IRIS FailSafe™ Version 2
Administrator's Guide（日本語版）

007–3901–006JP

制作スタッフ

著作 Jenn Byrnes、Susan Ellis、Lori Johnson、Steven Levine

編集 Susan Wilkening

イラスト Chrystie Danzer、Dany Galgani

製作 Glen Traefald

協力エンジニア Vidula Iyer、Ashwinee Khaladkar、Harald Kaul、Tony Kavadias、Linda Lait、Michael Nishimoto、Alain Renaud、Wesley Smith、Bill Sparks、Paddy Sreenivasan、Dan Stekloff、Rebecca Underwood、Manish Verma

カバー・デザイン: Sarah Bolles、Sarah Bolles Design、Dany Galgani、SGI Technical Publications

このガイドでの新機能

この改訂版には、以下の新しい情報が含まれています。

- SGI Origin 300、SGI Origin 3200C、SGI Onyx 300、および SGI Onyx 3200C システム用 L1 システム・コントローラ・ポートのサポート。111 ページの「ノードの定義」および215 ページの「Origin 300、Origin 3200C、Onyx 300、および Onyx 3200C のコンソール・サポート」を参照してください。
- 「ファイル(File)」メニューで FailSafe GUI からアクセスできる PCP (Performance Co-Pilot) の `hbvis`(1) および `rmvis`(1) ツール。85 ページの「GUI の概要」を参照してください。
- FailSafe および CXFS メタデータ・サーバ再設定に関する情報。275 ページの「CXFS メタデータ・サーバの再設定」を参照してください。
- `cluster_status`(1M) コマンドにおける変更。223 ページの「`cluster_status` を使ったシステムのステータスのモニタ」を参照してください。
- offline_detach の使用に関する説明。220 ページの「2 つのノードの使用の再開」を参照してください。
- 追加の FailSafe GUI トラブルシューティング情報。273 ページの「GUI が実行されない」を参照してください。
- アップデートされた FailSafe GUI アイコン。225 ページの「アイコンと状態の説明」を参照してください。
- FailSafe システムのソフトウェア階層、通信パス、およびクラスタ・データベースに関する情報は、293 ページの付録A「ソフトウェアの概要」に移動されました。

改訂情報

バージョン	説明
002	1999 年 12 月 FailSafe 2.0 ロールアップ・パッチと共に発行。IRIX 6.5.2 以降をサポート。
003	2000 年 11 月 IRIS FailSafe 2.1 リリースをサポート
004	2001 年 5 月 IRIS FailSafe 2.1.1 リリースをサポート
005	2001 年 10 月 IRIS FailSafe 2.1.2 リリースをサポート
006	2002 年 4 月 IRIS FailSafe 2.1.3 リリースをサポート

目次

図一覧

表一覧

このガイドについて

このガイドでは、IRIS FailSafe 高可用性システムの設定と管理について説明します。

このガイドは、IRIS FailSafe 製品のリリース 2.1.3 に関して作成されており、IRIX 6.5.16 以降をサポートしています。

対象読者

『IRIS FailSafe Version 2 Administrator's Guide』は、IRIS FailSafe システムの管理者の方向けに作成されています。IRIS FailSafe 管理者は、Origin サーバの操作に加え、オプションの Origin Vault、ファイバ・チャネル RAID、JBOD、TP9100、または TP9400 ストレージ・システムのうち IRIS FailSafe 設定で使用するものについても精通している必要があります。また、XLV と XFS についての詳しい知識も必要になります。

前提条件

Performance Co-Pilot (PCP) for FailSafe を使用するには、以下のライセンスが必要です。

- 2 つ以上の PCP Collector ライセンス (`PCPCOL`)、パフォーマンス・メトリックを収集する FailSafe クラスタの各ノードに対して 1 つずつ
- 表示ツールを実行するワークステーション用に 1 つの PCP Monitor ライセンス (`PCPMON`)

このガイドの構造

IRIS FailSafe の設定と管理の情報は、以下の章と付録に記載されています。

- 第1章「概要」では、IRIS FailSafe システムのコンポーネントの概要、およびそのハードウェアとソフトウェア・アーキテクチャを説明します。
- 第2章「設定計画」では、FailSafe クラスタの設定計画の方法を説明します。
- 第3章「インストールとシステムの準備」では、ノードを FailSafe 用に準備するために、クラスタのノードで実行する必要がある複数の手順について説明します。

- 第4章「管理ツール」では、FailSafe マネージャ GUI および `cmgr`(1M) コマンドの概要について説明します。
- 第5章「設定」では、FailSafe システムの設定方法について説明します。
- 第6章「設定例」では、FailSafe の 3 ノード設定の例と、その設定のバリエーションを示します。
- 第7章「IRIS FailSafe のシステム運用」では、FailSafe システムを運用およびモニタする方法について説明します。
- 第8章「設定のテスト」では、設定した FailSafe システムのテスト方法を説明します。
- 第9章「システムの回復とトラブルシューティング」では、FailSafe で使用されるログ・ファイルと、回復手順について説明します。
- 第10章「運用中のクラスタのアップグレードとメインテナンス」では、FailSafe クラスタをシャットダウンせずに実行しなければならない場合があるいくつかの手順について説明します。
- 第11章「Performance Co-Pilot for FailSafe」では、PCP を使用して FailSafe クラスタの可用性をモニタする方法を説明します。
- 付録B「IRIS FailSafe 1.2 から IRIS FailSafe 2.1.*x* へのアップグレード」では、アップグレード手順について説明します。
- 付録C「IRIS FailSafe 2.1.*x* ソフトウェア」では、クラスタの各コンポーネントにインストールするシステムの概要を説明します。
- 付録D「PCP (Peformance Co-Pilot) for FailSafe によってエクスポートされるメトリック」では、`pmdafsafe`(1) によって実装されるメトリックを示します。

関連ドキュメント

FailSafe 環境では、以下のドキュメントが役立ちます。

- 『IRIS FailSafe Version 2 Programmer's Guide』
- 『Performance Co-Pilot User's and Administrator's Guide』
- 『CXFS Version 2 Software Installation and Administration Guide』
- 『IRIS FailSafe 2.0 DMF Administrator's Guide』
- 『IRIS FailSafe 2.0 INFORMIX Administrator's Guide』
- 『IRIS FailSafe 2.0 Netscape Server Administrator's Guide』

- 『IRIS FailSafe Version 2 NFS Administrator's Guide』
- 『IRIS FailSafe 2.0 Oracle Administrator's Guide』
- 『IRIS FailSafe Version 2 Samba Administrator's Guide』
- 『IRIS FailSafe Version 2 TMF Administrator's Guide』
- 『Embedded Support PartnerUser Guide』

IRIS FailSafe のマン・ページは、以下のとおりです。

- `cdbBackup`(1M)
- `cdbRestore`(1M)
- `cmgr`(1M)
- `crsd`(1M)
- `failsafe`(7M)
- `fs2d`(1M)
- `ha_cilog`(1M)
- `ha_cmsd`(1M)
- `ha_exec2`(1M)
- `ha_fsd`(1M)
- `ha_gcd`(1M)
- `ha_ifd`(1M)
- `ha_ifdadmin`(1M)
- `ha_macconfig2`(1M)
- `ha_srmd`(1M)
- `ha_statd2`(1M)
- `haStatus`(1M)

各 IRIS FailSafe 製品にはリリース・ノートが付属します。リリース・ノートの名前は以下のとおりです。

リリース・ノート	製品
`cluster_admin`	Cluster administration services
`cluster_control`	Node control services
`cluster_services`	Cluster services
`failsafe2`	IRIS FailSafe 2.1.*X*
`failsafe2_dmf`	IRIS FailSafe/DMF
`failsafe2_informix`	IRIS FailSafe INFORMIX
`failsafe2_nfs`	IRIS FailSafe NFS
`failsafe2_oracle`	IRIS FailSafe Oracle
`failsafe2_samba`	IRIS FailSafe Samba
`failsafe2_tmf`	IRIS FailSafe/TMF
`failsafe2_web`	IRIS FailSafe Netscape Web

出版物の入手方法

SGI のドキュメントは、下記の SGI Technical Publications Library サイトで入手可能です。

`http://techpubs.sgi.com.`

表記規則

このドキュメントでは、以下の表記規則が使用されています。

表記規則	意味
`command`	この固定スペース・フォントは、コマンド、ファイル、ルーチン、パス名、シグナル、メッセージ、プログラミング言語の構造などのリテラル項目を示します。
`manpage(`*x*`)`	マン・ページのセクション ID は、マン・ページ名の後に括弧で囲んで示します。(1) はユーザ・コマンドを示し、(1M) は管理者コマンドを示します。

variable	イタリック体は、変数のエントリ、および定義される語や概念を示します。
user input	この太字体の固定スペース・フォントは、対話型セッションでユーザが入力するリテラル項目を示します。出力は、太字体ではない固定スペース・フォントで示されます。
[ ]	コマンドやディレクティブ行のオプション部分は、角括弧で囲みます。
...	省略記号は、先行する要素が繰返し可能であることを示します。

このガイドでは、*IRIS FailSafe* の簡略名として *FailSafe* という用語を使用しています。

ご意見とお問合わせ先

このマニュアルの技術的正確性、内容、または構成についてご意見等ございましたら、弊社までお問合わせください。コメントいただくマニュアルのタイトルとドキュメント番号も必ず一緒にお聞かせいただくようお願いいたします。オンラインの場合、ドキュメント番号はマニュアルの最初の部分に記載されています。印刷マニュアルの場合は、各ページの下部にドキュメント番号が記載されています。

ご連絡の際は、以下のいずれかの方法をご利用いただけます。

- 電子メールの場合は、

  `techpubs@sgi.com`

  までお送りください。

- 次の Technical Publications Library の World Wide Web ページからの場合は、「Feedback」オプションをクリックしてください。

  `http://techpubs.sgi.com`

- お客様相談窓口までご連絡いただき、SGI の問題追跡システムへの入力をお申付けください。

- 郵送の場合は、次の住所までお送りください。
  Technical Publications
  SGI
  1600 Amphitheatre Pkwy., M/S 535
  Mountain View, California 94043–1351

- FAX の場合は、+1 650 932 0801 の「Technical Publications」宛にお送りください。

いただいたコメントには迅速確実に対応いたします。

概要

この章では、IRIS FailSafe システムのコンポーネントおよび操作の概要について説明します。この章は、以下の節で構成されています。

293 ページの付録A「ソフトウェアの概要」も参照してください。

高可用性および IRIS FailSafe

ミッション・クリティカルなコンピューティングの世界において、情報およびコンピュータ・リソースの可用性は非常に重要です。システムの可用性は、いずれかのコンポーネントに異常が発生した後にそのコンポーネントを使用できない時間の長さに左右されます。以下のように、可用性の度合いはシステムのタイプによって異なります。

- フォールトトレラント・システム（継続的な可用性）。これらのシステムは、冗長なコンポーネントと特別な論理を使用することで、継続的な運用を保証し、データの整合性を提供します。これらのシステムの可用性の度合いは非常に高いものです。これらのシステムの中には、ハードウェアやソフトウェアのアップグレードのための機能停止にも耐えられるものがあります。ただし、このソリューションは非常に高価で、特別なハードウェアとソフトウェアが必要です。
- 高可用性システム。これらのシステムでは、市販の冗長なコンポーネントと特別なソフトウェアを使用することによって、シングル・ポイント異常を切抜けます。これらのシステムの可用性の度合いは、フォールトトレラント・システムに比べると下がりますが、コストは大幅に低くなります。通常は、クライアント/サーバ・アプリケーションに対してのみ高可用性が提供され、その冗長性は、共有リソースを使用したクラスタ・アーキテクチャに基づきます。

SGI IRIS FailSafe 製品には、高可用性サービスを提供するための一般的な機能が用意されています。また、複数のノードを含むクラスタに対する高可用性サービスも用意されています（*N* ノード設定）。

クラスタ内のノードの 1 つ、またはそのノードのコンポーネントの 1 つに異常が発生した場合、その異常ノードの高可用性サービスはクラスタ内の別のノードで再開されます。クライアントからは、代替ノードのサービスと、異常が発生する前の元のサービスは区別できず、元のサービスがクラッシュして、ただちに再開されたように見えます。つまり、クライアントでは、高可用性サービスが短時間中断されたことしかわかりません。

FailSafe 環境において、ノードは、ほかのノードのバックアップとして機能します。フォールトトレラント・システムのバックアップ・リソースには、異常の発生に備えたバックアップ用の冗長なハードウェアとしての機能しかありませんが、高可用性システムの各ノードのリソースは、これとは異なり、必ずしも高可用性サービスではないその他のアプリケーションを実行するために通常の運用時にも使用できます。すべての高可用性サービスは、クラスタ内の 1 つのノードによって同時に所有されます。

高可用性サービスは、FailSafe ソフトウェアによって監視されます。通常の運用中に、これらのコンポーネントのいずれかで異常が検出された場合、フェイルオーバー・プロセスが開始されます。FailSafe を使用すると、どのような状況でどのノードがサービスを引継ぐかを指定したフェイルオーバー・ポリシーを定義できます。このプロセスは、異常ノードのリセット（データの整合性を保つため）、フェイルオーバーしたサービスに必要な回復手順の実行、およびサービスを引継ぐノードでのサービスの迅速な再開で構成されます。

FailSafe では、選択的フェイルオーバーがサポートされています。これにより、個々の高可用性アプリケーションは、同じノードのその他の高可用性アプリケーションとは別にバックアップ・ノードにフェイルオーバーできます。

クラスタ環境

この節では、以下の内容について説明します。

用語

この節では、FailSafe を使用して高可用性サービスを設定およびモニタするために必要な用語を定義します。

クラスタ

クラスタは、単一のコンピュータ・リソースとして連携動作するよう設定されたシステム（ノード）の集合です。クラスタは単純な名前で識別されます。この名前は、プール内で一意でなければなりません。たとえば、クラスタとノードで同じ名前を使用することはできません。

ノード

ノードはオペレーティング・システム (OS: Operating System) の 1 個のイメージで、通常は個々のコンピュータです。ノードは、ストレージ・システムとクラスタ内のノードを接続するストレージ・エリア・ネットワーク (SAN: Storage Area Network) に接続されています。ノードが属することのできるクラスタは 1 つだけです。

この意味でのノードという用語は、SGI Origin 3000 または SGI 2000 システムのノードとは異なる意味を持ちます。

プール

プールは、特定のクラスタに参加可能なノードの集合です。特定のプールから設定できるのは 1 つのクラスタのみです。このクラスタに、使用可能なすべてのノードが含まれている必要はありません。ほかのプールを存在させることはできますが、各プールはお互いから分離した状態になり、ノードやクラスタ定義は共有されません。

図1-1 に、プールとクラスタの概念を示します。

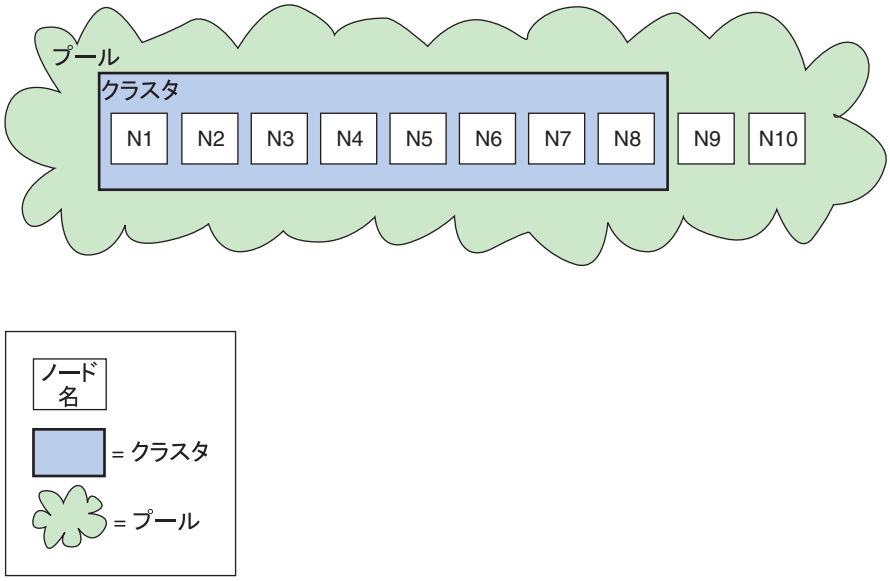


図1-1 プールとクラスタの概念

クラスタ・データベース

*クラスタ・データベース*には、すべてのノードとクラスタに関する設定情報が含まれます。fs2d デーモンは、プール内のすべてのノードへのクラスタ・データベース (CDB) の配布を管理します。

メンバーシップ

メンバーシップには、以下のタイプがあります。

- FailSafe メンバーシップ。FailSafe がリソース・グループをオンラインにできるクラスタ内の FailSafe ノードのリストです。
- fs2d データベース・メンバーシップ 。ユーザ空間メンバーシップとも呼ばれ、fs2d にアクセス可能な、プール内のノードのグループです。fs2d デーモンは、プール内のすべてのノードへのクラスタ・データベースの配布を管理します。fs2d で利用可能なノードは、クラスタ・データベースの更新を受信できるため、fs2d データベース・メンバーシップの一部となります。プール内で定義されたノードのサブセットである場合があります。
- プロセス・メンバーシップ。プロセス・グループを形成する、クラスタ内のプロセス・インスタンスのリストです。各ノードで複数のプロセス・グループを使用できます。

CXFS 同時実行の場合は、*CXFS メンバーシップ*もあります。CXFS についての詳細は、46 ページの「CXFS と IRIS FailSafe の同時実行」および『CXFS Version 2 Software Installation and Administration Guide』を参照してください。

上限量

上限量とは、クラスタを形成するために必要なノードの数のことで、メンバーシップに応じて以下のように変わります。

- FailSafe メンバーシップの場合、クラスタの形成と維持には、高可用性 (HA) サービスが開始されているクラスタ内のノードの **>50%**（過半数）が既知の状態（ハートビートおよびコントロール・ネットワークを使用して相互に正常にリセットまたは通信できる状態）である必要があります。
- `fs2d` データベース・メンバーシップの場合、クラスタの形成と維持には、プール内のノードの **50%**（半数）が `fs2d` デーモンで利用できる（したがってクラスタ・データベースの更新を受信できる）必要があります。

図1-2 に、FailSafe と `fs2d` メンバーシップの例を示します。この図には、以下の状況が示されています。

- プールは N1～N6 の 6 つのノードで構成されます。
- クラスタは、N1～N4 の 4 つのノードを含みます。
- N1～N4 の 4 つのノードで HA サービスが開始されています (HA services can only be started on nodes that have been defined as part of the cluster; however, not all nodes within the cluster must have HA services started.)
- 3 つのノード（N1～N3）は実行中で、3 つのノード（N4～N6）は停止しています。
- `fs2d` メンバーシップは、プール内の 6 つのノードのうちの 3 つ (50%) である N1～N3 で構成されています。
- FailSafe メンバーシップも、HA サービスが開始されている 4 つのノードのうちの 3 つであるノード N1～N3 で構成されています。

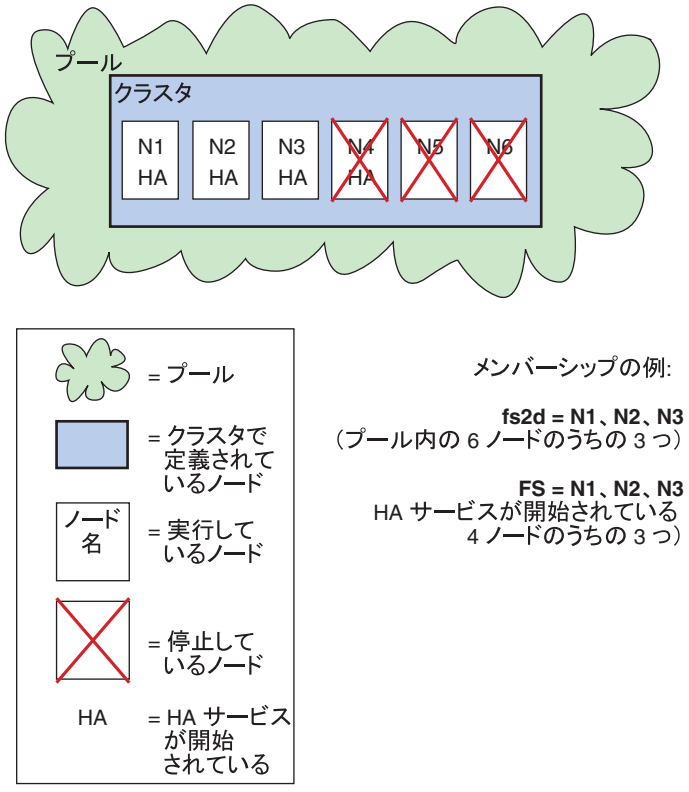


図1-2 FailSafe メンバーシップ

ネットワーク分割によって同数メンバーシップ（それぞれがクラスタの 50% で構成される 2 つのセットのノードが存在する状態）が発生した場合は、上限量を維持するために、タイブレーカー・ノードが含まれるセットのノードがもう一方のセットのノードのリセットを試みます。図1-3 に、以下を含むこの例を示します。

- ノード N1 と N2 がタイブレーカー・ノードを含むセットなので、これらの両方のノードがもう一方のセットのノード（N3 と N4）のリセットを試みます。
- リセットが正常に実行されて N1 と N2 のメンバーシップが確認された後、N3 と N4 に存在していたリソース・グループは、N1 と N2 に移動されます。ここでは、これらのノードのフェイルオーバー・ポリシーでこのように定義されていることを想定しています。

メモ: この図は、実際には 1 つの連続するイベントを示しています。それぞれの図は、異なる時点での状況を表しています。

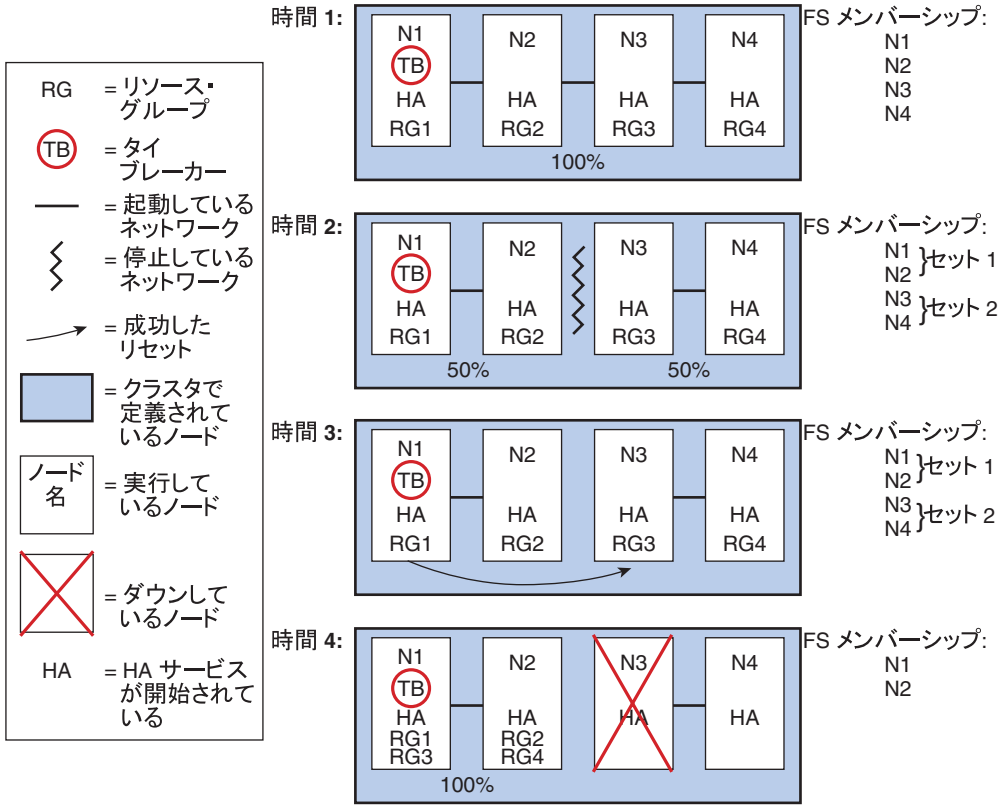


図1-3 タイブレーカー・ノードとメンバーシップ

プライベート・ネットワーク

プライベート・ネットワークは、クラスタ通信専用のネットワークで、管理者はアクセスできますが、ユーザはアクセスできません。

クラスタ・ソフトウェアは、プライベート・ネットワークを使用して、クラスタ・データベースを設定を機能させるために必要なハートビート・メッセージとコントロール・メッセージを送信します。ハートビート・メッセージの受信に遅れがあった場合、クラスタ・ソフトウェアでは、ノードが応答していないと判断し、そのノードを FailSafe メンバーシップから削除することがあります。

プライベート・ネットワークを使用するとパブリック・ネットワークのトラフィックが抑えられるため、不要なリセットや切断を回避するために役立ちます。

メッセージング・プロトコルではパケットの覗き見（表示）やなりすまし（ネットワーク上の特定のマシンを別のマシンに見せかけること）を防止できないので、プライベート・ネットワークの方がパブリック・ネットワークよりも安全です。

したがって、パブリック・ネットワークのパフォーマンスやセキュリティ特性が原因でクラスタに問題が発生する可能性があるという理由や、ハートビートがタイミングに大きく依存する（わずかな変動でも問題が発生する可能性があります）という理由から、すべてのノードが接続された専用のプライベート・ネットワークをお薦めしています。このプライベート・ネットワークは、ハートビート・メッセージやコントロール・メッセージを送信するために使用されます。

さらに、すべてのノードを同じローカル・ネットワーク・セグメント上に配置することが推奨されています。

メモ: プライベート・ネットワークにネットワーク上の問題がある場合は、FailSafe を使用する前に解決してください。

リソース

リソースは、クライアントまたはその他のリソースにサービスを提供する単一の物理的または論理的な実体です。たとえば、単一のディスク・ボリューム、特定のネットワーク・アドレス、Web サーバなどのアプリケーションがリソースの場合があります。通常、リソースは、長い間隔で見るとクラスタ内の複数のノードで使用できますが、同時に 1 つのノードにしか割当てることはできません。

リソースは、リソース名およびリソース・タイプによって識別されます。

リソース・タイプ

リソース・タイプは、リソースの特定のクラスです。フェイルオーバーの目的では、特定のリソース・タイプのリソースはすべて同じ方法で処理できます。各リソースは、常に 1 つのリソース・タイプのインスタンスです。

リソース・タイプは単純な名前で識別されます。この名前は、クラスタ内で一意でなければなりません。リソース・タイプは、特定のノードに対して定義したり、クラスタ全体に対して定義できます。特定のノードに対して定義されているリソース・タイプは、同じ名前のクラスタ全体のリソース・タイプ定義をオーバーライドします。このため、個々のノードは、クラスタ全体のリソース・タイプ定義のグローバル設定をオーバーライドできます。

FailSafe ソフトウェアには、定義済みリソース・タイプが多数含まれています。高可用性にするアプリケーションに適したリソース・タイプがある場合は、それらのタイプを流用できます。適切なスクリプトがない場合は、『IRIS FailSafe Version 2 Programmer's Guide』の手順を使用して、追加のリソース・タイプを作成できます。

リソース名

リソース名は、リソース・タイプの特定のインスタンスを識別します。リソース名は、特定のリソース・タイプに対して一意でなければなりません。

リソース・グループ

リソース・グループは、相互依存したリソースの集合です。リソース・グループは単純な名前で識別されます。この名前は、クラスタ内で一意でなければなりません。表1-1 に Webgroup というリソース・グループのリソースと、それに対応するリソース・タイプの例を示します。

表1-1 Webgroup リソース・グループの例

リソース	リソース・タイプ
10.10.48.22	IP_address
/fs1	filesystem
vol1	volume
web1	Netscape_web

リソース・グループのいずれかのリソースが目的の用途に使用できなくなった場合は、リソース・グループ全体が利用不可と見なされます。したがって、リソース・グループがフェイルオーバーの単位になります。

リソース・グループを重複させることはできません。つまり、2 つのリソース・グループに同じリソースを含めることはできません。

依存性

1 つのリソースは、その他の 1 つまたは複数のリソースに依存できます。その場合、リソースは、依存リソースも開始しないかぎり開始できません(つまり、使用可能にできません)。依存リソースは、同じリソース・グループの一部でなければなりません。また、依存リソースは、リソース依存リストで識別されます。リソース依存性は、リソースが定義されたときではなく、リソースがリソース・グループに追加されるときに確認されます。

リソースと同様に、リソース・タイプも、1 つまたは複数のリソース・タイプに依存できます。そのような依存性が存在する場合、各依存リソース・タイプのインスタンスを少なくとも 1 つ定義してください。リソース・タイプ依存リストには、あるリソース・タイプが依存するリソース・タイプの詳細がリストされています。

たとえば、Netscape_web というリソース・タイプに、 IP_address および volume というリソース・タイプに対してリソース・タイプ依存性を持たせることができます。 WS1 というリソースが Netscape_web リソース・タイプで定義されている場合は、WS1 を含むリソース・グループにも、タイプ IP_address のリ

ソースとタイプ volume のリソースが少なくとも 1 つずつ含まれていなければなりません。図1-4 に、これを示します。

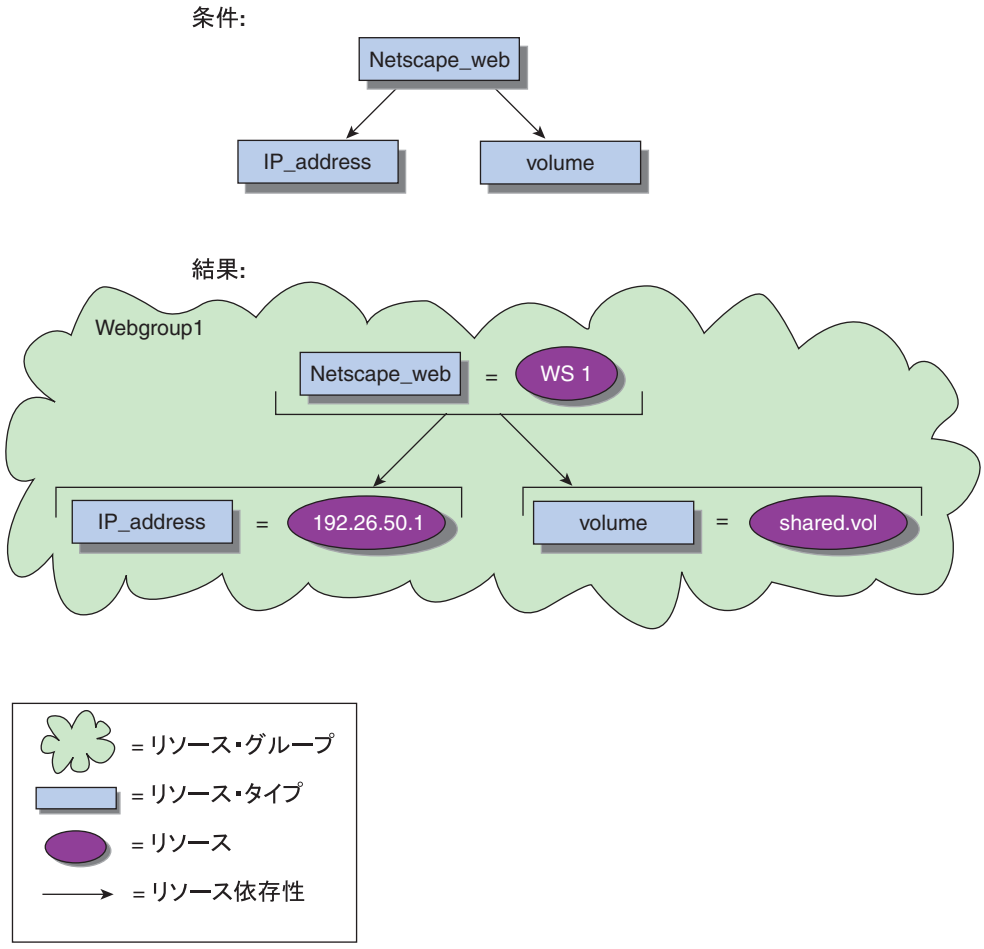


図1-4 リソース・タイプの依存性

フェイルオーバー

フェイルオーバーは、フェイルオーバー・ポリシーに従ってリソース・グループ（またはアプリケーション）を別のノードに割当てるプロセスです。フェイルオーバーは、リソース異常が発生したとき、FailSafe メン

バーシップが変更されたとき(ノードに異常が発生した場合やノードが開始された場合など)、または管理者が手動で要求したときに開始できます。

フェイルオーバー・ポリシー

フェイルオーバー・ポリシーは、フェイルオーバー先のノードを決定するときに FailSafe で使用される方法です。フェイルオーバー・ポリシーは以下で構成されます。

- フェイルオーバー・ドメイン
- フェイルオーバー属性
- フェイルオーバー・スクリプト

FailSafe では、フェイルオーバー・スクリプトからのフェイルオーバー・ドメイン出力をフェイルオーバー属性と共に使用して、リソース・グループを存在させるノードを決定します。

管理者は、各リソース・グループに対してフェイルオーバー・ポリシーを設定しなければなりません。フェイルオーバー・ポリシーの名前は、プール内で一意でなければなりません。

フェイルオーバー・ドメイン

フェイルオーバー・ドメインは、特定のリソース・グループを割当てることができるノードの順序付きリストです。フェイルオーバー・ドメインにリストされているノードは同じクラスタ内に存在しなければなりませんが、フェイルオーバー・ドメインにクラスタのすべてのノードを含める必要はありません。

初期フェイルオーバー・ドメインは、フェイルオーバー・ポリシーを作成するときに管理者が定義します。このリストは、フェイルオーバー・スクリプトによってランタイム・フェイルオーバー・ドメインに変換されます。FailSafe は、ランタイム・フェイルオーバー・ドメインをフェイルオーバー属性および FailSafe メンバーシップと共に使用して、リソース・グループを存在させるノードを決定します。FailSafe は、ランタイム・フェイルオーバー・ドメインを格納し、次のフェイルオーバー・スクリプト呼出しへの入力として使用します。ランタイムの状態やフェイルオーバー・スクリプトの内容によっては、初期フェイルオーバー・ドメインとランタイム・フェイルオーバー・ドメインは同一であることもあります。

一般的に FailSafe では、特定のリソース・グループは、ランタイム・フェイルオーバー・ドメインにリストされていて、FailSafe メンバーシップにも含まれる最初のノードに割当てられます。この割当てがいつ行われるかは、フェイルオーバー属性によって変わります。

フェイルオーバー属性

フェイルオーバー属性は、クラスタ内のリソース・グループの割当てを制御する文字列です。管理者は、システム属性(`Auto_Failback` や `Controlled_Failback` など)を指定する必要があります。また、オプションでサイト固有属性を指定できます。

フェイルオーバー・スクリプト

フェイルオーバー・スクリプトは、ランタイム・フェイルオーバー・ドメインを生成して ha_fsd プロセスに返すシェル・スクリプトです。ha_fsd プロセスによりフェイルオーバー属性が適用された後、返されたフェイルオーバー・ドメインの中で、現在の FailSafe メンバーシップにも存在する最初のノードが選択されます。

以下のフェイルオーバー・スクリプトは、FailSafe リリースに付属しています。

- ordered。これによって、初期フェイルオーバー・ドメインが変更されることはありません。このスクリプトを使用する場合、初期フェイルオーバー・ドメインとランタイム・フェイルオーバー・ドメインは同じものになります。
- round-robin。これによって、ラウンドロビン（循環）方式でリソース・グループ・オーナーが選択されます。このポリシーは、クラスタ内のすべてのノードで実行できるリソース・グループに対して使用できます。

これらのスクリプトがニーズを満たさない場合は、『IRIS FailSafe Version 2 Programmer's Guide』の情報を使用して、新しいフェイルオーバー・スクリプトを作成できます。

アクション・スクリプト

アクション・スクリプトは、リソースの開始、モニタ、および停止の方法を決定します。各リソース・タイプに対して 1 つのセットのアクションを指定する必要があります。

以下に、各リソース・タイプに指定できるアクション・スクリプトの完全なセットを示します。

- exclusive。リソースがまだ実行されていないことを確認します。
- start。リソースを開始します。
- stop。リソースを停止します。
- monitor。リソースをモニタします。
- restart。モニタ異常が発生した後、同じサーバでリソースを再開します。

リリースには、定義済みリソース・タイプのアクション・スクリプトが用意されています。これらのスクリプトが高可用性にするリソース・タイプに適さない場合は、必要に応じてコピーして変更することにより流用できます。適切なスクリプトがない場合は、『IRIS FailSafe Version 2 Programmer's Guide』の手順を使用して、追加のアクション・スクリプトを作成できます。

クラスタ・プロセス・グループ

クラスタ・プロセス・グループは分散型アプリケーションのアプリケーション・インスタンスのグループで、連携してサービスを提供します。各アプリケーション・インスタンスは、1 つまたは複数の UNIX プロセスで構成でき、1 つのノードだけで使用できます。

たとえば、各ノードの分散ロック・マネージャのインスタンスはプロセス・グループを形成します。プロセス・グループを形成することで、プロセス・メンバーシップに加え、信頼性の高い順序付きの原始的な通信サービスを実現できます。

メモ: UNIX のプロセス・グループとクラスタ・プロセス・グループの間に関係はありません。

プラグイン

プラグインは、リソース・タイプおよびアクション・スクリプトを含む、アプリケーションを高可用性にするために必要なソフトウェアのセットです。プラグインには、base FailSafe リリースに付属するプラグイン、SGI から購入できるオプションのプラグイン、『IRIS FailSafe Version 2 Programmer's Guide』の手順に従って作成できるカスタム・プラグインがあります。付属のプラグインおよびオプションのプラグインのリストについては、293 ページの「ソフトウェア階層」を参照してください。

ハードウェア・コンポーネント

図1-5に、FailSafe ハードウェア・コンポーネントの例（2 ノード・システムの場合）を示します。

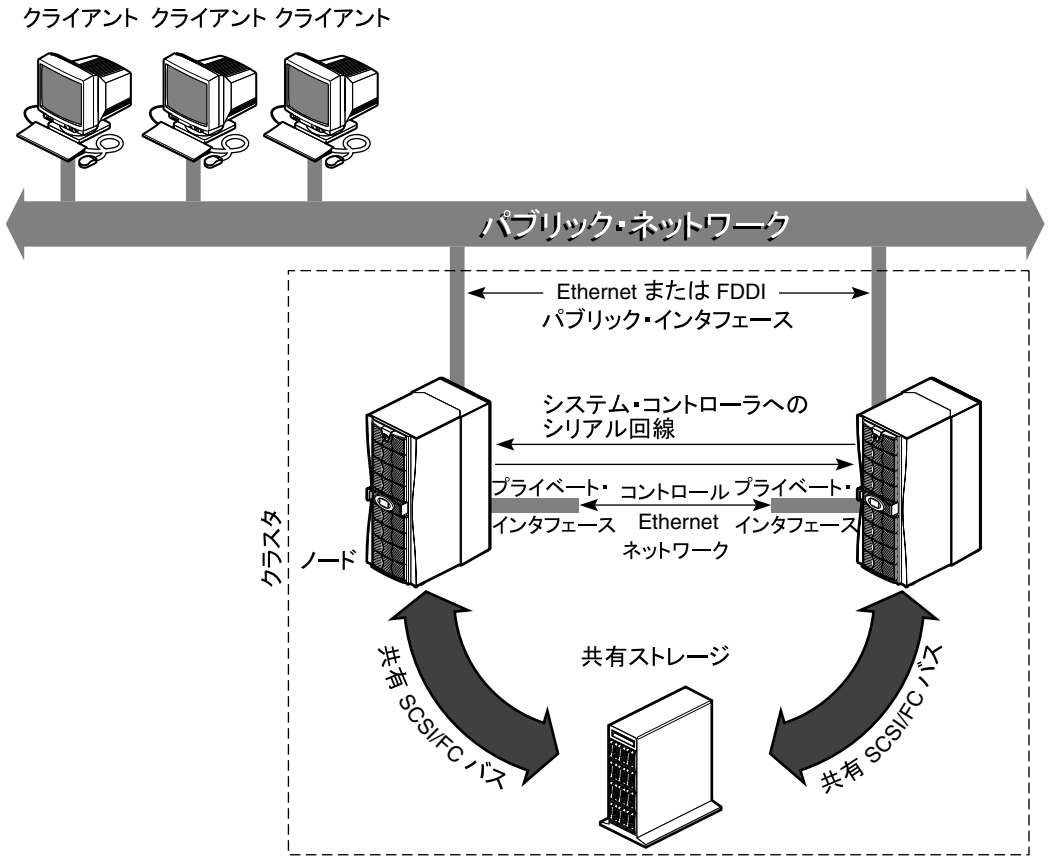


図1-5 システム・コンポーネントのサンプル

ハードウェア・コンポーネントは、以下のとおりです。

- 最高 8 つまでの SGI 2000 ノード、SGI 200 ノード、または SGI Origin 3000 ノード
- 各ノードにコントロール・ネットワーク用の複数のインタフェース

  コントロール・ネットワークのハートビート接続用に、各ノードに少なくとも 2 つの Ethernet インタフェースまたは FDDI インタフェースが必要です。各ノードは、ハートビート接続によってほかのノードの状

態をモニタします。この接続は、FailSafe ソフトウェアがノード間でコントロール・メッセージを渡すためにも使用されます。これらのインタフェースは、異なる IP アドレスを持ちます。

- 各ノードのシリアル・ポートと別のノードのリモート・システム・コントローラ・ポートを接続するシリアル回線

  異常ノードのサービスを引継ぐノードは、この回線を使用して、引継ぎ中に異常ノードを再起動します。この手順により、代替ノードが異常ノードを引継ぐときに、異常ノードが共有ディスクを使用しないようにします。

- クラスタ内のノードによって共有されるディスク・ストレージおよび SCSI バス／ファイバ・チャネル

  FailSafe システムのノードは、共有 Fast-Wide SCSI バスまたはファイバ・チャネル上のデュアルホスト・ディスク・ストレージを共有します。このストレージ接続は、異常が発生した場合にどちらかのノードがディスクを引継ぐことができるようにするために共有されます。ディスク・ストレージに必要なハードウェアは、以下のいずれかになります。

  – Origin SCSI JBOD/RAID

  – Origin FC RAID デスクサイド・ストレージ・システムまたは Origin RAID ラックマウント・ストレージ・システム。各シャーシ・アセンブリには、2 台のストレージ制御プロセッサ (SP: Storage-Control Processor) と、キャッシュが有効なディスク・モジュールが少なくとも 5 つあります。

  – TP9100

  – TP9400

  – TP9900

- クラスタのマシンをリセットするための EL-8+ (`FAILSAFE-N_NODE`) ハードウェア・コンポーネント、あるいはオプションで ST16XX ハードウェア・コンポーネントまたは EL-16 ハードウェア・コンポーネント

さらに、FORE Systems ATM カードを FORE Systems スイッチと共に使用すると、ATM LAN エミュレーション・フェイルオーバーがサポートされます。

メモ: FailSafe は、シングル・ポイント異常を切抜ける目的で設計されています。したがって、1 つのシステム・コンポーネントに異常が発生した場合は、複数のコンポーネントに異常が発生するのを避けるために、できるだけ早くシステム・コンポーネントの再開、修復、または交換を行ってください。

ディスク接続

FailSafe システムでは、以下のディスク接続をサポートします。

- RAID サポート
  - シングル・コントローラまたはデュアル・コントローラ
  - シングル・ハブまたはデュアル・ハブ
  - シングル・パスまたはデュアル・パス
- JBOD サポート
  - シングル・ボールトまたはデュアル・ボールト
  - シングル・ハブまたはデュアル・ハブ

SCSI ディスクは、2 台のマシンにのみ接続できます。ファイバ・チャネル・ディスクは、複数のマシンに接続できます。

サポートされている設定

FailSafe では、以下の高可用性設定をサポートします。

- 基本 2 ノード設定
- 複数のプライマリ・ノードと 1 つのバックアップ・ノードのスター設定
- リング設定

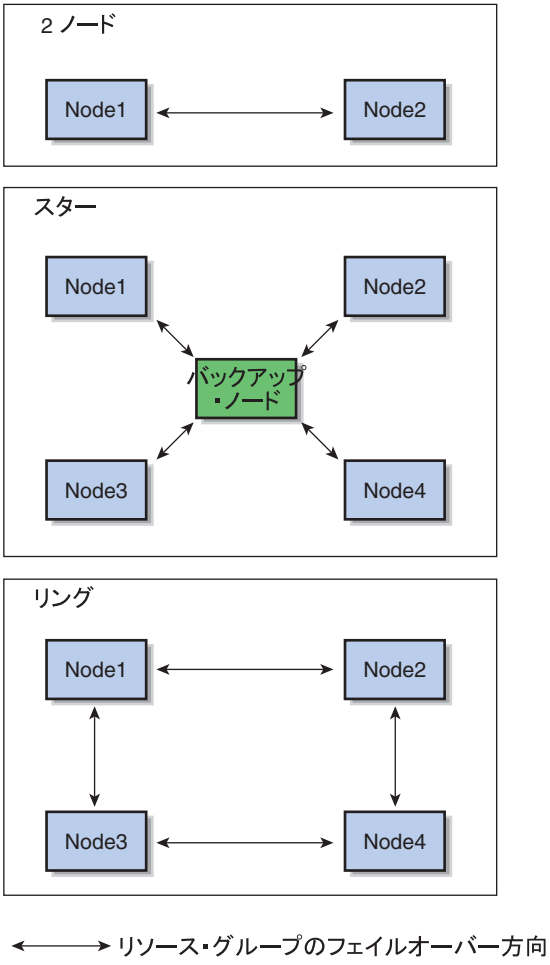


図1-6 設定のタイプ

これらの設定は、プロセッサおよびI/Oコントローラの冗長性を備えています。ストレージの冗長性は、複数ホストRAIDディスク・デバイス、およびプレックス化（ミラー化）されたディスクを使用することによって実現されます。

