7)

ocf:SAPDatabase — SAPデータベースリソースエージェント

書式

OCF_RESKEY_SID=string OCF_RESKEY_DIR_EXECUTABLE=string
OCF_RESKEY_DBTYPE=string OCF_RESKEY_NETSERVICENAME=string
OCF_RESKEY_DBJ2EE_ONLY=boolean OCF_RESKEY_JAVA_HOME=string
OCF_RESKEY_STRICT_MONITORING=boolean
OCF_RESKEY_AUTOMATIC_RECOVER=boolean
OCF_RESKEY_DIR_BOOTSTRAP=string OCF_RESKEY_DIR_SECSTORE=string
OCF_RESKEY_DB_JARS=string OCF_RESKEY_PRE_START_USEREXIT=string
OCF_RESKEY_POST_START_USEREXIT=string
OCF_RESKEY_PRE_STOP_USEREXIT=string
OCF_RESKEY_POST_STOP_USEREXIT=string SAPDatabase [start | stop | status | monitor | validate-all | meta-data | methods]

説明

SAPデータベースのリソーススクリプト。任意のタイプのSAPデータベースをHAリソースとして管理します。

サポートされているパラメータ

OCF_RESKEY_SID=SAPシステムID
固有のSAPシステム識別子。たとえばP01などです。

OCF_RESKEY_DIR_EXECUTABLE=sapstartsrvとsapcontrolのパス
sapstartsrvとsapcontrolの検索先の完全修飾パス。

OCF_RESKEY_DBTYPE=データベースベンダ
使用するデータベースベンダ名。ORA、DB6、ADAのいずれかに設定します。

OCF_RESKEY_NETSERVICENAME=リスナ名
Oracle TNSリスナ名。

OCF_RESKEY_DBJ2EE_ONLY=JAVAスタックのみインストール済み
ABAPスタックがSAPデータベースにインストールされていない場合は、これをTRUEに設定します。

OCF_RESKEY_JAVA_HOME=Java SDKのパス
DBJ2EE_ONLYパラメータをtrueに設定した場合のみ必要です。SAP WebAS Javaで使用するJava SDKのパスを入力します

OCF_RESKEY_STRICT_MONITORING=アプリケーションレベルの監視を起動
リソースエージェントによるデータベースの監視方法を制御します。trueに設定するとSAPツールを使用してデータベースへの接続をテストします。Oracleで使用すると、アーカイバがスタックした場合にフェールオーバーが発生してしまうため、使用しないでください。

OCF_RESKEY_AUTOMATIC_RECOVER=起動時の自動復旧を有効化または無効化
SAPDatabaseリソースエージェントが、失敗した開始の試みを自動的に1回復旧させようとします。RDBMSの強制的な中断、復旧コマンドの実行のいずれか、または両方によってこれを行います。

OCF_RESKEY_DIR_BOOTSTRAP=j2eeブートストラップディレクトリへのパス
J2EEインスタンスブートストラップディレクトリの検索先の完全修飾パス。たとえば/usr/sap/P01/J00/j2ee/cluster/bootstrapなどです。

OCF_RESKEY_DIR_SECSTORE=j2eeセキュアストレージディレクトリへのパス
J2EEセキュリティストアディレクトリの検索先の完全修飾パス。たとえば/usr/sap/P01/SYS/global/security/lib/toolsなどです。

OCF_RESKEY_DB_JARS=jdbcドライバのファイル名
データベース接続テスト用のjdbcドライバの完全修飾ファイル名。Javaエンジン6.40および7.00のbootstrap.propertiesファイルから自動的に読み込まれます。Javaエンジン7.10では必須パラメータです。

OCF_RESKEY_PRE_START_USEREXIT=開始前スクリプトへのパス
このリソースの開始前に実行するスクリプトまたはプログラムを検索する完全修飾パス。

OCF_RESKEY_POST_START_USEREXIT=開始後スクリプトへのパス
このリソースの開始後に実行するスクリプトまたはプログラムを検索する完全修飾パス。

OCF_RESKEY_PRE_STOP_USEREXIT=開始前スクリプトへのパス
このリソースの停止前に実行するスクリプトまたはプログラムを検索する完全修飾パス。

OCF_RESKEY_POST_STOP_USEREXIT=開始後スクリプトへのパス
このリソースの停止後に実行するスクリプトまたはプログラムを検索する完全修飾パス。

ocf:SAPInstance (7)

ocf:SAPInstance — SAPインスタンスリソースエージェント

書式

OCF_RESKEY_InstanceName=string OCF_RESKEY_DIR_EXECUTABLE=string
OCF_RESKEY_DIR_PROFILE=string OCF_RESKEY_START_PROFILE=string
OCF_RESKEY_START_WAITTIME=string
OCF_RESKEY_AUTOMATIC_RECOVER=boolean
OCF_RESKEY_PRE_START_USEREXIT=string
OCF_RESKEY_POST_START_USEREXIT=string
OCF_RESKEY_PRE_STOP_USEREXIT=string
OCF_RESKEY_POST_STOP_USEREXIT=string SAPInstance [start | recover | stop | status | monitor | validate-all | meta-data | methods]

説明

SAPのリソーススクリプト。SAPインスタンスをHAリソースとして管理します。

サポートされているパラメータ

OCF_RESKEY_InstanceName=インスタンス名:SID_INSTANCE_VIR-HOSTNAME
完全に修飾されたSAPインスタンス名。たとえばP01_DVEBMGS00_sapp01ciなどです。

OCF_RESKEY_DIR_EXECUTABLE=sapstartsrvとsapcontrolのパス
sapstartsrvとsapcontrolの検索先の完全修飾パス。

OCF_RESKEY_DIR_PROFILE=開始プロファイルのパス
SAP STARTプロファイルの検索先の完全修飾パス。

OCF_RESKEY_START_PROFILE=開始プロファイル名
SAP STARTプロファイル名。

OCF_RESKEY_START_WAITTIME=この時間の経過後、開始成功を確認する(J2EE-アドインを待機しない)
リソースエージェントが監視操作を実行するまでの秒数。監視がSUCCESSを返すと、開始はSUCCESSとして処理されます。これはたとえばJ2EE-Addinインスタンスのタイミング問題を解決する際に役立ちます。

OCF_RESKEY_AUTOMATIC_RECOVER=起動時の自動復旧を有効化または無効化
SAPInstanceリソースエージェントが、失敗した開始の試みを自動的に1回復旧しようとします。インスタンスプロセスの強制終了とcleanipcの実行によって行います。

OCF_RESKEY_PRE_START_USEREXIT=開始前スクリプトへのパス
このリソースの開始前に実行するスクリプトまたはプログラムを検索する完全修飾パス。

OCF_RESKEY_POST_START_USEREXIT=開始後スクリプトへのパス
このリソースの開始後に実行するスクリプトまたはプログラムを検索する完全修飾パス。

OCF_RESKEY_PRE_STOP_USEREXIT=開始前スクリプトへのパス
このリソースの停止前に実行するスクリプトまたはプログラムを検索する完全修飾パス。

OCF_RESKEY_POST_STOP_USEREXIT=開始後スクリプトへのパス
このリソースの停止後に実行するスクリプトまたはプログラムを検索する完全修飾パス。

ocf:scsi2reserve (7)

ocf:scsi2reserve — scsi-2の予約

書式

[OCF_RESKEY_scsi_reserve=string] [OCF_RESKEY_sharedisk=string] [OCF_RESKEY_start_loop=string] scsi2reserve [start | stop | monitor | meta-data | validate-all]

説明

scsi-2-reserveリソースエージェントは、SCSI-2の予約のためのプレースホルダです。scsi-2-reserveリソースの良好なインスタンスで、指定したSCSIデバイスの所有を示します。このリソースエージェントは、Linux特有のscsiresパッケージのscsi_reserveに依存しています。

サポートされているパラメータ

OCF_RESKEY_scsi_reserve= scsi_reserveコマンド
scsi_reserveはscsiresパッケージのコマンドラインです。SCSIデバイスでのSCSI-2予約の発行に役立ちます。

OCF_RESKEY_sharedisk= 共有ディスク。
予約可能な共有ディスク。

OCF_RESKEY_start_loop= 中断する前に再試行する回数。
中断する前に複数回試行します。Start_loopは再試行回数を示します。

ocf:SendArp (7)

ocf:SendArp — SendArpリソースエージェント

書式

[OCF_RESKEY_ip=string] [OCF_RESKEY_nic=string] SendArp [start | stop | monitor | meta-data | validate-all]

説明

このスクリプトはIPアドレスに対してGratuitous ARPを送信します

サポートされているパラメータ

OCF_RESKEY_ip=IPアドレス
arpパッケージの送信先IPアドレス。

OCF_RESKEY_nic=NIC
arpパッケージを送信するNIC。

ocf:ServeRAID (7)

ocf:ServeRAID — ServeRAIDリソースエージェント

書式

[OCF_RESKEY_serveraid=integer] [OCF_RESKEY_mergegroup=integer] ServeRAID [start | stop | status | monitor | validate-all | meta-data | methods]