FailSafeシステムを設定する場合、以下のリセット・モデルを使用できます。

- サーバ間。各サーバは、リセットのために別のサーバに直接接続されます。単方向の場合があります。

- ネットワーク。各サーバは、コントロール・ネットワークを通じて EL-16 multiplexer にシグナルを送信することによって、ほかのサーバをリセットできます。
- IRISconsole。各サーバは、IRISconsole がリセットを実行するよう要求できます。

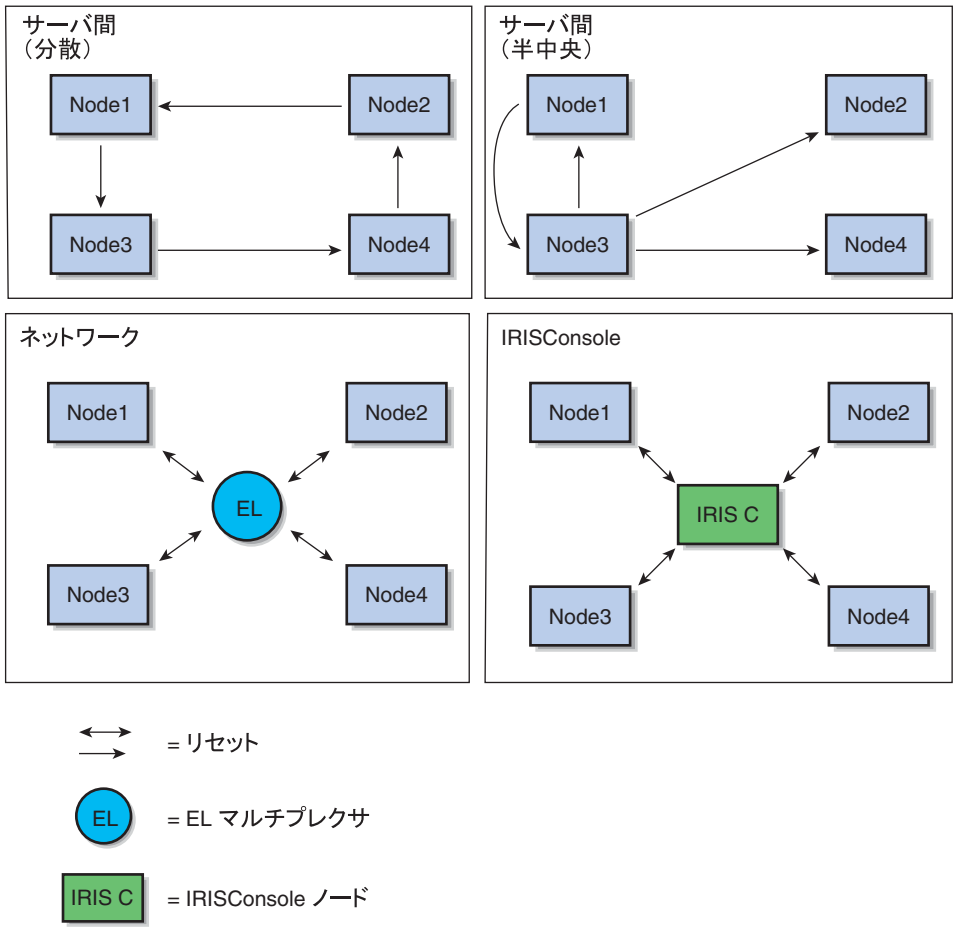


図1-7 リセット・タイプ

基本 2 ノード設定では、以下の処理が可能です。

- すべての高可用性サービスは、1 つのノードで実行されます。もう一方のノードは、バックアップ・ノードになります。フェイルオーバー後は、これらのサービスはバックアップ・ノードで実行されます。この場合、バックアップ・ノードはフェイルオーバー専用のホット・スタンバイです。ただし、高可用性サービスではないその他のアプリケーションをバックアップ・ノードで実行することは可能です。
- 高可用性サービスは、両方のノードで同時に実行されます。各サービスに対して、他方のノードがバックアップ・ノードとして機能します。たとえば、両方のノードが異なる NFS ファイルシステムをエクスポートできます。ただし、フェイルオーバーが発生した場合は、1 つのノードがすべての NFS ファイルシステムをエクスポートします。

追加機能

FailSafe には、高可用性システムをより柔軟かつ簡単に運用するための以下の機能が用意されています。

- 動的管理
- 細粒度フェイルオーバー
- ローカル再開

これらの機能の概要については、以下の節で説明します。

動的管理

FailSafe では、システムの実行中にさまざまな管理タスクを実行できます。

- アプリケーションのモニタ。FailSafe を実行し続けながら、アプリケーションのモニタをオンまたはオフにできます。これにより、FailSafe システムを停止せずにオンラインでアプリケーションをアップグレードできます。
- 管理リソース。FailSystem がオンラインの間にリソースを追加できます。
- FailSafe ソフトウェアのアップグレード。FailSafe クラスタ全体を停止せずに、1 ノードずつ FailSafe ソフトウェアをアップグレードできます。

細粒度フェイルオーバー

フェイルオーバーの単位はリソース・グループです。このため、リソース・グループのコンポーネント異常の影響は、コンポーネントが属するリソース・グループだけに抑えられ、同じノードのほかのリソース・グ

ループやサービスに影響が及ぶことはありません。ほかのリソース・グループを最初のノードで実行し続けながら、特定のリソース・グループをあるノードから別のノードにフェイルオーバーするプロセスを、細粒度フェイルオーバーと呼びます。

ローカル再開

FailSafe では、同じノードにリソース・グループをフェイルオーバーできます。この機能により、可能であれば、特定のアプリケーションのバックアップが同じマシンに作成される単一ノード・システムを設定できます。また、リソース・グループが別のノードにフェイルオーバーする前に、指定した回数のローカル再開が試行されるよう指定することもできます。

管理

すべての FailSafe 管理タスクは、FailSafe マネージャ・グラフィカル・ユーザ・インタフェース (GUI: Graphical User Interface) を使用して実行できます。この GUI には、FailSafe によって制御される高可用性クラスタを設定、管理、およびモニタするためのインタフェース・ガイドが付属しています。また、GUI では、画面ごとに表示されるヘルプ・テキストも用意されています。

管理タスクは、`cmgr`(1M) コマンドを使用して直接実行することもできます。このコマンドを使用すると、管理タスクに対してコマンド・ライン・インタフェースが提供されます。

詳細については、83 ページの 第4章「管理ツール」、99 ページの 第5章「設定」、および215 ページの 第7章「IRIS FailSafe のシステム運用」を参照してください。

高可用性リソース

この節では、FailSafe システムの以下の高可用性リソースについて説明します。

- ノード
- ネットワーク・インタフェースおよび IP アドレス
- ディスク

ノード

(たとえば、パリティ・エラーやバス・エラーなどが原因で) ノードがクラッシュまたはハングすると、FailSafe ソフトウェアによって検出されます。フェイルオーバー・ポリシーによって決定される別のノードは、異常発生ノードをリセットした後、その異常ノードのサービスを引継ぎます。

ノードに異常が発生した場合は、インタフェース、ストレージへのアクセス、およびサービスも使用できなくなります。FailSafe システムがこれらの異常ポイントを処理または取除く方法についての説明は、以降の各節を参照してください。

ネットワーク・インタフェースおよび IP アドレス

クライアントは、FailSafe クラスタが提供する高可用性サービスに IP アドレスを使用してアクセスします。各高可用性サービスが複数の IP アドレスを使用できます。IP アドレスは特定の高可用性サービスに関連付けられていないので、リソース・グループ内のすべてのリソースで共有できます。

FailSafe では、IP エイリアス・メカニズムを使用して、単一のネットワーク・インタフェースで複数の IP アドレスをサポートしています。クライアントは、サーバ・ノードにネットワーク・インタフェースが 1 つしかない場合でも、複数の IP アドレスを使用する高可用性サービスを使用できます。

IP エイリアス・メカニズムでは、複数のネットワーク・インタフェースを使用するノードがある FailSafe 設定を、単一のネットワーク・インタフェースを使用するノードでバックアップできます。異常が発生した場合、複数のネットワーク・インタフェースに設定されている IP アドレスは、他方のノードの単一のインタフェースに移動されます。

メモ: つまり、ホスト名は、移動されることのない IP アドレスにバインドされています。

FailSafe では、クラスタ内の各ネットワーク・インタフェースに、フェイルオーバーしない IP アドレスが必要です。固定 IP アドレスと呼ばれるこれらの IP アドレスは、ネットワーク・インタフェースをモニタするために使用されます。固定 IP アドレスは、FailSafe がインストールされる前に通常のシステムとして設定してネットワーク上に置いた場合に使用するアドレスと同じになります。

各固定 IP アドレスは、システムの起動時にネットワーク・インタフェースに対して設定される必要があります。クラスタ内のその他の IP アドレスは、すべて高可用性 (HA) IP アドレスとして設定されます。.

高可用性 IP アドレスは、ネットワーク・インタフェースに設定されます。これらのアドレスは、フェイルオーバーおよび回復プロセス中に FailSafe によって別のノードの別のネットワーク・インタフェースに移動されます。高可用性 IP アドレスは、FailSafe システムを設定するときに指定します。FailSafe では、`ifconfig` コマンドを使用して、ネットワーク・インタフェースに IP アドレスを設定したり、あるインタフェースから別のインタフェースに IP アドレスを移動します。

ネットワークの実装によっては、`ifconfig` コマンドだけでは IP アドレスをあるインタフェースから別のインタフェースに移動できない場合があります。FailSafe では、MAC (Medium Access Control) アドレス偽装（再 MAC）を使用して、これらのネットワーク実装をサポートします。再 MAC により、ネットワーク・インタフェースの物理 MAC アドレスが別のインタフェースに移動されます。これは、`macconfig` コマンドを使用して行われます。再 MAC は、FailSafe が IP アドレスの移動に使用する標準の `ifconfig` プロセスとは別に実行されます。再 MAC では、各 MAC アドレスに対して、パブリック・ネットワークへのネットワーク接続が 2 つ必要です。移動される各 MAC アドレスに対しては、専用のバックアップ・ネットワー

ク・インタフェースが 1 つ必要です。FailSafe で再 MAC が実行される場合は、タイプ `MAC_Address` のリソースが使用されます。

メモ: 再 MAC を使用できるのは Ethernet ネットワークだけです。FDDI ネットワークでは使用できません。

再 MAC は、*Gratuitous ARP パケット*というパケットがネットワークを通して渡されない場合に必要です。これらのパケットは、(フェイルオーバー・プロセスなどで) IP アドレスがインタフェースに追加されたときに自動的に生成され、IP アドレスと MAC アドレスの新しいマッピングがアナウンスされます。これにより、特定のインタフェースに特定の IP アドレスがあることが、ローカル・サブネットのクライアントに通知されます。続いて、クライアントは、IP アドレスの新しい MAC アドレスで内部 ARP キャッシュを更新します。IP アドレスは、単にインタフェースからインタフェースへと移動しただけです。Gratuitous ARP パケットがネットワークを通して渡されない場合、サブネット・クライアントの内部 ARP キャッシュは更新できません。このような場合に再 MAC が使用されます。これにより、元のインタフェースの MAC アドレスは新しいインタフェースに移動されます。したがって、IP アドレスと MAC アドレスの両方が新しいインタフェースに移動され、クライアントの内部 ARP キャッシュの更新は必要ありません。

再 MAC はデフォルトでは実行されないため、再 MAC が必要なプライマリ・インタフェースとセカンダリ・インタフェースの各ペアに対して実行されるよう指定しなければなりません。再 MAC が必要かどうかを判断する方法については、43 ページの「ネットワーク・インタフェースと HA IP アドレスの設定計画」の手順で説明されています。一般的には、ルータおよび PC/NFS クライアントではインタフェースの再 MAC が必要です。

再 MAC の結果、新しい MAC アドレスを受信したインタフェースの元の MAC アドレスは使用できなくなります。このため、各ネットワーク・インタフェースは、専用のバックアップ・インタフェースによってバックアップされる必要があります。このバックアップ・インタフェースは、クライアントがプライマリ・インタフェースとして使用することはできません。このインタフェースへのフェイルオーバーの後は、元の MAC アドレスに送信されたパケットは、ネットワーク上のすべてのノードで無視されます。各バックアップ・インタフェースでバックアップされるネットワーク・インタフェースの数は、1 つだけです。

ディスク

FailSafe システムでは、SCSI またはファイバ・チャネルに基づいてストレージをサポートします。

XLV プレキシングは、JBOD 設定でディスクをミラー化するために必要になります。高可用性アプリケーションがファイルシステムを使用する場合は、XFS ファイルシステムまたは CXFS ファイルシステムを使用する必要があります。CXFS ファイルシステムを使用する場合は、XVM ボリューム上になければなりません。

メモ: SAN 設定では SCSI ストレージも FIBRE JBOD もサポートされないため、CXFS は使用できません。

ストレージ・コンポーネントには、シングル・ポイント異常があってはなりません。すべてのデータは、RAID に存在するか、XLV を使用してミラー化する必要があります。冗長性を持たせるため、ストレージからサーバへのパスは少なくとも 2 つあることが推奨されます。

ファイバ・チャネル RAID ストレージ・システムの場合、ディスクまたはディスク・コントローラに異常が発生したときは、RAID ストレージ・システムの独自の機能によって引続きサービスを使用できるようになっています。

上記のすべてのストレージ・システムの場合で、ディスクまたはディスク・コントローラが失敗したときに、XLV または XVM の冗長なパスによって適宜サービスを引続き使用できるようになっています。

ストレージ・サブシステムに使用できる代替パスがない場合は、FailSafe によってフェイルオーバー・プロセスが開始されます。

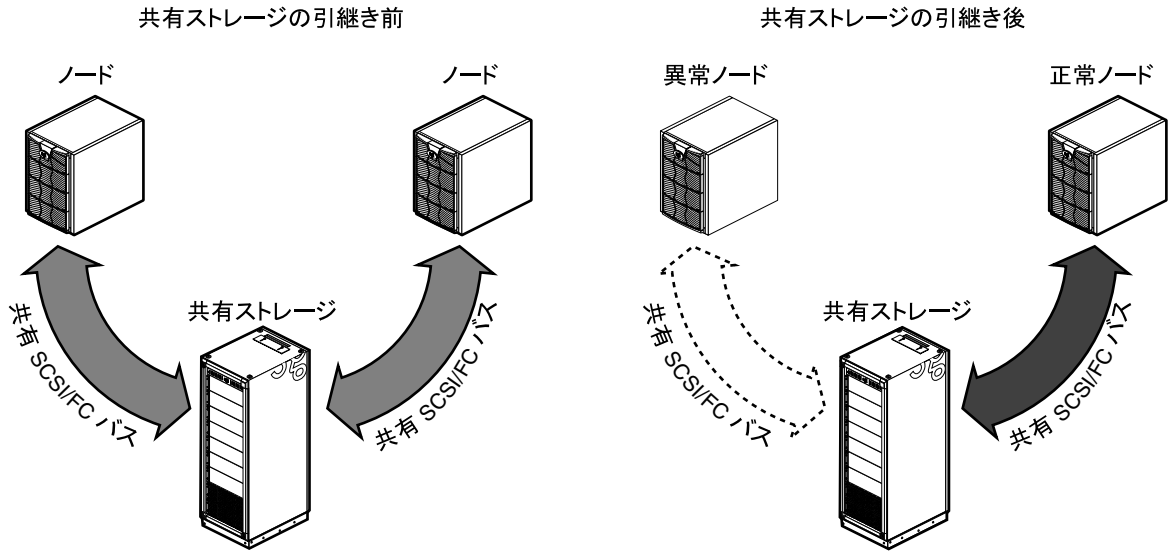


図1-8 2 ノード・システムにおけるディスク・ストレージ・フェイルオーバー

高可用性アプリケーション

各アプリケーションには、プライマリ・ノードが 1 つと、定義したフェイルオーバー・ポリシーに基づいてバックアップ・ノードとして使用できる最高 7 つまでの追加ノードがあります。プライマリ・ノードは、FailSafe が通常の状態である場合にアプリケーションを実行するノードです。FailSafe ソフトウェアによって高可用性アプリケーションの異常が検出されると、異常ノードで影響を受けるリソース・グループのすべてのリ

ソースが別のノードにフェイルオーバーされ、その異常ノードのリソースは停止されます。これらの操作が完了すると、リソースはバックアップ・ノードで開始されます。

プライマリ・ノード、リソース・グループのコンポーネント、フェイルオーバー・ポリシーなど、リソースに関するすべての情報は、GUI または `cmgr`(1M) を使用して FailSafe システムを設定するときに指定します。システムの設定についての詳細は、99 ページの 第5章「設定」を参照してください。リソースの異常は、モニタ・スクリプトによって検出されます。

FailSafe ソフトウェアには、アプリケーションを高可用性サービスにするためのフレームワークが用意されています。スクリプトを作成し、それらのスクリプトに基づいてシステムを設定することで、クライアント/サーバ・アプリケーションを高可用性アプリケーションにすることができます。詳細については、『IRIS FailSafe Version 2 Programmer's Guide』を参照してください。

フェイルオーバー・プロセスおよび回復プロセス

異常とは、ノードのクラッシュ、ハング、またはシャットダウンが発生した場合や、高可用性サービスが動作しなくなっている場合のことです。異常のあるノードを異常ノードと呼びます。別のノードによって、異常ノードで提供されている高可用性サービスのフェイルオーバーが実行されます。フェイルオーバーにより、異常ノードで提供されていた高可用性サービスを含むすべての高可用性サービスを、クラスタ内で使用可能な状態に保つことができます。

異常後のアクションの順序は、どのノードが異常を検出したかによって異なります。

同じノードで実行されている FailSafe ソフトウェアによって異常が検出された場合は、異常ノードで以下の操作が実行されます。

- 異常ノードで実行されている高可用性リソース・グループを停止する
- 定義されているフェイルオーバー・ポリシーに基づいて、高可用性リソース・グループを別のノードに移動する
- 以前に異常ノードによって提供されていたすべてのリソース・グループ・サービスの提供を開始するよう、新しいノードに要求する

リソース・グループを引継ぐノードでは、メッセージが受信されると、以下の操作が実行されます。

- リソース・グループの所有権を異常ノードから自分に転送する
- 異常ノードで実行されていたリソース・グループ・サービスの提供を開始する

別のノードで実行されている FailSafe ソフトウェアによって異常が検出された場合は、異常を検出したノードで以下の操作が実行されます。

- ノード間のシリアル接続を使用して、異常ノードの電源サイクルを行う(データの破壊を防ぐため)

- リソース・グループのフェイルオーバー・ポリシーに基づいて、そのリソース・グループの所有権を異常ノードからクラスタ内のほかのノードに転送する
- 異常ノードで実行されていたリソース・グループ・サービスの提供を開始する

異常ノードが正常な動作に戻ったときに自動的に高可用性サービスの提供を再開するかどうかは、定義されているフェイルオーバー・ポリシーによります。

詳細については、174 ページの「GUI を使ったフェイルオーバー・ポリシーの定義」を参照してください。

通常、異常の発生したノードは、自動的に再起動して高可用性サービスの提供を再開します。これは、一時的なエラー（および機器やソフトウェアのアップグレードのための計画的な機能停止）の場合に役立ちます。

スタートアップ中およびフェイルオーバー中の FailSafe の実行についての詳細は、303 ページの「FailSafe のアクション・スクリプトおよびフェイルオーバー・スクリプトの実行」を参照してください。

新しいクラスタの設定およびテストの概要

FailSafe クラスタ・ハードウェアをインストールしたら、次の一般的な手順に従って FailSafe システムを設定およびテストします。

1. この章を参照して、FailSafe の用語について理解します。
2. 第2章「設定計画」を参照して、クラスタでの高可用性アプリケーションと高可用性サービスの設定を計画します。
3. 第3章「インストールとシステムの準備」で説明されているとおりに、FailSafe で必要なさまざまな管理タスク（必須ソフトウェアのインストールなど）を実行します。
4. 99 ページの 第5章「設定」で説明されているとおりに、設定を定義します。
5. 以下の 3 段階でシステムをテストします。
   - FailSafe ソフトウェアを開始する前に個々のコンポーネントをテストする
   - システムの通常の動作をテストする
   - 異常をシミュレートして異常発生後のシステムの動作をテストする

設定計画

この章では、IRIS FailSafe クラスタでの高可用性 (HA) サービスの設定の計画方法について説明します。この章の主な節は、以下のとおりです。

- 「概要」
- 31 ページの「ディスク設定」
- 37 ページの「論理ボリューム設定」
- 40 ページの「ファイルシステム設定」
- 43 ページの「HA IP アドレス設定」
- 46 ページの「CXFS と IRIS FailSafe の同時実行」

概要

最初に、クラスタをどのように使用するかを決める必要があります。その後、クラスタで提供する高可用性 (HA) サービスのニーズに合わせてディスクとインタフェースを設定できます。

回答すべき質問

計画プロセスでは、以下の質問に対する答を用意しておく必要があります。

- ノードをどのように使用するか

  回答としては、ユーザへのホーム・ディレクトリの提供、Oracle データベースをサポートする特定のアプリケーションの実行、Netscape Web サービスの提供、ファイル・サービスの提供などの用途が挙げられます。

- 上記のどの用途を HA サービスとして提供するか

  SGI では、一部の高可用性アプリケーション向けに FailSafe ソフトウェア・オプションを開発しています。293 ページの「ソフトウェア階層」を参照してください。ほかのアプリケーションを HA サービスとして提供するには、『IRIS FailSafe Version 2 Programmer's Guide』で説明されているように、アプリケーションをモニタするシェル・スクリプトのセットを開発する必要があります。スクリプトの開発についてサポートが必要な場合は、日本 SGI の営業担当員、もしくは営業フリーダイヤルまでお問合わせください。

- 各 HA サービスに対してどのノードをプライマリ・ノードにするか

  プライマリ・ノードとは、サービス(ファイルシステムのエクスポート、Netscape サーバとしての使用、データベースの提供など)を提供するノードです。

- 各 HA サービスに対して、ソフトウェアとデータをどのように共有ディスクと非共有ディスクに分散するか

  フェイルオーバーされるディスク(共有ディスク)またはフェイルオーバーされないディスク(非共有ディスク)のどちらにソフトウェアを配置するかについては、各アプリケーションごとに必要条件と選択肢があります。

- 共有ディスクを RAID ストレージ・システムの一部にするか、それともプレックス化された XLV 論理ボリュームを持つ SCSI/ファイバ・チャネル・ストレージのディスクにするか

  共有ディスクは、RAID ストレージ・システムの一部にするか、またはプレックス化された XLV 論理ボリュームを持つ SCSI/ファイバ・チャネル・ディスク・ストレージに含める必要があります。

- 以下のどの方法で共有ディスクを設定するか

  – raw XLV 論理ボリュームとして。

  – XFS ファイルシステムが存在する XLV 論理ボリュームとして。

  – XVM 論理ボリュームを使用する CXFS ファイルシステムとして。FailSafe および CXFS の使用についての詳細は、46 ページの「CXFS と IRIS FailSafe の同時実行」を参照してください。

  どのボリュームやファイルシステムを選択するかは、そのディスク容量を使用するアプリケーションによって決まります。

- HA サービスのクライアントがどの IP アドレスを使用するか

  ノードを複数のネットワークに接続する場合や、1 つのネットワークとの複数のインタフェースが存在する場合があるので、各ノードで複数のインタフェースが必要になる場合があります。

- どのリソースをリソース・グループの一部にするか

  相互に依存するリソースはすべて同じリソース・グループに存在する必要があります。

- リソース・グループのフェイルオーバー・ドメイン (フェイルオーバー・ドメインについての詳細は、11 ページの「フェイルオーバー・ドメイン」を参照してください)。

  リソース・グループが存在できるクラスタ内のノードのリストは、フェイルオーバー・ドメインで指定されます。たとえば、リソース・グループの一部であるボリューム・リソースは、そのボリュームを構成するディスクにアクセスできるノードだけに存在できます。

- 高可用性 (HA) サービスのクライアントが使用できる各ネットワーク・インタフェースの HA IP アドレスの数

HA サービスのクライアントが使用する各ノード上のインタフェースに対して、少なくとも 1 つの HA IP アドレスが必要です。

- HA サービスのクライアントがフェイルオーバー・ドメインのノードのどの HA IP アドレスを使用できるようにするか
- フェイルオーバー・ドメインのノードにあり、HA サービスのクライアントが使用可能な各 HA IP アドレスについて、フェイルオーバー後に、その他のどのノードにその IP アドレスを割当てるか

HA サービスが使用するすべての HA IP アドレスは、リソース・グループ・サービスを引継ぐことができる各ノードの少なくとも 1 つのインタフェースにマップされている必要があります。HA IP アドレスは、リソース・グループのプライマリ・ノードのインタフェースから代替ノードのインタフェースにフェイルオーバーされます。

例

設定計画プロセスの例として、部門サーバである 2 ノードの FailSafe クラスタがある場合を取上げてみます。NFS のマウント用に 4 つの XFS ファイルシステムが使用できるようにして、それぞれが異なるセットのドキュメントを提供する 2 つの Netscape FastTrack サーバを使用したいと考えています。これらのアプリケーションは HA サービスになります。

各ノードが 2 つのファイルシステムと 1 つの Netscape サーバのプライマリ・ノードになるよう、2 つのノードにサービスを分散することを決めました。これらのファイルシステムと、(XFS ファイルシステム上の) Netscape サーバのドキュメントのルートは、それぞれプレックス化された専用の XLV 論理ボリューム上にあります。論理ボリュームは、両方のノードに接続されたファイバ・チャネル・ストレージ・システム内のディスクから作成されています。

リソース・グループは以下の 4 つです。

- `NFSgroup1`
- `NFSgroup2`
- `Webgroup1`
- `Webgroup2`

`NFSgroup1` と `NFSgroup2` が NFS リソース・グループで、 `Webgroup1` と `Webgroup2` が Web リソース・グループです。一方のノードが `NFSgroup1` と `Webgroup1` のプライマリ・ノードになり、もう一方のノードが `NFSgroup2` と `Webgroup2` のプライマリ・ノードになります。

各ノードでは、`ef0` と `ef1` の 2 つのネットワークが使用可能です。各ノードの `ef1` インタフェースは相互に接続されており、プライベート・ネットワークを形成しています。

図2-1 に、この設定を示します。

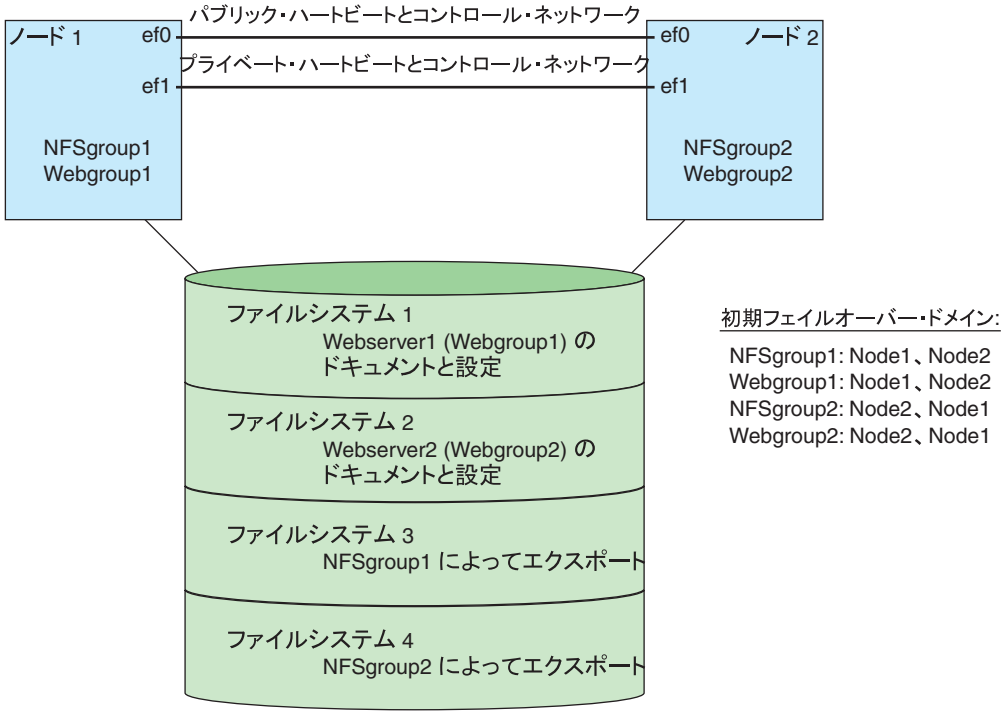


図2-1 4 つのリソース・グループでの設定例

ディスク設定

この節では、以下の内容について説明します。

- 「ディスク設定の計画」
- 37 ページの「ディスク設定パラメータ」

ディスク設定の計画

FailSafe クラスタ内の各ディスクに対して、共有ディスク(フェイルオーバーが可能)または非共有ディスクのどちらにするかを選択する必要があります。非共有ディスクはフェイルオーバーされません。

FailSafe クラスタ内のノードは、以下の必要条件に従う必要があります。

- システム・ディスクは非共有ディスクでなければなりません。
- FailSafe ソフトウェアは非共有ディスク上に存在しなければなりません。
- すべてのシステム・ディレクトリ(/tmp、 /var、/usr、/bin、 /dev など)は、非共有ディスクに存在する必要があります。

共有ディスクに配置できるのは、HA アプリケーション・データと設定データだけです。ディスクを共有または非共有のどちらにするかは、そのディスクを使用する HA サービスのニーズに応じて決まります。各 HA サービスには、サービスと関連付けられているデータの場所について以下の必要条件があります。

- 一部のデータは非共有ディスクに配置する必要があります。
- 一部のデータは共有ディスクには配置できません。
- 一部のデータは共有ディスクまたは非共有ディスクのどちらにも配置できます。

この項の後半の図に、フェイルオーバー前後の FailSafe クラスタの基本ディスク設定を示します。クラスタには、以下の基本ディスク設定を組合わせて含めることができます。

- 各ノード上の非共有ディスク
- Web サーバおよび NFS ファイル・サーバ・ドキュメントが含まれる複数の共有ディスク

メモ: フェイルオーバーの前後の各図には、1 つまたは 2 つのディスクだけが示されています。実際には、図中の各ディスクと同じ方法で多くのディスクを接続できます。したがって、各ディスクは複数のディスクのセットを表していると考えてかまいません。

図2-2 に、クラスタの 2 つのノードを示します。各ノードには、リソース・グループが含まれる非共有ディスクが 1 つあります。HA アプリケーションが非共有ディスクを使用する場合は、これらのアプリケーション

に必要なデータを両方のノードの非共有ディスクに複製しなければなりません。クライアントは、HA IP エイリアスを使用して共有ディスク内のデータにアクセスする必要があります。フェイルオーバーが実行されると、HA IP エイリアスがフェイルオーバーされます。

メモ: ホスト名は、移動されることのない IP アドレスにバインドされています。

最初に異常ノードにあったデータは、この HA IP エイリアスを使用してアクセスすることで、引続き代替ノードから使用できます。

図2-2 の設定には、以下の 2 つのリソース・グループが含まれています。

リソース・グループ	リソース・タイプ	リソース
Group1	IP_address	192.26.50.1
Group2	IP_address	192.26.50.2

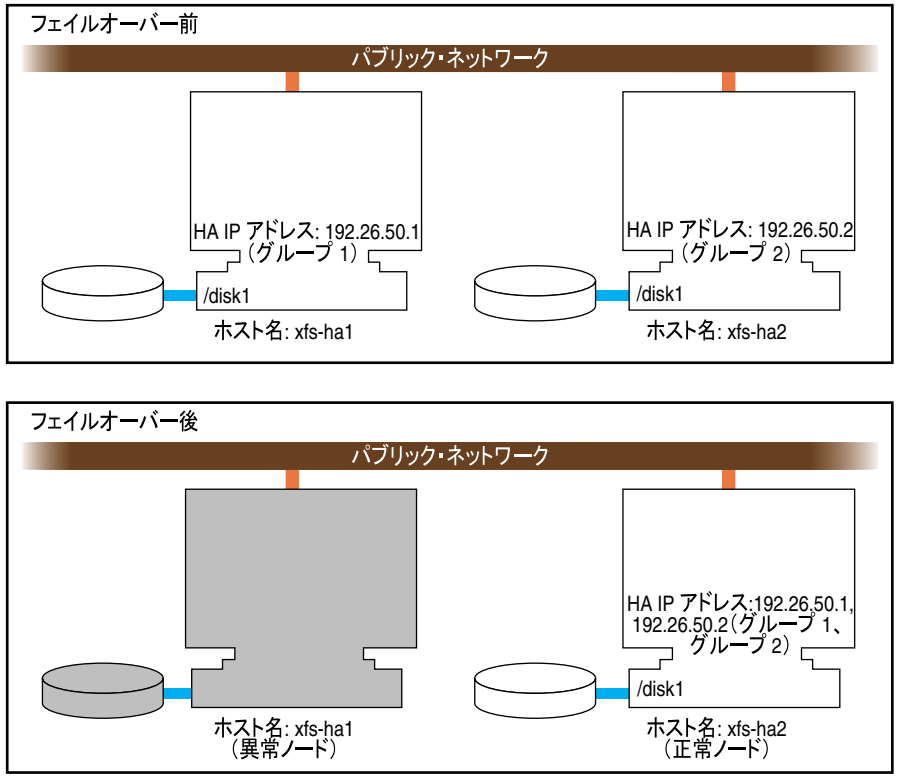


図2-2 非共有ディスク設定とフェイルオーバー

図2-3 に、1 つのリソース・グループ Group1 がある 2 ノード設定を示します。

リソース・グループ	リソース・タイプ	リソース	フェイルオーバー・ドメイン
Group1	IP_address	192.26.50.1	(xfs-ha1, xfs-ha2)
	filesystem	/shared	
	volume	shared_vol	

この設定では、フェイルオーバー前にディスクにアクセスするノードであるプライマリ・ノードは、リソース・グループ Group1 にあります。このノードとの接続は、実線で示されています。フェイルオーバー後にこのディスクにアクセスするバックアップ・ノードとの接続は、点線で示されています。したがって、このディスクはこれらのノードで共有されていることになります。アクティブ／バックアップ設定では、すべてのリソース・グループが同じプライマリ・ノードを使用します。バックアップ・ノードでは、フェイルオーバーが実行されるまで HA リソース・グループは実行されません。

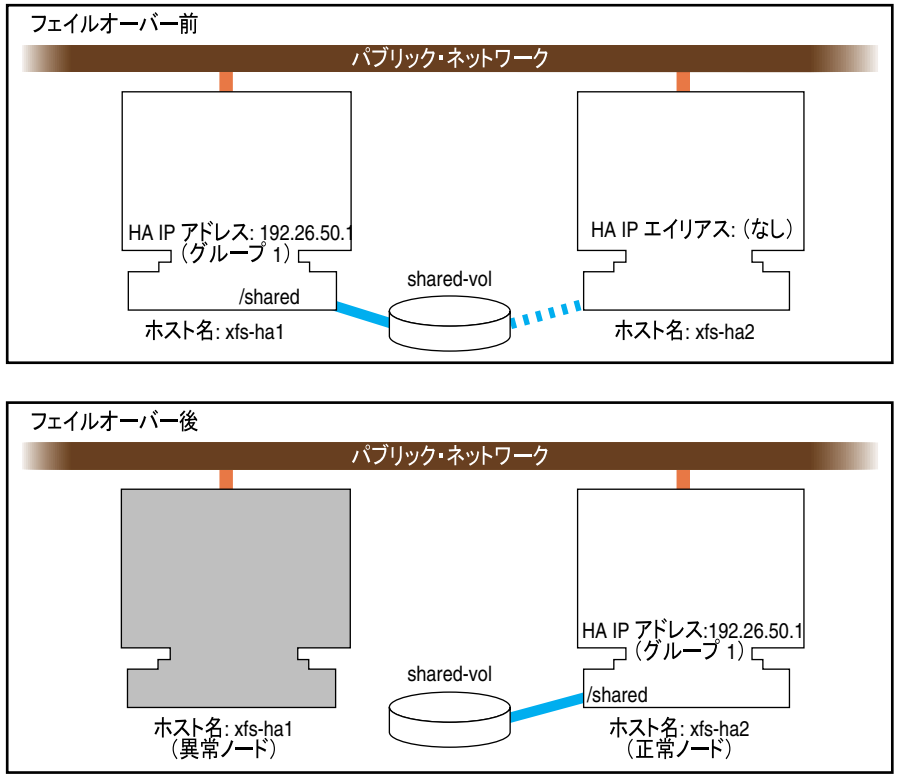


図2-3 アクティブ／バックアップ設定時の共有ディスク設定

図2-4 に、Group1 と Group2 の 2 つのリソース・グループがある 2 ノード・クラスタの 2 つの共有ディスクを示します。

リソース・グループ	リソース・タイプ	リソース	フェイルオーバー・ドメイン
Group1	IP_address	192.26.50.1	(xfs-ha1, xfs-ha2)
	filesystem	/shared1	
	volume	shared1_vol	
Group2	IP_address	192.26.50.2	(xfs-ha2, xfs-ha1)

リソース・グループ	リソース・タイプ	リソース	フェイルオーバー・ドメイン
	filesystem	/shared2	
	volume	shared2_vol	

この設定では、各ノードが 1 つのリソース・グループのプライマリ・ノードとして動作します。実線はフェイルオーバー前のプライマリ・ノードとの接続を示し、点線はバックアップ・ノードとの接続を示します。フェイルオーバー後は、正常ノードがすべてのリソース・グループを持ちます。

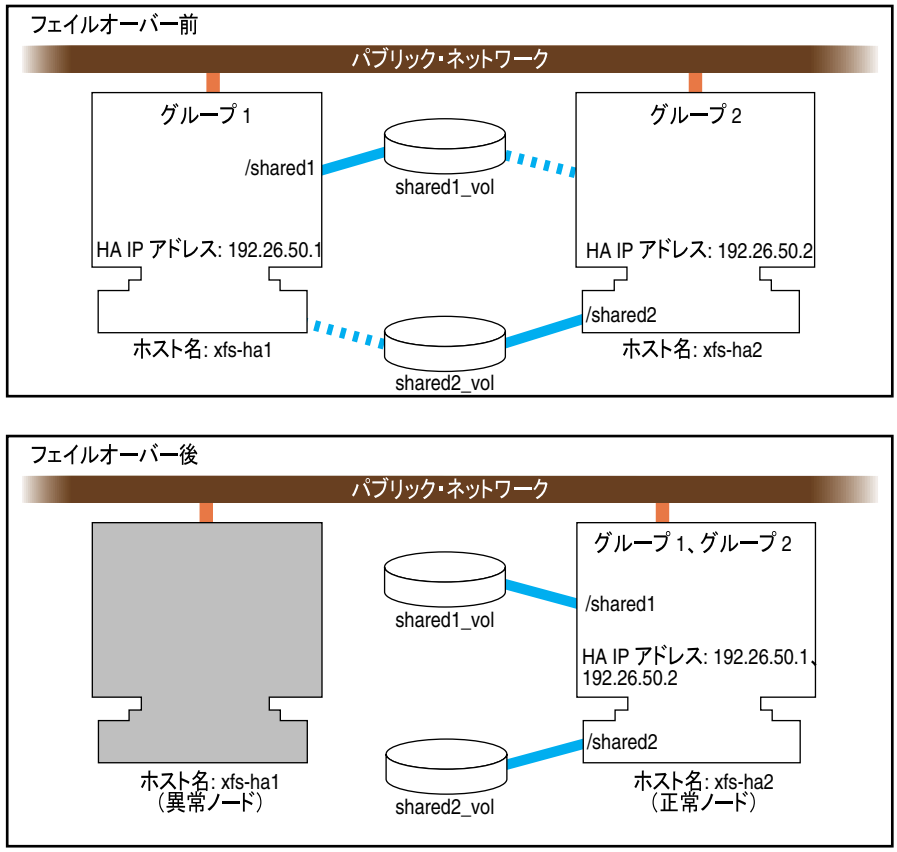


図2-4 デュアルアクティブ設定時の共有ディスク設定

各 HA サービスに関連付けられているさまざまな種類のデータに対して共有ディスクまたは非共有ディスクのどちらを選択するかについては、この章のその他の節や、『IRIS FailSafe 2.0 Oracle Administrator's

Guide』と『IRIS FailSafe 2.0 INFORMIX Administrator's Guide』の同様の節でさらに詳しく説明されています。

ディスク設定パラメータ

非共有ディスクに関連付けられている設定パラメータはなく、FailSafe システムの設定時に指定するものはありません。設定時には、共有ディスク上の XLV 論理ボリュームだけを指定します。詳細については、40 ページの「論理ボリュームの設定パラメータ」を参照してください。

FailSafe 設定で、(XVM 論理ボリュームを使用する) CXFS ファイルシステムの使用についての詳細は、46 ページの「CXFS と IRIS FailSafe の同時実行」を参照してください。

論理ボリューム設定

メモ: この節では、XLV 論理ボリュームを使用した論理ボリューム設定について説明します。(XVM 論理ボリュームを使用する) FailSafe ファイルシステムおよび CXFS ファイルシステムの共同実行についての詳細は、46 ページの「CXFS と IRIS FailSafe の同時実行」を参照してください。

この節では、以下の内容について説明します。

- 「XLV 論理ボリュームの計画」
- 39 ページの「論理ボリューム設定の例」
- 40 ページの「論理ボリュームの設定パラメータ」

XLV 論理ボリュームの計画

すべての共有ディスクには XLV 論理ボリュームが含まれている必要があります。共有ディスク上の XLV 論理ボリュームは、それ以外のディスクと同様に操作できますが、以下の規則に従う必要があります。

- HA アプリケーションが使用する共有ディスク上のすべてのデータは、XLV 論理ボリュームに格納されている必要があります。
- 単一の物理ボリューム上に複数の XLV ボリュームを作成する場合は、これらのすべてのボリュームが同じノードによって所有されている必要があります。たとえば、2 つの XLV ボリュームの一部である 2 つのパーティションがディスクにある場合は、両方の XLV ボリュームが同じリソース・グループの一部でなければなりません (XLV ボリュームの所有権についての詳細は、64 ページの「XLV 論理ボリュームおよび XFS ファイルシステムの作成」を参照してください)。

- ファイバ・チャネルまたはSCSIボールトの各ディスク、あるいはRAID論理ユニット番号 (LUN: logical unit number) は、1つのリソース・グループの一部である必要があります。したがって、ファイバ・チャネルまたはSCSIボールトのディスクとRAID LUNは、各リソース・グループに対して1つのセットに分割しなければなりません。ファイバ・チャネルまたはSCSIボールトのディスク、あるいはRAID LUN上に複数のボリュームを作成する場合は、該当するすべてのボリュームを1つのリソース・グループの一部にしてください。
- 複数のノードから同時に共有XLVボリュームにアクセスしないでください。同時にマウントすると、データが破壊されます。

FailSafeソフトウェアは、正常に動作するためにXLVの命名スキームに依存します。完全修飾XLVボリューム名では、以下のいずれかの形式を使用します。

pathname`/`*volname*
pathname`/`*nodename*`.`*volname*

以下に各構成要素を説明します。

- *pathname*は`/dev/xlv`です。
- *nodename*は、デフォルトでは、ボリュームが作成されているノードのホスト名と同じです。
- *volname*は、ボリュームの作成時に指定した名前です。通常、この構成要素は、いずれかのXLVツールでボリュームを操作する場合に使用します。

たとえば、共有ディスク上のディスク・パーティションを使用してボリューム`vol1`をノード`ha1`に作成した場合、このボリュームのrawキャラクタ・デバイス名は、`/dev/rxlv/vol1`になります。ただし、ピアの`ha2`上では、同じrawキャラクタ・ボリュームが`/dev/rxlv/ha1.vol1`と表示されます。この場合、`ha1`が*nodename*構成要素で、`vol1`が*volname*構成要素です。この例からわかるように、*nodename*構成要素がローカル・ホスト名と同じであると、この構成要素はデバイス・ノード名の一部として表示されません。

各ディスクまたはLUNのボリューム・ヘッダには、1つの*nodename*が格納されます。このため、*nodename*構成要素は、単一のディスク上に複数のボリューム要素を持つすべてのボリュームで同じになります。

注意: この規則に従わないと、FailSafeは正常に動作しません。

ボリューム・ヘッダの*nodename*構成要素は、フェイルオーバーおよび回復処理中にボリュームがノード間で移動されるときに、FailSafeによって変更されます。`xlv_assemble`コマンドで構築されるのは*nodename*がローカル・ホスト名に一致するボリュームだけなので、この点は重要です。その他のXLVユーティリティの中には、ボリュームを所有しているノードに関係なく、すべてのノードを参照(および変更)できるものもあります。

`volume`リソース・タイプのリソースのリソース名は、XLVボリューム名になります。

XLV 論理ボリュームを raw ボリューム（つまりファイルシステムなし）としてデータベース・データの格納に使用する場合、データベース・システムでは、/dev/xlv のデバイス名で特定のオーナー、グループ、およびモードが指定されていなければならないことがあります。XLV 論理ボリューム・デバイス名にデフォルト値と異なるオーナー、グループ、およびモードを指定する必要があるかどうかを判断するには、データベース・ベンダーが提供するドキュメントを参照してください（デフォルトはそれぞれ root、sys、および 0600 です）。

論理ボリューム設定の例

論理ボリューム設定の例として、Disk1 ～Disk5 という名前のディスク上に以下の論理ボリュームがある場合を取上げます。

- /dev/xlv/VolA（ボリューム A）には、 Disk1、および Disk2 の一部が含まれます。
- /dev/xlv/VolB（ボリューム B）には、Disk2 の残りと Disk3 が含まれます。
- /dev/xlv/VolC（ボリューム C）には、 Disk4 と Disk5 が含まれます。

VolA と VolB は 1 つのディスクを共有しているので、同じリソース・グループの一部でなければなりません。VolC は、任意のリソース・グループの一部にできます。図2-5 に、これを説明します。

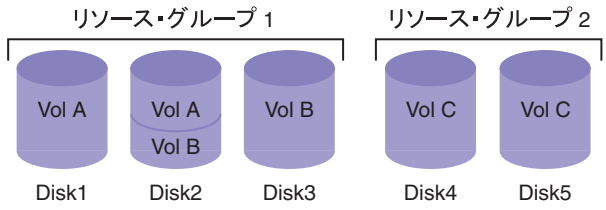


図2-5 論理ボリューム設定の例

論理ボリュームの設定パラメータ

以下に、XLV 論理ボリューム・リストの設定パラメータを示します。

- デバイス・ファイルのオーナー（デフォルト値: root ）
- デバイスのグループ（デフォルト値: sys）
- デバイスのモード（デフォルト値: 0600）

表2-1 に、個々の論理ボリュームのラベルとパラメータを示します。

表2-1 XLV 論理ボリュームの設定パラメータ

リソース属性	VolA	VolB	VolC	コメント
devname-owner	root	root	root	デバイス名のオーナー
devname-group	sys	sys	root	デバイス名のグループ
devname-mode	0600	0600	0600	デバイス名のモード

XLV 論理ボリュームの作成についての詳細は、64 ページの「XLV 論理ボリュームおよび XFS ファイルシステムの作成」を参照してください。

ファイルシステム設定

この節で説明するのは、XFS ファイルシステムを使用する FailSafe 用のファイルシステム設定です。FailSafe ファイルシステムと CXFS ファイルシステムの同時実行についての詳細は、46 ページの「CXFS と IRIS FailSafe の同時実行」を参照してください。

ファイルシステムの計画

FailSafe は、共有ディスク上の XFS ファイルシステムのフェイルオーバーをサポートしています。共有ディスクは、FailSafe クラスタのノードで共有されている、ファイバ・チャネルまたは SCSI JBOD、あるいは RAID のストレージ・システムに存在する必要があります。ファイバ・チャネルおよび SCSI JBOD のストレージ・システムでは、XLV ミラー化を使用しなければなりません。

以下に、FailSafe クラスタの共有ディスク上のファイルシステムで作業する場合に理解しておく必要がある特別な問題点を挙げます。

- フェイルオーバーされるすべてのファイルシステムは、XFS ファイルシステムでなければなりません。

- フェイルオーバーされるすべてのファイルシステムは、共有ディスク上の XLV 論理ボリュームに作成しなければなりません。
- 可用性を実現するには、クラスタ内のフェイルオーバーされるファイルシステムは、XLV プレキシング・ソフトウェアを使用してミラー・ディスク上に作成するか、またはファイバ・チャネル RAID ストレージ・システム上に作成しなければなりません。
- ファイルシステムのマウント・ポイントは、フェイルオーバー・ドメインに含まれるすべてのノードに作成してください。
- 各ノードごとにさまざまな IRIS FailSafe ファイルシステムをセットアップするときは、各ファイルシステムが異なるマウント・ポイントを使用するようにしてください。
- 共有ディスク上のファイルシステムを同時に複数のノードでマウントしないでください。同時にマウントすると、データが破壊されます。通常、すべてのファイルシステムは FailSafe によって共有ディスクにマウントされます。共有ディスクに手動でファイルシステムをマウントする場合は、そのファイルシステムがほかのノードで使用されていないことを確認してください。
- 共有ディスク上のファイルシステムは、 `/etc/fstab` ファイルで指定しないでください。これらのファイルシステムは、別のノードにマウントされていないことを確認した後で FailSafe によってマウントされます。

`filesystem` リソース・タイプのリソースの名前は、ファイルシステムのマウント・ポイントになります。

ファイルシステムがモード `wsync` でエクスポートされていない場合、ファイルシステムのフェイルオーバー中にクライアントが FailSafe NFS ファイルシステムにアクティブに書込んでいると、データが破壊される可能性があります。. このモードを使用する場合は、XFS ファイルシステムのローカル・マウントでも `wsync` マウント・モードを使用する必要があります。`wsync` を使用すると、パフォーマンスに大きく影響します。

注意: FailSafe クラスタでは、NFS を使用してファイルシステムを相互にマウントしないでください（つまり、ローカルにマウントされているファイルシステムを、NFS を使用して別のノードにマウントしないでください）。この設定は信頼性に欠け、FailSafe では動作しません。代わりに、同じ機能を提供する CXFS（クラスタ化された XFS）プラグインを使用してください。詳細については、『IRIS FailSafe Version 2 Administrator's Guide』を参照してください。

TCP 上で NFS を使用することは推奨されていません。クライアントの TCP 接続が切断されて再接続しなかった場合、フェイルオーバーの実行時にクライアントがハングする可能性があります。TCP ではなく UDP を使用することが推奨されています。NFS クライアントのデフォルトは TCP の可能性があるため、UDP を使用するようにそれらを再設定しなければならないことがある点に注意してください。これを実現する 1 つの方法は、`-p UDP` という内容が含まれる `/etc/config/nfsd.options` ファイルを作成することです。これにより、サーバが TCP マウント要求を受付けるのを防止します。

ファイルシステム設定の例

39 ページの「論理ボリューム設定の例」の節の設定例の続きとして、以下の XFS ファイルシステムがある場合を取上げます。

- VolA 上の xfsA は、モード rw および noauto で /sharedA にマウントされます。
- VolB 上の xfsB は、モード rw、 noauto および wsync で /sharedB にマウントされます。
- VolC 上の xfsC は、モード rw および noauto で /sharedC にマウントされます。

表2-2 に、各ファイルシステムのラベルと設定パラメータを示します。

表2-2 ファイルシステム設定パラメータ

属性	/sharedA	/sharedB	/sharedC	コメント
monitor-level	2	2	2	モニタリングには以下の 2 つのレベルがあります。1 – /etc/mtab ファイルをチェックします。2 – stat(1M) コマンドを使用して、ファイルシステムがマウントされているかどうかをチェックします。
volume-name	VolA	VolB	VolC	ファイルシステムが作成されている論理ボリュームのラベル
mode	rw, noauto	rw, noauto, wsync	rw, noauto	ファイルシステムのモード(/etc/fstab で指定されているモードと同じ)

図2-6 に、以下を示します。

- リソース・グループ 1 では、2 つの XFS ファイルシステム(xfsA および xfsB)と 2 つの XLV ボリューム(VolA および VolB)が使用されています。
- リソース・グループ 2 では、1 つの XFS ファイルシステム (xfsC) と 1 つの XLV ボリューム (VolC) が使用されています。

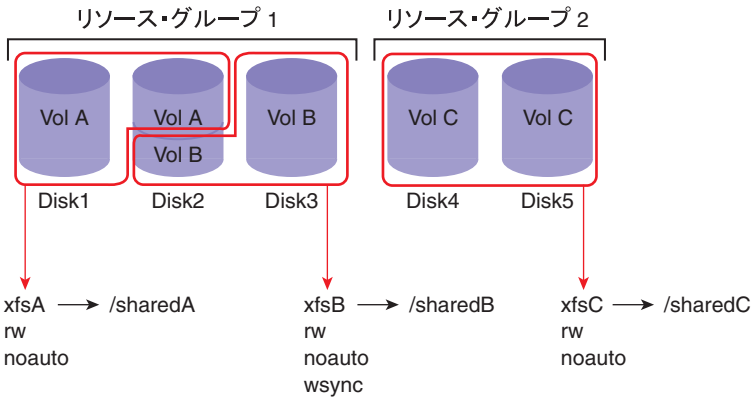


図2-6 ファイルシステムと論理ボリューム

XFS ファイルシステムの作成についての詳細は、「XLV 論理ボリュームおよび XFS ファイルシステムの作成」を参照してください。

HA IP アドレス設定

この節では、以下の内容について説明します。

- 「ネットワーク・インタフェースと HA IP アドレスの設定計画」
- 46 ページの「HA IP アドレス設定の例」
- 46 ページの「HA IP アドレスのローカル・フェイルオーバー」

ネットワーク・インタフェースと HA IP アドレスの設定計画

ノード間のプライベート・コントロール・ネットワーク用のインタフェース設定を計画するときは、以下のガイドラインを使用してください。

- 各インタフェースに 1 つの IP アドレスを使用します。
- 各ノードでプライベート・ネットワークとのインタフェース用に使用される HA IP アドレスは、パブリック・ネットワークの IP アドレスとは異なるサブネットに配置します。
- `/etc/hosts` で各 HA IP アドレスに対して IP 名を指定できます。

- これらの HA IP アドレスをプライベート・ネットワークに関連付ける一定の命名規則を使用すると便利です。たとえば、`priv-xfs-ha1` や `priv-xfs-ha2` のように、ホスト名の前に *private* を意味する `priv-` を付けます。

1 つまたは複数のパブリック・ネットワーク用のインタフェース設定を計画するときは、以下のガイドラインを使用します。

- 再 MAC が必要な場合、フェイルオーバーされる各インタフェースは、もう一方のノード上に専用のバックアップ・インタフェース(HA IP アドレスを持たないインタフェース)を必要とします。したがって、再 MAC が必要なインタフェースの各 HA IP アドレスに対して、そのインタフェース専用のフェイルオーバー・ドメインの各ノードに 1 つのインタフェースが必要になります。
- 各インタフェースには、固定アドレスとも呼ばれるプライマリ IP アドレスがあります。プライマリ IP アドレスはフェイルオーバーされません。
- ノードのホスト名は HA IP アドレスにできません。
- クライアントが HA サービスへのアクセスに使用するすべての HA IP アドレスは、HA サービスが属するリソース・グループの一部でなければなりません。
- 再 MAC が必要な場合は、すべての HA IP アドレスに同じバックアップ・インタフェースを指定しなければなりません。
- HA IP アドレスは適切に選択することが重要です。これらのアドレスは、HA サービスのユーザが使用する「ホスト名」であり、ノードの本当のホスト名ではありません。
- HA サービスのユーザは、`hostname` コマンドの出力ではなく HA IP アドレスを使用する必要があるので、HA IP アドレスをユーザに公開するための計画を立ててください。
- HA IP アドレスは、`/etc/config/netif.options` ファイルで設定したり、`/etc/config/ipaliases.options` ファイルで定義しないでください。

再 MAC が必要かどうかを判断するには、以下の手順に従ってください(再 MAC についての詳細は、「ネットワーク・インタフェースおよび IP アドレス」を参照してください)。この手順では、 `node1`、`node2`、および `node3` の 3 つのノードを使用する必要があります。 `node1` および `node2` は FailSafe クラスタのノードにできますが、これは必須ではありません。これらのノードは同じサブネット上に存在する必要があります。 `node3` は 3 番目のノードです。ルータが Gratuitous ARP パケットを受付けるかどうかを確認する必要がある場合(つまり再 MAC が必要ない場合)、 `node3` は、 `node1` と `node2` から見てルータの反対側になければなりません。

1. node1 のインタフェースの 1 つに HA IP アドレスを設定します。

   ```
   # /usr/etc/ifconfig interface inet ip_address netmask netmask up
   ```

   interface はノードへのアクセスに使用されるインタフェースで、*ip_address* は node1 の IP アドレスです。この IP アドレスはこの手順全体で使用されます。*netmask* は、この IP アドレスのネットマスクです。

2. node3 から、手順 1 で使用した HA IP アドレスに対して ping コマンドを実行します。

   ```
   # ping -c 2 ip_address
   PING 190.0.2.1 (190.0.2.1): 56 data bytes
   64 bytes from 190.0.2.1: icmp_seq=0 ttl=255 time=29 ms
   64 bytes from 190.0.2.1: icmp_seq=1 ttl=255 time=1 ms

   ----190.0.2.1 PING Statistics----
   2 packets transmitted, 2 packets received, 0% packet loss
   round-trip min/avg/max = 1/1/1 ms
   ```

3. node1 で次のコマンドを入力して、手順 1 で設定したインタフェースをシャットダウンします。

   ```
   # /usr/etc/ifconfig interface down
   ```

4. node2 で次のコマンドを入力して、HA IP アドレスを node2 に移動します。

   ```
   # /usr/etc/ifconfig interface inet ip_address netmask netmask up
   ```

5. node3 からこの HA IP アドレスに対して ping を実行します。

   ```
   # ping -c 2 ip_address
   ```

   ping(1) コマンドが失敗する場合は、Gratuitous ARP パケットが受付けられていないので、HA IP アドレスをフェイルオーバーするには再 MAC が必要になります。

HA IP アドレス設定の例

表2-3 に、これらの HA IP アドレスに対して指定できる FailSafe 設定パラメータを示します。

表2-3 HA IP アドレス設定パラメータ

リソース属性	リソース名: 192.26.50.1	リソース名: 192.26.50.2
ネットワーク・マスク	0xffffff00	0xffffff00
ブロードキャスト・アドレス	192.26.50.255	192.26.50.255
インタフェース	ef0	ef0

HA IP アドレスのローカル・フェイルオーバー

HA IP アドレスが同じノード内の 2 番目のインタフェースにフェイルオーバーされるようにシステムを設定できます(たとえば、 ef0 から ef1 など)。この設定の際に従わなければならない手順を示した設定例は、210 ページの「例: HA IP アドレスのローカル・フェイルオーバー」にあります。

CXFS と IRIS FailSafe の同時実行

クラスタ化された XFS ファイルシステムである CXFS を使用することで、コンピュータのグループは、高いパフォーマンスを保ったまま、一貫性のある方法で大量のデータを共有できます。ユーザは、CXFS クラスタ内で FailSafe を使用して、CXFS ファイルシステム上で実行される高可用性サービス(NFS や Web など)を提供できます。この組合わせにより、パフォーマンスの高い共有データ・アクセスが高可用性アプリケーションに提供されます。

CXFS 6.5.10 以降と IRIS FailSafe 2.1 以降(および関連するパッチ)を同じシステムにインストールして実行できます。これを同時実行と呼びます。これにより、アプリケーション・レベルの高可用性とクラスタ・ファイルシステムを利用できるようになります。

同時実行クラスタ内のノードのサブセットは、FailSafe ノードとして使用するように設定できます。したがって、同時実行クラスタでは、FailSafe を実行するノードを最大 8 つまで使用できます。

この節では、以下の内容について説明します。

- 47 ページの「同時実行クラスタのサイズ」
- 47 ページの「クラスタのタイプ」
- 47 ページの「CXFS メタデータ・サーバのノードのタイプ」

- 48 ページの「CXFS メタデータ・サーバとフェイルオーバー・ドメイン」
- 48 ページの「CXFS FailSafe のリソース・タイプ」
- 50 ページの「CXFS と FailSafe の異なる GUI」
- 50 ページの「CXFS と FailSafe の変換」
- 51 ページの「ネットワーク・インタフェース」

302 ページの「同時実行クラスタ内の通信パス」も参照してください。

同時実行クラスタのサイズ

CXFS クラスタ内のすべてのノードでは CXFS が実行され、これらのノードの最大 8 つでは FailSafe も実行できます。CXFS と FailSafe を実行している場合も、プール、クラスタ、およびクラスタ設定はそれぞれ 1 つだけです。

実作業用のクラスタは、重み付けされた 3 つ以上のノード（CXFS の重み）と 16 個以下のノードで設定することが推奨されています（リセット・ケーブルが接続されていて重み付けされた 2 つのノードだけのクラスタはサポートされていますが、この設定には特有の問題があります。『CXFS Version 2 Software Installation and Administration Guide』を参照してください）。

クラスタのタイプ

クラスタは、以下の 3 つのいずれかのタイプに設定できます。

- `FailSafe`。この場合、すべてのノードのタイプも `FailSafe` になります。
- `CXFS`。この場合、すべてのノードのタイプが `CXFS` になります。
- `CXFS and FailSafe`（同時実行）。この場合、すべてのノードは、`CXFS` と `CXFS and FailSafe` が混在したタイプになり、アプリケーション・レベルの高可用性を得るための FailSafe と CXFS が使用されます。

メモ: `FailSafe` タイプのノードだけで同時実行クラスタを設定することも可能ですが、この設定はサポートされていません。

CXFS メタデータ・サーバのノードのタイプ

使用可能なすべてのメタデータ・サーバ・ノードは、以下のいずれかのタイプでなければなりません。

- CXFS
- CXFS and FailSafe

CXFS メタデータ・サーバとフェイルオーバー・ドメイン

メタデータ・サーバのリストは、フェイルオーバー・ドメインのリストに完全に一致する必要があります（名前と名前の順序の一致）。

CXFS FailSafe のリソース・タイプ

FailSafe には、CXFS ファイルシステムを使用するアプリケーションをフェイルオーバーさせるときに使用できる CXFS リソース・タイプが用意されています。CXFS リソースは、CXFS ファイルシステムに依存するリソースが含まれるリソース・グループに追加しなければなりません。CXFS リソースの名前は、CXFS ファイルシステムのマウント・ポイントになります。

CXFS リソース・タイプは以下の特性を持ちます。

- CXFS ファイルシステムがローカル・ノードにマウントされるまで、CXFS ファイルシステムに依存するすべてのリソースを開始しません。
- CXFS リソース・タイプに対して start および stop アクション・スクリプトを実行しても、それぞれ CXFS ファイルシステムのマウントとアンマウントは行われません（start スクリプトは、CXFS ファイルシステムが使用可能になるまで待機します。 stop スクリプトは何も実行しませんが、FailSafe ではこのスクリプトが存在している必要があります）。CXFS ファイルシステムのマウントとアンマウントを行うには、ユーザは CXFS GUI または cmgr(1M) コマンドを使用する必要があります。
- 異常がないかどうか CXFS ファイルシステムをモニタします。
- CXFS メタデータ・サーバで実行する必要があるアプリケーションに対しては、CXFS タイプは、リソース・アプリケーション・フェイルオーバーの発生時に CXFS メタデータ・サーバを再設定します（オプション）。この場合、リソース・グループのアプリケーション・フェイルオーバー・ドメイン (AFD: Application Failover Domain) は、CXFS メタデータ・サーバおよびメタデータ・サーバのバックアップ・ノードで構成されている必要があります。

NFS サーバがエクスポートする CXFS ファイルシステムは、CXFS GUI または cmgr(1m) ツールを使用して、フェイルオーバー・ドメインのすべてのノードにマウントする必要があります。

たとえば、以下に、/FC/lun0_s6 にマウントされている CXFS ファイルシステムに基づいて NFS、CXFS、および statd_unlimited を作成するために使用するコマンドを示します（この例では、 test-cluster

という名前のクラスタが定義されていて、cxfs-fp という名前のフェイルオーバー・ポリシーと、このポリシーに基づく cxfs-group という名前のリソース・グループが作成済みであることを想定しています)。

```
cmgr> define resource /FC/lun0_s6 of resource_type CXFS in cluster test-cluster
Enter commands, when finished enter either "done" or "cancel"

Type specific attributes to create with set command:

Type Specific Attributes - 1: relocate-mds


No resource type dependencies to add

resource /FC/lun0_s6 ? set relocate-mds to false
resource /FC/lun0_s6 ? done


==========================================

cmgr> define resource /FC/lun0_s6 of resource_type NFS in cluster test-cluster
Enter commands, when finished enter either "done" or "cancel"

Type specific attributes to create with set command:

Type Specific Attributes - 1: export-info
Type Specific Attributes - 2: filesystem


No resource type dependencies to add

resource /FC/lun0_s6 ? set export-info to rw
resource /FC/lun0_s6 ? set filesystem to /FC/lun0_s6
resource /FC/lun0_s6 ? done


==========================================
cmgr> define resource /FC/lun0_s6/statmon of resource_type statd_unlimited in cluster test-cluster
Enter commands, when finished enter either "done" or "cancel"

Type specific attributes to create with set command:

Type Specific Attributes - 1: ExportPoint
```

```
Resource type dependencies to add:

Resource Dependency Type - 1: NFS


resource /FC/lun0_s6/statmon ? set ExportPoint to /FC/lun0_s6
resource /FC/lun0_s6/statmon ? add dependency /FC/lun0_s6 of type NFS
resource /FC/lun0_s6/statmon ? done

==========================================
cmgr> define resource_group cxfs-group in cluster test-cluster
Enter commands, when finished enter either "done" or "cancel"

resource_group cxfs-group ? set failover_policy to cxfs-fp
resource_group cxfs-group ? add resource /FC/lun0_s6 of resource_type NFS
resource_group cxfs-group ? add resource /FC/lun0_s6 of resource_type CXFS
resource_group cxfs-group ? add resource /FC/lun0_s6/statmon of resource_type statd_unlimited
resource_group cxfs-group ? done
```

CXFS と FailSafe の異なる GUI

cmgr(1M) コマンドは共通ですが、CXFS と FailSafe にはそれぞれ別のグラフィカル・ユーザ・インタフェース (GUI: Graphical User Interface) があります。CXFS 設定の管理は CXFS GUI で行い、FailSafe 設定の管理は FailSafe GUI で行ってください。cmgr ではどちらも管理できます。

CXFS と FailSafe の変換

CXFS GUI または cmgr(1M) を使用して、既存の FailSafe クラスタとノードのタイプを CXFS または CXFS and FailSafe に変換できます。FailSafe GUI を使用すると、並行して操作を実行できます。変換後のノードは、アプリケーション・レベルの高可用性を提供するために FailSafe で使用したり、クラスタ・ファイルシステムを提供するために CXFS で使用できます。

ただし、以下の点に注意してください。

- 対応する高可用性 (HA) サービスまたは CXFS サービスがアクティブな場合は、ノードのタイプを変更することはできません。まず、ノードのサービスを停止してください。
- クラスタは、そのノードに対してオンにするすべての機能 (FailSafe または CXFS、あるいはその両方) をサポートしていなければなりません。つまり、クラスタのタイプが CXFS の場合、すでにクラスタの一

部であるノードのタイプを FailSafe に変更することはできません。ただし、ノードがクラスタのすべての機能をサポートする必要はありません。つまり、CXFS and FailSafe クラスタで CXFS ノードを使用することができます。

ネットワーク・インタフェース

FailSafe では少なくとも 2 つのネットワーク・インタフェースが必要ですが、CXFS では、ハートビート・メッセージとコントロール・メッセージの両方に 1 つのインタフェースだけを使用します。

同じノードで FailSafe と CXFS を使用する場合、CXFS に対しては優先度 1 のネットワークだけが使用されるので、このネットワークは、ハートビート・メッセージとコントロール・メッセージの両方を使用できるように設定する必要があります。

メモ: CXFS は 2 番目のネットワークにフェイルオーバーされません。ノードが CXFS and FailSafe の場合、優先度 1 のネットワークに異常が発生すると、CXFS には異常が発生しますが、FailSafe サービスは 2 番目のネットワークに移動できます。

優先度 1 のネットワークがなくなったために CXFS がノードをリセットすると、そのノードは FailSafe によって FailSafe メンバーシップから削除されます。これにより、リソース・グループはクラスタ内の別の FailSafe ノードにフェイルオーバーされます。

インストールとシステムの準備

メモ: この章の手順では、27 ページの 第2章「設定計画」で説明されている計画を行ったものと仮定します。

IRIS FailSafe のインストールとシステムの準備には、以下の手順が必要です。

- 「ソフトウェアのインストール」
- 57 ページの「システム・ファイルの設定」
- 64 ページの「`corepluspid` システム・パラメータの設定」
- 64 ページの「NVRAM 変数の設定」
- 64 ページの「XLV 論理ボリュームおよび XFS ファイルシステムの作成」
- 66 ページの「ネットワーク・インタフェースの設定」
- 70 ページの「シリアル・ポートのリング・リセット用の設定」
- 71 ページの「パッチのインストール」
- 76 ページの「PCP (Performance Co-Pilot) ソフトウェアのインストール」
- 79 ページの「システムのテスト」

ソフトウェアのインストール

IRIS FailSafe base CD のインストールには、約 10 MB の空き容量が必要です。

必要なソフトウェアをインストールするには、以下の手順に従います。

1. プール内の各ノードで、『IRIX 6.5 Installation Instructions』および FailSafe 製品のリリース・ノートに従って、サポートされているリリースの IRIX にアップグレードします。

   # **relnotes failsafe2 [*chapter_number*]**

   特定のノードがアップグレードされたことを確認するには、次のコマンドを使用して、現在インストールされているシステムを表示します。

   # **uname -aR**

2. 設定に含まれるサーバとストレージ、および IRIX 改訂レベルに基づいて、最新の推奨パッチをインストールします。各プラットフォームの推奨パッチについての詳細は、`http://bits.csd.sgi.com/digest/patches/recommended/` を参照してください。

3. 各ノードに、オペレーティング・システムに適したバージョンのシリアル・ポート・サーバ・ドライバをインストールします。シリアル・ポート・サーバに付属の CD を使用してください。インストールの後、システムを再起動してください。

   詳細については、シリアル・ポート・サーバに付属の以下のドキュメントを参照してください。

   - 『EL Serial Port Server Installation Guide』(Digi Corporation によって提供されています)
   - 『EL Serial Port Server Installation Guide Errata』

4. 各ノードに、以下のソフトウェアをこの順序でインストールします。

   a. `sysadm_base.sw.dso`

   b. `sysadm_base.sw.server`

   c. `sysadm_cluster.sw.server`

   d. `cluster_admin.sw.base`

   e. `cluster_control.sw.base`

   f. `cluster_services.sw.base`

   g. `cluster_services.sw.cli`

   h. `cluster_control.sw.cli`

   i. `failsafe2.sw.cli`

   j. `sysadm_failsafe2.sw.server`

   k. `cluster_control.sw`

sysadm_base がインストールされると、tcpmux サービスが /etc/inetd.conf ファイルに追加されます。

メモ: sysadmdesktop がインストールされていないシステムに対しては、inst によって、必要なソフトウェアがないと報告されます。この競合を解決するには、このディストリビューションに含まれており sysadmdesktop.sw.base の機能のサブセットを提供する sysadm_base.sw.priv をインストールするか、IRIX ディストリビューションから sysadmdesktop.sw.base をインストールしてください。

---

sysadmdesktop.sw.base がすでに存在するシステムに sysadm_base.sw.priv をインストールしようとすると、 inst によって、サブシステムに互換性がないと報告されます。この競合を解決するには、sysadm_base.sw.priv をインストールしないようにします。sysadm_base.sw.priv がすでに存在するシステムに sysadmdesktop.sw.base をインストールしようとした場合も、同様の競合が発生します。

GUI の Web ベースのバージョンでノードを管理する場合は、以下のサブシステムをこの順序でインストールします。

a. java_eoe.sw、バージョン 3.1.1

b. sysadm_base.sw.client

c. sysadm_cluster.sw.client

d. sysadm_failsafe2.sw.client

e. sysadm_failsafe2.sw.web

注意: GUI は Java 1.1.8 でのみ動作します。これは、IRIX 6.5.*x* リリースに付属する Java のバージョンです。

SGI の Web サイトには Java 2 も掲載されていますが、このバージョンの Java は GUI では使用できません。1.1.8 以外のバージョンの Java を使用すると、GUI は正しく実行できません。

---

5. 各ノードに、以下の追加のソフトウェアをこの順序でインストールします。

   a. cluster_services.sw.

   b. failsafe2.sw.

   c. （必要な場合）nfs.ws.nfs（IRIX から。すでに存在している可能性があります）。NFS サーバを高可用性にするには、オプションの FailSafe/NFS ソフトウェアを購入する必要があります。

   d. failsafe2_nfs.sw.

e. （必要な場合）ns_admin.sw.server （Netscape から。すでに存在している可能性があります）。

f. （必要な場合）ns_fasttrack.sw.server または ns_enterprise.sw.server （Netscape から）。Netscape サーバを高可用性にするには、オプションの FailSafe/Web ソフトウェアを購入する必要があります。

g. failsafe2_web.sw.

6. IRIX デスクトップから管理ワークステーション（GUI クライアント）を実行する場合は、デスクトップに以下のサブシステムをインストールします。

   - sysadm_failsafe2.sw.desktop.
   - sysadm_failsafe2.sw.client.
   - sysadm_base.sw.client.
   - sysadm_cluster.sw.client.
   - java_eoe.sw、バージョン 3.1.1
   - 管理ワークステーションが、Java をサポートする Web ブラウザから GUI を起動する IRIX マシンの場合は、CXFS CD から java_plugin をインストールします（ただし、IRIX 上では Web ブラウザから GUI を起動する方法は推奨されていません。IRIX デスクトップから GUI クライアントを実行する方法が推奨されています）。

     java_plugin のすべてのサブシステムをインストールしようとすると、サブシステム（java_plugin.sw.swing101、java_plugin.sw.swing102、および java_plugin.sw.swing103）に互換性のないことが inst によって報告されます。これらの 3 つのサブシステムは GUI では使用されないので、インストールしないでください。

     Java Plug-in をインストールしたら、すべてのブラウザ・ウィンドウを閉じ、ブラウザを再起動してください。

7. 適切なノードに、ストレージ管理ソフトウェアやネットワーク・ボード・ソフトウェアなど、その他のオプションのソフトウェアをインストールします。

8. プレックス化された XLV 論理ボリュームをクラスタで使用している場合は、以下を実行します。

   a. 各ノードの /var/flexlm/license.dat ファイルにディスク・プレクシング・ライセンスをインストールします。XLV 論理ボリューム、および XFS プレクシングと XFS ファイルシステムについての詳細は、第2章「設定計画」を参照してください。

   b. xlv_mgr(1M)コマンドを使用して、クラスタ内の各ノードにライセンスが正常にインストールされていることを確認します。

```
# xlv_mgr
xlv_mgr> show config
```

ライセンスが正常にインストールされている場合、次の行が表示されます。

```
Plexing license: present
```

c. xlv_mgr を終了します。

9. FailSafe の推奨パッチをインストールします。

FailSafe パッチのインストールについての詳細は、71 ページの「パッチのインストール」を参照してください。

AutoLoad 変数を Yes に設定します。これは、64 ページの「NVRAM 変数の設定」で説明されているとおり、ホスト SCSI ID を設定する際に行うことができます。

メモ: クラスタまたはノードの各コンポーネントにインストールするシステムについての概要は、325 ページの付録C「IRIS FailSafe 2.1.*x* ソフトウェア」を参照してください。

システム・ファイルの設定

この節では、以下の内容について説明します。

- 「ホスト名解決: /etc/sys_id、 /etc/hosts、/etc/nsswitch.conf」
- 59 ページの「/etc/services」
- 60 ページの「/etc/config/cad.options」
- 60 ページの「/etc/config/fs2d.options」
- 63 ページの「/etc/config/cmond.options」

ホスト名解決: /etc/sys_id、 /etc/hosts、/etc/nsswitch.conf

注意: FailSafe クラスタを設定する前に、以下の規則を理解しておくことが非常に重要です。

FailSafe クラスタには、以下のホスト名解決規則と推奨事項が適用されます。

- ホスト名は、アンダースコア (_) で始めたり、空白文字を含めることはできません。

- /etc/sys_idファイルの値は、クラスタ内のすべてのノードに対して、/etc/hostsファイルのノードのプライマリ・ホスト名（ /etc/hostsのノードのIPアドレスの直後のフィールド）に一致しなければなりません。このフィールドは、ホスト名または完全修飾ドメイン名で指定できます。

  /etc/hostsファイルの形式は以下のとおりです。*primary_hostname*には、単純なホスト名または完全修飾ドメイン名を指定できます。

  IP_address primary_hostname aliases

  たとえば、/etc/hostsファイルに以下が含まれているとします。

```
# The public interface:
128.2.3.4  color-green.sgi.com color-green green


# The private interface:
192.0.1.1  color-green-private.sgi.com  color-green-private green-private
```

  /etc/sys_idファイルは、ホスト名 color-green または完全修飾ドメイン名 color-green.sgi.comで指定できます。エイリアス green を含めることはできません。

  この場合、ログイン画面の「サーバ」フィールド、および「新規ノードの定義(Define a new node)」ウィンドウの「ホスト名」フィールドには、ホスト名 color-green または完全修飾ドメイン名 color-green.sgi.com を入力します。

- nsd(1M) ネーム・サービス・デーモンを使用する場合は、NIS (Network Information Service) またはドメイン・ネーム・サービス (DNS: Domain Name Service) よりも先にローカル・ファイルがアクセスされるようシステムを設定する必要があります。つまり、/etc/nsswitch.conf の hosts の行の最初に files がリストされていなければなりません。例は、次のとおりです。

```
hosts:      files nis dns
```

  （nis と dns の順序は FailSafe では重要ではありません。files を最初にする必要があります。）

  /etc/config/netif.options ファイルでは、インタフェースの 1 つが /etc/sys_id ($HOSTNAME) の値と等しくなければなりません。

  USN (Unified Name Service) およびネーム・サービス・デーモンについての詳細については、nsd(1M) のマン・ページを参照してください。

- /etc/nsswitch.conf または /etc/hosts ファイルを変更した場合は、nsadmin restart コマンドを使用して nsd を再開する必要があります。これにより、キャッシュもフラッシュされます。

  これらのファイルの変更後に nsd(1M) を再開しなければならない理由は、nsd ネーム・サービス・デーモンは実際に /etc/hosts の内容を読取って、その内容をより高速に検索できる形式でメモリ・キャッシュに配置するためです。したがって、変更を反映させて新しい /etc/hosts の情報を RAM キャッシュに配置するには、nsd を再開しなければなりません。/etc/nsswitch.conf を

変更した場合は、管理対象のファイルのタイプ（hosts や passwd など）、情報を得るために呼出すサービス、およびこれらのサービスの呼出し順序を判断するために、nsd がこのファイルを再度読取る必要があります。

クラスタ・サービスがアクティブなときは、クラスタで実行中のノードの IP アドレスや、クラスタの最初のノードの IP アドレスを変更することはできません。

- /etc/hosts ファイルで完全修飾ドメイン名を使用する場合は、すべて完全修飾ドメイン名を使用してください。特定のノードの /etc/sys_id で完全修飾ドメイン名を使用した場合、クラスタ内のノードの /etc/hosts ファイルにそのホストの IP/ホスト名情報を定義するときは、すべてのノードでそのノードの完全修飾ドメイン名を使用する必要があります。

  通常、完全修飾ドメイン名を使用するかどうかは、クライアント（NFS など）でのクライアント・サーバ・プログラムの名前解決方法や、デフォルトの解決方法などに応じて決定します。

- 最初のノード（クラスタ・データベースを作成するノード）を定義した後で、 /etc/nsswitch.conf ファイルのホスト名解決方法を変更した場合は、データベースを再度作成する必要があります。

- CXFS との同時実行を使用する場合は、/etc/sys_id の値を IP アドレス・エイリアスに関連付ける /etc/hosts のエントリは追加しないでください。この場合は、プライマリ・アドレスを使用しなければなりません。

/etc/services

プール内の各ノードに cluster_admin 製品をインストールする前に、/etc/services ファイルを編集して、sgi-cad および sgi-crsd のエントリを含めます。これらのプロセスに割当てられたポート番号は、プール内のすべてのノードで同じである必要があります。

メモ: sgi-cad は、FailSafe ノード間での通信のために TCP ポートを必要とします。

以下に、sgi-cad および sgi-crsd の /etc/services エントリの例を示します。

```
sgi-crsd        7500/udp            # Cluster Reset Services Daemon
sgi-cad         9000/tcp            # Cluster Admin daemon
```

高可用性 (HA) サービスをノードで開始する前に、各ノードの /etc/services ファイルを、sgi-cmsd および sgi-gcd のエントリが含まれるように編集します。これらのプロセスに割当てられたポート番号は、クラスタ内のすべてのノードで同じである必要があります。

以下に、sgi-cmsd および sgi-gcd の /etc/services エントリの例を示します。

```
sgi-cmsd        7000/udp          # SGI FailSafe Membership Daemon
sgi-gcd         8000/udp          # SGI Group Communication Daemon
```

/etc/config/cad.options

/etc/config/cad.options ファイルには、プロセスの開始時に cad(1M) クラスタ管理デーモンによって読取られるパラメータのリストが含まれています。cad は、GUI にクラスタ情報を提供します。

cad.options ファイルでは、以下のオプションを設定できます。

--append_log	cad ログ情報を cad ログ・ファイルに上書きするのではなく、追加します。
--log_file *filename*	cad ログ・ファイル名。これは、-lf *filename* として指定することもできます。
-vvvv	詳細レベル。文字 v の数は、ログのレベルを示します。-v を設定すると、ログに記録されるメッセージの数は最小になります。-vvvv を設定すると、ログに記録されるメッセージの数は最大になります。

以下に、/etc/config/cad.options ファイルの例を示します。

```
-vv -lf /var/cluster/ha/log/cad_nodename --append_log
```

cmgr(1M) コマンドまたは GUI を使用して /etc/config/cad.options ファイルの内容を変更することはできません。

メモ: 初期設定時以外に cad.options ファイルを変更した場合、変更を反映させるには、 cad プロセスを再開する必要があります。これは、ノードを再起動するか、または次のコマンドを入力することで実行できます。

```
# /etc/init.d/cluster restart
```

実行中のクラスタでこのコマンドを実行した場合、クラスタは動作し続けますが、GUI と cad(1M) デーモンの接続は切断され、再接続するよう GUI によってプロンプトが表示されます。

/etc/config/fs2d.options

/etc/config/fs2d.options ファイルには、プロセスの開始時に fs2d デーモンによって読取られるパラメータのリストが含まれています。fs2d デーモンは、プール内の全ノードへのクラスタ・データベースの配布を管理するクラスタ・データベース・デーモンです。

表3-1 に、fs2d.options ファイルで設定できるオプションを示します。

表3-1 fs2d.options ファイル・オプション

オプション	説明
-logevents *event name*	選択されたイベントをログに記録します。使用されるイベントの名前には、all、internal、args、attach、 chandle、node、tree、 lock、datacon、trap、 notify、access、 storage などがあります。デフォルトは all です。
-logdest *log destination*	ログの記録先を設定します。使用されるログの記録先には、 all、stdout、stderr、 syslog、logfile などがあります。複数の記録先が指定されている場合、ログ・メッセージはそれらのすべてに書込まれます。 logfile を指定した場合は、-logfile オプションも指定しないと効果がありません。デフォルトは logfile です。
-logfile *filename*	ログ・ファイル名を設定します。 デフォルトは /var/cluster/ha/log/fs2d_log です。
-logfilemax *maximum size*	ログ・ファイルの最大サイズ(バイト単位)を設定します。ファイルが最大サイズを超過した場合は、既存の filename.old が削除された後、現在のファイルの名前が filename.old に変更され、新しいファイルが作成されます。単一のメッセージが複数のファイルに分割されることはありません。-logfile が設定されている場合、デフォルトは 10000000 です。
-loglevel *loglevel*	ログ・レベルを設定します。使用されるログ・レベルには、always、critical、error、warning、info、 moreinfo、freq、morefreq、 trace、busy などがあります。デフォルトは info です。
-trace *trace_class*	選択されたイベントをトレースします。使用されるトレース・クラスには、all、rpcs、 updates、transactions、monitor などがあります。このオプションを指定した場合は、-tracefile または -tracelog、あるいはその両方も指定してください。イベントの 1 つまたは複数のクラスのトレースを要求しても、-tracefile または -tracelog、あるいはその両方を指定しないかぎり、トレースは実行されません。デフォルトは transactions です。
-tracefile *filename*	トレース・ファイル名を設定します。デフォルトはありません。
-tracefilemax *maximum_size*	トレース・ファイルの最大サイズ(バイト単位)を設定します。ファイルが最大サイズを超過した場合は、既存の filename.old が削除された後、現在のファイルの名前が filename.old に変更され、新しいファイルが作成されます。
-[no]tracelog	-tracelog の場合はログの記録先をトレースし、-notracelog の場合はログの記録先をトレースしません。このオプションを設定すると、トレース・メッセージはログの記録先に送信されます。トレース・ファイルがある場合は、このファイルにもトレース・メッセージが書込まれます。デフォルトは -tracelog です。

オプション	説明
-[no]parent_timer	-parent_timer の場合は親が存在すると終了し、-noparent_timer の場合は終了しません。デフォルトは -noparent_timer です。
-[no]daemonize	-daemonize の場合はデーモンとして実行し、-nodaemonize の場合はデーモンとして実行しません。デフォルトは -daemonize です。
-l	デーモンとして実行しません。
-h	使用法メッセージを出力します。
-o help	使用法メッセージを出力します。

これらのオプションに対してデフォルト値を使用すると、レベル info 以下のすべてのログ・メッセージと、トランザクション・イベントのすべてのトレース・メッセージが /var/cluster/ha/log/fs2d_log ファイルに送信されるよう、システムが設定されます。ファイル・サイズが 10 MB に達した場合、このファイルは拡張子が .old で同じ名前のファイルに移動され、ログは同じ名前の新しいファイルにロールオーバーされます。単一のメッセージが複数のファイルに分割されることはありません。

メモ: 初期設定時以外に fs2d.options ファイルを変更した場合、変更を反映させるには、fs2d プロセスを再開する必要があります。これは、ノードを再起動するか、または次のコマンドを入力することで実行できます。

```
# /etc/init.d/cluster restart
```

実行中のクラスタでこのコマンドを実行した場合、クラスタは動作し続けますが、GUI と cad(1M) デーモンの接続は切断され、再接続するよう GUI によってプロンプトが表示されます。

例 1

以下に、ログおよびトレース情報を次のように送信する /etc/config/fs2d.options ファイルの例を示します。

- すべてのログ・イベントは /var/adm/SYSLOG に送信されます。
- RPC、更新、およびトランザクションのトレース情報は /var/cluster/ha/log/fs2d_ops1 に送信されます。

  ファイルのサイズが 100,000,000 バイトを超えると、このファイルの名前は /var/cluster/ha/log/fs2d_ops1.old に変更され、新しい /var/cluster/ha/log/fs2d_ops1ファイルが作成されます。単一のメッセージが複数のファイルに分割されることはありません

（改行して読みやすくしてあります。）

```
-logevents all -loglevel trace -logdest syslog -trace rpcs
-trace updates -trace transactions -tracefile /var/cluster/ha/log/fs2d_ops1
-tracefilemax 100000000
```

例 2

以下に、すべてのログとトレース・メッセージを 1 つのファイル /var/cluster/ha/log/fs2d_chaos6 に送信する /etc/config/fs2d.options ファイルの例を示します。このファイルでは、最大サイズが 100,000,000 バイトに指定されています。-tracelog は、トレース結果をログ・ファイルに送信します。

(改行して読みやすくしてあります。)

```
-logevents all -loglevel trace -trace rpcs -trace updates
-trace transactions -tracelog -logfile /var/cluster/ha/log/fs2d_chaos6
-logfilemax 100000000 -logdest logfile.
```

/etc/config/cmond.options

/etc/config/cmond.options ファイルには、プロセスの開始時に cmond(1M) クラスタ・モニタ・デーモンによって読取られるパラメータのリストが含まれています。また、cmond イベントをログに記録するファイルの名前も、このファイルで指定されます。 cmond により、プロセス・グループを開始、停止、およびモニタするためのフレームワークが提供されます。詳細については、cmond(1M) のマン・ページを参照してください。

cmond.options ファイルでは、以下のオプションを設定できます。

-L *log_level*	ログ・レベルを *log_level* に設定します。*log_level* の正しい値は、normal、critical、error、 warning、info、frequent、および all です。
-d	デバッグ・モードで実行します。
-l	レイジー・モード。cmond はクラスタ・データベースとの接続を検証しません。
-t *nap_interval*	cmond が、モニタするプロセス・グループの活動をチェックする時間周期(ミリ秒単位)
-s	メッセージのログを標準エラーに記録します。

デフォルトの cmond.options ファイルは、以下のオプションに設定されています。このデフォルトのオプション・ファイルでは、cmond イベントは /var/cluster/ha/log/cmond_log ファイルにログされます。

```
-L info -f /var/cluster/ha/log/cmond_log
```

corepluspid システム・パラメータの設定

systune(1M) コマンドを使用して、すべてのノードで corepluspid フラグを 1 に設定します。このフラグを設定すると、IRIX によってすべてのコア・ファイルにプロセス ID (PID) の接尾辞が付けられます。これにより、特定のコア・ダンプが別のプロセスのコア・ダンプによって上書きされるのを防ぎます。

NVRAM 変数の設定

FailSafe ノードのハードウェア・インストール中に、以下の 2 つの不揮発性ランダム・アクセス・メモリ (NVRAM: Non-Volatile Random-Access Memory) 変数を設定しなければなりません。

- AutoLoad 起動パラメータは、yes に設定してください。FailSafe では、ノードがリセットされたりノードの電源がオンになった場合、ノードが自動的に起動する必要があります。
- scsihostid 変数によって指定されるノードの SCSI ID は、すべて異なっていなければなりません。この変数が重要なのは、クラスタが共有 SCSI ストレージを使用して設定されている場合だけです。クラスタに共有ストレージがない場合や、クラスタで共有ファイバ・チャネル・ストレージを使用している場合は、 scsihostid の設定は重要ではありません。