説明

ServeRAIDのリソーススクリプト。共有ServeRAIDマージグループを有効または無効にします。

サポートされているパラメータ

OCF_RESKEY_serveraid=serveraid
ServeRAIDアダプタのアダプタ番号。

OCF_RESKEY_mergegroup=マージグループ
検討中の論理ドライブ。

ocf:sfex (7)

ocf:sfex — SF-EXリソースエージェント

書式

[OCF_RESKEY_device=string] [OCF_RESKEY_index=integer]
[OCF_RESKEY_collision_timeout=integer]
[OCF_RESKEY_monitor_interval=integer]
[OCF_RESKEY_lock_timeout=integer] sfex [start | stop | monitor | meta-data]

説明

SF-EXのリソーススクリプト。共有ストレージメディアを排他的に管理します。

サポートされているパラメータ

OCF_RESKEY_device=ブロックデバイス
　排他的な制御データを保存するブロックデバイスパス。

OCF_RESKEY_index=インデックス
　排他的な制御データが保存されているブロックデバイスの場所。1以上を指定します。デフォルトは1です。

OCF_RESKEY_collision_timeout=ロックの取得を待機
　ロック取得の衝突を検出した場合の待機時間。デフォルトは1秒です。

OCF_RESKEY_monitor_interval=監視間隔
　監視間隔(秒)。デフォルトは10秒です。

OCF_RESKEY_lock_timeout=ロックの有効期限
　ロックの有効期限(秒)。デフォルトは20秒です。

ocf:SphinxSearchDaemon (7)

ocf:SphinxSearchDaemon — searchdリソースエージェント

書式

OCF_RESKEY_config=string [OCF_RESKEY_searchd=string]
[OCF_RESKEY_search=string] [OCF_RESKEY_testQuery=string]
SphinxSearchDaemon [start | stop | monitor | meta-data | validate-all]

説明

これはsearchdリソースエージェントです。Sphinx検索デーモンを管理します。

サポートされているパラメータ

OCF_RESKEY_config=環境設定ファイル
searchd環境設定ファイル

OCF_RESKEY_searchd=searchdバイナリ
searchdバイナリ

OCF_RESKEY_search=検索バイナリ
監視アクションでの機能テスト用の検索バイナリ。

OCF_RESKEY_testQuery=テストクエリ
監視アクションでの機能テスト用のテストクエリ。クエリは索引内のどのドキュメントにも一致する必要はありません。単に検索デーモンが索引に対するクエリを行うことができ、正しく応答するかどうかテストすることを目的としています。

ocf:Squid (7)

ocf:Squid — SquidのRA

書式

[OCF_RESKEY_squid_exe=string] OCF_RESKEY_squid_conf=string
OCF_RESKEY_squid_pidfile=string OCF_RESKEY_squid_port=integer
[OCF_RESKEY_squid_stop_timeout=integer]
[OCF_RESKEY_debug_mode=string] [OCF_RESKEY_debug_log=string] Squid
[start | stop | status | monitor | meta-data | validate-all]

説明

Squidのリソースエージェント。SquidインスタンスをHAリソースとして管理します。

サポートされているパラメータ

OCF_RESKEY_squid_exe=実行可能ファイル
これは必須パラメータです。このパラメータはsquidの実行可能ファイルを指定します。

OCF_RESKEY_squid_conf=環境設定ファイル
これは必須パラメータです。このパラメータはこのRAが管理するsqidインスタンスの環境設定ファイルを指定します。

OCF_RESKEY_squid_pidfile=Pidファイル
これは必須パラメータです。このパラメータはこのRAが管理するsquidインスタンスのプロセスidファイルを指定します。

OCF_RESKEY_squid_port=ポート番号
これは必須パラメータです。このパラメータはこのRAが管理するsquidインスタンスのポート番号を指定します。複数のポートを使用している場合、そのうちの1つだけを指定する必要があります。

OCF_RESKEY_squid_stop_timeout=通常の停止方法を確認するために待機する秒数

これは省略可能なパラメータです。停止アクションでは、通常の停止方法が最初に使用されます。次に、このパラメータで指定した秒数の間、完了確認を待機します。デフォルト値は「10」です。

OCF_RESKEY_debug_mode=デバッグモード

このパラメータの指定は任意です。このパラメータに「x」または「v」が含まれている場合は、このRAがデバッグモードで実行されます。「x」が含まれている場合は、STDOUTとSTDERRの両方を「debug_log」で指定されたログファイルに転送し、内蔵されているシェルオプション「x」がオンになります。「v」に類しています。