これらの変数の設定は、以下のコマンドを使用してチェックできます。

```
# nvram AutoLoad
Y
# nvram scsihostid
0
```

これらの変数を設定するには、以下のコマンドを使用します。

```
# nvram AutoLoad yes
# nvram scsihostid number
```

number は、選択した SCSI ID です。ノードは、接続されているすべてのバスでその SCSI ID を使用します。したがって、ノードに接続されているどのデバイスでも、SCSI ユニット番号として *number* が指定されていないことを確認してください。scsihostid 変数の値を変更した場合は、システムを再起動して変更を反映してください。

XLV 論理ボリュームおよび XFS ファイルシステムの作成

XLV 論理ボリュームは、『IRIX Admin: Disks and Filesystems』の指示に従って作成できます。

メモ: この節では、XLV 論理ボリュームを使用した論理ボリューム設定について説明します。(XVM 論理ボリュームを使用する) FailSafe ファイルシステムおよび CXFS ファイルシステムの共同実行についての詳細は、46 ページの「CXFS と IRIS FailSafe の同時実行」を参照してください。CXFS ファイルシステムの作成についての詳細は、『CXFS Version 2 Software Installation and Administration Guide』を参照してください。XVM 論理ボリュームの作成についての詳細は、『XVM Volume Manager Administrator's Guide』を参照してください。

XLV 論理ボリュームおよび XFS ファイルシステムを作成するときは、以下の重要な点に留意してください。

- RAID ストレージ・システムに共有ディスクがない場合は、プレックス化された XLV 論理ボリュームを作成する必要があります。
- 各 XLV 論理ボリュームは、その論理ボリュームを使用するリソースのプライマリ・ノードであるノードによって所有されなければなりません(37 ページの「XLV 論理ボリュームの計画」を参照してください)。共有ディスク上のボリュームのオーナーの管理を簡略化するには、以下の推奨事項に従います。
  - 共有ディスク上のボリュームは、クラスタ内の 1 つのノードからのみ操作します。
  - あるノードですべてのボリュームを作成した後で、`xlv_mgr` を使用して、*nodename* を別のノードに変更できます。
- 作成した XLV 論理ボリュームを raw ボリューム(つまりファイルシステムなし)としてデータベース・データの格納に使用する場合、データベース・システムでは、(`/dev/rxlv` および `/dev/xlv` 内の)デバイス名に特定のオーナー、グループ、およびモードが設定されていなければならないことがあります。このような場合に該当するときは(データベース・ベンダーによって提供されているドキュメントを参照してください)、`chown`(1) コマンドおよび `chmod`(1) コマンドを使用して、必要に応じてオーナー、グループ、およびモードを設定します。
- `/etc/fstab` には、共有ディスク上の XFS ファイルシステムのファイルシステム・エントリは作成されません。共有ディスク上のファイルシステムは、FailSafe ソフトウェアによってマウントされます。ただし、システム管理を簡略化するために、FailSafe 用に設定されている XFS ファイルシステムをリストしたコメントを `/etc/fstab` に追加することを検討してください。したがって、システム管理者は、マウントされている FailSafe ファイルシステムを `df` コマンドの出力で確認して、ファイルシステムを `/etc/fstab` ファイルで検索すると、これらのファイルシステムが FailSafe によって管理されていることがわかります。
- 必ず、すべてのノードの各ファイルシステムのマウント・ポイント・ディレクトリを作成してください。

ネットワーク・インタフェースの設定

この節では、ネットワーク・インタフェースの設定方法について説明します。この手順では、図3-1 に示す例を使用します。

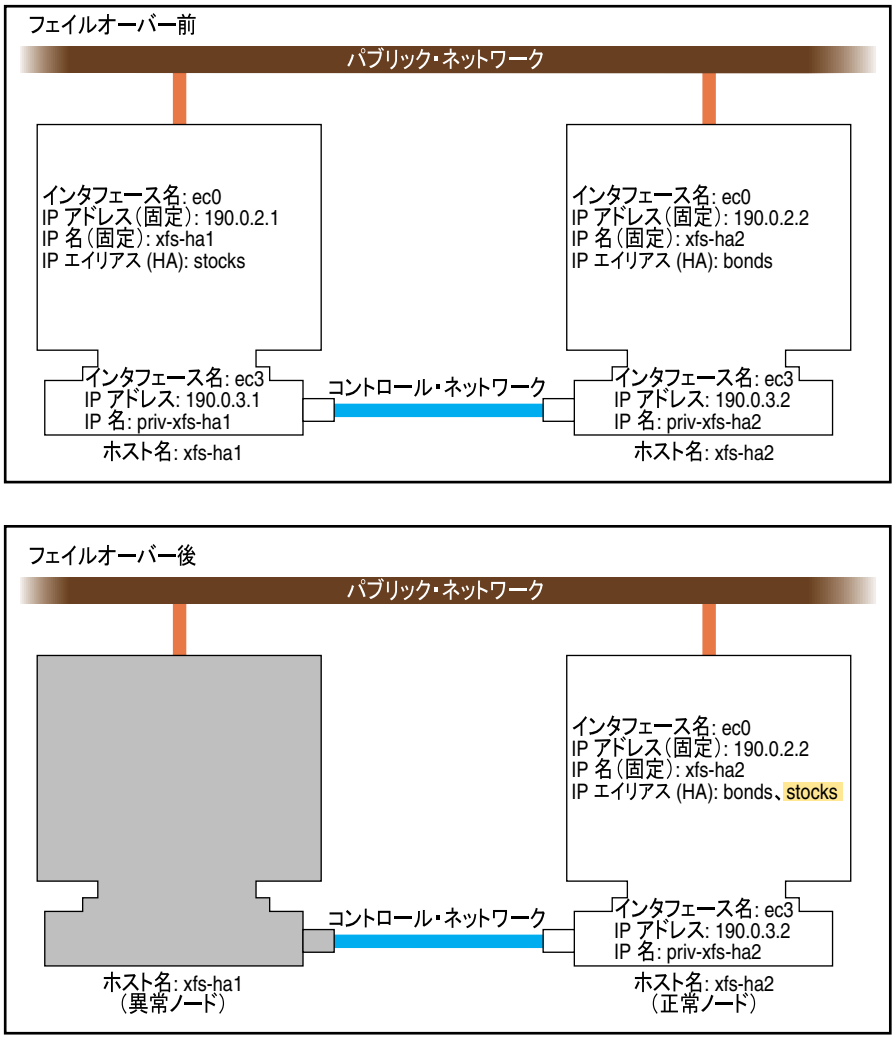


図3-1 インタフェース設定の例

1. 可能であれば、ノードのすべての IP アドレス、IP 名、および IP エイリアスを 1 つのノードの /etc/hosts に追加します。

   例は、次のとおりです。

   ```
   190.0.2.1 xfs-ha1.company.com xfs-ha1
   190.0.2.3 stocks
   190.0.3.1 priv-xfs-ha1
   190.0.2.2 xfs-ha2.company.com xfs-ha2
   190.0.2.4 bonds
   190.0.3.2 priv-xfs-ha2
   ```

   メモ：HA サービスによって排他的に使用される IP エイリアスは、ファイル /etc/config/ipaliases.options に追加されません。同様に、すべての IP エイリアスが HA サービスだけで使用される場合は、ipaliases chkconfig フラグを off にします。

   ---

2. 手順 1 のすべての IP アドレスを、クラスタ内のその他のノードの /etc/hosts に追加します。

3. 手順 1 および 2 で /etc/hosts に追加しなかった IP アドレス、IP 名、または IP エイリアスがある場合は、各ノードで以下のコマンドを入力することによって、すべてのノードで NIS が設定されていることを確認します。

   ```
   # chkconfig | grep yp
   ...
           yp              on
   ```

   出力で yp が off と表示される場合は、NIS を開始してください。詳細については、『NIS Administrator's Guide』を参照してください。

4. 手順 1 および 2 でノードの /etc/hosts に追加しなかった IP アドレス、IP 名、および IP エイリアスについては、各アドレスごとに次のコマンドを入力して、これらが NIS データベースに存在することを確認します。

   ```
   # ypmatch address hosts
   190.0.2.1 xfs-ha1.company.com xfs-ha1
   ```

   address は、IP アドレス、IP 名、または IP エイリアスです。ypmatch(1M) で *address* が一致しないと報告された場合は、該当するアドレスを NIS データベースに追加する必要があります。詳細については、『NIS Administrator's Guide』を参照してください。

5. 1 つのノードで、ノードのインタフェースとそれらの IP アドレスを /etc/config/netif.options ファイルに追加します。ただし、高可用性 (HA) IP アドレスは、 netif.options ファイルに追加されません。

図3-1 の例では、パブリック・インタフェース名および IP アドレスの行は以下のとおりです。

```
if1name=ec0
if1addr=$HOSTNAME
```

$HOSTNAME は、/etc/hosts で設定されている IP アドレスのエイリアスです。

パブリック・インタフェースがほかにもある場合は、インタフェースの名前と IP アドレスは、以下のような行に表示されます。

```
if2name=
if2addr=
```

この例では、コントロール・ネットワーク名および IP アドレスは以下のとおりです。

```
if3name=ec3
if3addr=priv-$HOSTNAME
```

この例のコントロール・ネットワーク IP アドレスである priv-$HOSTNAME は、/etc/hosts で設定されている IP アドレスのエイリアスです。

6. ノードに 9 つ以上のインタフェースがある場合は、if_num の値をインタフェースの数に変更します。図3-1 の例のようにインタフェースが 7 つ以下の場合は、行は次のようになります。

   ```
   if_num=8
   ```

7. その他のノードで、手順 5 および 6 を繰返します。

8. ルートがコントロール・ネットワークを通じてアドバタイズされるように、各ノードの /etc/config/routed.options ファイルを編集します。オプションのリストについては、routed(1M) のマン・ページを参照してください。

   例は、次のとおりです。

   ```
   -q -h -Prdisc_interval=45
   ```

メモ: -q オプションは、FailSafe が正しく機能するために必要です。このオプションは、クラスタに関係のないパケットでハートビート・ネットワークに負荷がかかることがないようにします。

これらのオプションは、以下を実行します。

- ルートのアドバタイズをオフにします。
- ホストまたはポイントツーポイント・ルートがアドバタイズされないようにします（同じ方向のネットワーク・ルートがある場合）。

- Router Discovery Advertisement が送信される標準周期を 45 秒（およびそれらの寿命を 135 秒）に設定します。

9. chkconfig(1M) コマンドを使用して、各ノードで IRIS FailSafe 2.*X* がオフであることを確認します。

```
# chkconfig | grep failsafe2
...
        failsafe2           off
...
```

特定のノードで failsafe2 が on に設定されている場合は、そのノードで次のコマンドを入力します。

```
# chkconfig failsafe2 off
```

さらに、Failsafe 1.*X* が存在する場合は、どのノードでも on に設定されていないことを確認する必要もあります。

```
# chkconfig | grep failsafe
...
        failsafe              off
...
```

いずれかのノードで failsafe が on の場合は、そのノードで次のコマンドを入力します。

```
# chkconfig failsafe off
```

10. 各ノードで電子メール・エイリアスを設定します。この電子メール・エイリアスは、FailSafe がクラスタ外のユーザやクラスタ内のその他のノードのユーザにクラスタが移行したことを通知する電子メールを送信するためのもです。

たとえば、xfs-ha1 および xfs-ha2 という 2 つのノードがある場合は、xfs-ha1 の /usr/lib/aliases に次の行を追加します。

```
fsafe_admin:operations@console.xyz.com,admin_user@xfs-ha2.xyz.com
```

xfs-ha2 では、次の行を /usr/lib/aliases に追加します。

```
fsafe_admin:operations@console.xyz.com,admin_user@xfs-ha1.xyz.com
```

選択したエイリアス（この場合は fsafe_admin）が、システムの設定時にメールの送信先アドレスに使用する値になります。この例では、operations がクラスタ外のユーザで、admin_user が各ノードのユーザです。

11. ノードで NIS を使用する(つまり、chkconfig(1M) を使用して yp が on に設定されている)場合、または BIND ドメイン名サーバ (DNS: Domain Name Server) を使用する場合は、ローカル名解決に切替えることをお薦めします。/etc/nsswitch.conf ファイルを以下のように変更します。

```
hosts:                  files nis dns
```

メモ: NIS や DNS をノードの IP アドレス・ルックアップ専用に使用すると、NIS サービスの信頼性が低下した場合に可用性が低下することが判明しています。

---

12. FDDI を使用している場合は、FDDIXpress リリース・ノートおよび『FDDIXPress Administration Guide』の説明に従って、新しい FDDI ステーションの設定と確認を完了します。
13. すべてのノードを再起動して、新しいネットワーク設定を反映させます。

シリアル・ポートのリング・リセット用の設定

リセット・シリアル・ケーブルが接続されている TTY ポートの getty プロセスは、リング・リセット設定が使用されている場合はオフになります。各ノードで以下の手順を実行します。

1. リセット・シリアル回線に使用するポートを決定します。
2. ファイル /etc/inittab を開いて、編集します。
3. 手順 1 のポート番号について、右側のコメントを参照してそのポートの行を検索します。
4. この行の 3 つ目のフィールドを off に変更します。例は、次のとおりです。

```
t2:23:off:/sbin/getty -N ttyd2 co_9600          # port 2
```

5. ファイルを保存します。
6. 以下のコマンドを入力して、変更を反映します。

```
# killall getty
# init q
```

メモ: IRISconsole または SGIconsole システムで実行されるリセット・デーモンを使用してマルチノード・クラスタを設定する場合は、FailSafe システムが実行しているリセット・デーモンと競合するため、IRISconsole または SGIconsole にはリセット・ポートを設定しないでください。リセット・ポートは、直接またはシリアル・マルチプレクサを使用してクラスタ内の別のノードのシステム・コントローラ・ポートに接続される tty ポートです。

FailSafe は、IRISconsole または SGIconsole 製品を使用して取得したコンソール接続には影響を与えません。

パッチのインストール

この節の手順では、FailSafe パッチをインストールする方法について説明します。パッチはすべてのノードにインストールします。

この節には、FailSafe イメージおよび FailSafe パッチを同時にインストールする手順と、既存の FailSafe クラスタに FailSafe パッチだけをインストールする手順が含まれています。

FailSafe 2.*x* と FailSafe パッチの同時インストール

FailSafe 2.*x* イメージとアップグレード・パッチを同時にインストールする場合は、各ノードでクラスタ・プロセスを停止して、パッチのインストール後に開始する必要があります。これは、FailSafe 2.*x* をインストールするとクラスタ・プロセスは自動的に開始されますが、パッチのインストールの際には自動的に停止されないためです。そのため、クラスタ・プロセスを再開しないかぎり、クラスタ・プロセスは、パッチが適用されていない共有ライブラリを実行し続けることになります。

各ノードで以下を実行します。

1. FailSafe 2.*x* イメージをノードにインストールします。これには以下の製品が含まれます。
   - `cluster_admin`
   - `cluster_control`
   - `cluster_services`
   - `failsafe2`
   - `sysadm_base`
   - `sysadm_failsafe2`
2. FailSafe 2.*x* パッチをインストールします。
3. UNIX シェルで、ノードのクラスタ・プロセスをすべて停止します。

   ```
   # /etc/init.d/cluster stop
   ```

4. クラスタ・プロセス(`cad`、`cmond`、`crsd`、および `fs2d`)が停止していることを確認します。

   ```
   # ps -ef | egrep '(cad|cmond|crsd|fs2d)'
   ```

5. ノードでクラスタ・プロセスを開始します。

   ```
   # /etc/init.d/cluster start
   ```

これで、FailSafe マネージャ GUI または `cmgr`(1M) コマンドを実行して、FailSafe クラスタを設定する準備が整いました。

既存の FailSafe 2.*x* クラスタでの FailSafe パッチのインストール

以下の手順を使用すると、クラスタ全体をシャットダウンしたり、クラスタによって提供される HA サービスを中断することなく、FailSafe パッチを各 FailSafe 2.*x* に順にインストールできます。

メモ: FailSafe パッチをインストールする前に、パッチのリリース・ノートをお読みください。これらのリリース・ノートには、この手順では説明されていない特殊な指示が記載されている場合があります。

FailSafe クラスタの各ノードに FailSafe パッチをインストールするには、以下の手順に従います。

1. ノードではないマシンに FailSafe GUI クライアント・ソフトウェアがインストールされている場合は、まずそのマシンにパッチ・クライアント・サブシステムをインストールします。以下に、GUI クライアント・ソフトウェア・サブシステムを示します。*xxxxxxx* はパッチ番号です。

   - `patchSG`*`xxxxxxx`*`.sysadm_base_sw.client`
   - `patchSG`*`xxxxxxx`*`.sysadm_failsafe2_sw.client`
   - `patchSG`*`xxxxxxx`*`.sysadm_failsafe2_sw.desktop`

2. パッチをインストールするノードを選択します。そのノードで FailSafe GUI または `cmgr`(1M) コマンドを開始します。

   便宜上、GUI はアップグレードしないノードに接続してください。

   メモ: アップグレードするノードに接続すると、後半の FailSafe HA サービスを停止する手順で、FailSafe から GUI に正確なステータスが報告されなくなります。また、クラスタ・サービスを停止するもう 1 つの後半の手順では、GUI が切断されてしまいます。

   次の `cmgr` コマンドを使用して、デフォルトのノードを指定します(この手順の以降のコマンドでは、クラスタ名がすでに設定されていることを想定しています)。

   ```
   cmgr> set cluster clustername
   ```

3. (オプション)インストール中もすべてのリソース・グループをノードで実行し続ける場合は、`detach` オプションを使用してリソース・グループをオフラインにします(つまり、リソース・グループを分離します)。この操作を実行すると、FailSafe は、ノードで実行され続けるリソースのモニタを停止し、リソース・グループの制御を行わなくなります。これを実行しなかった場合は、フェイルオーバー・ポリシーがそのように定義されていると想定され、次の手順でリソースがほかのノードに自動的に移行されます。

   GUI を使用している場合は、「リソース・グループをオフラインにする (Take Resource Group Offline)」タスクを実行し、「分離のみ (Detach Only)」チェックボックスをオンにします。

   `cmgr` を使用している場合は、次のコマンドを実行します。

   ```
   cmgr> admin offline_detach resource_group groupname
   ```

4. ノードで HA サービスを停止します。FailSafe マネージャがそのノードに接続されていた場合、FailSafe HA サービスが停止すると、FailSafe は現在のクラスタとノードの状態を報告できなくなります。インストール中にクラスタの状態をモニタするには、アップグレードしないノードに FailSafe マネージャを接続してください。

FailSafe GUI を使用している場合は、「FailSafe HA サービスの停止(Stop FailSafe HA Services)」タスクを実行し、「1 ノードのみ(One Node Only)」フィールドで、パッチを適用するノードを指定します。

cmgr を使用している場合は、次のコマンドを実行します。

```
cmgr> stop ha_services on node nodename
```

前のオプションの手順をスキップした場合、FailSafe は、このノードからすべてのリソース・グループを移行しようとします。ただし、そのリソース・グループのフェイルオーバー・ドメインに使用可能なノードがほかにない場合、この移行は失敗します。移行に失敗した場合は、前の手順を完了するか、別のノードにリソース・グループを移動してください。

GUI を使用している場合は、「リソース・グループの移動(Move Resource Group)」タスクを実行し、「フェイルオーバー・ドメイン・ノード(Failover Domain Node)」フィールドで、パッチを適用しないノードを指定します。

cmgr を使用している場合は、次のコマンドを実行します。

```
cmgr> admin move resource_group groupname to node nodename
```

5. アップグレードするノードの UNIX シェルで、クラスタ・プロセスをすべて停止します。

   ```
   # /etc/init.d/cluster stop
   ```

   GUI を使用しているときに、「ネットワーク切断(Connection lost)」ダイアログが表示されたら、「いいえ(No)」をクリックします。引続き GUI を使用する場合は、GUI を再起動して、パッチを適用しないノードに接続します。

6. クラスタ・プロセス(cad、cmond、crsd、および fs2d)が停止していることを確認します。

   ```
   # ps -ef | egrep '(cad|cmond|crsd|fs2d)'
   ```

7. chkconfig(1M) を使用して、cluster フラグをオフにします。

   ```
   # chkconfig cluster off
   ```

   メモ: failsafe2 フラグを使用してノードで HA サービスをオフにすることはできません。HA サービスを停止するには、GUI または cmgr コマンドを使用する必要があります。これらのコマンドは、プール内の任意のノードから実行できます。必要な場合は、force オプションを使用できます。詳細については、191 ページの「FailSafe HA サービスの停止」を参照してください。

   ---

8. ノードにパッチをインストールします。

9. chkconfig を使用して、cluster フラグをオンにします。

   ```
   # chkconfig cluster on
   ```

10. ノードでクラスタ・プロセスを開始します。

    # **/etc/init.d/cluster start**

11. ノードで HA サービスを開始します。

    GUI を使用しており、Web ブラウザで GUI 実行している場合は、以下を実行します。

    a. ブラウザを終了します。

    b. 直前にパッチを適用したノードで Web サーバを再起動します。

    c. GUI を再起動して、パッチを適用したノードに接続します。

    d. 「FailSafe HA サービスの開始(Start FailSafe HA Services)」タスクを実行し、直前にパッチを適用したノードを「1 ノードのみ(One Node Only)」フィールドで指定します。

       GUI によって、FailSafe HA サービスがクラスタでアクティブであると報告される場合は、パッチが適用されていないクライアントを使用します。この場合は、 cmgr コマンドを代わりに実行するか、パッチが適用されているクライアントで GUI を実行するか、またはパッチが適用されているノードから Web ブラウザで GUI を実行します。

    cmgr を使用している場合は、次のコマンドを実行します。

    cmgr> **start ha_services on node *nodename***

12. リソース・グループをモニタし、アップグレードしたノードでこれらのリソース・グループがオンラインになっていることを確認します。グループのリソースのタイプや数によっては、この作業には数分かかる場合があります。

    GUI を使用している場合は、ツリー表示で「表示(View)」->「ノードが所有するグループ(Groups owned by Nodes)」を選択します。リソース・グループのアイコンがオンラインのステータスを示していることを確認します。

    メモ: アップグレードしたノードで HA サービスを再開する場合は、ノードとクラスタが通常の「アクティブ(active)」状態に戻るのに数分かかる場合があります。

    ---

    cmgr を使用している場合は、次のコマンドを実行します。

    cmgr> **show status of resource_group *groupname***

その他のノードに対して、上記のプロセスを繰返します。GUI を使用している場合は、直前にアップグレードしたノードに再接続することを忘れないでください。すべてのノードに対して上記の手順が完了したら、アップグレードしたクラスタのモニタと管理を続行し、必要に応じてその他の新しいノードを定義できます。

PCP (Performance Co-Pilot) ソフトウェアのインストール

以下のように、PCP for FailSafe は、コレクタ・エージェントまたはモニタ・クライアントとして導入できます。

- コレクタ・エージェントは、統計を収集する FailSafe クラスタ自体のノードであるコレクタ・ホストにインストールされます。FailSafe クラスタの各ノードは、通常はコレクタ・ホストとして指定されます。
- モニタ・クライアントは、モニタ・ホストにインストールされます。通常、モニタ・ホストは、ディスプレイが接続されていて、IRIS Desktop を実行中のワークステーションです。

コレクタ・ホストのインストール

指定したコレクタ・ホストに PCP for FailSafe をインストールするには、以下のソフトウェア・コンポーネントがすでにインストールされている必要があります。

- IRIX 6.5.11 以降の `pcp_eoe.sw` サブシステム
- IRIS FailSafe 2.1 以降
- PCP 2.1 以降

これらの各ノードには、コレクタ・ライセンス (`PCPCOL`) もインストールされていなければなりません。

このソフトウェアをインストールしたら、各コレクタ・ホストに PCP for FailSafe の以下のサブシステムをインストールしてください。表3-2 に、モニタ・ホストに必要なサブシステムとおおよそのサイズを示します。

表3-2 PCP for FailSafe コレクタ・サブシステム

サブシステム	サイズ (KB 単位)
`pcp_fsafe.man.pages`	40
`pcp_fsafe.man.relnotes`	32
`pcp_fsafe.sw.collector`	128

必要なサブシステムをコレクタ・ホストにインストールするには、以下を実行します。

1. 使用可能なドライブに FailSafe CD-ROM を挿入して、マウントします。ローカル CD-ROM ドライブ、またはネットワーク上の別のホストのリモート CD-ROM ドライブにアクセスできます。
2. `root` としてログインします。
3. `inst`(1) コマンドを開始します。

```
# inst
```

4. インストール場所を指定します。

   - ローカル CD-ROM ドライブからインストールしている場合は、次のように入力します。

     ```
     Inst> from /CDROM/dist
     ```

   - リモート・ドライブからインストールしている場合は、次のように入力します。*host* には、マウントされた FailSafe CD-ROM を含む CD-ROM ドライブが接続されているホストの名前を指定します。

     ```
     Inst> from host:/CDROM/dist
     ```

5. pcp_fsafe パッケージでデフォルトのサブシステムを選択します。デフォルトのサブシステムは、複数のコレクタ・ホストへのインストールを簡単にするために用意されています。

   ```
   Inst> install default
   ```

6. 競合がないことを確認します。

   ```
   Inst> conflicts
   ```

7. ソフトウェアをインストールします。

   ```
   Inst> go
   ```

8. /var/pcp/pmdas/fsafe ディレクトリに移動します。

   ```
   # cd /var/pcp/pmdas/fsafe
   ```

9. Install ユーティリティを実行します。これにより、FailSafe パフォーマンス・メトリックが PCP パフォーマンス・メトリックの名前空間にインストールされます。

   ```
   # ./Install
   ```

10. fsafe PMDA (Performance Metrics Domain Agent) のインストールに適した設定を選択します。

    - collector - このシステムでパフォーマンス統計を収集します。
    - monitor - このシステムで、ローカル・システムまたはリモート・システム、あるいはその両方をモニタできるようにします。
    - both - このシステムに対して、コレクタとモニタを設定できるようにします。

    たとえば、コレクタだけを選択するには、次のように入力します。

    ```
    Please enter c(ollector) or m(onitor) or b(oth) [b] c
    ```

コレクタ・ホストからのパフォーマンス・メトリックの削除

コレクタ・ホストから PCP for FailSafe を削除したい場合は、そのホストのパフォーマンス・メトリック名前空間から PCP for FailSafe メトリックを削除する必要があります。これは、以下のコマンドを実行することによって、 pcp_fsafe サブシステムを削除する前に行うことができます。

1. /var/pcp/pmdas/fsafe ディレクトリに移動します。

   ```
   # cd /var/pcp/pmdas/fsafe
   ```

2. Remove ユーティリティを実行します。

   ```
   # ./Remove
   ```

モニタ・ホストのインストール

指定したモニタ・ホストに PCP for FailSafe をインストールするには、以下のソフトウェア・コンポーネントがすでにインストールされている必要があります。

- IRIX 6.5.11 以降の pcp_eoe.sw サブシステム(サブシステム pcp_eoe.sw.monitor も含みます)
- PCP 2.1 以降(サブシステム pcp.sw.monitor も含みます)

モニタ・ホストには、モニタ・ライセンス (PCPMON) もインストールされていなければなりません。

このソフトウェアをインストールしたら、表3-3 に示す PCP for FailSafe のサブシステムを各モニタ・ホストにインストールします。

表3-3 PCP for FailSafe モニタ・サブシステム

サブシステム	サイズ(KB 単位)
pcp_fsafe.man.pages	40
pcp_fsafe.man.relnotes	32
pcp_fsafe.sw.monitor	516

PCP for FailSafe に必要なサブシステムをモニタ・ホストにインストールするには、以下の手順を実行します。

1. 使用可能なドライブに PCP for FailSafe CD-ROM 挿入して、マウントします。ローカル CD-ROM ドライブ、またはネットワーク上の別のホストのリモート CD-ROM ドライブにアクセスできます。
2. root としてログインします。

3. inst(1) を開始します。

   ```
   # inst
   ```

4. インストール場所を指定します。

   - ローカル CD-ROM ドライブからインストールしている場合は、次のように入力します。

     ```
     Inst> from /CDROM/dist
     ```

   - リモート・ドライブからインストールしている場合は、次のように入力します。*host* には、マウントされた PCP for FailSafe CD-ROM を含む CD-ROM ドライブが接続されているホストの名前を指定します。

     ```
     Inst> from host:/CDROM/dist
     ```

5. モニタ設定用の pcp_fsafe パッケージで、必要なサブシステムを選択します。

   ```
   Inst> keep pcp_fsafe.sw.collector
   Inst> install pcp_fsafe.sw.monitor
   ```

6. PCP for FailSafe をインストールする前に、競合がないことを確認します。

   ```
   Inst> conflicts
   ```

7. ソフトウェアをインストールします。

   ```
   Inst> go
   ```

システムのテスト

この節では、システムをテストする以下の方法について説明します。

- 「プライベート・ネットワーク・インタフェース」
- 80 ページの「シリアル・リセット接続」

プライベート・ネットワーク・インタフェース

プール内の各ノードの各プライベート・ネットワークに対して、次を入力します。*nodeIPaddress* は、ノードの IP アドレスです。

```
# /usr/etc/ping -c 3 nodeIPaddress
```

以下のような標準の ping(1M) 出力が表示されます。

```
PING IPaddress (190.x.x.x: 56 data bytes
```

```
64 bytes from 190.x.x.x: icmp_seq=0 tt1=254 time=3 ms
64 bytes from 190.x.x.x: icmp_seq=1 tt1=254 time=2 ms
64 bytes from 190.x.x.x: icmp_seq=2 tt1=254 time=2 ms
```

ping が失敗する場合は、以下の手順に従います。

1. ifconfig を使用して、ネットワーク・インタフェースが UP に設定されていることを確認します。例は、次のとおりです。

```
# /usr/etc/ifconfig ec3
ec3: flags=c63<UP,BROADCAST,NOTRAILERS,RUNNING,FILTMULTI,MULTICAST>
inet 190.x.x.x netmask 0xffffff00 broadcast 190.x.x.x
```

   出力の最初の行にある UP は、インタフェースが設定されていることを示します。

2. ケーブルが正しく接続されていることを確認します。

各ノードでこの手順を繰返します。

シリアル・リセット接続

シリアル・リセット接続をテストするには、以下の手順に従います。

1. ノードとシリアル・マルチプレクサの電源がオンになっていることを確認します。
2. プール内のノードの 1 つで cmgr(1M) コマンドを開始します。

   ```
   # cmgr
   ```

3. 各ノードで HA サービスを停止します。

   ```
   stop ha_services for cluster clustername
   ```

   例は、次のとおりです。

   ```
   cmgr> stop ha_services for cluster fs6-8
   ```

   ノードが正常に非アクティブ状態に移行して FailSafe プロセスが終了するまで待ちます。このプロセスには数分かかる場合があります。

4. 以下のいずれかを入力して、シリアル接続をテストします。

   - クラスタ全体をテストするには、次を入力します。

     ```
     test serial in cluster clustername
     ```

例は、次のとおりです。

```
cmgr> test serial in cluster fs6-8
Status: Testing serial lines ...
Status: Checking serial lines using crsd (cluster reset services) from node fs8
Success: Serial ping command OK.

Status: Checking serial lines using crsd (cluster reset services) from node fs6
Success: Serial ping command OK.

Status: Checking serial lines using crsd (cluster reset services) from node fs7
Success: Serial ping command OK.

Notice: overall exit status:success, tests failed:0, total tests executed:1
```

- 個々のノードをテストするには、次のように入力します。

  test serial in cluster *clustername* node *machinename*

  例は、次のとおりです。

```
cmgr> test serial in cluster fs6-8 node fs7
Status: Testing serial lines ...
Status: Checking serial lines using crsd (cluster reset services) from node fs6
Success: Serial ping command OK.

Notice: overall exit status:success, tests failed:0, total tests executed:1
```

- ping だけを使用して個々のノードをテストするには、次を入力します。

  admin ping node *nodename*

  例は、次のとおりです。

  ```
  cmgr> admin ping node fs7

  ping operation successful
  ```

5. コマンドが失敗する場合は、すべてのケーブルが正しく接続されていることを確認してから、再度コマンドを実行します。
6. クラスタのほかのノードでこの手順を繰返します。

管理ツール

IRIS FailSafe 管理タスクは、FailSafe マネージャ・グラフィカル・ユーザ・インタフェース (GUI: Graphical User Interface) または `cmgr`(1M) コマンドを使用して実行できます。FailSafe システムの設定とモニタを行う際にこれらのツールが使用する下層のソフトウェア・コマンド・ライン・インタフェース (CLI: Command Line Interface) は同じですが、GUI は、実作業用システムでは特に重要な機能を備えています。85 ページの「GUI の概要」を参照してください。

FailSafe マネージャ GUI

FailSafe Manager GUI lでは、FailSafe のクラスタおよびノードの設定、管理、およびモニタを行うことができます。

GUI の起動

FailSafe デーモンが開始されている場合に、クラスタの正確なステータスを参照するには、すべての FailSafe クラスタ・デーモンが実行されているノードに接続するようにしてください(100 ページの「クラスタ・デーモンが実行されていることの確認」を参照してください)。クラスタ内で FailSafe クラスタ・デーモンが開始されていない場合は、プール内の任意のノードに接続できます。

メモ: クラスタの表示は、どのノードから GUI を実行するかによって変わります。変更を加えたときは、次の変更を行う前に、ツリー表示に変更内容が表示されるまで待ってください。ツリー表示に変更内容が表示されるまでは、その変更がクラスタ全体に伝わったことは保証されていません。クラスタ・データベースに変更を加えると、その都度クラスタ全体のステータス情報が送信されます。したがって、設定が大きくなるほど送信に時間がかかります。

変更は、特定の時点で実行されている GUI の 1 つのインスタンスだけから行ってください。2 つ目の GUI インスタンス(`fstask` の 2 つ目の呼出し)から変更を行うと、最初のインスタンスで加えた変更が上書きされる場合があります。これは、異なる GUI インスタンスが異なるときに独立して更新されるためです。時間が経てば、独立した GUI インスタンスによって同じ情報が提供されるようになります。ただし、「ファイル(File)」メニューからアクセスした複数のウィンドウは、すべて単一の GUI インスタンスの一部なので、これらのどのウィンドウからでも変更を加えることができます。

すべてのタスクを実行するために必要な特権を持つことができるよう、GUI には `root` としてログインする必要があります。ただし、IRIX Interactive Desktop System Administration (`sysadmdesktop`) 製品

の一部である特権マネージャを使用すると、システム管理者が一部またはすべての特権を任意のユーザに許可できます。詳細については、『Personal System Administration Guide』を参照してください。

GUI を起動するには、以下のいずれかの方法を使用します。

- 次のコマンド・ラインを入力します。

  ```
  # /usr/sbin/fstask
  ```

  メモ: クラスタを定義する前に `fstask` を呼出すと、エラー・メッセージが表示されます。クラスタの定義中の場合は、このエラー・メッセージは無視してかまいません。

  `fsdetail` コマンド・ラインは `fstask` と同じ機能を実行します。従来の操作性を維持するために両方のコマンドが保持されています。

- Toolchest から「FailSafe マネージャ(FailSafe Manager)」を選択します。

  FailSafe のインストール後は、Toolchest を再起動して、FailSafe のエントリを Toolchest に表示させる必要があります。Toolchest を再起動するには、以下のコマンドを入力します。

  ```
  % killall toolchest
  % /usr/bin/X11/toolchest &
  ```

  このコマンドを有効にするには、『IRIS FailSafe Installation and Maintenance Instructions』に説明されているように、クライアント・システムに `sysadm_failsafe2.sw.desktop` がインストールされている必要があります。

- Web ブラウザで `http://`*server*`/FailSafeManager/`(*server* は、管理するプールまたはクラスタ内のノードの名前)と入力して、<Enter> キーを押します。表示される Web ページで「シールド」アイコンをクリックします。

  この方法で FailSafe マネージャを起動できるのは、Java プラグインをインストールしてすべての Java プロセスを終了してから、ブラウザを再起動して Java を有効にした場合だけです。「シールド」アイコンが表示されるまでに長い時間がかかる場合は、「プラグインなし(non plug-in)」リンクをクリックすることができますが、ブラウザ固有の Java で実行することにより、操作上の不具合が発生する可能性があります。

  この方法で GUI を起動できるのは、IRIX 以外のシステムから GUI を実行する場合です。IRIX システムで GUI を実行している場合は、Toolchest または `/usr/sbin/fstask` コマンドを使用する方法が推奨されています。

GUI の概要

FailSafe マネージャ GUI を使用すると、単一のポイントからクラスタ全体を管理できます。これは、クラスタの設定と管理を行いやすくするタスクへのアクセスを提供します。設定ガイド・タスクは、より大きな目的を達成するために一まとめにされたタスクのグループで構成されます。たとえば、「新規クラスタの設定(Set Up a New Cluster)」を使用すると、新規クラスタの作成手順を順番に実行でき、タイトルをクリックするだけで必要な個々のタスクを起動できます。

「ヘルプ(Help)」ボタンをクリックしてオンライン・ヘルプを利用できます。また、青いテキストをクリックして、そのテキストの概念や入力フィールドの詳細を参照することもできます。

「ファイル(File)」メニューでは、以下を実行できます。

- GUI の現在のインスタンスの複数のウィンドウを表示する
- `/var/adm/SYSLOG` システム・ログ・ファイルおよび `/var/sysadm/salog` システム管理ログ・ファイル (GUI からアクセスしたコマンドが示されています) を表示する
- PCP (Performance Co-Pilot) を起動して、リソース・モニタ (`rmvis`(1)) とハートビート・モニタ (`hbvis`(1)) を実行する
- 現在のウィンドウを閉じる
- GUI を完全に終了する

「編集(Edit)」メニューでは、ツリー表示の内容を展開および縮小できます。また、自動的に画面を展開して、プールまたはクラスタに追加された新しいノードを反映するように選択することもできます。

「タスク(Tasks)」メニューには以下が含まれます。

- 「タスクの検索(Find Tasks)」キーワードを使用して特定のタスクを検索できます。
- 「設定ガイド(Guided Configuration)」クラスタの設定、ファイルシステムの定義、既存のクラスタの変更、およびステータスの確認を行うためのタスクセットが含まれます。
- 「ノード(Nodes)」ノードを定義および管理するためのタスクが含まれます。
- 「クラスタ(Cluster)」クラスタを定義および管理するためのタスクが含まれます。
- 「リソース・タイプ(Resource Types)」`volume` のような高可用性リソース・タイプを設定するためのタスクが含まれます。
- 「リソース(Resources)」個々のリソースを設定するためのタスクが含まれます。
- 「フェイルオーバー・ポリシー(Failover Policies)」FailSafe でリソース・グループの高可用性を維持する方法を決定するためのタスクが含まれます。

- 「リソース・グループ(Resource Groups)」リソース・グループを定義および管理するためのタスクが含まれます。
- 「FailSafe HA サービス(FailSafe HA Services)」高可用性 (HA) サービスの開始と停止、FailSafe タイブレーカー・ノードの設定、およびログ設定を行うことができます。
- 「診断(Diagnostics)」接続性、リソース、およびフェイルオーバー・ポリシーをテストするためのタスクが含まれます。

デフォルトでは、ウィンドウはツリー表示とアイテム表示の 2 つのセクションに分かれています。ウィンドウの中央にある矢印を使用して画面を切り替えることができます。現在のツリーでコンポーネントを検索するには、コンポーネントの名前を入力して、「検索(Find)」ボタンをクリックします。

ツリー表示でアイテムの選択を解除するには、ツリー表示でアイテムの名前以外の場所をクリックします。

クラスタ・コンポーネントの表示

クラスタ内のノード、プール内のノード(定義されているすべてのノード)、またはクラスタ内のファイルシステムのうち、表示したいものを「表示(View)」の選択肢から選択します。

コンポーネントの詳細の表示

コンポーネントの詳細を表示するには、ツリー表示でそのアイコン名をクリックします。すると、設定とステータスの詳細が右側のアイテム表示に表示されます。アイテム表示でアイテムの詳細を参照するには、名前を選択すると(選択した名前は青色で表示されます)、新しいウィンドウに詳細が表示されます。用語集に定義されている用語も、青色で表示されます。

以下のコンポーネントについての詳細は、アイコンを選択することにより表示できます。

- クラスタ
- ノード
- リソース・タイプ
- リソース
- リソース・グループ
- フェイルオーバー・ポリシー

タスクの実行

個々のタスクを実行するには、以下の手順に従います。

1. 「タスク(Task)」メニューからタスク名を選択するか、またはツリー表示でマウスの右ボタンをクリックします。例は、次のとおりです。以下のように選択します。

   「タスク(Task)」
   >「設定ガイド(Guided Configuration)」
   >「新規クラスタの設定(Set Up a New Cluster)」

   「タスク(Task)」ウィンドウが表示されます。

   メモ: 青いテキストをクリックすると、その概念や入力フィールドに関する詳細を参照できます。

   ---

2. 該当するフィールドに情報を入力して「OK」をクリックし、タスクを完了します。タスクの中には複数のページで構成されるものもあります。このような場合は、「次へ(Next)」をクリックして次のウィンドウに移動し、そのウィンドウで情報を入力してから「OK」をクリックします。どのタスクでも、「OK」をクリックするまでクラスタ設定は更新されません。

   タスクが正常に完了したことを確認するダイアログ・ボックスが表示されます。

3. 必要に応じて、タスクの起動を続行します。

画面

図4-1 に、「FailSafe マネージャ(FailSafe Manager)」ウィンドウを示します。図4-2 には、アイテム表示の詳細を示します。

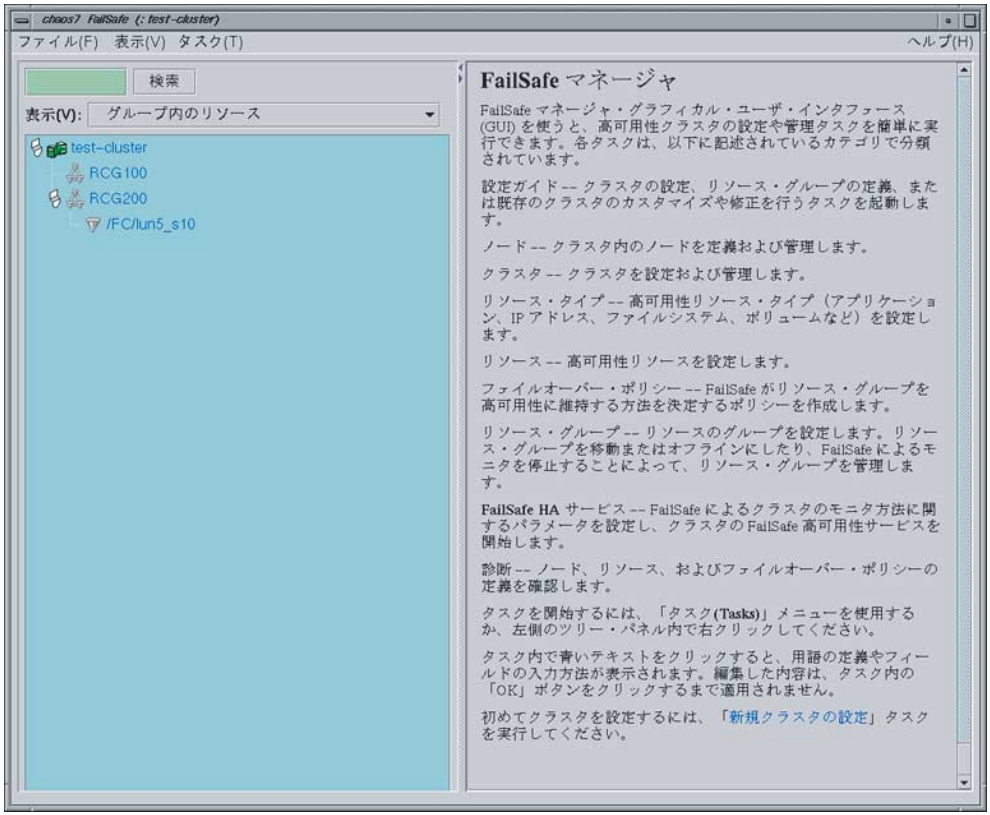


図4-1 FailSafe マネージャ GUI

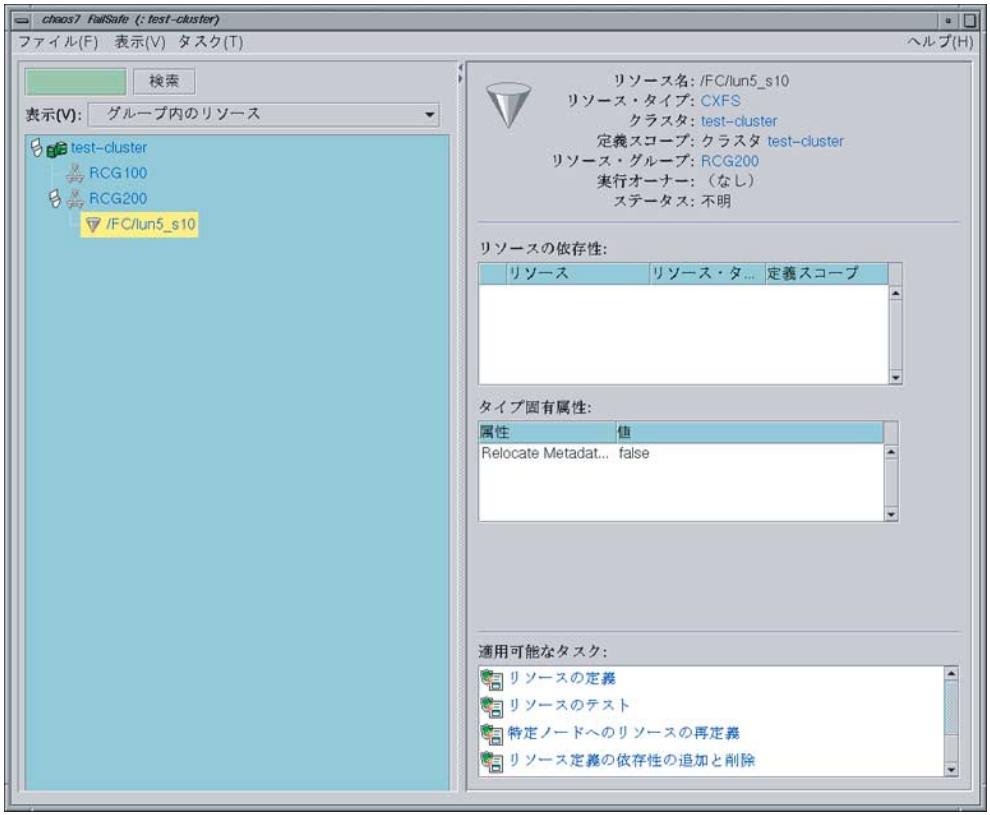


図4-2 GUI でのリソースの詳細の表示

cmgr コマンド

cmgr(1M) コマンドの機能は GUI よりも限られており、IRIX システム上にかぎり、コマンド・ライン・インタフェースを使用して FailSafe システムを設定および管理できます。また、ヘルプや書式付き出力は最低限しか用意されておらず、クエリーを実行した場合を除き、動的にステータスを参照することはできません。ただし、経験が豊富な FailSafe の管理者の方にとっては、基本的な FailSafe 設定タスクや実作業環境で独立した単一のタスクを実行する場合、またはスクリプトを実行して特定のクラスタ管理タスクを自動化する場合は、cmgr の方が便利でしょう。