OCF_RESKEY_debug_log=デバッグログの宛先

このパラメータはオプションで省略することができます。このパラメータはデバッグログの宛先ファイルを指定し、このRAをデバッグモードで実行する場合にのみ機能します。デバッグモードについては、「debug_mode」を参照してください。必要な値が指定されていない場合、次のルールに従って作成されます。「/var/log/」をディレクトリ部分とし、「syslog_ng_conf」をベース名部分、「.log」をサフィックスとして環境設定ファイルの基本名を作成します。

ocf:Stateful (7)

ocf:Stateful — ステートフルリソースエージェントの例

書式

`OCF_RESKEY_state`=string `Stateful` [start | stop | monitor | meta-data | validate-all]

説明

2つの状態を実装するリソースエージェントの例です

サポートされているパラメータ

`OCF_RESKEY_state`=状態ファイル
    リソース状態を保存する場所

ocf:SysInfo (7)

ocf:SysInfo — SysInfoリソースエージェント

書式

[OCF_RESKEY_pidfile=string] [OCF_RESKEY_delay=string] SysInfo [start | stop | monitor | meta-data | validate-all]

説明

これはSysInfoリソースエージェントです。ノードのサンプルLinux出力の各種属性を記録(CIBに)します。arch: i686 os: Linux-2.4.26-gentoo-r14 free_swap: 1999 cpu_info: Intel(R) Celeron(R) CPU 2.40GHz cpu_speed: 4771.02 cpu_cores: 1 cpu_load: 0.00 ram_total: 513 ram_free: 117 root_free: 2.4 Sample Darwin output: arch: i386 os: Darwin-8.6.2 cpu_info: Intel Core Duo cpu_speed: 2.16 cpu_cores: 2 cpu_load: 0.18 ram_total: 2016 ram_free: 787 root_free: 13 Units: free_swap: Mb ram_*: Mb root_free: Gb cpu_speed (Linux): bogomips cpu_speed (Darwin): Ghz

サポートされているパラメータ

OCF_RESKEY_pidfile=PIDファイル
　PIDファイル

OCF_RESKEY_delay=ダンパー遅延
　変数を安定させる間隔

ocf:tomcat (7)

ocf:tomcat — tomcatリソースエージェント

書式

OCF_RESKEY_tomcat_name=string OCF_RESKEY_script_log=string
[OCF_RESKEY_tomcat_stop_timeout=integer]
[OCF_RESKEY_tomcat_suspend_trialcount=integer]
[OCF_RESKEY_tomcat_user=string] [OCF_RESKEY_statusurl=string]
[OCF_RESKEY_java_home=string] OCF_RESKEY_catalina_home=string
OCF_RESKEY_catalina_pid=string
[OCF_RESKEY_tomcat_start_opts=string]
[OCF_RESKEY_catalina_opts=string]
[OCF_RESKEY_catalina_rotate_log=string]
[OCF_RESKEY_catalina_rotatetime=integer] tomcat [start | stop | status | monitor | meta-data | validate-all]

説明

tomcatのリソーススクリプト。Tomcatインスタンスを HAリソースとして管理します。

サポートされているパラメータ

OCF_RESKEY_tomcat_name=リソース名
　リソース名

OCF_RESKEY_script_log=このスクリプトのログ宛先
　このスクリプトのログの宛先。

OCF_RESKEY_tomcat_stop_timeout=停止時のタイムアウト
　停止時のタイムアウト

OCF_RESKEY_tomcat_suspend_trialcount=停止を待機する再試行回数
　停止を待機する再試行回数

OCF_RESKEY_tomcat_user=リソースを開始するユーザ名
　リソースを開始するユーザ名

OCF_RESKEY_statusurl=状態確認用のURL
　状態確認用のURL

OCF_RESKEY_java_home=Javaのホームディレクトリ
　Javaのホームディレクトリ

OCF_RESKEY_catalina_home=Tomcatのホームディレクトリ
　Tomcatのホームディレクトリ

OCF_RESKEY_catalina_pid=TomcatのPIDファイル名
　TomcatのPIDファイル名

OCF_RESKEY_tomcat_start_opts=Tomcatの開始オプション
　Tomcatの開始オプション

OCF_RESKEY_catalina_opts=Catalinaオプション
　Catalinaオプション

OCF_RESKEY_catalina_rotate_log=catalina.outフラグの切り替え
　catalina.outフラグを切り替えます

OCF_RESKEY_catalina_rotatetime=catalina.outの切り替え間隔
　catalina.outの切り替え間隔

ocf:VIPArip (7)

ocf:VIPArip — RIP2プロトコルによる仮想IPアドレス

書式

OCF_RESKEY_ip=string [OCF_RESKEY_nic=string] VIPArip [start | stop | monitor | validate-all | meta-data]