この節では、cmgr コマンドを使用して FailSafe の管理タスクを実行する方法を説明します。このコマンドを使用するには、root としてログインする必要があります。

cmgr コマンドも、GUI と同じ下層の FailSafe コマンドを使用します。

cmgr を使用するには、以下のいずれかを入力します。

```
# /usr/cluster/bin/cmgr
# /usr/cluster/bin/cluster_mgr
```

サポートが必要な場合は、コマンド・ラインで -p オプションを使用することもできます。90 ページの「プロンプト・モードの使用」を参照してください。

このコマンドを入力すると、以下が表示されます。

```
Welcome to SGI Cluster Manager Command-Line Interface
cmgr>
```

コマンド・プロンプトが表示されたら、クラスタ・マネージャのコマンドを入力できます。

? または help と入力すると、いつでもヘルプを表示できます。

ヘルプの利用

コマンド・プロンプトが表示されたら、以下のサブコマンドを入力できます。? または help と入力すると、いつでも cmgr のヘルプを表示できます。

プロンプト・モードの使用

cmgr(1M) コマンドには、FailSafe コンポーネントの定義と変更を行うコマンドで必要な値を入力するよう求めるプロンプトを表示するオプションが用意されています。以下のいずれかの方法で、プロンプト・モードで実行できます。

- 以下の例のように、cmgr コマンドを入力するときに -p オプションを指定します。

  ```
  # cmgr -p
  ```

- 通常の対話型モードにいるときに、以下の例のように set prompting on コマンドを実行します。

  ```
  cmgr> set prompting on
  ```

  この方法でプロンプト・モードに入ると、個々の cmgr コマンドを実行するときにプロンプト・モードのオンオフを切替えることができます。

プロンプト・モードを終了するには、次のコマンドを入力します。

```
cmgr> set prompting off
```

たとえば、プロンプト・モードではない場合に、ノードを定義するために以下のコマンドを入力すると、以下に示すような単一のプロンプトが表示されます。

```
cmgr> define node cm1a
Enter commands, when finished enter either "done" or "cancel"

cm1a?
```

cm1a? プロンプトで、個々のノード定義コマンドを以下の形式で入力します(ノードの定義についての詳細は、114 ページの「cmgr を使ったノードの定義」を参照してください)。例は、次のとおりです。

```
cm1a? set hostname to hostname
```

ノードを定義するには、一連のコマンドが必要です。ただし、プロンプト・モードで cmgr を実行している場合は、以下の例に示されているように、必要な各コマンドを入力するよう求めるプロンプトが表示されます。

```
cmgr> define node cm1a
Enter commands, you may enter "done" or "cancel" at any time to exit

Node Name [cm1a]? cm1a

Hostname[optional]? cm1a
Is this a FailSafe node <true|false> ? true
Is this a CXFS node <true|false> ? false
Node ID ? 1
Partition ID[optional] ? (0)
Reset type <powerCycle> ? (powerCycle)
Do you wish to define system controller info[y/n]:y
Sysctrl Type <msc|mmsc|l2>? (msc) msc
Sysctrl Password [optional]? ( )
Sysctrl Status <enabled|disabled>? enabled
Sysctrl Owner? cm2
Sysctrl Device? /dev/ttyd2
Sysctrl Owner Type <tty> [tty]?
Number of Network interfaces [2]? 2
NIC 1 - IP Address? cm1
NIC 1 - Heartbeat HB (use network for heartbeats) <true|false>? true
NIC 1 - (use network for control messages) <true|false>? true
NIC 1 - Priority <1,2,...>? 1
NIC 2 - IP Address? cm2
NIC 2 Heartbeat HB (use network for heartbeats) <true|false>? true
```

```
NIC 2 - (use network for control messages) <true|false>? false
NIC 2 - Priority <1,2,...>? 2
```

アクションの完了とキャンセル

クラスタのコンポーネントを作成または変更する場合は、以下のいずれかのコマンドを入力できます。

- cancel: 現在のモードを中止して、変更を破棄します。
- done: 現在の定義または変更を確定して、cmgr プロンプトに戻ります。

cmgr 内でのコマンド・ライン編集

cmgr コマンドでは、以下の cmgr コマンド・ライン編集コマンドがサポートされています。

h [*n*] または history [*n*]	コマンド・ライン履歴を表示します。オプションの *n* は、記憶される数値コマンドを設定するために使用できます。
!!	以前のコマンドを参照します。この方法では、自動的に以前のコマンドが繰返されます。
!*n*	コマンド・ライン *n* を参照します。
!-*n*	現在のコマンド・ラインから *n* を引いたものを参照します。
!*string*	*string* で始まる最後のコマンドを参照します。
exit	シェルを終了します。
Ctrl-W	前の語を削除します。
Ctrl-D	現在の文字を削除します。
Ctrl-A	行の先頭に移動します。
Ctrl-E	行の最後に移動します。
Ctrl-F	1 文字先に移動します。
Ctrl-B	1 文字前に移動します。
Ctrl-H	前の文字を削除します。
Ctrl-N	履歴の古い方に移動します。
Ctrl-K	カーソルから行の最後までを消去します。
Ctrl-L	画面をクリアし、プロンプトを再表示します。
Ctrl-P	履歴の新しい方に移動します。
Ctrl-U	カーソルから行の先頭までを消去します。

Ctrl-R	入力行を再表示します。
Esc-f	1 語先に移動します。
Esc-b	1 語前に移動します。
Esc-d	次の語を削除します。
Esc-DEL	前の語を削除します。

実行に長時間かかるタスク

クラスタを定義し、HA サービスを停止するためのタスクは、完了するまでに数分を必要とする、実行に長時間かかるタスクです。cmgr コマンドを使用すると、このようなタスクの中間ステータスが提供されます。例は、次のとおりです。

```
cmgr> stop ha_services in cluster nfs-cluster
Making resource groups offline
Stopping HA services on node node1
Stopping HA services on node node2
```

スタートアップ・スクリプト

環境変数 CMGR_START_FILE を設定して、スタートアップ cmgr スクリプトを指定できます。この変数で指定したスタートアップ・スクリプトは、-p オプションが指定されているかどうにかかわらず、 cmgr の起動時に実行されます。cmgr スタートアップ・ファイルで使用できるのは、cmgr の set および show コマンドだけです。

以下に、cmgr_rc という名前の cmgr スタートアップ・スクリプト・ファイルの例を示します。

```
set cluster test-cluster
show status of resource_group oracle_rg
```

このファイルをスタートアップ・スクリプトとして指定するには、IRIX プロンプトで次のコマンドを実行します。

```
# setenv CMGR_START_FILE /cmgr_rc
```

cmgr を起動すると、cmgr_rc スクリプトが実行されます。デフォルトのクラスタが test-cluster に設定されて、クラスタ test-cluster のリソース・グループ oracle_rg のステータスが表示されます。

コマンド・ラインでのサブコマンドの入力

一部の cmgr サブコマンドは、次の形式を使用してコマンド・ラインに直接入力できます。

```
cmgr -c "subcommand"
```

subcommand には、適切なオペランドと共に以下のいずれかを入力できます。

- `admin`: ノードのリセットなど、特定のアクションを実行できるようにします。
- `delete`: クラスタまたはノードを削除します。
- `help`: ヘルプ情報を表示します。
- `show`: クラスタまたはノードに関する情報を表示します。
- `start`: FailSafe HA サービスを開始し、再起動時に HA サービスが自動的に再開されるよう設定します。
- `stop`: FailSafe HA サービスを停止し、再起動時に HA サービスが自動的に再開されないよう設定します。
- `test`: 接続性をテストします。

たとえば、クラスタに関する情報を表示するには、次を入力します。

```
# cmgr -c "show clusters"
1 Cluster(s) defined
      eagan
```

詳細については、99 ページの 第5章「設定」および `cmgr`(1M) のマン・ページを参照してください。

スクリプト・ファイルの使用

以下のように `-f` オプションを使用して入力ファイルを指定することで、`cmgr` コマンドを連続して実行できます。

```
cmgr -f input_file
```

または、ファイルの 1 行目に以下を含めておき、そのファイルをスクリプトとして実行することができます。

```
#!/usr/cluster/bin/cmgr -f
```

ファイルの各行は、有効な `cmgr` コマンド・ライン、コメント行(`#` で始まります)、または空白行でなければなりません。複数レベルのコマンドを完了するために `done` コマンド・ラインを含める必要があります。また、ファイルの最後には `quit` コマンド・ラインを含める必要があります。

入力ファイルのいずれかの行が失敗すると、`cmgr` は終了します。以下のように、`-i` オプションを `-f` オプションと共に使用すると、行の失敗を無視するように選択して処理を続行できます。

```
cmgr -if input_file
```

または、以下のようにスクリプトの 1 行目にこのオプションを含めることもできます。

```
#!/usr/cluster/bin/cmgr -if
```

メモ: cmgr コマンド・ラインをスクリプトの 1 行目として使用するときに -i を含める場合は、まったく同じ構文 (-if) を必ず使用してください。

たとえば、ファイル /tmp/showme に以下が含まれるとします。

```
fs6# more /tmp/showme
show clusters
show nodes in cluster fs6-8
quit
```

この場合は、次のコマンドを実行でき、以下の出力が得られます。

```
fs6# /usr/cluster/bin/cmgr -if /tmp/showme

1 Cluster(s) defined
        fs6-8


Cluster fs6-8 has following 3 machine(s)
        fs6
        fs7
        fs8
```

または、スクリプトの 1 行目に cmgr コマンド・ラインを含めて、そのスクリプトに実行アクセス権を指定し、 showme 自体を実行できます。

```
fs6# more /tmp/showme
#!/usr/cluster/bin/cmgr -if
#
show clusters
show nodes in cluster fs6-8
quit

fs6# /tmp/showme

1 Cluster(s) defined
         fs6-8


Cluster fs6-8 has following 3 machine(s)
```

```
fs6
fs7
fs8
```

テンプレート・スクリプト

システムのさまざまなコンポーネントを設定するために変更できるスクリプトのテンプレート・ファイルは、/var/cluster/cmgr-templates ディレクトリにあります。

各テンプレート・ファイルには、特定のオブジェクトを作成するための cmgr コマンドのリストに加え、各フィールドを説明するコメントも含まれています。また、オプションのフィールドのデフォルト値も提供されています。

表4-1 に、var/cluster/cmgr-templates ディレクトリ内にある cmgr のテンプレート・スクリプトを示します。

表4-1 cmgr のテンプレート・スクリプト

ファイル名	説明
cmgr-create-cluster	クラスタの作成
cmgr-create-failover_policy	フェイルオーバー・ポリシーの作成
cmgr-create-node	ノードの作成
cmgr-create-resource_group	リソース・グループの作成
cmgr-create-resource_type	リソース・タイプの作成
cmgr-create-resource-*resource type*	タイプが *resource type* のリソースを作成するためのスクリプト・テンプレート

FailSafe 設定を作成するには、複数のテンプレートを 1 つのファイルに連結して、生成されたスクリプトを実行できます。複数のテンプレート・スクリプトの情報を連結してクラスタ設定を準備する場合、各テンプレート・スクリプトの末尾にある quit は、最後の quit を除いて削除してください。1 つの cmgr スクリプトでは、quit 行は 1 つしか使用しないでください。

たとえば、1 つの volume、1 つの filesystem 、1 つの IP_address、および 1つの NFS リソースが含まれる NFS リソース・グループを持つ 3 ノード設定の場合は、以下のファイルを連結して、最後のテンプレート・スクリプト以外の各テンプレート・スクリプトの末尾にある quit を削除します。

- 3 つの cmgr-create-node ファイル

- 以下の各ファイル 1 つ
  - cmgr-create-cluster
  - cmgr-create-failover_policy
  - cmgr-create-resource_group
  - cmgr-create-resource-volume
  - cmgr-create-resource-filesystem
  - cmgr-create-resource-IP_address
  - cmgr-create-resource-NFS

cmgr 内部からのシェルの呼出し cmgr

cmgr の内部からシェルを呼出すことができます。シェルを呼出すには、次のコマンドを入力します。

```
cmgr> sh
```

シェルを終了して cmgr プロンプトに戻るには、シェル・プロンプトで exit と入力します。

設定

この章では、FailSafe マネージャ・グラフィカル・ユーザ・インタフェース (GUI: Graphical User Interface) または cmgr(1M) コマンドを使用してクラスタを設定するために必要な手順の概要を説明します。

メモ: 初期インストールの場合は、GUI の設定ガイド・タスクセットを使用することが強く推奨されています。104 ページの「GUI の設定ガイド」を参照してください。

ネットワーク・パーティションがある場合でもデータベースの最新版コピーが使用可能になるよう、FailSafe 管理はすべてプールの 1 つのノードから行うことをお薦めします。

以下の節では、従う必要のある準備手順、理解しておくべき情報、GUI の設定ガイド、および GUI と cmgr を使用するさまざまな個々のタスクについて説明します。

準備手順

クラスタ・プロセスは、FailSafe とクラスタ・サブシステムを IRIX CD からインストールしたときに自動的に開始されます。クラスタ・デーモンを開始する前に、クラスタ・デーモンが実行されていることを確認してください。以下の手順を完了して、初期クラスタの設定準備を整えます。

- 99 ページの「Cluster chkconfig フラグが On であることの確認」
- 100 ページの「クラスタ・デーモンの開始」
- 100 ページの「クラスタ・デーモンが実行されていることの確認」
- 101 ページの「ノードのホスト名の判断」

設定作業中は、ログ・ファイルにさまざまな情報メッセージが示されます。

Cluster chkconfig フラグが On であることの確認

次のフラグが on に設定されていることが chkconfig(1M) の出力に表示されることを確認します。

```
# chkconfig
        Flag                State
        ====                =====
        cluster             on
```

このように表示されない場合は、on に設定します。例は、次のとおりです。

```
# /etc/chkconfig cluster on
```

クラスタ・デーモンの開始

以下を入力してクラスタ・デーモンを開始します。

```
# chkconfig cluster on
# /etc/init.d/cluster start
```

高可用性 (HA) サービスを開始すると、base FailSafe システムでは以下のデーモンも開始されます（オプションのプラグインなしに）。

- ha_fsd
- ha_cmsd
- ha_gcd
- ha_srmd
- ha_ifd

クラスタ・デーモンが実行されていることの確認

ソフトウェアを**初めてインストール**するときは、以下のクラスタ・デーモンが実行されている必要があります。

- fs2d
- cmond
- cad
- crsd

実行されているデーモンを判断するには、次を入力します。

```
ps -ef | grep cluster
```

以下に、初期デーモンだけが実行されている場合の出力の例を示します。ここでは、見やすいように空白を削除してデーモンを強調表示してあります。

```
# ps -ef | grep cluster
root 31431     1 0 12:51:36 ?     0:14 /usr/lib32/cluster/cbe/fs2d /var/cluster/cdb/cdb.db #
root 31456 31478 0 12:53:01 ?     0:03 /usr/cluster/bin/crsd -l
```

```
root 31475 31478 0 12:53:00 ?     0:08 /usr/cluster/bin/cad -l -lf /var/cluster/ha/log/cad_log --append_log
root 31478     1 0 12:53:00 ?     0:00 /usr/cluster/bin/cmond -L info -f /var/cluster/ha/log/cmond_log
root 31570 31408 0 14:01:52 pts/0 0:00 grep cluster
```

これらのプロセスが表示されない場合は、ログを参照して、考えられる問題を確認してください。デーモンを再開する必要がある場合は、次を入力します。

```
# /etc/init.d/cluster restart
```

ノードのホスト名の判断

クラスタを初期設定する場合、GUI にログインするときとプールのノードを定義するときは、IP アドレスまたは /etc/sys_id の値(/etc/hosts のノードの IP アドレスのプライマリ名に一致していなければなりません)を使用する必要があります。hostname(1) コマンドを実行すると、/etc/sys_id の値が表示されます。例は、次のとおりです。

```
# hostname fs6
```

また、nsd(1M) を使用する場合は、NIS (Network Information Service) またはドメイン・ネーム・サービス (DNS: Domain Name Service) よりも先にローカル・ファイルがアクセスされるようシステムを設定する必要があります。57 ページの「ホスト名解決: /etc/sys_id、 /etc/hosts、/etc/nsswitch.conf」を参照してください。

注意: これらのファイルが正しく設定されており、ノードに対してプライマリ名を入力することは非常に重要です。53 ページの「ソフトウェアのインストール」を参照してください。

命名の制約

FailSafe システムのさまざまなコンポーネントの名前を指定する際、アンダースコア (_) で始まる名前や、空白として使用される文字を含む名前を付けることはできません。さらに、FailSafe コンポーネントの名前には、スペース、印刷できない文字、*、?、\、# などを含むこともできません。

次は、FailSafe コンポーネントの名前に対して使用できる文字のリストです。

- 英数文字
- /
- .
- -(ハイフン)

- _（アンダースコア）
- :
- “
- =
- @
- ’

これらの文字の制約は、システムの設定にGUIまたはcmgrのどちらを使用した場合にも当てはまります。

タイムアウト値およびモニタ周期の設定

FailSafeシステムのコンポーネントを設定する際、異常発生時の高可用性 (HA) システムのアプリケーション・ダウンタイムを決定するさまざまなタイムアウト値とモニタ周期を設定します。システムに設定するのに適切な値を決定するには、次の方程式を使用してください。

```
application_downtime = failure_detection + time_to_handle_failure + failure_recovery_time
```

異常検出は、検出された異常のタイプによって異なります。

- ノードが使用できなくなると、ノード・タイムアウト時間後にノード異常検出が起こります。このタイムアウト時間はパラメータの1つで、変更することが可能です。ノード異常と考えられる異常（ハートビート異常やOS異常など）は、すべてこの異常のカテゴリに分類されます。ノード・タイムアウトのデフォルト値は、15秒です。
- リソース異常が発生すると、リソースのモニタ異常も発生します。これにかかる時間は、以下によって決定されます。
  - リソース・タイプのモニタ周期
  - リソース・タイプのモニタ・タイムアウト
  - リソース・タイプに定義された再開数（再開モードがオンに設定されている場合）

  リソース・タイプの値の設定についての詳細は、141ページの「GUIを使ったリソース・タイプの定義」を参照してください。

これらの値を小さくすると、フェイルオーバー時間は短くなりますが、FailSafeのオーバーヘッドが大幅に増え、システム・パフォーマンスに影響を与えるほか、フェイルオーバーが誤って実行される場合もあります。

異常を処理する時間は、ユーザには制御できません。一般に、これには数秒かかります。

異常から回復する時間は、FailSafe が以下を実行するのに必要な時間の合計によって決まります。

- フェイルオーバー・ポリシー・スクリプトの実行（約 5 秒）
- リソース・グループの全リソースに対する stop アクション・スクリプトの実行。異常が発生したノードはリセットされるので、これはノード異常の場合は必須ではありません。
- リソース・グループの全リソースに対する start アクション・スクリプトの実行。

cmgr を使ったデフォルトの設定

特定の cmgr コマンドでは、クラスタ、ノード、またはリソース・タイプを指定する必要があります。FailSafe システムのコンポーネントを設定する前に、明示的な値を指定しなかった場合に使用されるこれらのコンポーネントの値に対してデフォルトを設定できます。デフォルト値は、cmgr の現在のセッションに対してのみ有効となります。

変更するには、以下の cmgr コマンドを使用します。

- デフォルトのクラスタ:

  set cluster *clustername*

  例は、次のとおりです。

  cmgr> **set cluster test-cluster**

- デフォルトのノード:

  set node *nodename*

  例は、次のとおりです。

  cmgr> **set node node1**

- デフォルトのリソース・タイプ:

  set resource_type *resource_type_name*

  例は、次のとおりです。

  cmgr> **set resource_type IP_address**

現在のデフォルト値を表示するには、次のコマンドを使用します。

show set defaults

GUI の設定ガイド

GUI には、FailSafe クラスタを設定しやすくする設定ガイド・タスクセットが付属しています。

クラスタの表示は、どのノードから GUI を実行するかによって変わります。変更を加えたときは、次の変更を行う前に、ツリー表示に変更内容が表示されるまで待ってください。ツリー表示にアイコンが表示されるまでは、その変更がクラスタ全体に伝わったことは保証されていません。

変更は、特定の時点で実行されている GUI の 1 つのインスタンスだけから行ってください。2 つ目のインスタンス(`fstask` の 2 つ目の呼出し)から変更を行うと、最初のインスタンスで加えた変更が上書きされる場合があります。ただし、「ファイル(File)」メニューからアクセスした複数のウィンドウは、すべて単一の GUI インスタンスの一部なので、これらのどのウィンドウからでも変更を加えることができます。

新規クラスタの設定

メモ: タスク内で青いテキストをクリックすると、その概念や入力フィールドに関する詳細を参照できます。どのタスクでも、「OK」をクリックするまでクラスタ設定は更新されません。

「設定ガイド(Guided Configuration)」の「新規クラスタの設定(Set Up a New Cluster)」タスクセットを使用すると、新しいクラスタを作成するために必要な手順を実行できます。このタスクセットには、ほかの箇所で詳しく説明されているタスクが含まれます。

GUI を使用すると、クラスタとそのコンポーネントを簡単に表示できます。実行状況を確認し、ノードを追加するのが早すぎることがないようにしてください。

以下を実行します。

1. 以下のように選択します。

   「タスク(Task)」
   > 「設定ガイド(Guided Configuration)」
   > 「新規クラスタの設定(Set Up a New Cluster)」

2. 「ノードの定義(Define a Node)」をクリックして、接続先のノードを定義します。すべてのノードの定義で、`/etc/sys_id` に示されているホスト名が使用されます(101 ページの「ノードのホスト名の判断」を参照してください)。111 ページの「ノードの定義」を参照してください。

3. (オプション)ツリー表示に最初のノードのアイコンが表示されたら、手順 2「ノードの定義」をクリックして、クラスタのその他のノードを定義します。ホスト名 /IP アドレスのペアおよびネットワークの優先度は、クラスタ内の各ノードに対して同じでなければなりません。

メモ: ツリー表示にこのノードのアイコンが表示されるまで、別のノードは定義しないでください。ノードを追加するのが早すぎると(データベースにノードを含めることができる前にノードを追加すると)、エラーが発生します。

各ノードに対してこの手順を繰返します。大規模なクラスタの場合は、最初の 3 つのノードを定義してから次の手順に進むことが推奨されています。つまり、正常な小さいクラスタを作成してから残りのノードを追加します。

4. 「クラスタの定義(Define a Cluster)」をクリックして、クラスタ定義を作成します。132 ページの「クラスタの定義」を参照してください。ツリー表示にクラスタが表示されていることを確認します(「表示(View)」->「クラスタ内のノード(Nodes in Cluster)」を選択してください)。
5. 「クラスタ内のノードの追加と削除(Add/Remove Nodes in Cluster)」をクリックして、新しいクラスタにノードを追加します。120 ページの「GUI を使ったクラスタ内のノードの追加と削除」を参照してください。

   「次へ(Next)」をクリックして、タスクの 2 番目の画面に移動します。
6. (オプション)「接続性のテスト(Test Connectivity)」をクリックして、ノードが物理的に接続されていることを確認します。250 ページの「GUI を使った接続性のテスト」を参照してください。このテストでは、`/etc/.rhosts` ファイルが正しく設定されている必要があります。
7. 「HA サービスの開始(Start HA Services)」をクリックします。
8. 「閉じる(Close)」をクリックします。「閉じる」をクリックすると、タスクセットは終了します。これはタスクを元に戻すものではありません。

高可用性リソース・グループの設定

メモ: タスク内で青いテキストをクリックすると、その概念や入力フィールドに関する詳細を参照できます。どのタスクでも、「OK」をクリックするまでクラスタ設定は更新されません。

「高可用性リソース・グループの設定(Set Up a Highly Available Resource Group)」タスクセットを使用すると、リソース・グループを定義するために必要な手順を実行できます。このタスクセットには、ほかの箇所で詳しく説明されているタスクが含まれます。

以下を実行します。

1. 新しいリソースを定義します。161 ページの「新規リソースの定義」を参照してください。

2. 必要なリソース依存性を追加します。168 ページの「リソース定義の依存性の追加と削除」を参照してください。
3. リソースおよび依存性を確認します。251 ページの「GUI を使ったリソースのテスト」を参照してください。
4. リソースが実行できる場所を指定したフェイルオーバー・ポリシーを定義します。173 ページの「フェイルオーバー・ポリシーの定義」を参照してください。
5. フェイルオーバー・ポリシーをテストします。251 ページの「GUI を使ったフェイルオーバー・ポリシーのテスト」を参照してください。
6. 前の手順で定義したフェイルオーバー・ポリシーを使用するリソース・グループを定義します。184 ページの「リソース・グループの定義」を参照してください。
7. リソース・グループ内でリソースを追加または削除します。251 ページの「GUI を使ったフェイルオーバー・ポリシーのテスト」を参照してください。
8. HA サービスを開始すると開始されるよう、リソース・グループ内のリソースを設定します。239 ページの「リソース・グループをオンラインにする」を参照してください。
9. FailSafe HA サービスが開始されていない場合は、そのサービスを開始します。190 ページの「GUI を使った FailSafe HA サービスの開始」を参照してください。

各リソース・グループに対して上記の手順を繰返します。

FailSafe の既存の CXFS クラスタの設定

CXFS もインストールされている場合、このタスクセットは GUI に表示されます。

メモ: タスク内で青いテキストをクリックすると、その概念や入力フィールドに関する詳細を参照できます。どのタスクでも、「OK」をクリックするまでクラスタ設定は更新されません。

「FailSafeの既存の CXFS クラスタの設定(Set Up an Existing CXFS Cluster for FailSafe)」タスクセットを使用すると、既存の CXFS ノードおよびクラスタを FailSafe に切替えるために必要な手順を実行できます。このタスクセットには、ほかの箇所で詳しく説明されているタスクが含まれます。

FailSafe および CXFS のデータベースは共通のものが 1 つだけ存在します。特定のノードが両方の製品に適用される場合は、変更内容が両方の製品に対して適切であることを確認してください。

以下を実行します。

1. 「CXFS クラスタの FailSafe への切り替え(Convert a CXFS Cluster to FailSafe)」をクリックします。これにより、クラスタ・タイプは CXFS and FailSafe に変わります。137 ページの「GUI を使った CXFS クラスタの FailSafe への切替え」を参照してください。
2. CXFS GUI を使用して、切替えるノードの CXFS サービスを停止します。『CXFS Version 2 Software Installation and Administration Guide』を参照してください。
3. 「CXFS ノードの FailSafe への切り替え(Convert a CXFS Node to FailSafe)」をクリックして、ローカル・ノード(接続しているノード)に切替えます。切替えたノードのタイプは、CXFS and FailSafe または CXFS になります。125 ページの「GUI を使った CXFS ノードの FailSafe への切替え」を参照してください。
4. 「CXFS ノードの FailSafe への切り替え(Convert a CXFS Node to FailSafe)」をクリックして、別のノードに切替えます。切替えたいノードに対して、この手順を繰返します。
5. 「HA サービスの開始(Start HA Services)」をクリックします。

クラスタ・ノードの修正またはアップグレード

以下のタスクを使用して、ノードを修正またはアップグレードすることができます。

- クラスタ・ノードの FailSafe HA サービスを停止します。191 ページの「FailSafe HA サービスの停止」を参照してください。
- ノードで必要なメインテナンスを実行します。これは、必要な場合にのみ行ってください(245 ページの「GUI を使ったノードのリセット」を参照してください)。
- ツリー表示で、クラスタのコンポーネントの状態をモニタします。223 ページの「システムのステータス」を参照してください。

既存のクラスタの変更

クラスタ管理者にイベントを通知する方法の変更など、ほとんどの変更は、HA サービスがアクティブなときでも実行できます。ただし、接続性をテストするときは、まずクラスタ・サービスを停止してください。

以下を参照してください。

- 135 ページの「クラスタ定義の変更」
- 111 ページの「ノードの定義」
- 250 ページの「GUI を使った接続性のテスト」
- 120 ページの「GUI を使ったクラスタ内のノードの追加と削除」

- 194 ページの「FailSafe HA パラメータの設定」

ノード使用の最適化

クラスタのパフォーマンスは、特定のノードのハードウェアを活用することによって向上させることができます。たとえば、クラスタ内の 1 つのノードが、ほかのノードよりも大きいディスクまたは高速な CPU を持っている場合があります。

状況によっては、以下のタスクが便利な場合があります。

- リソース・グループがよりパワフルなノードで常に実行されるようにします。このためには、そのようなノードをフェイルオーバー・ドメインで最初にリストし、回復属性として `Automatic` を選択します。179 ページの「GUI を使ったフェイルオーバー・ポリシー定義の変更」を参照してください。
- 188 ページの「リソース・グループの移動」.
- 特定のノードでカスタム定義を持つリソースを作成します。167 ページの「特定ノードへのリソースの再定義」を参照してください。
- （クラスタ全体ではなく）選択したノードに対してのみ定義されたリソース・タイプを作成します。167 ページの「特定ノードへのリソースの再定義」を参照してください。

カスタム・リソースの定義

カスタム・リソースを定義するには、以下のタスクを使用できます。

- 141 ページの「リソース・タイプの定義」
- 167 ページの「特定ノードへのリソースの再定義」
- 152 ページの「リソース・タイプの依存性の追加と削除」
- 161 ページの「新規リソースの定義」
- 168 ページの「リソース定義の依存性の追加と削除」
- 251 ページの「GUI を使ったリソースのテスト」

FailSafe 異常検出のカスタマイズ

以下を実行して、リソース・グループが FailSafe によってモニタおよびフェイルオーバーされる方法をカスタマイズできます。

- ノード・タイムアウトまたはハートビート周期（FailSafe によってメッセージがノード間で送信される時間間隔）を変更します。194 ページの「FailSafe HA パラメータの設定」を参照してください。
- リソース・タイプ別に使用されるモニタ・アクション・タイムアウトおよび再開アクション・タイムアウトを変更します。155 ページの「リソース・タイプ定義の変更」を参照してください。

リソース・グループのフェイルオーバー処理のカスタマイズ

クラスタまたはリソース・グループのフェイルオーバー処理を変更するには、さまざまなタスクを使用できます。

- フェイルオーバーが必要な場合がクラスタによって検出される方法を変更するには、194 ページの「FailSafe HA パラメータの設定」を参照してください。
- ノード、およびフェイルオーバー・ドメイン内のノードの順序を変更するには、179 ページの「フェイルオーバー・ポリシー定義の変更」を参照してください。
- リソース・グループで使用されるリソース・タイプのモニタ設定を変更するには、155 ページの「リソース・タイプ定義の変更」を参照してください。

また、カスタム・フェイルオーバー・ポリシー・スクリプトを作成することもできます。

1. 『IRIS FailSafe Version 2 Programmer's Guide』を使用して、カスタム・フェイルオーバー・スクリプトを作成します。
2. 作成したスクリプトを `/var/cluster/ha/policies` ディレクトリ内に配置します。
3. FailSafe マネージャ GUI を再起動します。
4. 新しいカスタム・フェイルオーバー・スクリプトを使用するために、希望のフェイルオーバー・ポリシーを変更します。179 ページの「フェイルオーバー・ポリシー定義の変更」を参照してください。
5. GUI のツリー表示で「表示(View)」->「ノードが所有するグループ(Groups owned by Nodes)」を選択します。
6. あるノードから別のノードにリソース・グループを移動して、フェイルオーバーをシミュレートすることにより、スクリプトをテストします。ツリー表示でリソース・グループの処理を監視し、フェイルオーバー処理が予想どおりに行われていることを確認します。188 ページの「リソース・グループの移動」を参照してください。

リソースのフェイルオーバー処理のカスタマイズ

リソースのフェイルオーバー処理は、既存のアクション・スクリプトの編集または新しいスクリプトの作成を行うことによってカスタマイズできます。以下を実行します。

1. 変更するアクション・スクリプトのコピーを作成します。各リソース・タイプのアクション・スクリプトは、/var/cluster/ha/resource_types に含まれています。
2. コピーを編集するか、新しいスクリプトを作成します。『IRIS FailSafe Version 2 Programmer's Guide』を参照してください。
3. 編集したスクリプトまたは新しいスクリプトを、/var/cluster/ha/resource_types 内の適切なサブディレクトリに配置します。
4. FailSafe マネージャ GUI を再起動します。
5. そのリソース・タイプの新しいスクリプトを利用します。141 ページの「リソース・タイプの定義」および155 ページの「リソース・タイプ定義の変更」を参照してください。
6. 新しいリソース・タイプを使用してリソースを定義します。161 ページの「新規リソースの定義」を参照してください。
7. FailSafe によって新しいカスタム・リソースが管理できることを確認します。251 ページの「GUI を使ったリソースのテスト」を参照してください。
8. 新しいリソースを追加します。120 ページの「クラスタ内のノードの追加と削除」を参照してください。

クラスタでのリソース負荷の再分散

リソース・グループを設定し、それらがどのようにフェイルオーバーするかを監視した後、クラスタ内のノード全体で負荷を分散させるためにリソース・グループを異なって配布したい場合があります。

1. 現在の負荷を分析します。たとえば、Toolchest から IRIX システム・マネージャ・ツールを呼出した後、「システム・パフォーマンス(System Performance)」カテゴリを選択してグラフィカル・システム・モニタ・ウィンドウを起動し、「システム・リソースの表示(View System Resources)」タスクを選択して、さまざまなシステム負荷統計を表示します。詳細については、gr_osview(1) のマン・ページを参照してください。
2. ノード全体でリソース・グループを再分散する場合は、188 ページの「リソース・グループの移動」を参照してください。
3. 異なる順序でノードを使用する新しいフェイルオーバー・ポリシー、または異なるノードを使用する新しいフェイルオーバー・ポリシーを作成する場合は、以下を実行します。
   - ノードをさらに効率的に使用する新しいフェイルオーバー・ポリシーを作成します。173 ページの「フェイルオーバー・ポリシーの定義」を参照してください。
   - リソース・グループに対して新しいフェイルオーバー・ポリシーを使用します。185 ページの「リソース・グループ定義の変更」を参照してください。

- 新しいフェイルオーバー・ポリシーをアクティブにするために、リソース・グループを移動します。関連付けられているリソース・グループが移動されると、FailSafe は、フェイルオーバー・ポリシーを使用してのみ開始されます。188 ページの「リソース・グループの移動」を参照してください。

ノード・タスク

ノードはオペレーティング・システム (OS: Operating System) の 1 個のイメージで、通常は個々のコンピュータです。ノードは、ストレージ・システムとクラスタ内のノードを接続する SAN に接続されています。ノードが属することのできるクラスタは 1 つだけです。

この意味でのノードという用語は、SGI Origin 3000 または SGI 2000 システムのノードとは異なる意味を持ちます。

この節では、以下の設定タスクについて説明します。

- 「ノードの定義」
- 120 ページの「クラスタ内のノードの追加と削除」
- 121 ページの「ノード定義の変更」
- 125 ページの「CXFS ノードの FailSafe への切替え」
- 127 ページの「ノードの削除」
- 129 ページの「ノードの表示」

ノードの定義

この節では、ノードを定義する方法について説明します。

GUI を使ったノードの定義

定義する最初のノードは、クラスタ管理を実行するためにログインしたノードでなければなりません。`/etc/sys_id` に示されているホスト名を使用してください。

メモ: タスク内で青いテキストをクリックすると、その概念や入力フィールドに関する詳細を参照できます。どのタスクでも、「OK」をクリックするまでクラスタ設定は更新されません。

ノードを定義するには、以下を実行します。

1. 以下を入力します。

- 「ホスト名(Hostname)」: `mynode.company.com`(すべてのノードで解決される場合は、`mynode` に省略できます)など、定義するノードのホスト名。 `/etc/sys_id` に示されているホスト名を使用してください。`/etc/sys_id` ファイルに示されているとおりにホスト名を表示するには、`hostname`(1) コマンドを使用します。57 ページの「ホスト名解決: `/etc/sys_id`、`/etc/hosts`、`/etc/nsswitch.conf`」を参照してください。
- 「論理名(Logical Name)」: ホスト名と同じ名前、ホスト名の簡略名(`lilly` など)、またはまったく別の名前(`nodeA` など)。論理名は、アンダースコア (_) で始めたり、空白として使用される文字を含めたり、255 文字より長くすることはできません。

メモ: ノードの名前を変更する場合は、そのノードを削除した後、新しいノードを定義しなければなりません。

- 「受信クラスタ・メッセージ用ネットワーク(Networks for Incoming Cluster Messages)」: 以下を実行します。
  - 「ネットワーク(Network)」: プライベート・ネットワークの IP アドレスまたはホスト名を入力します。ホスト名は、 `/etc/hosts` ファイルで解決されなければなりません。ネットワークの優先度は、クラスタ内の各ノードに対して同じでなければなりません。ホスト名の使用についての詳細は、57 ページの「ホスト名解決: `/etc/sys_id`、 `/etc/hosts`、`/etc/nsswitch.conf`」を参照してください。プライベート・ネットワークが必要である理由については、7 ページの「プライベート・ネットワーク」を参照してください。
  - 「受信メッセージ(Messages to Accept)」: 適切なタイプを選択します。ネットワークを一時的に定義して、ネットワークでメッセージを受信しないようにする場合は、「なし(None)」設定を使用できます。
  - ネットワークをリストに追加するには、「追加(Add)」をクリックします。

    後でこのネットワークを変更する場合は、リストでそのネットワークをクリックして選択し、「変更(Modify)」をクリックします。

    リストからネットワークを削除する場合は、リストでそのネットワークをクリックして選択し、「削除(Delete)」をクリックします。
- 「ノード ID(Node ID)」: プール内のノード全体で一意な、1〜32767 の範囲の整数(オプション)。数字を指定しないと、FailSafe によって自動的に ID が計算されます。デフォルトの ID は、マシンのシリアル・ナンバーなどのマシン固有の情報を基にした 5 桁の数字で、連続する数字ではありません。ノードを定義した後にノード ID 番号を変更しないでください。
- 「パーティション ID(Partition ID)」: パーティションが設定された SGI Origin 3000 システムのパーティションを一意に定義します(オプション)。システムにパーティションが設定されていない場合、このフィールドは空のままにしてください。

メモ: このパーティション ID 値を決定するには、`mkpart`(1M) コマンドを使用してください。

– `-n` オプションは、パーティション ID をリストします(システムにパーティションが設定されていない場合は 0 になります)。

– `-l` オプションは、さまざまなパーティションのブリックをリストします(GUI では *rack#.slot#* 形式を使用してください)。

例は、次のとおりです(読みやすくするために出力を一部省略してあります)。

```
# mkpart -n
Partition id = 1
# mkpart -l
partition: 3 = brick: 003c10 003c13 003c16 ...
partition: 1 = brick: 001c10 001c13 001c16 ...
```

「パーティション ID(Partition ID)」フィールドには、以下のいずれかを入力できます。

```
1 001.10
```

「次へ(Next)」をクリックして、次の画面に移動します。

- ノードをリセットするためにシステム・コントローラ・ポートを使用するかどうかを選択できます。ノードをリセットするために FailSafe でシステム・コントローラを使用できるようにする場合は、「リセット・パラメータの設定(Set Reset Parameters)」チェックボックスをオンにし、次の情報を指定します。

  – このノード:

    - 「ポート・タイプ(Port Type)」:「L1」(Origin 300、Origin 3200C、Onyx 300、および Onyx 3200C システム用 L1 システム・コントローラ)、「L2」(Origin 3400、Origin 3800、NUMAlink モジュール付き Origin 300、および Onyx 3000 シリーズ用 L2 システム・コントローラ)、「MSC」(Origin 200、Onyx2 deskside、および SGI 2100/2200 deskside システム用モジュール・システム・コントローラ)、または「MMSC」(rackmount SGI 2400、SGI 2800、および Onyx2 システム用マルチモジュール・システム・コントローラ)を選択します。215 ページの「Origin 300、Origin 3200C、Onyx 300、および Onyx 3200C のコンソール・サポート」も参照してください。

    - 「ポート・パスワード(Port Password)」: 特権コマンドに対するシステム・コントローラのパスワードで、ノードの `root` パスワードや PROM パスワードではありません。マシンによっては、システム管理者がこのパスワードを設定していない可能性があります。システム・コントローラ・ポート・パスワードを設定または変更する場合は、ご使用のノードのハードウェア・マニュアルを参照してください。

- 「ポートを一時的に無効にする(Temporarily Disable Port)」: リセット情報を指定して、現時点ではリセット機能を使用できないようにする場合は、このボックスをオンにします。このボックスがオンになっていると、FailSafe はノードをリセットできません。

– オーナー(リセット・コマンドを送信するノード):

- 「論理名(Logical Name)」: リモート・リセット・コマンドを送信するノードの名前。シリアル・ケーブルは、システム・コントローラ・ポートを通じて、定義されているノードとオーナー・ノードに物理的に接続されていなければなりません。ランタイム時に、ノードはプールで定義される必要があります。

  論理名を選択したり、まだ定義されていないノードの論理名を入力できます。ただし、ノードは、そのノードの接続性診断タスクを実行する前に定義しなければなりません。

- 「TTY デバイス(TTY Device)」: システム・コントローラが接続されているオーナー・ノードの端末ポート (TTY) の名前(/dev/ttyd2 など)。オーナー・ノードからノードをリモートで制御できるよう、ケーブルのもう一方はこのノードのシステム・コントローラ・ポートに接続します。

リセット機能をまったく使用しない場合は、「リセット・パラメータの設定(Set Reset Parameters」ボックスをクリックして、選択を解除します(オフにします)。

2. 「OK」をクリックして、タスクを完了します。

ネットワーク・インタフェース・フィールドへの入力として、ホスト名または IP アドレスを使用できます。ただし、ホスト名を使用するにはノードに DNS が必要となるため、実際の IP アドレスを使用することをお薦めします。

メモ: 最初のノードのアイコンがツリー表示で表示されるまで、2 番目のノードは追加しないでください。クラスタ・データベースに変更を加えると、その都度クラスタ全体のステータス情報が送信されます。したがって、設定が大きくなるほど送信に時間がかかります。

---

cmgr を使ったノードの定義

ノードを定義するには、以下のコマンドを使用します。

```
define node logical_hostname
    set hostname to hostname
    set nodeid to nodeID
    set partition_id to partitionID
    set reset_type to powerCycle
    set sysctrl_type to msc|mmsc|l2|l1_(based_on_node_hardware)
    set sysctrl_password to password
    set sysctrl_status to enabled|disabled
```

```
set sysctrl_owner to node_sending_reset_command
set sysctrl_device to /dev/ttyd2
set sysctrl_owner_type to tty_device
set is_failsafe to true|false
set is_cxfs to true|false
set weight to 0|1
add nic IP_address_or_hostname_(if_DNS)
        set heartbeat to true|false
        set ctrl_msgs to true|false
        set priority to integer
remove nic IP_address_or_hostname_(if_DNS)
```

上記のコマンド書式について、次の点に注意してください。

- node は、ホスト名（mynode.company.com など）と同じであるか、ホスト名の簡略名（mynode など）、またはまったく別の名前（nodeA など）です。論理名は、アンダースコア (_) で始めたり、空白として使用される文字を含めたり、255 文字より長くすることはできません。
- hostname は、定義されているノードで hostname(1) コマンドを実行することによって返されるホスト名です。プール内のその他のノードでは、/etc/hosts または名前解決メカニズムによって、このホスト名を正しく解決できなければなりません。*hostname* のデフォルトは、*logical_hostname* の値であるため、*logical_hostname* に対してホスト名またはホスト名の簡略名以外の値を使用する場合は、このコマンドの値を入力してください。
- nodeid は、プール内のノード全体で一意な、1〜32767 の範囲の整数です。数字を指定しないと、FailSafe によって自動的に ID が計算されます。デフォルトの ID は、マシンのシリアル・ナンバーなどのマシン固有の情報を基にした 5 桁の数字で、連続する数字ではありません。ノードを定義した後にノード ID 番号を変更しないでください。
- partition_id は、パーティションが設定された SGI Origin 3000 システムのパーティションを一意に定義します。

メモ: この値を決定するには、mkpart(1M) コマンドを使用してください。

– -n オプションは、パーティション ID をリストします(システムにパーティションが設定されていない場合は 0 になります)。

– -l オプションは、さまざまなパーティションのブリックをリストします(cmgr では *rack#.slot#* 形式を使用してください)。

例は、次のとおりです(読みやすくするために出力を一部省略してあります)。

```
# mkpart -n
Partition id = 1
# mkpart -l
partition: 3 = brick: 003c10 003c13 003c16 ...
partition: 1 = brick: 001c10 001c13 001c16 ...
```

「パーティション ID(Partition ID)」フィールドには、以下のいずれかを入力できます。

```
1 001.10
```

システムにパーティションが設定されていない場合は、0 という値を使用します。

パーティション ID の設定を解除するには、0 または none という値を使用します。

- reset_type に対しては、 powerCycle だけが正しい値です。
- sysctrl_type はシステム・コントローラ・タイプで、表5-1 に示すように、ノードのハードウェアに基づいています。
- sysctrl_password はシステム・コントローラ・ポートのパスワードで、ノードの root パスワードや PROM パスワードではありません。ノードによっては、システム管理者がこのパスワードを設定していない可能性があります。システム・コントローラ・パスワードを設定または変更する場合は、ご使用のノードのハードウェア・マニュアルを参照してください。
- sysctrl_status は、 enabled または disabled のいずれかです。ここでは、システム・コントローラに関する情報を指定できますが、この値を disabled(つまり、FailSafe でノードをリセットできません)に設定すると、一時的に無効にできます。FailSafe がノードをリセットできるようにするには、disabled と入力します。
- sysctrl_owner は、システム・コントローラ・ポートを介してこのノードをリセットできるノードの論理名です。ノードがハートビート・メッセージに応答していないこと、または要求に正しく応答していないことが検出された場合、このノードは別のノードをリセットできます。オーナーのシリアル・ポートの 1 つと、定義されているノードのシステム・コントローラ・ポートがシリアル・ケーブルで物理的に接続されている必要があります。オーナーは、プール内のノードにしてください。まだ定義されていない

ノードの名前を指定できますが、オーナーは、ノードの接続性診断タスクを実行する前、さらにクラスタがアクティブになる前に、ノードとして定義する必要があります。

- sysctrl_device はシステム・コントローラ・デバイスで、/dev/ttyd2 だけが正しい値です。
- sysctrl_owner_type は、システム・コントローラが接続されているオーナー・ノードの端末ポート (TTY) の名前(/dev/ttyd2 など)です。一方の端からノードをリモートで制御できるよう、ケーブルのもう一方はこのノードのシステム・コントローラ・ポートに接続します。
- is_failsafe および is_cxfs は、ノード・タイプを指定します。このノードで FailSafe だけを実行している場合は、is_cxfs を false に、 is_failsafe を true にそれぞれ設定します。同時実行クラスタ内にあるこのノードで CXFS と FailSafe の両方を実行している場合は、両方の値を true に設定してください。
- weight はノードの重みで、CXFS に対してのみ使用されます。
- nic は、プライベート・ネットワークの IP アドレスまたはホスト名です。ホスト名は、/etc/hosts ファイルで解決されなければなりません。

  ネットワーク・インタフェースは 8 つまで指定できますが、優先度が最も高いネットワークのみが使用されます(フェイルオーバーはありません)。このネットワークはプライベートにする必要があります。7 ページの「プライベート・ネットワーク」を参照してください。

  ネットワークの優先度は、クラスタ内の各ノードに対して同じでなければなりません。ホスト名の使用についての詳細は、57 ページの「ホスト名解決: /etc/sys_id、 /etc/hosts、/etc/nsswitch.conf」を参照してください。プライベート・ネットワークが必要である理由については、7 ページの「プライベート・ネットワーク」を参照してください。

表5-1 システム・コントローラ・タイプ

l1	l2	mmsc	msc
Origin 300（215 ページの「Origin 300、Origin 3200C、Onyx 300、および Onyx 3200C のコンソール・サポート」も参照してください。）	Origin 3400	SGI 2400 rackmount	Origin 200
Origin 3200c	Origin 3800	SGI 2800 rackmount	Onyx2 deskside
Onyx 300	Origin 300（NUMAlink モジュール付き）	Onyx2 rackmount	SGI 2100 deskside
Onyx 3200c	Onyx 3000 シリーズ		SGI 2200 deskside

ネットワーク・インタフェースを定義するには、add nic コマンドを使用します。このコマンドを入力すると、次のプロンプトが表示されます。

```
NIC - nic#?
```

このプロンプトが表示されたら、以下のコマンドを使用して、コントロール・ネットワークに対してフラグを指定します。

```
set heartbeat to true|false
set ctrl_msgs to true|false
set priority to integer
```

次のコマンドを使用して、ノード名プロンプトからネットワーク・コントローラを削除できます。

```
remove nic IP_address
```

ノードの定義が終了したら、done と入力します。

以下の例では、コントローラが 1 つある cm1a という FailSafe ノードを定義します。

```
cmgr> define node cm1a
Enter commands, you may enter "done" or "cancel" at any time to exit

cm1a? set hostname to cm1a
cm1a? set nodeid to 1
cm1a? set reset_type to powerCycle
cm1a? set sysctrl_type to msc
cm1a? set sysctrl_password to [ ]
```

```
cm1a? set sysctrl_status to enabled
cm1a? set sysctrl_owner to cm2
cm1a? set sysctrl_device to /dev/ttyd2
cm1a? set sysctrl_owner_type to tty
cm1a? set is_failsafe to true
cm1a? set is_cxfs to true
cm1a? add nic cm1
Enter network interface commands, when finished enter "done"
or "cancel"

NIC - cm1 > set heartbeat to true
NIC - cm1 > set ctrl_msgs to true
NIC - cm1 > set priority to 0
NIC - cm1 > done
cm1a? done
```

-p オプションを使用して cmgr を呼出した場合、または set prompting on コマンドを入力した場合、表示は次の例のようになります。

```
cmgr> define node cm1a
Enter commands, when finished enter either "done" or "cancel"

Hostname[optional]? cm1a
Is this a FailSafe node <true|false> ? true
Is this a CXFS node <true|false> ? false
Node ID ? 1
Reset type <powerCycle> ? (powerCycle)
Do you wish to define system controller info[y/n]:y
Sysctrl Type <msc|mmsc|l2|l1>? (msc) msc
Sysctrl Password [optional]? ( )
Sysctrl Status <enabled|disabled>? enabled
Sysctrl Owner? cm2
Sysctrl Device? /dev/ttyd2
Sysctrl Owner Type <tty> [tty]?
Number of Network interfaces [2]? 2
NIC 1 - IP Address? cm1
NIC 1 - Heartbeat HB (use network for heartbeats) <true|false>? true
NIC 1 - (use network for control messages) <true|false>? true
NIC 1 - Priority <1,2,...>? 1
NIC 2 - IP Address? cm2
NIC 2 Heartbeat HB (use network for heartbeats) <true|false>? true
NIC 2 - (use network for control messages) <true|false>? false
NIC 2 - Priority <1,2,...>? 2
```

クラスタ内のノードの追加と削除

この節では、ノードを追加または削除する方法について説明します。

GUI を使ったクラスタ内のノードの追加と削除

プールにノードを追加して、クラスタを定義したら、クラスタ内に含めるノードを指定できます。

メモ: クラスタ・アイコンがツリー表示に表示されるまで、ノードを追加したり削除しないでください(「表示(View)」->「クラスタ内のノード(Nodes in Cluster)」を選択してください)。

以下を実行します。

1. 希望のノードを追加または削除します。
   - ノードを追加するには、その論理名を「利用可能なノード(Available Nodes)」メニューから選択し、「追加(Add)」をクリックします。ノード名は、「クラスタに追加するノード(Nodes to Go into Cluster)」リストに表示されます。
   - ノードを削除するには、その論理名を「クラスタに追加するノード(Nodes to Go into Cluster)」リストでクリックします(論理名が強調表示されます)。その後、「削除(Remove)」をクリックします。
2. 「OK」をクリックして、タスクを完了します。

ノード定義の変更

この節では、ノード定義を変更する方法について説明します。

GUI を使ったノード定義の変更

メモ: ノードの名前を変更する場合は、そのノードを削除した後、新しいノードを定義しなければなりません。

ノードを変更するには、以下を実行します。

1. 「論理名(Logical Name)」: ノードの論理名を選択します。論理名を選択すると、このノードの情報がさまざまなフィールドに入力されます。
2. 該当するフィールドの情報を、以下のように変更します。
   - 「受信クラスタ・メッセージ用ネットワーク(Networks for Incoming Cluster Messages)」: ネットワークの優先度は、クラスタ内の各ノードに対して同じでなければなりません。
     - 「ネットワーク(Network)」: 受信クラスタ・メッセージ用ネットワークを追加する場合は、IP アドレスまたはホスト名を「ネットワーク」テキスト・フィールドに入力し、「追加(Add)」をクリックします。
       - すでにリストに存在しているネットワークを変更する場合は、リストでそのネットワークをクリックして選択します。その後、「変更(Modify)」をクリックします。これにより、ネットワークは、リストからテキスト入力領域に移動されます。これで、ネットワークを変更できます。ネットワークをリストに追加し直すには、「追加」をクリックします。
       - ネットワークを削除する場合は、優先度リストでそのネットワークをクリックして選択します。その後、「削除(Delete)」をクリックします。
       - ネットワークの優先度を変更する場合は、優先度リストでそのネットワークをクリックして選択します。その後、上下の矢印をクリックして、リスト内でネットワークを別の位置に移動します。
     - 「パーティション ID(Partition ID)」: パーティションが設定された SGI Origin 3000 システムのパーティションを一意に定義します(オプション)。システムにパーティションが設定されていない場合、このフィールドは空のままにしてください。

メモ: このパーティション ID 値を決定するには、`mkpart`(1M) コマンドを使用してください。

- `-n` オプションは、パーティション ID をリストします（システムにパーティションが設定されていない場合は 0 になります）。
- `-l` オプションは、さまざまなパーティションのブリックをリストします（`cmgr` では *rack#.slot#* 形式を使用してください）。

  例は、次のとおりです（読みやすくするために出力を一部省略してあります）。

```
# mkpart -n
Partition id = 1
# mkpart -l
partition: 3 = brick: 003c10 003c13 003c16 ...
partition: 1 = brick: 001c10 001c13 001c16 ...
```

  「パーティション ID(Partition ID)」フィールドには、以下のいずれかを入力できます。

```
1 001.10
```

「次へ(Next)」をクリックして、次のページに移動します。

- ノードをリセットするためにシステム・コントローラ・ポートを使用するかどうかを選択できます。ノードをリセットするために FailSafe でシステム・コントローラを使用できるようにする場合は、「リセット・パラメータの設定(Set Reset Parameters)」チェックボックスをオンにし、次の情報を指定します。
  - このノード:
    - 「ポート・タイプ(Port Type)」:「L1」（Origin 300、Origin 3200C、Onyx 300、および Onyx 3200C システム用 L1 システム・コントローラ)、「L2」（Origin 3400、Origin 3800、NUMAlink モジュール付き Origin 300、および Onyx 3000 シリーズ用 L2 システム・コントローラ）、「MSC」（Origin 200、Onyx2 deskside、および SGI 2100/2200 deskside システム用モジュール・システム・コントローラ）、または「MMSC」（rackmount SGI 2400、SGI 2800、および Onyx2 システム用マルチモジュール・システム・コントローラ）を選択します。215 ページの「Origin 300、Origin 3200C、Onyx 300、および Onyx 3200C のコンソール・サポート」も参照してください。
    - 「ポート・パスワード(Port Password)」: システム・コントローラ・ポートのパスワードで、ノードの `root` パスワードや PROM パスワードでは**ありません**。マシンによっては、システム管理者がこのパスワードを設定していない可能性があります。システム・コントローラ・ポート・パスワードを設定または変更する場合は、ご使用のノードのハードウェア・マニュアルを参照してください。

- 「ポートを一時的に無効にする(Temporarily Disable Port)」: リセット情報を指定して、現時点ではリセット機能を使用できないようにする場合は、このボックスをオンにします。このボックスがオンになっていると、FailSafe はノードをリセットできません。

– オーナー (リセット・コマンドを送信するノード):

- 「論理名(Logical Name)」: リモート・リセット・コマンドを送信するノードの名前。シリアル・ケーブルは、システム・コントローラ・ポートを通じて、定義されているノードとオーナー・ノードに物理的に接続されていなければなりません。ランタイム時に、ノードはプールで定義される必要があります。

  論理名を選択したり、まだ定義されていないノードの論理名を入力できます。ただし、ノードは、そのノードの接続性診断タスクを実行する前に定義しなければなりません。

- 「TTY デバイス(TTY Device)」: システム・コントローラが接続されているオーナー・ノードの端末ポート (TTY) の名前 (/dev/ttyd2 など)。オーナー・ノードからノードをリモートで制御できるよう、ケーブルのもう一方はこのノードのシステム・コントローラ・ポートに接続します。

リセット機能をまったく使用しない場合は、「リセット・パラメータの設定(Set Reset Parameters」ボックスをクリックして、選択を解除します (オフにします)。

3. 「OK」をクリックして、タスクを完了します。

cmgr を使ったノードの変更

既存のノードを変更するには、以下のコマンドを入力します。

```
modify node logical_hostname
    set hostname to hostname
    set nodeid to nodeID
    set partition_id to partitionID
    set reset_type to powerCycle
    set sysctrl_type to msc|mmsc|l2|l1_(based_on_node_hardware)
    set sysctrl_password to password
    set sysctrl_status to enabled|disabled
    set sysctrl_owner to node_sending_reset_command
    set sysctrl_device to /dev/ttyd2
    set sysctrl_owner_type to tty_device
    set is_failsafe to true|false
    set is_cxfs to true|false
    set weight to 0|1
    add nic IP_address_or_hostname_(if_DNS)
            set heartbeat to true|false
```

```
            set ctrl_msgs to true|false
            set priority to integer
    remove nic IP_address_or_hostname_(if_DNS)
```

コマンドは、ノードの定義に使用されるものと同じです。ノード定義時に指定した情報は、ノード ID を除いてすべて変更可能です。コマンドについての詳細は、114 ページの「cmgr を使ったノードの定義」を参照してください。

注意: ノードを定義した後にノード ID 番号を変更しないでください。

パーティションの設定例

以下に、SGI Origin 3000 システムのパーティション設定の例を示します。

```
# cmgr
Welcome to SGI Cluster Manager Command-Line Interface

cmgr> modify node n_preston
Enter commands, when finished enter either "done" or "cancel"

n_preston ? set partition_id to 1
n_preston ? done

Successfully modified node n_preston
```

プロンプト表示ありでこの機能を実行するには、以下を入力します。

```
# cmgr -p
Welcome to SGI Cluster Manager Command-Line Interface

cmgr> modify node n_preston
Enter commands, you may enter "done" or "cancel" at any time to exit

Hostname[optional] ? (preston.engr.sgi.com)
Is this a FailSafe node <true|false> ? (true)
Is this a CXFS node <true|false> ? (true)
Node ID[optional] ? (606)
Partition ID[optional] ? (0) 1
Reset type <powerCycle> ? (powerCycle)
Do you wish to modify system controller info[y/n]:n
Number of Network Interfaces ? (2)
NIC 1 - IP Address ? (preston)
```

```
NIC 1 - Heartbeat HB (use network for heartbeats) <true|false> ? (true)
NIC 1 - (use network for control messages) <true|false> ? (true)
NIC 1 - Priority <1,2,...> ? (1)
NIC 2 - IP Address ? (192.168.168.1)
NIC 2 - Heartbeat HB (use network for heartbeats) <true|false> ? (true)
NIC 2 - (use network for control messages) <true|false> ? (true)
NIC 2 - Priority <1,2,...> ? (2)
Node Weight ? (1)

Successfully modified node n_preston

cmgr> show node n_preston
Logical Machine Name: n_preston
Hostname: preston.engr.sgi.com
Node Is FailSafe: true
Node Is CXFS: true
Nodeid: 606
Partition id: 1
Reset type: powerCycle
ControlNet Ipaddr: preston
ControlNet HB: true
ControlNet Control: true
ControlNet Priority: 1
ControlNet Ipaddr: 192.168.168.1
ControlNet HB: true
ControlNet Control: true
ControlNet Priority: 2
Node Weight: 1
```

パーティション ID の設定を解除するには、0 または none という値を使用します。

CXFS ノードの FailSafe への切替え

この節では、CXFS にも適用するように FailSafe ノードを切替える方法について説明します。

GUI を使った CXFS ノードの FailSafe への切替え

CXFS もインストールされている場合、このタスクは GUI に表示されます。

既存の CXFS ノード（タイプ CXFS）を CXFS and FailSafe または CXFS のタイプに切替えるには、以下を実行します。

1. CXFS GUI を使用して、切替えるノードの CXFS サービスを停止します。『CXFS Version 2 Software Installation and Administration Guide』を参照してください。
2. ノードを切替えます。
   - 「論理名(Logical Name)」: ノードの論理名を選択します。
   - 「CXFS の設定を維持する(Keep CXFS Settings)」:
     - タイプ `CXFS and FailSafe` に切替えるには、チェックボックスをオンにします。
     - タイプ `FailSafe` に切替えるには、チェックボックスをオフのままにします。
   - 「OK」をクリックして、タスクを完了します。

メモ: ノードの名前を変更する場合は、そのノードを削除した後、新しいノードを定義しなければなりません。

その他のパラメータを変更するには、121 ページの「GUI を使ったノード定義の変更」を参照してください。変更内容が FailSafe と CXFS の両方に対して適切であることを確認してください。

`cmgr` を使ったノードの CXFS または FailSafe への切替え

既存の CXFS ノードを FailSafe にも適用するよう切替えるには、`modify` コマンドを使用して設定を変更します。

メモ: 対応する高可用性 (HA) サービスまたは CXFS サービスがアクティブな場合は、ノードの FailSafe または CXFS をオフにできません。まず、ノードのサービスを停止してください。

通常モードの場合:

```
cmgr> modify node cxfs6
Enter commands, when finished enter either "done" or "cancel"

cxfs6 ? set is_FailSafe to true
cxfs6 ? done

Successfully modified node cxfs6
```

プロンプト・モードの場合:

```
cmgr> modify node cxfs6
Enter commands, you may enter "done" or "cancel" at any time to exit
```

```
Hostname[optional] ? (cxfs6.americas.sgi.com)
Is this a FailSafe node <true|false> ? (false) true
Is this a CXFS node <true|false> ? (true)
Node ID[optional] ? (13203)
Partition ID[optional] ? (0)
Reset type <powerCycle> ? (powerCycle)
Do you wish to modify system controller info[y/n]:n
Number of Network Interfaces ? (1)
NIC 1 - IP Address ? (cxfs6)
NIC 1 - Heartbeat HB (use network for heartbeats) <true|false> ? (true)
NIC 1 - (use network for control messages) <true|false> ? (true)
NIC 1 - Priority <1,2,...> ? (1)
Node Weight ? (0)

Successfully modified node cxfs6
```

ノードの削除

この節では、ノードを削除する方法について説明します。

GUI を使ったノードの削除

プールからノードを削除する前に、クラスタからそのノードを削除してください。詳細については、135 ページの「クラスタ定義の変更」を参照してください。

ノードを削除するには、以下を実行します。

1. 「削除するノード(Node to Delete)」: 削除するノードの論理名を選択します。
2. 「OK」をクリックして、タスクを完了します。

`cmgr` を使ったノードの削除

ノードを削除するには、次のコマンドを使用します。

```
delete node hostname
```

ノードを削除できるのは、現在そのノードがクラスタの一部でない場合だけです。クラスタに現在ノードが含まれている場合は、そのクラスタをまず変更して、ノードを削除しなければなりません。

たとえば、次のように設定されている cxfs6-8 というクラスタがあるとします。

```
cmgr> show cluster cxfs6-8
Cluster Name: cxfs6-8
Cluster Is FailSafe: true
Cluster Is CXFS: true
Cluster ID: 20
Cluster HA mode: normal
Cluster CX mode: normal
FileSystem Device Name: /dev/cxvm/dks2d72s0
FileSystem Mount Point: /mnts/fs1
FileSystem Mount Options:
FileSystem Status: enabled
FileSystem Force flag: false
        FileSystem Server Node: cxfs6
        FileSystem Server Rank: 0
        FileSystem Server Node: cxfs7
        FileSystem Server Rank: 1
FileSystem Device Name: /dev/cxvm/dks2d73s0
FileSystem Mount Point: /mnts/fs2
FileSystem Mount Options: enabled
FileSystem Status: enabled
FileSystem Force flag: false
        FileSystem Server Node: cxfs8
        FileSystem Server Rank: 0


Cluster cxfs6-8 has following 3 machine(s)
        cxfs6
        cxfs7
        cxfs8
```

cxfs8 ノードを削除するには、プロンプト・モードで以下を実行します(CXFS サービスと FailSafe HA サービスがそのノードで停止されていると想定しています)。

```
cmgr> modify cluster cxfs6-8
Enter commands, when finished enter either "done" or "cancel"

Is this a FailSafe cluster <true|false> ? (true)
Is this a CXFS cluster <true|false> ? (true)
Cluster Notify Cmd [optional] ?
Cluster Notify Address [optional] ?
Cluster HA mode <normal|experimental>[optional] ? (normal)
```

```
Cluster ID ? (20)
Number of Cluster FileSystems ? (0)

Current nodes in cluster cxfs6-8:
Node - 1: cxfs6
Node - 2: cxfs7
Node - 3: cxfs8

Add nodes to or remove nodes from cluster cxfs6-8
Enter "done" when completed or "cancel" to abort

cxfs6-8 ? remove node cxfs8
cxfs6-8 ? done
Successfully modified cluster cxfs6-8

cmgr> show cluster cxfs6-8
Cluster Name: cxfs6-8
Cluster Is FailSafe: true
Cluster Is CXFS: true
Cluster ID: 20
Cluster HA mode: normal


Cluster cxfs6-8 has following 2 machine(s)
        cxfs6
        cxfs7
```

プールから cxfs8 を削除するには、次を入力します。

```
cmgr> delete node cxfs8

IMPORTANT: NODE cannot be deleted if it is a member of a cluster.
The LOCAL node can not be deleted if some other nodes are still defined.


Deleted machine (cxfs6).
```

ノードの表示

この節では、ノードを表示する方法について説明します。

GUI を使ったノードの表示

ノードを定義したら、以下を表示できます。

1. 定義されているノード(「プール内のノード(Nodes in Pool)」
2. 特定のクラスタのメンバーであるノード(「クラスタ内のノード(Nodes In Cluster)」)
3. ノードの属性

ファイルシステムを表示するには、「表示(View)」->「ファイルシステム(Filesystems)」を選択します。

詳細なステータスおよび設定情報を表示するには、ツリー表示で任意の名前またはアイコンをクリックします。

cmgr を使ったノードの表示

ノードを定義したら、次のコマンドを使用して、そのノードのパラメータを表示できます。

```
show node nodename
```

ノード cm1a の show node コマンドによる表示は、次のとおりです。

```
cmgr> show node cm1
Logical Machine Name: cm1
Hostname: cm1
Node Is FailSafe: true
Node is CXFS: false
Nodeid: 1
Reset type: powerCycle
System Controller: msc
System Controller status: enabled
System Controller owner: cm2
System Controller owner device: /dev/ttyd2
System Controller owner type: tty
ControlNet Ipaddr: cm1
ControlNet HB: true
ControlNet Control: true
ControlNet Priority: 0
```

定義されているすべてのノードのリストは、次のコマンドを使用して表示できます。

```
show nodes in pool
```

例は、次のとおりです。

```
cmgr> show nodes in pool

3 Machine(s) defined
        cxfs8
        cxfs6
        cxfs7
```

デフォルトのクラスタが指定されている場合は、このコマンドを使用する際にクラスタを指定する必要はありません。例は、次のとおりです。

```
cmgr> set cluster cxfs6-8
cmgr> show nodes

Cluster cxfs6-8 has following 3 machine(s)
        cxfs6
        cxfs7
        cxfs8
```

クラスタ・タスク

クラスタは、クラスタとして定義されているプール内のノードのセットです。クラスタは単純な名前で識別されます。この名前は、プール内で一意でなければなりません。たとえば、クラスタとノードで同じ名前を使用することはできません。

クラスタ内のすべてのノードはプールにも存在しますが、プール内のすべてのノードがクラスタに存在するとはかぎりません。つまり、クラスタは、プール内のノードのサブセットで構成されている場合があります。各プールのクラスタは 1 つだけです。

この節では、以下のクラスタ設定タスクについて説明します。

- 132 ページの「クラスタの定義」
- 135 ページの「クラスタ定義の変更」
- 137 ページの「CXFS クラスタの FailSafe への切替え」
- 138 ページの「クラスタの削除」
- 139 ページの「クラスタの表示」

クラスタの定義

この節では、クラスタを定義する方法について説明します。

GUI を使ったクラスタの定義

クラスタは、プライベート・ネットワークによって互いに連結されたノードの集合です。クラスタは、単純な名前で識別されます。特定のノードは、1 つだけのクラスタのメンバーにできます。

クラスタを定義するには、以下を実行します。

1. 以下を入力します。
   - 「クラスタ名(Cluster Name)」: クラスタの論理名。名前は 255 文字以下にしてください。
   - 「クラスタ・モード(Cluster Mode)」: クラスタは、通常、デフォルトの通常 (`Normal`) モードに設定します。

     試験中 (`Experimental`) にモードを設定すると、ノードをリセットさせることなくクラスタをデバッグできるよう、リセットがオフになります。試験中 (`Experimental`) モードは、デバッグする場合にのみ使用することをお薦めします。
   - 「管理者に通知(Notify Administrator)」(クラスタおよびノードのステータス変更について):
     - 「メールで(By e-mail)」: これを選択した場合、電子メール・プログラム(デフォルトでは `/usr/sbin/Mail`)と、管理者の電子メール・アドレスを指定する必要があります。複数のアドレスを指定するには、各アドレスをカンマで区切ります。FailSafe では、ノードまたはクラスタのステータスが変更されるたびに、電子メールがそれらのアドレスに送信されます。アドレスを指定しなかった場合、通知は送信されません。
     - 「コマンドで(By other command)」: これを選択した場合、ノードまたはクラスタのステータスが変更されるたびに実行されるコマンドを指定する必要があります。
     - 「なし(Never)」: これを選択した場合、通知が送信されないことが指定されます。
2. 「OK」をクリックして、タスクを完了します。このタスクの実行には時間がかかり、完了までに数分かかる場合があります。

`cmgr` を使ったクラスタの定義

`cmgr` を使用してクラスタを定義するときは、同じコマンドを使用して、クラスタの定義およびクラスタへのノードの追加を行います。一般情報については、132 ページの「クラスタの定義」を参照してください。

クラスタを定義するには、以下のコマンドを使用します。

```
define cluster clustername
    set is_failsafe to true|false
    set is_cxfs to true|false
    set notify_cmd to notify_command
    set notify_addr to email_address
    set ha_mode to normal|experimental
    set cx_mode to normal|experimental
    add node node1name
    add node node2name
    ...
```

上記のコマンド書式について、次の点に注意してください。

- `cluster` は、クラスタの論理名です。論理名は、アンダースコア (_) で始めたり、空白として使用される文字を含めたり、255 文字より長くすることはできません。
- `is_failsafe` および `is_cxfs` は、クラスタ・タイプを指定します。FailSafe だけを実行している場合は、 `is_cxfs` を `false` に、`is_failsafe` を `true` にそれぞれ設定します。同時実行クラスタを実行している場合は、両方の値を `true` に設定してください。
- `notify_cmd` は、ノードまたはクラスタのステータスが変更されるたびに実行されるコマンドです。
- `notify_addr` は、クラスタおよびノードのステータス変更が通知されるアドレスです。複数のアドレスを指定するには、各アドレスをカンマで区切ります。FailSafe では、ノードまたはクラスタのステータスが変更されるたびに、電子メールがそれらのアドレスに送信されます。アドレスを指定しなかった場合、通知は送信されません。`notify_addr` コマンドを使用する場合は、*notify_command* として電子メール・プログラム(`/usr/sbin/Mail` など)を指定する必要があります。
- `set ha_mode` および `set cx_mode` は、通常は通常 (`normal`) に設定します。試験中 (`experimental`) にモードを設定すると、ノードをリセットさせることなくクラスタをデバッグできるよう、リセットがオフになります。試験中 (`experimental`) モードは、デバッグする場合にのみ使用することをお薦めします。 `set cx_mode` コマンドは CXFS にのみ適用し、 `set ha_mode` コマンドは IRIS FailSafe にのみ適用します。

このタスクの実行には時間がかかり、完了までに数分かかる場合があります。また、FailSafe では、ノードにインストールされているリソース・タイプが新しいクラスタに追加されます。このプロセスには時間がかかります。

以下に、プロンプト表示ありのコマンドを示します。

```
cmgr> define cluster clustername
Enter commands, you may enter "done" or "cancel" at any time to exit
```

```
Is this a FailSafe cluster <true|false> ? true|false
Is this a CXFS cluster  <true|false> ? true|false
Cluster Notify Cmd [optional] ?
Cluster Notify Address [optional] ?
Cluster HA mode <normal|experimental> [optional] ? normal
No nodes in cluster clustername

Add nodes to or remove nodes from cluster clustername
Enter "done" when completed or "cancel" to abort

clustername ? add node node1name
clustername ? add node node2name
...
clustername ? done
Creating resource type MAC_address
Creating resource type IP_address
Creating resource type filesystem
Creating resource type volume
Successfully defined cluster clustername
```

クラスタは、デフォルトの normal モードに設定します。試験中 (Experimental) にモードを設定すると、ノードをリセットさせることなくクラスタをデバッグできるよう、リセットがオフになります。試験中 (Experimental) モードは、デバッグする場合にのみ使用することをお薦めします。ただし、実作業用のクラスタでは experimental モードを使用しないでください。このモードは、日本 SGI のカスタマ・サポートから指示があった場合にのみ使用してください。SGI では、カスタマが experimental を使用することを推奨していません。

例は、次のとおりです。

```
cmgr> define cluster fs6-8
Enter commands, you may enter "done" or "cancel" at any time to exit

Is this a FailSafe cluster <true|false> ? true
Is this a CXFS cluster  <true|false> ? false
Cluster Notify Cmd [optional] ?
Cluster Notify Address [optional] ?
Cluster HA mode <normal|experimental> [optional] ?

No nodes in cluster fs6-8

Add nodes to or remove nodes from cluster fs6-8
Enter "done" when completed or "cancel" to abort
```

```
fs6-8 ? add node fs6
fs6-8 ? add node fs7
fs6-8 ? add node fs8
fs6-8 ? done
Creating resource type MAC_address
Creating resource type IP_address
Creating resource type filesystem
Creating resource type volume
Successfully defined cluster fd6-8
```

プロンプト表示なしでこれを行うには、以下を入力します。

```
cmgr> define cluster fs6-8
Enter commands, you may enter "done" or "cancel" at any time to exit

cluster fs6-8? set is_failsafe to true
cluster fs6-8? add node fs6
cluster fs6-8? add node fs7
cluster fs6-8? add node fs8
cluster fs6-8? done
Creating resource type MAC_address
Creating resource type IP_address
Creating resource type filesystem
Creating resource type volume
Successfully defined cluster fs6-8
```

クラスタ定義の変更

この節では、クラスタ定義を変更する方法について説明します。

GUI を使ったクラスタ定義の変更

クラスタの状態の変更をクラスタ管理者に通知する方法を変更するには、以下を実行します。

1. 「クラスタ名(Cluster Name)」: クラスタの名前を選択します。
2. 「クラスタ・モード(Cluster Mode)」: クラスタは、通常、デフォルトの通常 (Normal) モードに設定します。試験中 (Experimental) モードについての詳細は、132 ページの「クラスタの定義」を参照してください。
3. 「管理者に通知(Notify Administrator)」(クラスタおよびノードのステータス変更について):

- 「メールで(By e-mail)」: これを選択した場合、電子メール・プログラム(デフォルトでは /usr/sbin/Mail)と、管理者の電子メール・アドレスを指定する必要があります。複数のアドレスを指定するには、各アドレスをカンマで区切ります。FailSafe では、ノードまたはクラスタのステータスが変更されるたびに、電子メールがそれらのアドレスに送信されます。アドレスを指定しなかった場合、通知は送信されません。
- 「コマンドで(By other command)」: これを選択した場合、ノードまたはクラスタのステータスが変更されるたびに実行されるコマンドを指定する必要があります。
- 「なし(Never)」: これを選択した場合、通知が送信されないことが指定されます。

4. 「OK」をクリックします。

クラスタを構成しているノードを変更するには、120 ページの「GUI を使ったクラスタ内のノードの追加と削除」を参照してください。

メモ: クラスタの名前を変更する場合は、そのクラスタを削除した後、新しいクラスタを定義しなければなりません。

---

cmgr を使ったクラスタ定義の変更

コマンドは、以下のとおりです。

```
modify cluster clustername
    set is_failsafe to true|false
    set is_cxfs to true|false
    set notify_cnd to command
    set notify_addr to email_address
    set ha_mode to normal|experimental
    set cx_mode to normal|experimental
    add node node1name
    add node node2name
    ...
    remove node node1name
    remove node node2name...
```

たとえば、クラスタ testcluster にノード newnode を追加するには、以下を入力します。

```
cmgr> modify cluster mycluster
cluster testcluster? add node newnode
cluster testcluster? done
cmgr>
```

CXFS クラスタの FailSafe への切替え

この節では、FailSafe にも適用されるよう CXFS クラスタを切替える方法について説明します。

GUI を使った CXFS クラスタの FailSafe への切替え

CXFS もインストールされている場合、このタスクは GUI に表示されます。

既存の CXFS クラスタ(つまり、タイプ CXFS)からの情報を切替えて、FailSafe(つまり、タイプ CXFS and FailSafe)にも適用されるクラスタを作成するには、以下を実行します。

1. 「クラスタ名(Cluster Name)」: クラスタの名前を選択します。
2. 「OK」をクリックして、タスクを完了します。

クラスタは、FailSafe と CXFS の両方に適用されるようになります。クラスタを構成しているノードを変更するには、120 ページの「クラスタ内のノードの追加と削除」を参照してください。

メモ: クラスタの名前を変更する場合は、そのクラスタを削除した後、新しいクラスタを定義しなければなりません。

---

cmgr を使った CXFS クラスタの FailSafe への切替え

cmgr を使用してクラスタを切替えるには、 modify cluster コマンドに続けて以下のコマンドを使用します。

```
modify cluster clustername
    set is_failsafe to true|false
    set is_cxfs to true|false
    set clusterid to clusterID
```

たとえば、FailSafe にも適用されるよう CXFS クラスタ TEST を切替えるには、以下を入力します。

```
cmgr> modify cluster TEST
Enter commands, when finished enter either "done" or "cancel"

TEST ?set is_failsafe to true
```

クラスタは、そのノードに関してオンにするすべての機能(FailSafe または CXFS、あるいはその両方)をサポートしていなければなりません。つまり、クラスタがタイプ CXFS の場合は、クラスタの一部であるノードをタイプ FailSafe または CXFS and FailSafe に変更することはできません。ただし、ノードがクラスタのすべての機能をサポートする必要はありません。つまり、タイプ CXFS and FailSafe のクラスタでタイプ CXFS のノードを使用できます。

クラスタの削除

この節では、クラスタを削除する方法について説明します。

GUI を使ったクラスタの削除

ノードが含まれているクラスタは削除できません。したがって、最初に、クラスタに含まれているノードをクラスタから移動してください。詳細については、135 ページの「クラスタ定義の変更」を参照してください。

クラスタを削除するには、以下を実行します。

1. 「削除するクラスタ(Cluster to Delete)」: クラスタの名前を選択します。
2. 「OK」をクリックして、タスクを完了します。

cmgr を使ったクラスタの削除

クラスタを削除するには、次のコマンドを使用します。

```
delete cluster clustername
```

ただし、ノードが含まれているクラスタは削除できません。したがって、ノードで HA サービスをまず停止した後、クラスタを再定義して、ノードが含まれないようにする必要があります。

通常モードの場合:

```
cmgr> modify cluster fs6-8
Enter commands, when finished enter either "done" or "cancel"

fs6-8 ? remove node fs6
fs6-8 ? remove node fs7
fs6-8 ? remove node fs8
fs6-8 ? done
Successfully modified cluster fs6-8

cmgr> delete cluster fs6-8

cmgr> show clusters

cmgr>
```

プロンプト表示のある例:

```
cmgr> modify cluster fs6-8
Enter commands, you may enter "done" or "cancel" at any time to exit
```

```
Cluster Notify Cmd [optional] ?
Cluster Notify Address [optional] ?
Cluster HA mode <normal|experimental>[optional] ? (normal)

Current nodes in cluster  fs6-8:
Node - 1: fs6
Node - 2: fs7
Node - 3: fs8

Add nodes to or remove nodes from cluster fs6-8
Enter "done" when completed or "cancel" to abort

fs6-8 ? remove node fs6
fs6-8 ? remove node fs7
fs6-8 ? remove node fs8
fs6-8 ? done
Successfully modified cluster fs6-8

cmgr> delete cluster fs6-8

cmgr> show clusters

cmgr>
```

クラスタの表示

この節では、クラスタを表示する方法について説明します。

GUI を使ったクラスタの表示

GUI を使用すると、クラスタとそのコンポーネントを簡単に表示できます。**「表示(View)」**の選択肢から、クラスタ内の要素を選択してチェックできます。クラスタの詳細を表示するには、クラスタ名またはアイコンをクリックします。

すると、ステータスの詳細が右側のアイテム表示に現れます。

cmgr を使ったクラスタの表示

クラスタを定義したら、以下のコマンドを使用して、そのクラスタ内のノードを表示できます。

```
show cluster clustername
show clusters
```

例は、次のとおりです。

```
cmgr> show clusters

1 Cluster(s) defined
        nfs-cluster

cmgr> show cluster nfs-cluster
Cluster Name: nfs-cluster
Cluster Is FailSafe: true
Cluster Is CXFS: false
Cluster HA mode: normal

Cluster nfs-cluster has following 2 machine(s)
        hans2
        hans1
```

また、cluster_status コマンドを使用することもできます。

リソース・タイプ・タスク

リソース・タイプは、リソースの特定のクラスです。フェイルオーバーの目的では、特定のリソース・タイプのリソースはすべて同じ方法で処理できます。各リソースは、常に 1 つのリソース・タイプのインスタンスです。

この節では、以下のリソース・タイプ・タスクについて説明します。

- 「リソース・タイプの定義」
- 149 ページの「特定ノードへのリソース・タイプの再定義」
- 152 ページの「リソース・タイプの依存性の追加と削除」
- 155 ページの「リソース・タイプのロード」
- 155 ページの「リソース・タイプ定義の変更」
- 160 ページの「リソース・タイプの削除」
- 160 ページの「リソース・タイプの表示」

リソース・タイプの定義

この節では、リソース・タイプを定義する方法について説明します。

GUI を使ったリソース・タイプの定義

FailSafe ソフトウェアには、定義済みリソース・タイプが多数含まれています。クラスタ内のリソース・タイプは、`/usr/cluster/bin/cdb-create-resource-type` スクリプトを使用して、ノードにインストールされた FailSafe のプラグインに対して作成されます。クラスタの設定時に作成しなかったリソース・タイプは、155 ページの「GUI を使ったリソース・タイプのロード」で説明されているとおり、後から `resource type install` コマンドを使用して追加できます。

HA サービスにするアプリケーションに適したリソース・タイプがある場合は、それらの定義済みリソース・タイプを流用できます。適切なリソース・タイプがない場合は、追加のリソース・タイプを定義できます。リソース・タイプの定義についての詳細は、『IRIS FailSafe Version 2 Programmer's Guide』を参照してください。このマニュアルでは、その情報の概要が説明されています。

新しいリソース・タイプを定義するには、以下を実行します。

1. 「リソース・タイプ(Resource Type)」: 新しいリソース・タイプの名前を、最大 255 文字までで指定します。

   「次へ(Next)」をクリックして、次の画面に移動します。

2. 必須アクションの設定を指定します(時間の値はミリ秒単位です):

   - 「開始／停止順位(Start/Stop Order)」: その他のタイプのリソースに関連した、このタイプのリソースのアクション・スクリプトを実行する順序。
     - リソースは、この値が小さい順に開始されます。
     - リソースは、この値が大きい順に停止されます。

     可能な順序範囲についての詳細は、『IRIS FailSafe Version 2 Programmer's Guide』を参照してください。
   - 「開始タイムアウト(Start Timeout)」: このタイプのリソースを開始するまでの最大待機時間。
   - 「停止タイムアウト(Stop Timeout)」: このタイプのリソースを停止するまでの最大待機時間。
   - 「排他タイムアウト(Exclusive Timeout)」: このタイプのリソースがすでに実行されていないことを確認するまでの最大待機時間。
   - 「モニタ・タイムアウト(Monitoring Timeout)」: このタイプのリソースをモニタする最大待機時間。
   - 「モニタ周期(Monitor Interval)」: `monitor` アクション・スクリプトの実行間隔。これは、`monitor` アクション・スクリプトに対してのみ有効です。

- 「モニタ開始時刻(Monitoring Start Time)」: リソースの開始からリソースのモニタの開始までの時間。

  「次へ(Next)」をクリックして、次の画面に移動します。

3. 必要に応じて、オプションのアクションに対する設定を指定します。

   - 「再開を有効(Restart Enabled)」: リソースの再開を有効にするボックスをオンにします。モニタ異常が発生した後、このタイプのリソースを現在のノードで自動的に再開させる場合は、再開を有効にします。再開を有効にすると、アプリケーション・ダウンタイムの時間を短縮できます。

     たとえば、リソースのモニタ・アクションが失敗したことが FailSafe で検出されたとします。

     – 再開が無効になっている場合、FailSafe では、直ちにフェイルオーバー・ドメインの別のノードにグループ全体を移動しようと試みます。アプリケーションは、グループ全体がフェイルオーバーされるまでダウンしたままになります。

     – 再開が有効になっている場合、FailSafe では、リソース・グループの残りが実行されている現在のノードでリソースを再開しようと試みます。これが成功すると、リソース・グループは、リソースが再開されるとすぐに利用可能になります。失敗した場合にのみ、FailSafe では、フェイルオーバー・ドメインの別のノードにグループ全体を移動しようと試みます。

     ローカル再開フラグは、ローカル・フェイルオーバーを有効にします。

     – ローカル再開が有効になっていて、リソース・モニタ・スクリプトが失敗した場合は、SRMD によって、該当するリソースの再開スクリプトが実行されます。

     – 再開スクリプトが成功した場合、SRMD ではリソースのモニタを続行します。

     – 再開スクリプトが失敗した場合、または再開カウントがなくなった場合は、SRMD から FSD にリソース・グループ・モニタ・エラーが送信されます。FSD 自体は、ローカル・フェイルオーバーに関与しません。

     1 つのリソースが再開されても、リソース・グループ内のその他のリソースは再開されません。GUI または `cmgr` を使用してリソースのローカル再開を行うことはできません。

     リソース・タイプの再開カウンタをリセットする必要がある場合は、リソース・グループをメインテナンス・モードにして、メインテナンス・モードから削除できます。このプロセスにより、リソース・グループ内のすべてのリソースのカウンタが再開されます。リソース・グループをメインテナンス・モードにする方法についての詳細は、243 ページの「リソース・グループのモニタの中断と再開」を参照してください。

   - 「再開タイムアウト(Restart Timeout)」: モニタ異常が発生した後、リソースを再開するまでに待機する最大時間。

   - 「再開カウント(Restart Count)」: 現在のノードでこのタイプのリソースを再開しようと FailSafe が試みる最大回数。0 より大きい整数を入力してください。

- 「検査を有効(Probe Enabled)」: このタイプのリソースがノードで設定されていることを FailSafe で確認させる場合にオンにします。
- 「検査タイムアウト(Probe Timeout)」: このタイプのリソースがノードで設定されていることを確認しようと FailSafe が試みる最大時間。

4. タイプ固有属性の設定を変更します。リソース・タイプに固有の属性を指定してください。各属性に対しては以下を指定する必要があります。

   - 「属性キー(Attribute key)」: 属性の名前。
   - 「データ型(Data Type)」:「文字列(String)」または「整数(Integer)」のいずれかを選択します。
   - 「デフォルト値(Default Value)」: デフォルト値を指定します(オプション)。

   たとえば、NFS には以下の属性が必要です。

   - `export-point`。エクスポート・ディスク名を定義する値を取得します。この名前は、`exportfs`(1M) コマンドへの入力として使用されます。例は、次のとおりです。

     ```
     export-point = /this_disk
     ```

   - `export-info`。ファイルシステムのエクスポート・オプションを定義する値を取得します。これらのオプションは、`exportfs`(1M) コマンドで使用されます。例は、次のとおりです。

     ```
     export-info = rw,wsync,anon=root
     ```

   - `filesystem`。raw ファイルシステムを定義する値を取得します。この名前は、`mount`(1M) コマンドへの入力として使用されます。例は、次のとおりです。