説明

RIP2プロトコルによる仮想IPアドレス。このスクリプトはquagga/ripdで異なるサブネット内のIP別名を管理します。IP別名を追加または削除できます。

サポートされているパラメータ

OCF_RESKEY_ip=別なサブネットのIPアドレス
「192.168.1.1」などの、別なサブネット内のIPv4アドレス。

OCF_RESKEY_nic=ルート情報をブロードキャストするためのNIC
ルート情報をブロードキャストするためのNIC。ripdはこのNICを使用してルート情報をブロードキャストします

ocf:VirtualDomain (7)

ocf:VirtualDomain — 仮想ドメインを管理します。

書式

OCF_RESKEY_config=string [OCF_RESKEY_hypervisor=string]
[OCF_RESKEY_force_stop=boolean]
[OCF_RESKEY_migration_transport=string]
[OCF_RESKEY_monitor_scripts=string] VirtualDomain [start | stop | status | monitor | migrate_from | migrate_to | meta-data | validate-all]

説明

libvirtdが管理している仮想ドメイン(コンテキストに応じてdomU、仮想マシン、仮想環境など)のリソースエージェント。

サポートされているパラメータ

OCF_RESKEY_config=仮想ドメイン環境設定ファイル
この仮想ドメインのlibvirt環境設定ファイルへの絶対パス。

OCF_RESKEY_hypervisor=Hypervisor URI
接続先のHypervisor URI。サポートしているURI形式の詳細は、libvirtのドキュメントを参照してください。デフォルトはシステムに依存します。

OCF_RESKEY_force_stop=停止時にシャットダウンを強制
停止中のドメインを強制的にシャットダウン(「強制終了」)します。仮想ドメイン(または仮想化バックエンド)がGraceful Shutdownをサポートしていない場合にのみ有効にしてください。

OCF_RESKEY_migration_transport=リモートhypervisorトランスポート
ママイグレーション中にリモートhypervisorに接続するために使用するトランスポート。使用可能なトランスポートの詳細は、libvirtのドキュメン

トを参照してください。このパラメータを省略すると、リソースはlibvirtのデフォルトトランスポートを使用してリモートhypervisorに接続します。

OCF_RESKEY_monitor_scripts=監視スクリプトをスペースで区切ったリスト

仮想ドメイン内のサービスを追加で監視するには、このパラメータに監視するスクリプトのリストを追加します。注:監視スクリプトを使用すると、すべての監視スクリプトが正常に終了するまで、操作の開始とマイグレーションは完了しません。この遅延に対処するため、この操作のタイムアウトを必ず設定しておいてください。

ocf:WAS6 (7)

ocf:WAS6 — WAS6リソースエージェント

書式

[OCF_RESKEY_profile=string] WAS6 [start | stop | status | monitor | validate-all | meta-data | methods]

説明

WAS6のリソーススクリプト。Websphere Application Server (WAS6)をHAリソースとして管理します。

サポートされているパラメータ

OCF_RESKEY_profile=プロファイル名
　WASプロファイル名。

ocf:WAS (7)

ocf:WAS — WASリソースエージェント

書式

[OCF_RESKEY_config=string] [OCF_RESKEY_port=integer] WAS [start | stop | status | monitor | validate-all | meta-data | methods]

説明

WASのリソーススクリプト。Websphere Application Server (WAS)をHAリソースとして管理します。

サポートされているパラメータ

OCF_RESKEY_config=環境設定ファイル
　WAS環境設定ファイル。

OCF_RESKEY_port=ポート
　WAS-(snoop)-port-number。

ocf:WinPopup (7)

ocf:WinPopup — WinPopupリソースエージェント

書式

[OCF_RESKEY_hostfile=string] WinPopup [start | stop | status | monitor | validate-all | meta-data]

説明

WinPopupのリソーススクリプト。切り替えが発生するたびに、WinPopupsメッセージをsysadminのワークステーションに送信します。

サポートされているパラメータ

OCF_RESKEY_hostfile=ホストファイル
　WinPopupメッセージの送信先ホストが含まれたファイル。

ocf:Xen (7)

ocf:Xen — Xen DomUsを管理します

書式

[OCF_RESKEY_xmfile=string] [OCF_RESKEY_name=string]
[OCF_RESKEY_allow_migrate=boolean]
[OCF_RESKEY_shutdown_timeout=boolean]
[OCF_RESKEY_allow_mem_management=boolean]
[OCF_RESKEY_reserved_Dom0_memory=string]
[OCF_RESKEY_monitor_scripts=string] Xen [start | stop | migrate_from | migrate_to | monitor | meta-data | validate-all]