     ```
     filesystem = /dev/xlv/xlv_object
     ```

   「追加(Add)」をクリックして属性を追加します。必要に応じて、その他の属性に対してもこの操作を繰返します。

5. 「OK」をクリックして、タスクを完了します。

cmgr を使ったリソース・タイプの定義

以下のコマンドを使用します。

```
define resource_type RT_name on node nodename [in cluster clustername]

define resource_type RT_name [in cluster clustername]
   set order to start/stop_order_number
   set restart_mode to 0|1
```

```
set restart_count to number_of_attempts
add action action_script_name
        set exec_time to exclusive_timeout
        set monitor_interval to monitor_interval
        set monitor_time to monitor_time
add type_attribute type-specific_attribute_name
        set data_type to string|integer
        set default_value to default
add dependency RT_name
remove action action_script_name
remove type_attribute type-specific_attribute_name
remove dependency dependency_name
```

上記のコマンド書式について、次の点に注意してください。

- resource_type は、定義するリソース・タイプの名前（最大 255 文字まで）です。
- order は、その他のタイプのリソースに関連した、このタイプのリソースのアクション・スクリプトを実行する順序です。
  - リソースは、この値が小さい順に開始されます。
  - リソースは、この値が大きい順に停止されます。

  可能な順序範囲についての詳細は、『IRIS FailSafe Version 2 Programmer's Guide』を参照してください。
- restart_mode は、次のとおりです。
  - 0 = モニタ異常発生時に再開しない（再開の無効化）
  - 1 = 固定数回再開する（再開の有効化）

  モニタ異常が発生した後、このタイプのリソースを現在のノードで自動的に再開させる場合は、再開を有効にします。再開を有効にすると、アプリケーション・ダウンタイムの時間を短縮できます。
- restart_count は、現在のノードでこのタイプのリソースを再開しようと FailSafe が試みる最大回数です。0 より大きい整数を入力してください。
- action は、アクション・スクリプトの名前（exclusive、start、stop 、monitor、または restart）です。詳細については、12 ページの「アクション・スクリプト」を参照してください。以下の時間の値は、ミリ秒単位です。
  - exec_time は、アクション・スクリプトを実行できる最大時間です。

  - monitor_interval は、monitor アクション・スクリプトの実行間隔です(これは、monitor アクション・スクリプトに対してのみ有効です)。
  - monitor_time は、リソースの開始からリソースのモニタの開始までの時間です。
- type_attribute は、タイプ固有属性です。
  - data_type は、string または integer のいずれかです。
  - default_value は、属性のデフォルト値です。
- dependency は、指定したリソース・タイプ (*RT_name*) に依存性を追加します。

デフォルトでは、リソース・タイプはクラスタ全体に適用されます。リソース・タイプを特定のノードに制限したい場合は、プロンプトが表示されたらノード名を入力します。再開モードを有効にする場合は、プロンプトが表示されたら 1 と入力します。

通常モードの例については、次のファイル内にある cmgr コマンドの create resource type テンプレートを参照してください。

/var/cluster/cmgr-templates/cmgr-create-resource_type

メモ: 以下のプロンプト・モードの例には、newresourcetype という新しいリソース・タイプの 2 つのアクション・スクリプト(start と stop)のプロンプトと応答だけを示します。

```
cmgr> define resource_type newresourcetype

(Enter "cancel" at any time to abort)

Node[optional]?
Order ? 300
Restart Mode ? (0)

DEFINE RESOURCE TYPE OPTIONS

        0) Modify Action Script.
        1) Add Action Script.
        2) Remove Action Script.
        3) Add Type Specific Attribute.
        4) Remove Type Specific Attribute.
        5) Add Dependency.
        6) Remove Dependency.
        7) Show Current Information.
```

```
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:1

No current resource type actions

Action name ? start
Executable timeout (in milliseconds) ? 40000

        0) Modify Action Script.
        1) Add Action Script.
        2) Remove Action Script.
        3) Add Type Specific Attribute.
        4) Remove Type Specific Attribute.
        5) Add Dependency.
        6) Remove Dependency.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:1

Current resource type actions:
        start

Action name stop
Executable timeout?  (in milliseconds) 40000

        0) Modify Action Script.
        1) Add Action Script.
        2) Remove Action Script.
        3) Add Type Specific Attribute.
        4) Remove Type Specific Attribute.
        5) Add Dependency.
        6) Remove Dependency.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:3
```

```
No current type specific attributes

Type Specific Attribute ? integer-att
Datatype ? integer
Default value[optional] ? 33

        0) Modify Action Script.
        1) Add Action Script.
        2) Remove Action Script.
        3) Add Type Specific Attribute.
        4) Remove Type Specific Attribute.
        5) Add Dependency.
        6) Remove Dependency.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:3

Current type specific attributes:
        Type Specific Attribute - 1: integer-att

Type Specific Attribute ? string-att
Datatype ? string
Default value[optional] ? rw

        0) Modify Action Script.
        1) Add Action Script.
        2) Remove Action Script.
        3) Add Type Specific Attribute.
        4) Remove Type Specific Attribute.
        5) Add Dependency.
        6) Remove Dependency.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:5

No current resource type dependencies

Dependency name ? filesystem
```

```
        0) Modify Action Script.
        1) Add Action Script.
        2) Remove Action Script.
        3) Add Type Specific Attribute.
        4) Remove Type Specific Attribute.
        5) Add Dependency.
        6) Remove Dependency.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:7

Current resource type actions:
        Action - 1: start
        Action - 2: stop

Current type specific attributes:
        Type Specific Attribute - 1: integer-att
        Type Specific Attribute - 2: string-att

No current resource type dependencies

Resource dependencies to be added:
        Resource dependency - 1: filesystem

        0) Modify Action Script.
        1) Add Action Script.
        2) Remove Action Script.
        3) Add Type Specific Attribute.
        4) Remove Type Specific Attribute.
        5) Add Dependency.
        6) Remove Dependency.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:9
Successfully defined resource_type newresourcetype

cmgr> show resource_types
```

```
template
MAC_address
newresourcetype
IP_address
filesystem
volume

cmgr> exit
#
```

特定ノードへのリソース・タイプの再定義

この節では、特定のノードに適用されるリソース・タイプを定義する方法について説明します。

GUI を使った特定ノードへのリソース・タイプの再定義

このタスクを使用して、クラスタ全体に対して定義された既存のリソース・タイプを取得し、ローカル・ノードで使用できるよう再定義できます。

特定のノードに対して再定義されているリソース・タイプは、同じ名前のクラスタ全体の定義をオーバーライドします。このため、個々のノードは、クラスタ全体のリソース・タイプ定義のグローバル設定をオーバーライドできます。この機能を使用して、特定のノードに異なるスクリプト・タイムアウトを指定したり、クラスタ内の 1 つのノードだけのリソースを再開できます。

たとえば、IP_address リソースでは、デフォルトでローカル再開が有効になっています。特定のノードにローカル再開のない IP_address タイプが必要な場合は、再開モード(0 に設定)以外はすべて同じパラメータを持つ IP_address クラスタ全体のリソース・タイプのコピーを作成できます。

以下を実行します。

1. 「クラスタ・ノード(Cluster Node)」: ローカル・ノードの名前がデフォルトで入力されます。
2. 「クラスタ全体に対するリソース・タイプ(Clusterwide Resource Type)」: ローカル・ノードに対して再定義するリソース・タイプの名前を選択します。

   「次へ(Next)」をクリックして、次の画面に移動します。
3. 必要に応じて、必須アクションの設定を変更します(時間の値はミリ秒単位です):
   - 「開始／停止順位(Start/Stop Order)」: その他のタイプのリソースに関連した、このタイプのリソースのアクション・スクリプトを実行する順序。
     - リソースは、この値が小さい順に開始されます。

– リソースは、この値が大きい順に停止されます。

可能な順序範囲についての詳細は、『IRIS FailSafe Version 2 Programmer's Guide』を参照してください。

- 「開始タイムアウト(Start Timeout)」: このタイプのリソースを開始するまでの最大待機時間。
- 「停止タイムアウト(Stop Timeout)」: このタイプのリソースを停止するまでの最大待機時間。
- 「排他タイムアウト(Exclusive Timeout)」: このタイプのリソースがすでに実行されていないことを確認するまでの最大待機時間。
- 「モニタ・タイムアウト(Monitoring Timeout)」: このタイプのリソースをモニタする最大待機時間。
- 「モニタ周期(Monitor Interval)」: `monitor` アクション・スクリプトの実行間隔。これは、`monitor` アクション・スクリプトに対してのみ有効です。
- 「モニタ開始時刻(Monitoring Start Time)」: リソースの開始からリソースのモニタの開始までの時間。

「次へ(Next)」をクリックして、次の画面に移動します。

4. 必要に応じて、オプションのアクションに対する設定を変更します。

- 「再開を有効(Restart Enabled)」: リソースの再開を有効にするボックスをオンにします。モニタ異常が発生した後、このタイプのリソースを現在のノードで自動的に再開させる場合は、再開を有効にします。再開を有効にすると、アプリケーション・ダウンタイムの時間を短縮できます。

  たとえば、リソースのモニタ・アクションが失敗したことが FailSafe で検出されたとします。

  – 再開が無効になっている場合、FailSafe では、直ちにフェイルオーバー・ドメインの別のノードにグループ全体を移動しようと試みます。アプリケーションは、グループ全体がフェイルオーバーされるまでダウンしたままになります。

  – 再開が有効になっている場合、FailSafe では、リソース・グループの残りが実行されている現在のノードでリソースを再開しようと試みます。これが成功すると、リソース・グループは、リソースが再開されるとすぐに利用可能になります。失敗した場合にのみ、FailSafe では、フェイルオーバー・ドメインの別のノードにグループ全体を移動しようと試みます。

  ローカル再開フラグは、ローカル・フェイルオーバーを有効にします。

  – ローカル再開が有効になっていて、リソース・モニタ・スクリプトが失敗した場合は、SRMD によって、該当するリソースの再開スクリプトが実行されます。

  – 再開スクリプトが成功した場合、SRMD ではリソースのモニタを続行します。

– 再開スクリプトが失敗した場合、または再開カウントがなくなった場合は、SRMD から FSD にリソース・グループ・モニタ・エラーが送信されます。FSD 自体は、ローカル・フェイルオーバーに関与しません。

1 つのリソースが再開されても、リソース・グループ内のその他のリソースは再開されません。GUI または `cmgr` を使用してリソースのローカル再開を行うことはできません。

リソース・タイプの再開カウンタをリセットする必要がある場合は、リソース・グループをメインテナンス・モードにして、メインテナンス・モードから削除できます。このプロセスにより、リソース・グループ内のすべてのリソースのカウンタが再開されます。リソース・グループをメインテナンス・モードにする方法についての詳細は、243 ページの「リソース・グループのモニタの中断と再開」を参照してください。

- 「再開タイムアウト(Restart Timeout)」: モニタ異常が発生した後、リソースを再開するまでに待機する最大時間。
- 「再開カウント(Restart Count)」: 現在のノードでこのタイプのリソースを再開しようと FailSafe が試みる最大回数。0 より大きい整数を入力してください。
- 「検査を有効(Probe Enabled)」: このタイプのリソースがノードで設定されていることを FailSafe で確認させる場合にオンにします。
- 「検査タイムアウト(Probe Timeout)」: このタイプのリソースがノードで設定されていることを確認しようと FailSafe が試みる最大時間。

5. タイプ固有属性の設定を変更します。リソース・タイプに固有の属性を指定してください。各属性に対しては以下を指定する必要があります。

   - 「属性キー(Attribute key)」: 属性の名前を指定します。
   - 「データ型(Data Type)」:「文字列(String)」または「整数(Integer)」のいずれかを選択します。
   - 「デフォルト値(Default Value)」: デフォルト値を指定します(オプション)。

   たとえば、NFS には以下の属性が必要です。

   - `export-point`。エクスポート・ディスク名を定義する値を取得します。この名前は、`exportfs`(1M) コマンドへの入力として使用されます。例は、次のとおりです。

     ```
     export-point = /this_disk
     ```

   - `export-info`。ファイルシステムのエクスポート・オプションを定義する値を取得します。これらのオプションは、`exportfs`(1M) コマンドで使用されます。例は、次のとおりです。

     ```
     export-info = rw,wsync,anon=root
     ```

- filesystem。raw ファイルシステムを定義する値を取得します。この名前は、mount(1M) コマンドへの入力として使用されます。例は、次のとおりです。

  ```
  filesystem = /dev/xlv/xlv_object
  ```

「追加(Add)」をクリックして属性を追加します。必要に応じて、その他の属性に対してもこの操作を繰返します。

6. 「OK」をクリックして、タスクを完了します。

cmgr を使ったノード固有リソース・タイプの定義

cmgr を使用する場合も、クラスタ全体のリソース・タイプを定義するときと類似した方法で、ノード固有リソース・タイプを再定義できます。ただし、コマンド・ラインでノードを指定する点が異なります。ノード固有リソース・タイプを定義するには、次のコマンドを使用します。

```
define resource_type RT_name on node nodename [in cluster clustername]
```

リソース・タイプの依存性の追加と削除

この節では、リソース・タイプに依存性を追加する方法について説明します。

GUI を使ったリソース・タイプの依存性の追加と削除

リソースと同様に、リソース・タイプも、1 つまたは複数のリソース・タイプに依存できます。そのような依存性が存在する場合、各依存リソース・タイプのインスタンスを少なくとも 1 つ定義してください。たとえば、Netscape_web というリソース・タイプに、IP_address および volume というリソース・タイプに対してリソース・タイプ依存性を持たせることができます。ws1 というリソースが Netscape_web リソース・タイプで定義されている場合は、ws1 を含むリソース・グループにも、タイプ IP_address のリソースとタイプ volume のリソースが少なくとも 1 つずつ含まれていなければなりません。図5-1 に、これらの依存性を示します。

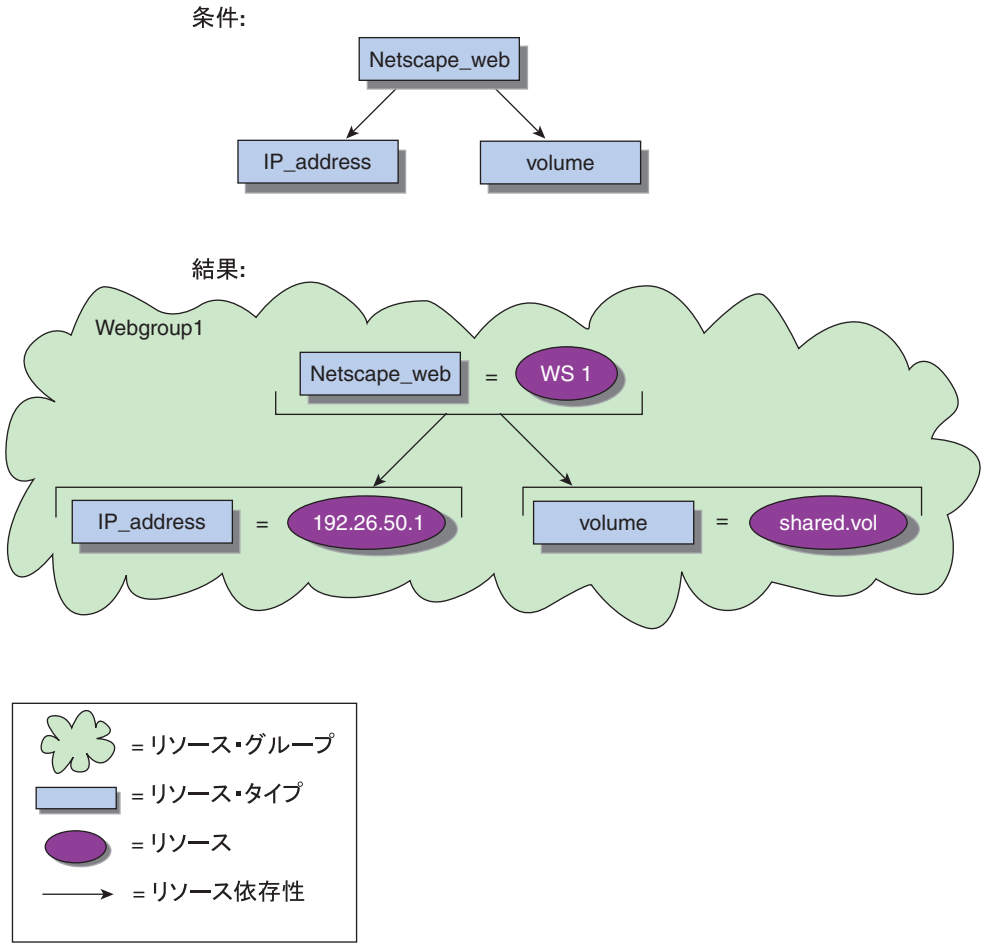


図5-1 依存性

次の情報を入力します。

1. 「リソース・タイプ(Resource Type)」: リソース・タイプを選択します。
2. 「依存性タイプ(Dependency Type)」: 依存性タイプを選択します。依存性をリストに追加するには「追加(Add)」をクリックし、依存性をリストから削除するには「削除(Delete)」をクリックします。
3. 「OK」をクリックして、タスクを完了します。

cmgr を使ったリソース・タイプの依存性の追加と削除

cmgr を使用している場合は、リソース・タイプを定義または変更する際に依存性を追加したり削除します。

たとえば、 nfs-cluster の NFS リソース・タイプが、filesystem リソース・タイプに対してリソース・タイプ依存性を持っているとします。filesystem ではなく IP_address リソース・タイプに対して依存性を持つよう NFS リソース・タイプを変更するには、以下を実行します。

```
cmgr> show resource_type NFS in cluster nfs-cluster

Name: NFS
Predefined: true
....

Resource type dependencies
        filesystem

cmgr> modify resource_type NFS in cluster nfs-cluster
Enter commands, when finished enter either "done" or "cancel"

resource_type NFS ? remove dependency filesystem
resource_type NFS ? add dependency IP_address
resource_type NFS ? done
Successfully modified resource_type NFS

cmgr> show resource_type NFS in cluster nfs-cluster

Name: NFS
Predefined: true
....

Resource type dependencies
        IP_address
```

リソース・タイプのロード

この節では、リソース・タイプをインストール(ロード)する方法について説明します。

GUI を使ったリソース・タイプのロード

クラスタを定義すると、使用可能なリソース・タイプ定義のセットが FailSafe によってインストールされます。これらの定義には、デフォルト値が含まれています。SGI 提供の標準リソース・タイプ定義をクラスタに追加インストールする必要がある場合や、標準リソース・タイプ定義を削除した後で再インストールする場合、そのリソース・タイプ定義をクラスタにロードできます。

ロードするリソース・タイプ定義は、クラスタに既存のものではいけません。

cmgr を使ったリソース・タイプのロード

クラスタにリソース・タイプをインストールするには、次のコマンドを使用します。

```
install resource_type RT_name [in cluster clustername]
```

リソース・タイプ定義の変更

この節では、リソース・タイプを変更する方法について説明します。

GUI を使ったリソース・タイプの変更

リソース・タイプの変更プロセスは、リソース・タイプの定義プロセスと類似しています。

以下を入力します(時間の値はミリ秒単位です)。

1. 「リソース・タイプ(Resource Type)」: 変更するリソース・タイプの名前を選択します。

   「次へ(Next)」をクリックして、次の画面に移動します。すると、各フィールドの現在の設定がデフォルトで入力されます。

2. 「開始/停止順位(Start/Stop Order)」: その他のタイプのリソースに関連した、このタイプのリソースのアクション・スクリプトを実行する順序。

   - リソースは、この値が小さい順に開始されます。
   - リソースは、この値が大きい順に停止されます。

     可能な順序範囲についての詳細は、『IRIS FailSafe Version 2 Programmer's Guide』を参照してください。

3. 「開始タイムアウト(Start Timeout)」: このタイプのリソースを開始するまでの最大待機時間。

4. 「停止タイムアウト(Stop Timeout)」: このタイプのリソースを停止するまでの最大待機時間。
5. 「排他タイムアウト(Exclusive Timeout)」: このタイプのリソースがすでに実行されていないことを確認するまでの最大待機時間。
6. 「モニタ・タイムアウト(Monitoring Timeout)」: このタイプのリソースをモニタする最大待機時間。
7. 「モニタ周期(Monitor Interval)」: `monitor` アクション・スクリプトの連続実行間。これは、`monitor` アクション・スクリプトに対してのみ有効です。
8. 「モニタ開始時刻(Monitoring Start Time)」: リソースの開始からリソースのモニタの開始までの時間。

   「次へ(Next)」をクリックして、次の画面に移動します。
9. 「再開を有効(Restart Enabled)」: リソースの再開を有効にするボックスをオンにします。モニタ異常が発生した後、このタイプのリソースを現在のノードで自動的に再開させる場合は、再開を有効にします。再開を有効にすると、アプリケーション・ダウンタイムの時間を短縮できます。

   たとえば、リソースのモニタ・アクションが失敗したことが FailSafe で検出されたとします。

   - 再開が無効になっている場合、FailSafe では、直ちにフェイルオーバー・ドメインの別のノードにグループ全体を移動しようと試みます。アプリケーションは、グループ全体がフェイルオーバーされるまでダウンしたままになります。
   - 再開が有効になっている場合、FailSafe では、リソース・グループの残りが実行されている現在のノードでリソースを再開しようと試みます。これが成功すると、リソース・グループは、リソースが再開されるとすぐに利用可能になります。失敗した場合にのみ、FailSafe では、フェイルオーバー・ドメインの別のノードにグループ全体を移動しようと試みます。

   ローカル再開フラグは、ローカル・フェイルオーバーを有効にします。

   - ローカル再開が有効になっていて、リソース・モニタ・スクリプトが失敗した場合は、SRMD によって、該当するリソースの再開スクリプトが実行されます。
   - 再開スクリプトが成功した場合、SRMD ではリソースのモニタを続行します。
   - 再開スクリプトが失敗した場合、または再開カウントがなくなった場合は、SRMD から FSD にリソース・グループ・モニタ・エラーが送信されます。FSD 自体は、ローカル・フェイルオーバーに関与しません。

   1 つのリソースが再開されても、リソース・グループ内のその他のリソースは再開されません。GUI または `cmgr` を使用してリソースのローカル再開を行うことはできません。

   リソース・タイプの再開カウンタをリセットする必要がある場合は、リソース・グループをメインテナンス・モードにして、メインテナンス・モードから削除できます。このプロセスにより、リソース・グループ内のすべてのリソースのカウンタが再開されます。リソース・グループをメインテナンス・モードにする方法についての詳細は、243 ページの「リソース・グループのモニタの中断と再開」を参照してください。

10. 「再開タイムアウト(Restart Timeout)」: モニタ異常が発生した後、リソースを再開するまでに待機する最大時間。

11. 「再開カウント(Restart Count)」: 現在のノードでこのタイプのリソースを再開しようと FailSafe が試みる最大回数。0 より大きい整数を入力してください。

12. 「検査を有効(Probe Enabled)」: このタイプのリソースがノードで設定されていることを FailSafe で確認させる場合にオンにします。

13. 「検査タイムアウト(Probe Timeout)」: このタイプのリソースがノードで設定されていることを確認しようと FailSafe が試みる最大時間。

14. 「タイプ固有属性(Type-Specific Attributes)」: リソース・タイプに固有の新しい属性を指定するか、既存の属性をその名前を選択することによって変更します。各属性に対しては以下を指定する必要があります。

    - 「属性キー(Attribute key)」: 属性の名前を指定します。
    - 「データ型(Data Type)」:「文字列(String)」または「整数(Integer)」のいずれかを選択します。
    - 「デフォルト値(Default Value)」: 属性のデフォルト値を指定します(オプション)。

    メモ: タイプ固有属性は、そのタイプの既存のリソースがある場合、変更できません。

    「追加(Add)」をクリックして属性を追加するか、「変更(Modify)」をクリックして属性を変更し、必要に応じて、その他の属性に対してもこれらの操作を繰返します。「OK」をクリックして、定義を完了します。

cmgr を使ったリソース・タイプの変更

リソース・タイプを変更するには、以下のコマンドを使用します。

```
modify resource_type RT_name [in cluster clustername]
   set order to start/stop_order_number
   set restart_mode to 0|1
   set restart_count to number_of_attempts
   add action action_script_name
           set exec_time to exclusive_timeout
           set monitor_interval to monitor_interval
           set monitor_time to monitor_time
   modify action action_script_name
           set exec_time to exclusive_timeout
           set monitor_interval to monitor_interval
           set monitor_time to monitor_time
```

```
add type_attribute type-specific_attribute_name
        set data_type to string|integer
        set default_value to default
add dependency RT_name
remove action action_script_name
remove type_attribute type-specific_attribute_name
remove dependency dependency_name
```

リソース・タイプの変更は、定義に使用するコマンドと同じコマンドを使用して行います。161 ページの「新規リソースの定義」を参照してください。

リソース・タイプ・タイムアウトの現在の値を表示して、どのアクション・タイムアウトでも変更することができます。

以下に、statdリソース・タイプのモニタ実行可能タイムアウトを 40 秒から 60 秒に増やす方法の例を示します。

```
#cmgr> modify resource_type statd in cluster test-cluster
Enter commands, when finished enter either "done" or "cancel"

resource_type statd ? modify action monitor
Enter action parameters, when finished enter "done" or "cancel"

Current action monitor parameters:
        exec_time : 40000ms
        monitor_interval : 20000ms
        monitor_time : 50000ms

Action - monitor ? set exec_time to 60000
Action - monitor ? done
resource_type statd ? done
Successfully modified resource_type statd
```

以下に、プロンプト・モードでリソース・タイプ・タイムアウトを変更する方法の例を示します。

```
#cmgr> modify resource_type statd in cluster test-cluster


(Enter "cancel" at any time to abort)

Node[optional] ?
Order ? (411)
Restart Mode ? (0)
```

```
MODIFY RESOURCE TYPE OPTIONS

        0) Modify Action Script.
        1) Add Action Script.
        2) Remove Action Script.
        3) Add Type Specific Attribute.
        4) Remove Type Specific Attribute.
        5) Add Dependency.
        6) Remove Dependency.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:0
Current resource type actions:
        stop
        exclusive
        start
        restart
        monitor

Action name ? monitor
Executable timeout (in milliseconds) ? (40000ms) 60000
Monitoring Interval (in milliseconds) ? (20000ms)
Start Monitoring Time (in milliseconds) ? (50000ms)

        0) Modify Action Script.
        1) Add Action Script.
        2) Remove Action Script.
        3) Add Type Specific Attribute.
        4) Remove Type Specific Attribute.
        5) Add Dependency.
        6) Remove Dependency.
        7) Show Current Information.
        8) Cancel. (Aborts command)
        9) Done. (Exits and runs command)

Enter option:9
Successfully modified resource_type statd
```

リソース・タイプの削除

この節では、リソース・タイプを削除する方法について説明します。

GUI を使ったリソース・タイプの削除

GUI を使用してリソース・タイプを削除するには、以下を入力します。

1. 「削除するリソース・タイプ(Resource Type to Delete)」: 削除するリソース・タイプの名前を選択します。

   メモ: ローカル・ノードに対して再定義されているリソース・タイプを選択した場合、そのリソース・タイプの特定の定義が削除され、クラスタ全体のリソース・タイプが代わりに使用されます。クラスタ全体のリソース・タイプは、そのタイプのリソースがある場合、削除できません。

2. 「OK」をクリックして、タスクを完了します。

cmgr を使ったリソース・タイプの削除

リソース・タイプを削除するには、次のコマンドを使用します。

```
delete resource_type RT_name [in cluster clustername]
```

リソース・タイプの表示

この節では、リソース・タイプを表示する方法について説明します。

GUI を使ったリソース・タイプの表示

「表示(View)」->「リソース・タイプ(Resource Types)」を選択します。その後、ツリー表示で任意のリソース・タイプ・アイコンをクリックして、そのリソース・タイプのパラメータをチェックします。

cmgr を使ったリソース・タイプの表示

リソース・タイプを表示するには、以下のコマンドを使用します。

1. 定義済みリソース・タイプ (*RT_name*) のパラメータを表示する場合:

   ```
   show resource_type RT_name [in cluster clustername]
   ```

2. 定義済みリソース・タイプをすべて表示する場合:

   ```
   show resource_types [in cluster clustername]
   ```

3. インストールされている定義済みリソース・タイプをすべて表示する場合:

```
show resource_types installed
```

リソース・タスク

リソースは、クライアントまたはその他のリソースにサービスを提供する単一の物理的または論理的な実体です。通常、リソースはクラスタ内の複数のノードで使用できますが、特定の時点でリソースを制御できるノードは 1 つです。たとえば、単一のディスク・ボリューム、特定のネットワーク・アドレス、Web ノードなどのアプリケーションがリソースである場合があります。

この節では、以下のリソース・タスクについて説明します。

新規リソースの定義

この節では、新しいリソースを定義する方法について説明します。

GUI を使った新規リソースの定義

リソースは、リソース・タイプおよびリソース名によって識別されます。リソース名は、リソース・タイプの特定のインスタンスを識別します。フェイルオーバーの目的では、特定のリソース・タイプのリソースはすべて同じ方法で処理できます。

デフォルトでは、リソースはクラスタ全体に適用されます。ただし、ローカル・ノードだけにリソースを適用させるために「特定ノードへのリソースの再定義(Redefine a Resource for a Specific Node)」タスクを使用することができます(167 ページの「特定ノードへのリソースの再定義」を参照してください)。

以下に対して適切な値を指定します。

1. 「リソース・タイプ(Resource Type)」: リソース・タイプの名前。

   リソース・タイプは、特定の論理ノードに対して定義したり、クラスタ全体に対して定義できます。ノードに対して定義されるリソース・タイプは、同じ名前のクラスタ全体のリソース・タイプ定義をオーバー

ライドします。これにより、個々のノードは、クラスタ全体のリソース・タイプ定義のグローバル設定をオーバーライドできます。

定義するリソースのタイプ。FailSafe システムには、GUI にリストされている定義済みリソース・タイプが含まれています（293 ページの「ソフトウェア階層」も参照してください）。また、独自のリソース・タイプを定義することもできます。

FailSafe ソフトウェアには、定義済みリソース・タイプが多数含まれています。高可用性サービスにするアプリケーションに適したリソース・タイプがある場合は、それらのタイプを流用できます。適切なリソース・タイプがない場合は、追加のリソース・タイプを定義できます。

2. 「リソース(Resource)」: 定義するリソースの名前（最大 255 文字まで）で、アンダースコアで始めることはできません。リソースは、クライアントまたはその他のリソースにサービスを提供する単一の物理的または論理的な実体です。例として、単一のディスク・ボリューム、特定のネットワーク高可用性 (HA) IP アドレス、Web サーバのような特定のアプリケーションが挙げられます。特定のリソース・タイプには、別の命名要件がある場合があります。以下の節を参照してください。

   FailSafe 設定では、リソースを 100 個まで定義できます。

   「次へ(Next)」をクリックして、次の画面に移動します。

3. 「タイプ固有属性(Type-specific attributes)」: このリソースに適用する属性を入力します。以下の節では、base FailSafe リリースに付属している各リソース・タイプの属性について説明します。その他の属性は FailSafe Plug-in リリースに付属しており、説明は、これらのリリースに付属のドキュメントに記載されています。作成した新しいリソース・タイプに対しては、属性を指定できます。

4. 「OK」をクリックして、タスクを完了します。

CXFS 属性

CXFS リソースは、/shared_CXFS などの、CXFS ファイルシステムのマウント・ポイントです。relocate-mds フィールドは、CXFS ファイルシステムのメタデータ・サーバを再設定する (true) かしない (false) かを指定します。

filesystem 属性

filesystem リソースは、XFS ファイルシステムでなければなりません。

高可用性でなければならない XFS ファイルシステムは、filesystem リソースとして設定する必要があります。filesystem リソースとして使用するすべての XFS ファイルシステムは、共有ディスクの XLV ボリュームで作成してください。

filesystem リソースを定義する場合、リソースの名前は、ファイルシステムのマウント・ポイントにする必要があります。たとえば、XLV ボリューム xlv_vol に作成されて /shared1 ディレクトリにマウントされている XFS ファイルシステムのリソース名は、/shared1 になります。

ファイルシステムを定義する際、以下のパラメータを指定してください。

- volume-name: ファイルシステムに関連付けられている XLV ボリュームの名前。たとえば、XLV ボリューム `xlv_vol` に作成されたファイルシステムの場合は、ボリューム名属性も `xlv_vol` になります。
- mount-options: ファイルシステムをマウントするために使用されるマウント・オプション。これらは、`mount`(1M) コマンドの `-o` オプションに渡されなければならないマウント・オプションです。使用可能なオプションのリストは、`fstab`(4) で提供されています。
- monitoring-level: ファイルシステムに使用されるモニタリング・レベル。`mtab`(4) のマン・ページに説明されているとおり、モニタリング・レベル 1 は、ファイルシステムが `/etc/mtab` に存在するかどうかをチェックするよう指定します。モニタリング・レベル 2 は、`stat`(1M) コマンドを使用してファイルシステムがマウントされているかどうかをチェックするよう指定します。モニタリング・レベル 2 は、時間内に完了する場合は、レベル 1 よりも厳しく信頼性の高いチェックです。ロードされているシステムの中には、このレベルで問題があることが知られているものもあります。

`IP_address` 属性

`IP_address` リソースは、リソース・グループ内の HA サービスにアクセスするためにクライアントが使用する IP アドレスです。これらの HA IP アドレスは、異常が検出されると、リソース・グループのその他のリソースと共に、あるノードから別のノードに移動されます。

`IP_address` リソースのリソース名は、ドット (.) 表記法で指定します。名前解決が必要な IP 名は使用しないでください。たとえば、`192.26.50.1` は、`IP_address` リソース・タイプの有効なリソース名です。

FailSafe リソースとして定義する HA IP アドレスは、ノード・ホスト名の IP アドレスまたはノードのコントロール・ネットワークの IP アドレスと同じにすることはできません。

`IP_address` リソースを定義するときに、以下のパラメータをオプションとして指定できます。これらのいずれかのパラメータを指定する場合は、すべてのパラメータを指定する必要があります。

- NetworkMask: HA IP アドレスの ネットワーク・マスク。
- interfaces: HA IP アドレスを設定できるインタフェースのコンマで区切ったリスト。この順序付きリストは、この HA IP アドレスが割当てられる可能性のあるすべてのノードの全インタフェースのスーパーセットです。これらのインタフェースが同じノードにある場合は、HA IP アドレスのローカル再開を設定するために複数のインタフェースを指定できます。

  インタフェースのリストの順序によって、ノードのローカル再開に使用される HA IP アドレスを決定する優先順序が決まります。
- BroadcastAddress: HA IP アドレスのブロードキャスト・アドレス。

MAC_address 属性

MAC アドレスは、ネットワーク・インタフェースのリンク・レベル・アドレスです。MAC アドレスをフェイルオーバーする場合は、専用のネットワーク・インタフェースが必要です。

MAC アドレスのリソース名は、インタフェースの MAC アドレスです。MAC アドレスは、ha_macconfig2(1M) コマンドを使用して取得できます。

再 MAC が必要なインターフェースである interface-name という属性を指定する必要があります。

再 MAC プロセスを実行できるのは Ethernet インタフェースだけです。

volume 属性

volume リソース・タイプは、リソース・グループのリソースによって使用される XLV ボリュームです。

volume リソースを定義する場合、リソース名は XLV ボリュームの名前にする必要があります。XLV デバイス・ファイル名は、リソース名として指定しないでください。たとえば、ボリュームのリソース名は xlv_vol にすることができますが、/dev/xlv/xlv_vol や /dev/dsk/xlv/xlv_vol にすることはできません。

XLV ボリュームがノード上で構成されると、 /dev/xlv にファイルが作成されます。FailSafe クラスタにボリューム・リソースを設定した場合でも、フェイルオーバーが起こらないかぎり、そのボリュームを表示できるノードは一度に 1 つだけです。

フェイルオーバーの後は、2 つの異なるノードの /dev/xlv のボリューム名を表示することができます。これは、XLV ボリュームがシャットダウンすると、ファイル名がそのディレクトリから削除されるためです。したがって、複数のノードで、そのディレクトリにボリューム・ファイル名が存在する可能性があります。ただし、一度に 1 つのノードでしか、ボリュームはまとめられません。ボリュームがまとめられているマシンを確認するには、 xlv_mgr(1M) を使用してください。

ボリュームを定義する際、以下のパラメータをオプションとして指定できます。

- devname-group: XLV デバイス・ファイルのグループ名。XLV デバイス・ファイルのデフォルトのグループ名は、sys グループです。
- devname-owner: XLV デバイス・ファイルのオーナーのユーザ名 (ログイン名)。XLV デバイス・ファイルのデフォルトのオーナーは、root です。
- devname-mode: 8 進数の表記法で指定される、デバイス・ファイルのアクセス権。XLV デバイス・ファイルのアクセス権のデフォルト値は、600 モードです。

cmgr を使った新規リソースの定義

クラスタ全体のリソースを定義するには、次のコマンドを使用します。

```
define resource resourcename [of resource_type RT_name] [in cluster clustername]
    set key to attribute_value
    add dependency dependencyname of type RT_name
    remove dependency dependencyname of type RT_name
```

上記のコマンド書式について、次の点に注意してください。

- `set key` は属性の名前を指定し、*attribute_value* はその値を設定します。
- `add dependency` は、指定したリソース・タイプ (*RT_name*) の依存性を追加します。
- `remove dependency` は、指定したリソース・タイプの依存性を削除します。

このコマンドを使用してリソースを定義する際、ノードに固有ではないクラスタ全体のリソースを定義します。

`set key to attribute_value` の正しい値は、161 ページの「新規リソースの定義」で説明されているように、定義するリソースのタイプによって異なります。リソース属性を定義するための形式を判断する方法についての詳細は、165 ページの「cmgr を使ったリソース属性の指定」を参照してください。

リソースとその依存性を定義し終わったら、done と入力して、cmgr プロンプトに戻ります。

例は、次のとおりです。

```
cmgr> define resource /hafs1/nfs/statmon of resource_type statd_unlimited in cluster nfs-cluster
resource /hafs1/nfs/statmon? set ExportPoint to /hafs1/subdir
resource /hafs1/nfs/statmon? done
```

cmgr スクリプトの次のセクションにより、リソース・タイプ statd_unlimited のリソースが定義されます。

```
define resource /hafs1/nfs/statmon of resource_type statd_unlimited in cluster nfs-cluster
        set ExportPoint to /hafs1/subdir
done
```

cmgr を使ったリソース属性の指定

特定のリソース・タイプに対して設定が必要なユーザ固有属性を指定できる形式を確認するには、次のコマンドを入力して、そのリソース・タイプの完全な定義を表示できます。

```
show resource_type RT_name [in cluster clustername]
```

たとえば、リソース・タイプ volumes のリソースに対して定義する属性を表示するには、次のコマンドを入力します。

```
cmgr> show resource_type volume in cluster test-cluster
```

結果画面の下部に、次の情報が表示されます。

```
...
Type specific attribute: devname-group
        Data type: string
        Default value: sys
Type specific attribute: devname-owner
        Data type: string
        Default value: root
Type specific attribute: devname-mode
        Data type: string
        Default value: 600
...
```

この画面には、ボリュームのグループ ID、デバイス・オーナー、およびデバイス・ファイルのアクセス権を指定できる形式が反映されます。

- devname-group は、XLV デバイス・ファイルのグループ ID を指定します。
- devname_owner は、XLV デバイス・ファイルのオーナーを指定します。
- devname_mode は、デバイス・ファイルのアクセス権を指定します。

たとえば、リソース名 A に対してグループ ID を sys に設定するには、次のコマンドを入力します。

```
resource A? set devname-group to sys
```

表5-2 では、定義済みの FailSafe リソース・タイプに対し set *key* to *attribute_value* コマンドを使用して指定する属性の概要を説明します。

表5-2 リソース・タイプ属性

リソース・タイプ	属性	説明
CXFS	relocate-mds	CXFS ファイルシステムのメタデータ・サーバを再設定するかどうかを指定します。CXFS リソースの名前は、CXFS ファイルシステムのマウント・ポイントです（例: /shared_CXFS）。
filesystem	volume-name	ファイルシステムに関連付けられている xlv ボリュームの名前を指定します。
	mount-options	ファイルシステムをマウントするために使用されるマウント・オプションを指定します。
IP_address	NetworkMask	IP アドレスのサブネット・マスクを指定します。
	interfaces	IP アドレスを設定できるインタフェースのコンマで区切ったリストを指定します。

リソース・タイプ	属性	説明
	BroadcastAddress	IP アドレスのブロードキャスト・アドレスを指定します。
MAC_address	interface-name	再 MAC が必要なインタフェースの名前を指定します。
volume	devname-group	xlv デバイス・ファイルのグループ ID を指定します。
	devname_owner	xlv デバイスのオーナーを指定します。
	devname_mode	デバイス・ファイルのアクセス権を指定します。

特定ノードへのリソースの再定義

この節では、特定のノードへのリソースの再定義について説明します。

GUI を使った特定ノードへのリソースの再定義

特定のノードに対する既存のリソースは、そのノード(ローカル・ノード)からのみ再定義できます。再定義できるのは、クラスタ全体に対して定義された既存のリソースだけです。

この機能は、IP_address リソースに対して異種クラスタを設定するときに使用したい場合があります。たとえば、あるサーバの Gigabit Ethernet インタフェース eg0 と、別のサーバの 100baseT インタフェース ef0 に、タイプ IP_address のリソース 192.26.50.2 を設定できます。192.26.50.2 のクラスタ全体のリソース定義では、 interfaces フィールドが ef0 に設定され、最初のノードのノード固有定義では、 interfaces フィールドに eg0 が設定されます。

以下に対して適切な値を指定します。

1. 「クラスタ・ノード(Cluster Node)」: GUI が現在実行されているノードの名前。この名前はデフォルトで入力されます。このノードに対するリソースのみ再定義できます。別のノードに対してリソースを再定義するには、そのノードで GUI を呼出す必要があります。

注意: 変更は、特定の時点で GUI の 1 つのインスタンスだけから行ってください。2 つ目の GUI インスタンス(fstask の 2 つ目の呼出し)から変更を行うと、最初のインスタンスで加えた変更が上書きされる場合があります。これは、異なる GUI インスタンスが異なるときに独立して更新されるためです。時間が経てば、独立した GUI インスタンスによって同じ情報が提供されるようになります。ただし、「ファイル(File)」メニューからアクセスした複数のウィンドウは、すべて単一の GUI インスタンスの一部なので、これらのどのウィンドウからでも変更を加えることができます。

2. 「リソース・タイプ(Resource Type)」: リソース・タイプを選択します。

3. 「クラスタ全体に対するリソース(Clusterwide Resource)」: このノードに対して再定義するリソースの名前。

   「次へ(Next)」をクリックして、次の画面に移動します。

4. 「タイプ固有属性(Type-specific attributes)」: 必要に応じて、各属性の情報を変更します。各属性についての詳細は、161 ページの「GUI を使った新規リソースの定義」を参照してください。

5. 「OK」をクリックして、タスクを完了します。

cmgr を使った特定ノードへのリソースの再定義

cmgr を使用する場合も、クラスタ全体のリソースを定義するときと同じ方法で、クラスタ全体のリソースがノード固有になるように再定義できます。ただし、define resource コマンドでノードを指定する点が異なります。

ノード固有リソースを定義するには、次のコマンドを使用します。

```
define resource resourcename of resource_type RT_name on node nodename
[in cluster clustername]
```

デフォルトのクラスタが指定されている場合は、このコマンドでクラスタを指定する必要はなく、cmgr によってデフォルトが使用されます。

リソース定義の依存性の追加と削除

この節では、リソースの依存性を追加および削除する方法について説明します。

GUI を使ったリソース定義の依存性の追加と削除

1 つのリソースは、その他の 1 つまたは複数のリソースに依存できます。その場合、リソースは、依存リソースも開始しないかぎり開始できません(つまり、使用可能にできません)。依存リソースは、同じリソース・グループの一部でなければなりません。

リソースを定義する際、ほかのリソースに依存するリソースを定義できます。たとえば、Web サーバは、HA IP アドレスとファイルシステムの両方に依存できます。そして、ファイルシステムはボリュームに依存できます。図5-2 に、これを示します。

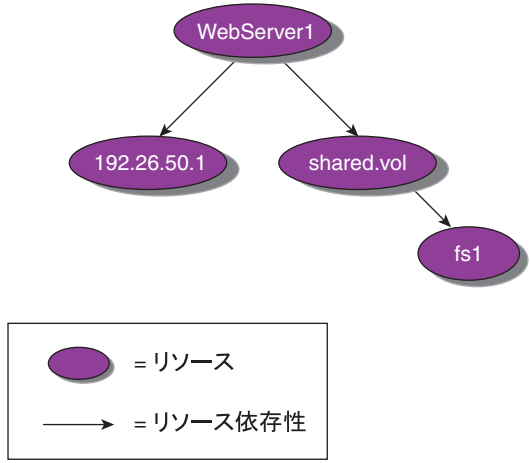


図5-2 リソース依存性の例

リソースは、互いに依存させることはできません。たとえば、リソース A がリソース B に依存している場合、リソース B をリソース A に依存させることはできません。さらに、循環依存性を定義することもできません。たとえば、リソース A がリソース B に依存しており、リソース B がリソース C に依存している場合、リソース C をリソース A に依存させることはできません。図5-3 に、これを示します。