説明

Xen Hypervisorのリソースエージェント。クラスタリソースの開始と停止をXenの作成とシャットダウンにそれぞれ関連付けることで、Xen仮想マシンインスタンスを管理します。名前に関する注記。環境設定ファイルから名前を抽出しようとします(xmfile属性)。簡単な割り当てステートメントを使用すれば問題ありません。そうでなければ、他の変数に応じてダイナミックに名前を割り当てるといった何らかの関連するpython処理がある場合はこの検出を試みます。そのときに名前の属性を設定してください。環境設定ファイルが共有ストレージに置かれている場合など、環境設定ファイルを検出できない異常な状況が発生した場合にもこの操作を行う必要があります。どの方法でも名前を抽出できない場合は、インスタンスidを元に戻して後方互換性を維持します。並行仮想化されたゲストも、meta_attribute allow_migrateを有効にしてマイグレートできます。

サポートされているパラメータ

OCF_RESKEY_xmfile=Xen制御ファイル
この仮想マシンのXen制御ファイルへの絶対パス。

OCF_RESKEY_name=Xen DomU名
仮想マシン名。

OCF_RESKEY_allow_migrate=ライブマイグレーションの使用
このブールパラメータによって、並行仮想化マシンのライブマイグレーションが行えます。

OCF_RESKEY_shutdown_timeout=シャットダウンエスカレーションタイムアウト
Xenエージェントは最初にxmシャットダウンを使用して順序正しくシャットダウンを試みます。これがタイムアウト時間までに成功しなかった場合は、エージェントはxm destroyにエスカレーションして、ノードを強制終了します。設定されていない場合、停止アクションのタイムアウトの3分の2にデフォルト設定されます。この値を0に設定すると、すぐに強制終了が行われます。

OCF_RESKEY_allow_mem_management=ダイナミックメモリ管理の使用
このパラメータによって、Dom0およびDomUsに使用する開始と停止アクションのメモリをダイナミックに調整できます。デフォルト設定ではメモリをダイナミックに調整しません。

OCF_RESKEY_reserved_Dom0_memory=最小Dom0メモリ
メモリ管理を使用する場合、コノパラメータはdom0に予約される最小メモリ量を定義します。デフォルトの最小メモリは512MBです。

OCF_RESKEY_monitor_scripts=監視スクリプトをスペースで区切ったリスト
許可されていないドメイン内のサービスを追加で監視するには、このパラメータに監視するスクリプトのリストを追加します。注意:この場合、少なくともDomUがすべてのサービスを開始するまでに要する時間を、監視操作の開始遅延として必ず設定しておく必要があります。

ocf:Xinetd (7)

ocf:Xinetd — Xinetdリソースエージェント

書式

[OCF_RESKEY_service=string] Xinetd [start | stop | restart | status | monitor | validate-all | meta-data]

説明

Xinetdのリソーススクリプト。xinetdが管理しているサービスを開始または停止します。xinetdデーモン自体が実行している必要があります。独自に開始や停止は行いません。重要:クラスタが管理しているサービスが、有効になっている唯一のものである場合、-stayaliveオプションをxinetdに指定しておかないと、ハートビートの停止時に終了してしまいます。あるいはエコーなどの内部サービスを有効にしておくこともできます。

サポートされているパラメータ

OCF_RESKEY_service=サービス名
    winetdが管理するサービス名。

パート V. 付録

GNU利用許諾契約書

A

この付録には、GNU一般公衆利用許諾契約書(GPL)とGNUフリー文書利用許諾契約書(GFDL)が含まれています。

GNU General Public License

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The 「Program」, below, refers to any such program or work, and a 「work based on the Program」 means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term 「modification」.) Each licensee is addressed as 「you」.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and 「any later version」, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the 「copyright」 line and a pointer to where the full notice is found.

one line to give the program's name and an idea of what it does. Copyright (C) yyyy name of author

```
This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.


This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.


You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type `show w'. This is free software, and you are welcome
to redistribute it under certain conditions; type `show c'
for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a 「copyright disclaimer」 for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright
interest in the program `Gnomovision'
(which makes passes at compilers) written
by James Hacker.


signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License [`http://www.fsf.org/licenses/lgpl.html`] instead of this License.

GNU Free Documentation License

Version 1.2, November 2002

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of 「copyleft」, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The 「Document」, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as 「you」. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A 「Modified Version」 of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A 「Secondary Section」 is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The 「Invariant Sections」 are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The 「Cover Texts」 are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A 「Transparent」 copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not 「Transparent」 is called 「Opaque」.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The 「Title Page」 means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, 「Title Page」 means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section 「Entitled XYZ」 means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as 「Acknowledgements」, 「Dedications」, 「Endorsements」, or 「History」.) To 「Preserve the Title」 of such a section when you modify the Document means that it remains a section 「Entitled XYZ」 according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled 「History」, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled 「History」 in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the 「History」 section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled 「Acknowledgements」 or 「Dedications」, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled 「Endorsements」. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled 「Endorsements」 or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled 「Endorsements」, provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled 「History」 in the various original documents, forming one section Entitled 「History」; likewise combine any sections Entitled 「Acknowledgements」, and any sections Entitled 「Dedications」. You must delete all sections Entitled 「Endorsements」.

COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled 「Acknowledgements」, 「Dedications」, or 「History」, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License 「or any later version」 applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (c) YEAR YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
only as published by the Free Software Foundation;
with the Invariant Section being this copyright notice and license.
A copy of the license is included in the section entitled “GNU
Free Documentation License”.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

用語集

アクティブ/アクティブ、アクティブ/パッシブ
: サービスがノード上で実行される方法についてのコンセプト。アクティブ/パッシブシナリオは、1つ以上のサービスがアクティブノード上で実行され、パッシブノードはアクティブノードの失敗を待機します。これに対して、アクティブ/アクティブは、各ノードが同時ににアクティブ、またはパッシブであることを意味します。

クラスタ
: ハイパフォーマンスクラスタは、アプリケーションロードを共有し、処理を迅速に実行する、コンピュータのグループ(現実または仮想)です。高可用性クラスタは、サービスの最大可用性を確保することを第一に設計されています。

クラスタパーティション
: 1つ以上のノードとその他のクラスタ間で通信が失敗した場合は、常にクラスタパーティションが発生します。クラスタパーティションのノードはまだアクティブで、相互に通信できますが、どのノードと通信できないかを認識していません。その他のパーティションの損失を確認できないため、スプリットブレインシナリオが作成されました(スプリットブレイン (298 ページ)も参照)。

コンセンサスクラスタメンバーシップ(CCM)
: CCMは、どのノードがクラスタを構成するか決定し、この情報をクラスタで共有します。ノードまたはクォーラムの新規追加および損失は、CCMによって通知されます。CCMモジュールはクラスタの各ノード上で実行されます。

クラスタ情報ベース(CIB)
: クラスタ構成全体と状態の表現(ノードメンバーシップ、リソース、制約など)。XMLで書かれ、メモリに常駐しています。マスタCIBはDCで保持され、その他のノードに複製されます。

クラスタリソースマネージャ(CRM)
: すべての非ローカルインタラクションを調整する、主要な管理エンティティ。クラスタの各ノードには固有のCRMがありますが、DC上で実行されるノードには決定をその他の非ローカルCRMに中継し、入力を処理するために選択されたCRMがあります。CRMは、いくつかのコンポーネン

トと通信します。固有のノードとその他のノードのローカルリソースマネージャ、非ローカルCRM、管理コマンド、フェンシング機能、メンバシップ層です。

指定コーディネータ(DC)
「マスタ」ノード。このノードには、CIBのマスタコピーが保持されます。その他すべてのノードは、現在のDCから構成とリソース割り当て情報を取得します。DCは、メンバーシップの変更後、クラスタ内のすべてのノードから選抜されます。

Distributed Replicated Block Device (DRBD)
DRBDは、高可用性クラスタを構築するためのブロックデバイスです。ブロックデバイス全体が専用ネットワーク経由でミラーリングされ、ネットワークRAID-1として認識されます。

フェールオーバー
リソースまたはノードが1台のマシンで失敗し、影響を受けるリソースが別のノードで起動されたときに発生します。

フェンシング
非クラスタメンバーによる共有リソースへのアクセスを防止するコンセプトを示します。「誤動作」しているノードを終了(シャットダウン)して問題の発生を防止する、状態が不明なノードからリソースをロックする、またはその他の方法で実現されます。さらに、フェンシングはノードとリソースフェンシングに分類されます。

Heartbeatリソースエージェント
HeartbeatリソースエージェントはHeartbeatバージョン1で広く使用されていました。あまり使用されなくなりましたが、バージョン2でサポートされています。Heartbeatリソースエージェントは`start`、`stop`、`status`操作を実行でき、`/etc/ha.d/resource.d`または`/etc/init.d`の下にあります。Heartbeatリソースエージェントの詳細は、`http://www.linux-ha.org/HeartbeatResourceAgent`を参照してください。

ローカルリソースマネージャ(LRM)
ローカルリソースマネージャ(LRM)は、リソース上の操作の実行を担当します。リソースエージェントスクリプトを使用して処理を実行します。LRMはそれ自身ではポリシーを認識していないという点で、「ダム」です。何をすべきか認識させるにはDCが必要です。

LSBリソースエージェント
: LSBリソースエージェントは標準LSB初期化スクリプトです。LSB初期化スクリプトは高可用性コンテキストでの使用に限定されません。あらゆるLSB準拠Linuxシステムは LSB初期化スクリプトを使用して、サービスを制御します。あらゆるLSBリソースエージェントは`start`、`stop`、`restart`、`status`、`force-reload`オプションをサポートし、オプションで`try-restart`および`reload`も使用できます。LSBリソースエージェントは`/etc/init.d`にあります。LSBリソースエージェントの詳細と実際の仕様は、`http://www.linux-ha.org/LSBResourceAgent`および`http://www.linux-foundation.org/spec/refspecs/LSB_3.0.0/LSB-Core-generic/LSB-Core-generic/iniscrptact.html`を参照してください。

node (ノード)
: クラスタのメンバーでユーザからは参照できないコンピュータ(現実または仮想)。

pingd
: pingデーモン。ICMP pingを使用して、クラスタ外部の1台以上のサーバに常に接続します。

ポリシーエンジン(PE)
: ポリシーエンジンはCIBでのポリシー変更を実装するために必要な処理を計算します。この情報はトランザクションエンジンに渡され、次にポリシー変更がクラスタセットアップで実装されます。PEは常にDC上で実行されます。

OCFリソースエージェント
: OCFリソースエージェントはLSBリソースエージェント(初期化スクリプト)と同様です。任意のOCFリソースエージェントは`start`、`stop`、`status`(場合によっては`monitor`と呼ばれる)オプションをサポートする必要があります。また、`metadata`オプションをサポートし、リソースエージェントタイプの説明をXMLで返します。追加オプションをサポートできますが、必須ではありません。OCFリソースエージェントは`/usr/lib/ocf/resource.d/`*`provider`*にあります。OCFリソースエージェントの詳細と仕様のドラフトは、`http://www.linux-ha.org/OCFResourceAgent`および`http://www.opencf.org/cgi-bin/viewcvs.cgi/specs/ra/resource-agent-api.txt?rev=HEAD`を参照してください。

クォーラム
: クラスタでは、クラスタパーティションは、ノード(投票)の大多数を保有する場合、クォーラムを持つ(「定数に達している」)と定義されます。クォーラムはただ1つのパーティションで識別されます。複数の切断されたパーティションまたはノードが処理を続行してデータおよびサービスが破損されないようにする、アルゴリズムの一部です(スプリットブレイン)。クォーラムはフェンシングの前提条件で、このためクォーラムは一意になります。

resource
: Heartbeatに認識されている、任意のタイプのサービスまたはアプリケーション。IPアドレス、ファイルシステム、データベースなどです。

リソースエージェント(RA)
: リソースエージェント(RA)は、プロキシとして動作してリソースを管理するスクリプトです。リソースエージェントには、OCF(Open Cluster Framework)リソースエージェント、LSBリソースエージェント(標準LSB初期化スクリプト)、Heartbeatリソースエージェント(Heartbeat v1リソース)の3種類があります。

シングルポイント障害(SPOF)
: シングルポイント障害(SPOF)は、失敗した場合、クラスタ全体の失敗につながるクラスタのコンポーネントです。

スプリットブレイン
: クラスタノードが(ソフトウェアまたはハードウェア障害によって)互いに認識しない2つ以上のグループに分割される場合のシナリオです。スプリットブレイン状態によってクラスタ全体に悪影響が及ばないようにするため、STONITHが救済する必要があります。「パーティションされたクラスタ」シナリオとも呼ばれます。

STONITH
: 「Shoot the other node in the head」の略で、基本的に誤動作しているノードを停止させ、クラスタでの問題発生を防止するものです。

遷移エンジン(TE)
: 遷移エンジン(TE)はPEからポリシーでぃれくてぃぶを取得し、これを実行します。TEは常にDC上で実行されます。ここから、その他のノードのローカルリソースマネージャにどのようなアクションを実行するか指示します。