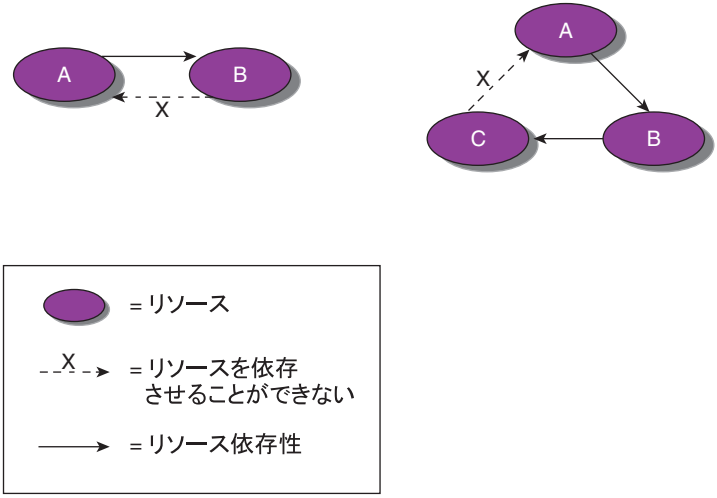


図5-3 リソースは互いに依存させることはできない

以下に対して適切な値を指定します。

1. 「リソース・タイプ(Resource Type)」: リソース・タイプの名前を選択します。
2. 「リソース(Resource)」: リソース名を選択します。
3. 「依存性タイプ(Dependency Type)」: 依存性リストに対して追加または削除するリソース・タイプを選択します。
4. 「依存性名(Dependency Name)」: 依存性リストに対して追加または削除するリソース名を選択します。表示されているタイプと名前をリストに追加するには、「追加(Add)」をクリックします。
5. 「OK」をクリックして、タスクを完了します。

cmgr を使ったリソース定義の依存性の追加と削除

リソース定義の依存性を追加または削除するには、 `modify resource` コマンドを使用します。例は、次のとおりです。

```
cmgr> modify resource /hafs1/expdir of resource_type NFS in cluster nfs-cluster
Enter commands, when finished enter either "done" or "cancel"

Type specific attributes to modify with set command:
Type Specific Attribute - 1: export-info
Type Specific Attribute - 2: filesystem   Resource type dependencies to add or remove:

Resource dependency - 1: /hafs1         type: filesystem


resource /hafs1/expdir ? add dependency 100.102.10.101 of type IP_address
resource /hafs1/expdir ? done
Successfully modified resource /hafs1/expdir
```

リソース定義の変更

この節では、リソース定義を変更する方法について説明します。

GUI を使ったリソース定義の変更

変更できるのは、リソースのタイプ固有属性だけです。リソースを定義した後に、リソースの名前を変更することはできません。リソースの名前を変更するには、そのリソースを削除した後、新しいリソースを定義しなければなりません。

以下に対して適切な値を指定します。

1. 「リソース・タイプ(Resource Type)」: リソース・タイプの名前を選択します。
2. 「リソース(Resource)」: 変更するリソースの名前を選択します。

   「次へ(Next)」をクリックして、次の画面に移動します。
3. 「タイプ固有属性(Type-specific attributes)」: 必要に応じて、属性を変更します。属性についての詳細は、161 ページの「GUI を使った新規リソースの定義」を参照してください。
4. 「OK」をクリックして、タスクを完了します。

メモ:リソース属性の中には、そのリソースが含まれるリソース・グループを再びオンラインにするまで変更が反映されないものがあります。たとえば、タイプ NFS のリソースのエクスポート・オプションを変更した場合、そのリソースをオンラインにするまで変更は反映されません。

---

cmgr を使ったリソース定義の変更

リソースを変更するには、以下のコマンドを使用します。

```
modify resource resourcename [of resource_type RT_name] on node
nodename [in cluster clustername]

modify resource resourcename [of resource_type RT_name] [in cluster clustername]
    set key to attribute_value
    add dependency dependencyname of type typename
    remove dependency dependencyname of type typename
```

リソースの変更は、定義に使用したコマンドと同じコマンドを使用して行います。161 ページの「新規リソースの定義」を参照してください。

リソースの削除

この節では、リソースを削除する方法について説明します。

GUI を使ったリソースの削除

リソースは、リソース・グループの一部である場合、削除できない可能性があります。185 ページの「リソース・グループ定義の変更」を参照してください。

リソースを削除するには、以下を入力します。

1. 「リソース・タイプ(Resource Type)」: リソース・タイプを選択します。

2. 「削除するリソース(Resource to Delete)」: 削除するリソースの名前を選択します。
3. 「OK」をクリックして、タスクを完了します。

GUI が接続されているノードに再定義されたリソースを選択した場合は、削除操作により、再定義されたリソース定義が削除され、クラスタ全体のリソース定義にも影響を与えます。

クラスタ全体のリソース定義を選択した場合は、削除操作によりこの定義が削除され、リソース・グループで使用できなくなります。クラスタ全体のリソース定義の削除は、そのリソースがリソース・グループの一部である場合、失敗します。

cmgr を使ったリソースの削除

リソース定義を削除するには、次のコマンドを使用します。

```
delete resource resourcename of resource_type RT_type [in cluster clustername]
```

リソースの表示

以下を表示することができます。

- 特定の定義済みリソースの属性
- 指定したリソース・グループのすべての定義済みリソース
- 指定したリソース・タイプのすべての定義済みリソース

注意: クラスタ・データベースを変更できるのは、root だけです。ただし、データベース情報の表示は、どのユーザでも GUI を使用して実行できます。このため、クラスタ・データベースには機密情報を含めないようにすることが推奨されます。

GUI を使ったリソースの表示

GUI を使用すると、リソースをツリー表示で簡単に表示できます。定義済みリソースをすべて表示するには、「表示(View)」->「グループ内のリソース(Resources in Groups)」を選択します。これらのリソースのステータスは、アイコンに表示されます (グレーはオフラインを示します)。また、「表示(View)」->「タイプごとのリソース(Resources by Type)」または「表示(View)」->「ノードが所有するリソース(Resources owned by Nodes)」を選択することもできます。

cmgr を使ったリソースの表示

リソースを表示するには、以下のコマンドを使用します。

- 単一のリソースのパラメータを表示する場合:

  ```
  show resource resourcename of resource_type RT_name
  ```

- リソース・グループの定義済みリソースをすべて表示する場合:

  ```
  show resources in resource_group RG_name [in cluster clustername]
  ```

- 指定したクラスタ内の特定のリソース・タイプの定義済みリソースをすべて表示する場合:

  ```
  show resources of resource_type RT_name [in cluster clustername]
  ```

フェイルオーバー・ポリシー・タスク

フェイルオーバー・ポリシーは、フェイルオーバー先のノードを決定するときに FailSafe で使用される方法です。フェイルオーバー・ポリシーは以下で構成されます。

- フェイルオーバー・ドメイン
- フェイルオーバー属性
- フェイルオーバー・スクリプト

FailSafe では、フェイルオーバー・スクリプトからのフェイルオーバー・ドメイン出力をフェイルオーバー属性と共に使用して、リソース・グループを存在させるノードを決定します。

管理者は、各リソース・グループに対してフェイルオーバー・ポリシーを設定しなければなりません。フェイルオーバー・ポリシーの名前は、プール内で一意でなければなりません。

この節では、以下のフェイルオーバー・ポリシー・タスクについて説明します。

- 「フェイルオーバー・ポリシーの定義」
- 179 ページの「フェイルオーバー・ポリシー定義の変更」
- 182 ページの「フェイルオーバー・ポリシーの削除」
- 183 ページの「フェイルオーバー・ポリシーの表示」

フェイルオーバー・ポリシーの定義

この節では、フェイルオーバー・ポリシーを定義する方法について説明します。

GUI を使ったフェイルオーバー・ポリシーの定義

リソースをリソース・グループに設定する前に、そのリソース・グループに適用するフェイルオーバー・ポリシーを決定しなければなりません。フェイルオーバー・ポリシーを定義するには、次の情報を指定します。

1. **「フェイルオーバー・ポリシー(Failover Policy)」:** フェイルオーバー・ポリシーの名前を、最大 63 文字までで入力します。この名前はプール内で一意でなければなりません。

2. **「スクリプト(Script)」:** 既存のフェイルオーバー・スクリプトの名前を選択します。このスクリプトは、ランタイム・フェイルオーバー・ドメインを生成して、それを FailSafe プロセスに返します。FailSafe プロセスにより、フェイルオーバー属性が適用された後、返されたフェイルオーバー・ドメインの中で現在の FailSafe メンバーシップにも存在する最初のノードが選択されます。

   - `ordered` は、FailSafe リリースに付属しています。`ordered` スクリプトによって初期フェイルオーバー・ドメインが変更されることはありません。このスクリプトを使用する場合、初期フェイルオーバー・ドメインとランタイム・フェイルオーバー・ドメインは同じものになります。

   - `round-robin` も、FailSafe リリースに付属しています。`round-robin` スクリプトでは、ラウンドロビン(循環)方式でリソース・グループ・オーナーが選択されます。このポリシーは、クラスタ内のすべてのノードで実行できるリソース・グループに対して使用できます。

   フェイルオーバー・スクリプトは、`/var/clusters/ha/policies` ディレクトリに保存されます。`ordered` スクリプトがニーズを満たさない場合は、新しいフェイルオーバー・スクリプトを定義して、`/var/clusters/ha/policies` ディレクトリに配置できます。FailSafe GUI を使用している場合、GUI によってスクリプトが自動的に検出され、使用できる選択肢として表示されます。必要なリソース・グループに対して新しいフェイルオーバー・スクリプトを使用するために、クラスタ・データベースを設定できます。フェイルオーバー・スクリプトの定義についての詳細は、『IRIS FailSafe Version 2 Programmer's Guide』を参照してください。

3. **「フェイルバック (Failback)」:** フェイルオーバー属性の名前を選択します。

   フェイルオーバー属性はフェイルオーバー・スクリプトに渡される値で、FailSafe によって、特定のリソース・グループに使用されるランタイム・フェイルオーバー・ドメインを変更する目的で使用されます。

   フェイルオーバー属性の以下のクラスを指定できます。

   - 必須の属性: `Auto_Failback` または `Controlled_Failback`(互いに排他)
   - オプションの属性:
     - `Auto_Recovery` または `InPlace_Recovery`(互いに排他)
     - `Critical_RG`
     - `Node_Failures_Only`

メモ: 属性の開始条件は、以下のようにクラスによって異なります。

- 必須属性の場合は、クラスタがすでに HA サービスを提供しているときにノードが FailSafe メンバーシップに設定されることです。
- オプションの属性の場合は、HA サービスが開始されていて、クラスタ内の 1 つのノードだけでリソース・グループが実行されていることです。

表5-3 に、各属性について説明します。

表5-3 フェイルオーバー属性

クラス	名前	説明
必須	Auto_Failback	ノードがクラスタに設定されたときに、フェイルオーバー・ポリシーに基づいてリソース・グループがオンラインになるように指定します。この属性は、特定の種類の負荷分散が必要な場合に使用すると最も効果的です。この属性または Controlled_Failback 属性のいずれかを指定してください。
	Controlled_Failback	ノードがクラスタに設定されたときに、リソース・グループが同じノードに残るように指定します。この属性は、tcp を使用して通信するアプリケーションやデータベースなど、クライアント/サーバ・アプリケーションが高機能の回復メカニズムを備えている場合に使用すると最も効果的です。この属性または Auto_Failback 属性のいずれかを指定してください。
オプション	Auto_Recovery	リソース・グループがノードで実行されていることが排他チェックによってわかっている場合でも、フェイルオーバー・ポリシーに基づいてそのリソース・グループがオンラインになるように指定します。この属性はオプションで、InPlace_Recovery 属性とは互いに排他です。これらの属性のどちらも指定していない場合に、 Auto_Failback 属性が指定されているときは、デフォルトでこの属性が使用されます。
	InPlace_Recovery	リソース・グループが実行されているノードと同じノードでリソース・グループがオンラインになるように指定します。この属性はオプションで、Auto_Recovery 属性とは互いに排他です。これらの属性のどちらも指定していない場合に、 Controlled_Failback 属性が指定されているときは、デフォルトでこの属性が使用されます。

クラス	名前	説明
	Critical_RG	リソース・グループの解放異常が発生した場合でも、モニタ異常から正常に回復できるようにします。リソース・モニタが失敗すると、FailSafe は、リソース・グループをアプリケーション・フェイルオーバー・ドメインの別のノードに移動しようとします。FailSafe がリソース・グループのリソースの解放に失敗した場合、そのリソース・グループは、srmd executable error ステータスになります。リソース・グループのフェイルオーバー・ポリシーで Critical_RG フェイルオーバー属性が指定されている場合、FailSafe は、解放操作に失敗したノードをリセットし、フェイルオーバー・ポリシーに基づいてそのリソース・グループを別のノードに移動します。
	Node_Failures_Only	ノード異常が発生した場合のみフェイルオーバーを可能にします。この属性は、ローカル・ノードのリソース再開には影響を与えません。リソース・グループでリソース・モニタ異常が発生した場合は、フェイルオーバーは行われません。この属性は、DMF などの階層ストレージ管理システムを使用しているカスタマにとって便利です。この場合、リソース・モニタ異常は自動回復されないままカスタマに報告されるので、オペレータが必要に応じて回復アクションを手動で実行することが可能です。

フェイルオーバー・ポリシーの例については、『IRIS FailSafe Version 2 Programmer's Guide』を参照してください。

4. 「回復(Recovery)」: 回復属性を選択します。

   - 「FailSafe による選択(Let FailSafe Choose)」は、状況に最適な属性が FailSafe によって決定されることを意味します。
   - 「自動(Automatic)」: フェイルオーバー・ドメインの初期ノードでグループがオンラインにされることを意味します。
   - 「イン・プレイス(In Place)」は、グループがすでに部分的に割当てられているノードでそのグループがオンラインにされることを意味します。

5. 「クリティカル・リソース・グループ(Critical Resource Group)」: オンにして、選択を切替えます。この属性を選択すると、リソース・グループの解放異常が発生した場合でも、モニタ異常から正常に回復できるようになります。

   リソース・モニタが失敗すると、FailSafe は、リソース・グループをフェイルオーバー・ドメインの別のノードに移動しようとします。

   - FailSafe がリソースの解放に失敗した場合、そのリソース・グループは srmd executable error という状態になります。

- 「クリティカル・リソース・グループ(Critical Resource Group)」の状態を選択した場合、FailSafe は、解放操作に失敗したノードをリセットし、フェイルオーバー・ポリシーに基づいてそのリソース・グループを別のノードに移動します。

6. 「ノード異常のみ(Node Failures Only)」: この属性は、リソース・モニタ異常のフェイルオーバーを制御します。この属性を選択した場合、ノード異常が発生した場合にのみリソース・グループ回復(つまり、フェイルオーバー・ドメイン内の別のノードへのフェイルオーバー)が実行されます。

7. 「ほかの属性(Other Attributes)」: フェイルオーバーに使用される追加の属性を入力します。これらのオプションの属性は、ユーザが作成して `/var/cluster/ha/policies` ディレクトリに配置できるユーザ定義フェイルオーバー・スクリプトによって決定されます。

8. 「フェイルオーバー・ドメインの順序付きノード(Ordered Nodes in Failover Domain)」: フェイルオーバー・ドメインは、特定のリソース・グループを割当てることができるノードの順序付きリストです。フェイルオーバー・ドメインにリストされているノードはクラスタに対して定義されなければなりませんが、フェイルオーバー・ドメインにクラスタのすべてのノードを含める必要はありません。フェイルオーバー・ドメインは、クラスタ内のリソース・グループを静的に負荷分散する目的でも使用できます。

   例:

   - 4 ノード・クラスタでは、特定の XLV ボリュームにアクセスできる 2 つのノードのセットを、その XLV ボリュームが含まれるリソース・グループのフェイルオーバー・ドメインに設定できます。
   - `venus`、 `mercury`、および `pluto` というノードで構成されるクラスタでは、リソース・グループ `RG1` および `RG2` に対して以下の初期フェイルオーバー・ドメインを設定できます。
     - `RG1: mercury, venus, pluto`
     - `RG2: pluto, mercury`

   初期フェイルオーバー・ドメインは、フェイルオーバー・ポリシーを設定するときに管理者が定義します。初期フェイルオーバー・ドメインは、クラスタを初めて起動するときに使用されます。初期フェイルオーバー・ドメインによって指定される順序付きリストは、フェイルオーバー・スクリプトによってランタイム・フェイルオーバー・ドメインに変換されます。異常が発生するたびに、フェイルオーバー・スクリプトは現在のランタイム・フェイルオーバー・ドメインを利用し、場合によっては変更します(`ordered` フェイルオーバー・スクリプトについては、順序は変更されません)。初期フェイルオーバー・ドメインが再度使用されることはありません。負荷やフェイルオーバー・スクリプトの内容などのランタイムの状態によっては、初期フェイルオーバー・ドメインおよびランタイム・フェイルオーバー・ドメインは同じこともあります。

   たとえば、 `N1`、`N2`、および `N3` という 3 つのノードを含むクラスタがあり、ノード異常がフェイルオーバーの原因ではなく、初期フェイルオーバー・ドメインは次のとおりであるとします。

   ```
   N1 N2 N3
   ```

   ランタイム・フェイルオーバー・ドメインは、フェイルオーバー・スクリプトによって異なります。

- ordered の場合:

  ```
  N1 N2 N3
  ```

- round-robin の場合:

  ```
  N2 N3 N1
  ```

- カスタマイズされたフェイルオーバー・スクリプトの場合、スクリプトの内容によって、あらゆる順序を取る可能性があります。

  ```
  N1 N2 N3                  N1 N3 N2
  N2 N1 N3                  N2 N3 N1
  N3 N1 N2                  N3 N2 N1
  ```

  FailSafe は、ランタイム・フェイルオーバー・ドメインを格納し、次のフェイルオーバー・スクリプト呼出しへの入力として使用します。

9. 「OK」をクリックして、タスクを完了します。

フェイルオーバー・ポリシーおよびフェイルオーバー・スクリプトの詳細は、『IRIS FailSafe Version 2 Programmer's Guide』を参照してください。このマニュアルでは、独自のフェイルオーバー・ポリシーとフェイルオーバー・スクリプトの作成が重点的に説明されています。

cmgr を使ったフェイルオーバー・ポリシーの定義

フェイルオーバー・ポリシーについての詳細は、174 ページの「GUI を使ったフェイルオーバー・ポリシーの定義」を参照してください。

フェイルオーバー・ポリシーを定義するには、以下を使用します。

```
define failover_policy policyname
    set attribute to attributename
    set script to scriptname
    set domain to nodename
```

次のプロンプトが表示されます。

```
failover_policy policyname?
```

フェイルオーバー・ポリシーを定義するときは、必要に応じて属性とドメインをいくつでも設定できます。ただし、add attribute コマンドや add domain コマンドで異なる値を設定する必要があります。また、次の形式の 1 つのコマンドで複数のドメインを指定できます。

```
set domain to node1 node2 node3 ...
```

フェイルオーバー・ポリシーのコンポーネントは、『IRIS FailSafe Version 2 Programmer's Guide』と、174 ページの「GUI を使ったフェイルオーバー・ポリシーの定義」の概要で詳細に説明されています。

たとえば、属性が Auto_Failback、 Auto_Recovery、および Critical_RG で、node2 node1 というフェイルオーバー・ドメインの fp_ord というフェイルオーバー・ポリシーがあるとします。プライマリ・ノードは node2 で、バックアップ・ノードは node1 です。以下は、通常モードでフェイルオーバー・ポリシーを定義する場合の例です。

```
cmgr> define failover_policy fp_ord
Enter commands, when finished enter either "done" or "cancel"

failover_policy fp_ord? set attribute to Auto_Failback
failover_policy fp_ord? set attribute to Auto_Recovery
failover_policy fp_ord? set attribute to Critical_RG
failover_policy fp_ord? set script to ordered
failover_policy fp_ord? set domain to node2 node1
failover_policy fp_ord? done
```

フェイルオーバー・ポリシー定義の変更

この節では、フェイルオーバー・ポリシーを変更する方法について説明します。

GUI を使ったフェイルオーバー・ポリシー定義の変更

フェイルオーバー・ポリシーの削除プロセスは、新しいポリシーの定義プロセスと類似しています。174 ページの「GUI を使ったフェイルオーバー・ポリシーの定義」を参照してください。

以下を実行します。

1. 「フェイルオーバー・ポリシー(Failover Policy)」: フェイルオーバー・ポリシーの名前を選択します。
2. 「スクリプト(Script)」: メニューを使用して、既存のフェイルオーバー・スクリプトの名前を選択します。

- `ordered` は、FailSafe リリースに付属しています。`ordered` スクリプトによって初期ドメインが変更されることはありません。このスクリプトを使用する場合、初期ドメインとランタイム・ドメインは同じものになります。
- `round-robin` も、FailSafe リリースに付属しています。`round-robin` スクリプトでは、ラウンドロビン（循環）方式でリソース・グループ・オーナーが選択されます。このポリシーは、クラスタ内のすべてのノードで実行できるリソース・グループに対して使用できます。

フェイルオーバー・スクリプトは、`/var/clusters/ha/policies` ディレクトリに保存されます。`ordered` スクリプトがニーズを満たさない場合は、新しいフェイルオーバー・スクリプトを定義して、`/var/clusters/ha/policies` ディレクトリに配置できます。FailSafe GUI を使用している場合、GUI によってスクリプトが自動的に検出され、使用できる選択肢として表示されます。必要なリソース・グループに対して新しいフェイルオーバー・スクリプトを使用するために、クラスタ・データベースを設定できます。フェイルオーバー・スクリプトの定義についての詳細は、『IRIS FailSafe Version 2 Programmer's Guide』を参照してください。

3. **「フェイルバック (Failback)」:** フェイルオーバー属性の名前を選択します。フェイルオーバー属性の以下のクラスを指定できます。
   - 必須の属性: `Auto_Failback` または `Controlled_Failback`（互いに排他）
   - オプションの属性:
     - `Auto_Recovery` または `InPlace_Recovery`（互いに排他）
     - `Critical_RG`
     - `Node_Failures_Only`

メモ: 属性の開始条件は、以下のようにクラスによって異なります。

- 必須属性の場合は、クラスタがすでに HA サービスを提供しているときにノードが FailSafe メンバーシップに設定されることです。
- オプションの属性の場合は、HA サービスが開始されていて、クラスタ内の 1 つのノードだけでリソース・グループが実行されていることです。

---

175 ページの表5-3 に、各属性について説明します。

フェイルオーバー・ポリシーの例については、『IRIS FailSafe Version 2 Programmer's Guide』を参照してください。

4. **「回復(Recovery)」:** 回復属性を選択するか、状況に最適な属性が FailSafe によって選択されるようにします。

- `Automatic` は、フェイルオーバー・ドメインの初期ノードでグループがオンラインにされることを意味します。
- `InPlace` は、グループがすでに部分的に割当てられているノードでそのグループがオンラインにされることを意味します。

5. 「クリティカル・リソース・グループ(Critical Resource Group)」: オンにして、選択を切替えます。この属性を選択すると、リソース・グループの解放異常が発生した場合でも、モニタ異常から正常に回復できるようになります。

   リソース・モニタが失敗すると、FailSafe は、リソース・グループをフェイルオーバー・ドメインの別のノードに移動しようとします。

   - FailSafe がリソースの解放に失敗した場合、そのリソース・グループは `srmd executable error` という状態になります。
   - 「クリティカル・リソース・グループ(Critical Resource Group)」の状態を選択した場合、FailSafe は、解放操作に失敗したノードをリセットし、フェイルオーバー・ポリシーに基づいてそのリソース・グループを別のノードに移動します。

6. 「ノード異常のみ(Node Failures Only)」: この属性は、リソース・モニタ異常のフェイルオーバーを制御します。この属性を選択した場合、ノード異常が発生した場合にのみリソース・グループ回復(つまり、フェイルオーバー・ドメイン内の別のノードへのフェイルオーバー)が実行されます。

7. 「ほかの属性(Other Attributes)」: フェイルオーバーに使用される追加の属性を入力します。これらのオプションの属性は、ユーザが作成して `/var/cluster/ha/policies` ディレクトリに配置できるユーザ定義フェイルオーバー・スクリプトによって決定されます。

8. 「フェイルオーバー・ドメインの順序付きノード(Ordered Nodes in Failover Domain)」: フェイルオーバー・ドメインは、特定のリソース・グループを割当てることができるノードの順序付きリストです。フェイルオーバー・ドメインにリストされているノードはクラスタに対して定義されなければなりませんが、フェイルオーバー・ドメインにクラスタのすべてのノードを含める必要はありません。フェイルオーバー・ドメインは、クラスタ内のリソース・グループを静的に負荷分散する目的でも使用できます。

   初期フェイルオーバー・ドメインは、フェイルオーバー・ポリシーを設定するときに管理者が定義します。初期フェイルオーバー・ドメインは、クラスタを初めて起動するときに使用されます。初期フェイルオーバー・ドメインによって指定される順序付きリストは、フェイルオーバー・スクリプトによってランタイム・フェイルオーバー・ドメインに変換されます。異常が発生するたびに、フェイルオーバー・スクリプトは現在のランタイム・フェイルオーバー・ドメインを利用し、場合によっては変更します(`ordered` フェイルオーバー・スクリプトについては、順序は変更されません)。初期フェイルオーバー・ドメインが再度使用されることはありません。負荷やフェイルオーバー・スクリプトの内容などのランタイムの状態によっては、初期フェイルオーバー・ドメインおよびランタイム・フェイルオーバー・ドメインは同じこともあります。

FailSafe は、ランタイム・フェイルオーバー・ドメインを格納し、次のフェイルオーバー・スクリプト呼出しへの入力として使用します。

9. 「OK」をクリックして、タスクを完了します。

フェイルオーバー・ポリシーおよびフェイルオーバー・スクリプトの詳細は、『IRIS FailSafe Version 2 Programmer's Guide』を参照してください。このマニュアルでは、独自のフェイルオーバー・ポリシーとフェイルオーバー・スクリプトの作成が重点的に説明されています。

cmgr を使ったフェイルオーバー・ポリシー定義の変更

フェイルオーバー・ポリシーを変更するには、次のコマンドを使用します。

```
modify failover_policy policyname
```

フェイルオーバー・ポリシーの変更は、定義に使用するコマンドと同じコマンドを使用して行います。178 ページの「cmgr を使ったフェイルオーバー・ポリシーの定義」を参照してください。

フェイルオーバー・ポリシーの削除

この節では、フェイルオーバー・ポリシーを削除する方法について説明します。

GUI を使ったフェイルオーバー・ポリシーの削除

このタスクを使用して、フェイルオーバー・ポリシーを削除できます。フェイルオーバー・ポリシーを削除しても、ポリシーのフェイルオーバー・ドメイン内のクラスタ・ノードは削除されません。

メモ: リソース・グループによって現在使用されているフェイルオーバー・ポリシーは削除できません。最初に「リソース・グループの変更(Modify Resource Group)」タスクを使用して、そのリソース・グループの別のフェイルオーバー・ポリシーを選択する必要があります。

以下を実行します。

1. 「削除するフェイルオーバー・ポリシー(Failover Policy to Delete)」: ポリシーを選択します。
2. 「OK」をクリックして、タスクを完了します。

cmgr を使ったフェイルオーバー・ポリシーの削除

フェイルオーバー・ポリシーを削除するには、次のコマンドを使用します。

```
delete failover_policy policyname
```

フェイルオーバー・ポリシーの表示

FailSafe を使用して、以下を表示できます。

- 指定したフェイルオーバー・ポリシーのコンポーネント
- すべてのフェイルオーバー・ポリシー
- すべてのフェイルオーバー・ポリシー属性
- すべてのフェイルオーバー・ポリシー・スクリプト

GUI を使ったフェイルオーバー・ポリシーの表示

ツリー表示で定義済みフェイルオーバー・ポリシーをすべて表示するには、「表示(View)」->「フェイルオーバー・ポリシー(Failover Policies)」を選択します。アイテム表示で特定のポリシーについての詳細を表示するには、ツリー表示でそのポリシーの名前を選択します。

cmgr を使ったフェイルオーバー・ポリシーの表示

フェイルオーバー・ポリシーを表示するには、以下のコマンドを使用します。

- フェイルオーバー・ポリシーをすべて表示する場合:

  ```
  show failover policies
  ```

- 特定のフェイルオーバー・ポリシーのパラメータを表示する場合:

  ```
  show failover_policy policyname
  ```

- フェイルオーバー・ポリシー属性をすべて表示する場合:

  ```
  show failover_policy attributes
  ```

- フェイルオーバー・ポリシー・スクリプトをすべて表示する場合:

  ```
  show failover_policy scripts
  ```

リソース・グループ・タスク

リソース・グループは、相互依存したリソースの集合です。リソース・グループは単純な名前で識別されます。この名前は、クラスタ内で一意でなければなりません。

この節では、以下のリソース・グループ・タスクについて説明します。

リソース・グループの定義

この節では、リソース・グループを定義する方法について説明します。

GUI を使ったリソース・グループの定義

複数のリソースは、リソース・グループにまとめて設定されます。リソース・グループは、相互依存したリソースの集合です。リソース・グループのいずれかのリソースが目的の用途に使用できなくなった場合は、リソース・グループ全体が利用不可と見なされます。したがって、FailSafe では、リソース・グループがフェイルオーバーの単位になります。

たとえば、Web ノード自体、外部世界との通信に使用する HA IP アドレス、コンテンツが含まれるディスク・ボリュームなど、Web ノードの操作に必要なすべてのリソースをリソース・グループに含めることができます。

リソース・グループを定義するときは、フェイルオーバー・ポリシーを指定します。フェイルオーバー・ポリシーにより、異常発生時のリソース・グループの処理が制御されます。

以下を実行します。

1. 「フェイルオーバー・ポリシー(Failover Policy)」: フェイルオーバー・ポリシーの名前を選択します。このポリシーは、異常発生時にリソース・グループのサービスを引継ぐノードを決定します。
2. 「リソース・グループ名(Resource Group Name)」: リソース・グループの名前を、最大 63 文字までで指定します。
3. 「OK」をクリックして、タスクを完了します。

グループにリソースを追加するには、187 ページの「グループ内のリソースの追加と削除」を参照してください。

メモ: FailSafe では、オンラインにするリソースが含まれないリソース・グループは使用できません。

任意の数のリソース・グループに、リソースを最大 100 個定義できます。

cmgr を使ったリソース・グループの定義

リソース・グループを定義するには、次のコマンドを使用します。

```
define resource_group RG_name [in cluster clustername]
    set failover_policy to policyname
    add resource resourcename of resource_type RT_name
    remove resource resourcename of resource_type RT_name
```

上記のコマンド書式について、次の点に注意してください。

- failover_policy は、フェイルオーバー・ポリシー名を指定します。
- resource は、リソース名を指定します。
- resource_type は、リソース・タイプを指定します。

例は、次のとおりです。

```
cmgr> define resource_group group1 in cluster filesystem-cluster
Enter commands, when finished enter either "done" or "cancel"

resource_group group1? failover_policy to fp_ord
resource_group group1? add resource 10.154.99.99 of resource_type IP_address
resource_group group1? add resource havol of resource_type volume
resource_group group1? add resource /hafs of resource_type filesystem
resource_group group1? done
```

cmgrを使用したリソース・グループ作成の例については、213 ページの「例: リソース・グループの作成」を参照してください。

リソース・グループ定義の変更

この節では、リソース・グループを変更する方法について説明します。

GUI を使ったリソース・グループ定義の変更

このタスクを使用して、リソース・グループのフェイルオーバー・ポリシーを変更することにより、そのリソース・グループを変更できます。

以下を実行します。

1. 「リソース・グループ(Resource Group)」: リソース・グループを選択します。
2. 「フェイルオーバー・ポリシー(Failover Policy)」: フェイルオーバー・ポリシーを選択します。
3. 「OK」をクリックして、タスクを完了します。

リソース・グループの内容を変更するには、187 ページの「グループ内のリソースの追加と削除」を参照してください。

cmgr を使ったリソース・グループ定義の変更

リソース・グループを変更するには、以下のコマンドを使用します。

```
modify resource_group RG_name [in cluster clustername]
    set failover_policy to policyname
    add resource resourcename of resource_type RT_name
    remove resource resourcename of resource_type RT_name
```

例は、次のとおりです。

```
cmgr> modify resource_group WS1 in cluster test-cluster
```

リソース・グループの変更は、定義に使用するコマンドと同じコマンドを使用して行います。185 ページの「cmgr を使ったリソース・グループの定義」を参照してください。

リソース・グループの削除

この節では、リソース・グループを削除する方法について説明します。

GUI を使ったリソース・グループの削除

このタスクを使用して、オフライン・リソース・グループを削除できます。グループを削除しても、そのグループのメンバーである個々のリソースは削除されません。

メモ: オンラインのリソース・グループは削除できません。

以下を実行します。

1. 「削除するグループ(Group to Delete)」: 削除するリソース・グループの名前を選択します。オフライン・リソース・グループのみがリストされます。
2. 「OK」をクリックして、タスクを完了します。

cmgr を使ったリソース・グループの削除

リソース・グループを削除するには、次のコマンドを使用します。

```
delete resource_group  RG_name [in cluster clustername]
```

例は、次のとおりです。

```
cmgr> delete resource_group WS1 in cluster test-cluster
```

グループ内のリソースの追加と削除

このタスクを使用して、リソースを追加または削除することにより、リソース・グループを変更できます。

メモ:リソースを含まないリソース・グループをオンラインのまま使用することはできないため、いったんリソース・グループがオンラインになったら、そのリソース・グループからすべてのリソースを削除することはできません。同様に、リソースがない場合にリソース・グループをオンラインにすることはできません。

リソースは最小単位で追加または削除する必要があります。つまり、相互依存したリソースは一緒に追加または削除しなければなりません。

以下を実行します。

1. 「リソース・グループ(Resource Group)」: リソース・グループを選択します。グループ内の既存のリソースのリストが表示されます。
2. グループにリソースを追加する場合:
   - 「リソース・タイプ(Resource Type)」: リソース・タイプを選択します。
   - 「リソース名(Resource Name)」: リソース名を選択します。
   - 「追加(Add)」をクリックします。
3. グループ内のリソースを変更する場合:
   - 表示ウィンドウからその名前を選択します。
   - 「変更(Modify)」をクリックします。
4. グループからリソースを削除する場合:
   - 表示ウィンドウからその名前を選択します。
   - 「削除(Delete)」をクリックします。

5. 「OK」をクリックして、タスクを完了します。

リソース・グループの移動

この節では、リソース・グループを移動する方法について説明します。

GUI を使ったリソース・グループの移動

このタスクを使用して、リソース・グループが現在実行されているノードからグループのフェイルオーバー・ドメイン内の別のノードに、そのリソース・グループ(およびそのリソースすべて)を移動できます。

以下を入力します。

1. 「移動するグループ(Group to Move)」: 移動するリソース・グループの名前を選択します。

   「次へ(Next)」をクリックして、次の画面に移動します。

2. 「フェイルオーバー・ドメイン・ノード(Failover Domain Node)」: リソース・グループを移動する先のノードを選択します。ノードは、このリソース・グループのフェイルオーバー・ドメイン内になければなりません。

   この手順はオプションですが、ノードを選択しなかった場合、リソース・グループは、FailSafe によって、フェイルオーバー・ドメインで次に利用可能なノードに移動されます。

3. 「OK」をクリックして、タスクを完了します。

cmgr を使ったリソース・グループの移動

リソース・グループを移動するには、次のコマンドを使用します。

```
admin move resource_group FG_name [in cluster clustername] [to node nodename]
```

たとえば、ノード primary で実行されているリソース・グループ nfs-group1 を、nfs-cluster クラスタ内のノード backup に移動するには、以下を実行します。

```
cmgr> admin move resource_group nfs-group1 in cluster nfs-cluster to node backup
```

ユーザがノードを指定しなかった場合、リソース・グループの移動先ノードを決定するためにそのリソース・グループのフェイルオーバー・ポリシーが使用されます。

リソース・グループの表示

この節では、リソース・グループを表示する方法について説明します。

GUI を使ったリソース・グループの表示

定義済みリソース・グループのパラメータや、クラスタに対して定義されているリソース・グループをすべて表示することができます。

cmgr を使ったリソース・グループの表示

リソース・グループを表示するには、以下のコマンドを使用します。

- 特定のリソース・グループを表示する場合:

  ```
  show resource_group RG_name [in cluster clustername]
  ```

  例は、次のとおりです。

  ```
  cmgr> show resource_group small-rg in cluster test-cluster
  Resource Group: small-rg
          Cluster: test-cluster
          Failover Policy: test_fp

  Resources:
          100.101.10.101  (type: IP_address)
          /hafs  (type: filesystem)
          havol  (type: volume)
  ```

- リソース・グループをすべて表示する場合:

  ```
  show resource_groups [in cluster clustername]
  ```

  例は、次のとおりです。

  ```
  cmgr> show resource_groups in cluster test-cluster

    Resource Groups:
           bar-rg
           foo-rg
           small-rg
  ```

FailSafe HA サービス・タスク

FailSafe システムを設定し、そのコンポーネントに対して診断テストを実行した後、高可用性 (HA) サービスを開始して FailSafe をアクティブにすることができます。HA サービスは、クラスタ内のすべてのノード、または指定した 1 つのノードのみで開始できます。

この節では、以下のタスクについて説明します。

- 「FailSafe HA サービスの開始」
- 191 ページの「FailSafe HA サービスの停止」
- 194 ページの「FailSafe HA パラメータの設定」
- 196 ページの「ログの設定」

FailSafe HA サービスの開始

この節では、FailSafe HA サービスを開始する方法について説明します。

GUI を使った FailSafe HA サービスの開始

FailSafe HA サービスは、クラスタ内のすべてのノード、または指定した 1 つのノードのみで開始できます。

1. 「**クラスタ名 (Cluster Name)**」: クラスタの名前がデフォルトで指定されます。
2. 「**1 ノードのみ(One Node Only)**」: HA サービスを 1 つのノードのみで開始させる場合は、その名前を選択します。このフィールドを空白のままにすると、HA サービスはクラスタ内のすべてのノードで開始されます。

注意: ノードのサブセットで HA サービスを開始するときは、開始されているノードでのみリソース・グループが実行されていることを確認してください。たとえば、クラスタにノード `N1`、 `N2`、および `N3` が含まれている場合に、ノード `N1` と `N2` だけで HA サービスを開始し、ノード `N3` では開始しないときは、リソース・グループがノード `N3` で実行されていないことを確認します。HA サービスが開始されていないノードでは、FailSafe によって排他チェックが実行されません。

HA サービスを開始すると、以下のアクションが実行されます。

- クラスタ内のすべてのノード (または選択したノードのみ) が有効になります。
- クラスタ・データベースの変更後、成功したことが FailSafe によってユーザに伝えられます。
- ローカル `cmond` が `fs2d` デーモンから通知を受信します。
- ローカル `cmond` によってすべての HA プロセス (`cmsd`、`gcd`、`srmd`、 `fsd`) と `ifd` が開始されます。
- `cmond` によって `failsafe2 chkconfig` フラグが `on` に設定されます。

cmgr を使った HA サービスの開始

HA サービスを開始するには、次のコマンドを使用します。

```
start ha_services [on node nodename] [for cluster clustername]
```

例は、次のとおりです。

- クラスタ全体で HA サービスを開始する場合:

```
cmgr> start ha_services for cluster test-cluster
```

- ノード N1 だけで HA サービスを開始する場合:

```
cmgr> start ha_services on node N1 for cluster test-cluster
```

FailSafe HA サービスの停止

この節では、FailSafe HA サービスを停止する方法について説明します。

GUI を使った FailSafe HA サービスの停止

HA サービスは、クラスタ内のすべてのノード、または指定した 1 つのノードのみで停止できます。

メモ: このタスクの実行には時間がかかり、完了までに数分かかる場合があります。

ノードまたはクラスタの停止は複数の手順を含む複雑な操作であり、数分かかる場合があります。停止操作を中止すると、ノードおよびリソースが意図していない状態のままになる可能性があります。

リソース・グループが安定した正常な状態でない場合、ノードまたはクラスタの HA サービスを停止すると、操作に失敗する可能性があります。移行中のリソース・グループは、HA サービス停止コマンドが失敗する原因となります。ほとんどの場合、このコマンドは、リソース・グループが安定した状態になった後で成功します。

ノードまたはクラスタを正常に停止すると、リソース・グループとすべての HA サービスはなくなります。

クラスタ内のすべてのノードの HA サービスを連続して停止することと、クラスタ全体に対して HA サービスを停止することは同じではありません。前者の場合、リソース・グループはオンラインのままで高可用性が維持されますが、後者の場合はオフラインになります。詳細については、続く節を参照してください。

HA サービスを停止すると、FailSafe デーモンによって以下のアクションが実行されます。

- シャットダウン要求が fsd に送信されます。
- すべてのリソース・グループが fsd によって解放され、「オンライン準備済み(ONLINE-READY)」状態になります。
- クラスタ・データベースのクラスタ内にあるすべてのノードが無効になります(一度に 1 ノードずつで、ローカル・ノードが最後になります)。

- FailSafe は、ノードを無効にする前に FailSafe メンバーシップからそのノードが削除されるまで待機します。
- シャットダウンは、すべてのノードが FailSafe メンバーシップの一部でない場合のみ成功します。
- ノードが無効になると、cmond はクラスタ・データベースから通知を受信します。
- ローカル cmond からすべての HA プロセスと ifd に SIGTERM が送信されます。
- すべての HA プロセスがクリーンアップされ、「再開しない(don't restart)」コードで終了します。
- その他のすべての cmsd デーモンにより、無効にされたノードが FailSafe メンバーシップから削除されます。

あるノードの HA サービスが停止されると、そのノードのオンライン・リソース・グループは、フェイルオーバー・ポリシーに基づいて、HA サービスがアクティブであるノードに移動されます。HA サービスがクラスタで停止された場合は、オンライン・リソース・グループがオフラインになり、高可用性でなくなります。

190 ページの「GUI を使った FailSafe HA サービスの開始」の注意を参照してください。

1 ノードでの HA サービスの停止

1 つのノードの HA サービスを停止するには、以下を入力します。

- 「強制(Force)」: エラーが発生しても(通常はエラーが発生して停止操作を防止します)、サービスを強制的に停止する場合は、チェックボックスをオンにします。

  ノードの停止操作では、すべてのリソース・グループをそのノードから別のノードに移動してから、クラスタ内でそのノードを無効にし、最後にすべての HA プロセスを終了します。

  ノードの HA サービスが停止されると、そのノードが所有するすべてのリソース・グループは、これらのリソース・グループを HA 状態に保つことのできるクラスタ内のほかのノードに移動されます。これらのリソース・グループを引継ぐことができるノードがない場合、この操作は失敗します。このような状態は、クラスタ内の最後ノードの HA サービスを停止するときにそのノードがシャットダウンされていると必ず発生します。

  この状況では、Force オプションを指定すると、リソース・グループを移動または解放できない場合でもノードをシャットダウンできます。この場合、通常、リソース・グループは、同じノードに非高可用性の状態で割当てられたままとなります。 force オプションを使用すると、結果としてノードがリセットされる場合があります。リソース・グループがクラスタ内の最後のノードに割当てられたままにするには、すべてのオンライン・リソース・グループを分離します。

  シャットダウンされているノードが所有するリソース・グループをオフラインに移動する場合は、ノードを停止する前に行ってください。

- 「**クラスタ名(Cluster Name)**」: クラスタの名前がデフォルトで指定されます。

- 「**1 ノードのみ(One Node Only)**」: ノード名を選択します。
- 「OK」をクリックして、タスクを完了します。

クラスタ内のすべてのノードでの HA サービスの停止

クラスタ全体で HA サービスを停止すると、すべてのリソース・グループが解放された後、クラスタ内のすべてのノードが無効になり、最後にすべての HA プロセスが終了します。

クラスタが停止され、そのクラスタの FailSafe HA サービスが停止すると、リソース・グループはオフラインに移動されるか、または割当て解除されます。リソース・グループが割当てられたままにする場合は、クラスタを停止する前にそのリソース・グループを分離してください。

クラスタ内のすべてのノードの HA サービスを連続して停止することと、クラスタ全体に対して HA サービスを停止することは同じではありません。前者の場合、リソース・グループはオンラインのままで高可用性が維持されますが、後者の場合はオフラインになります。

すべてのノードの HA サービスを停止するには、以下を入力します。

- 「強制(Force)」: エラーが発生しても強制的に終了する場合は、チェックボックスをオンにします。
- 「**クラスタ名 (Cluster Name)**」: クラスタの名前がデフォルトで指定されます。
- 「**1 ノードのみ(*One Node Only*)**」: このフィールドは空白のままにします。
- 「OK」をクリックして、タスクを完了します。

`cmgr` を使った FailSafe HA サービスの停止

FailSafe HA サービスを停止するには、次のコマンドを使用します。

```
stop ha_services [on node nodename] [for cluster clustername] [force]
```

`force` オプションは、エラーが発生しても停止が行われるようにします。

このタスクの実行には時間がかかり、完了までに数分かかる場合があります。`cmgr` コマンドを使用すると、このようなタスクの中間ステータスが提供されます。例は、次のとおりです。

```
cmgr> stop ha_services in cluster nfs-cluster
Making resource groups offline
Stopping HA services on node node1
Stopping HA services on node node2
```

FailSafe HA パラメータの設定

この節では、FailSafe HA パラメータを設定する方法について説明します。

GUI を使った FailSafe HA パラメータの設定

このタスクを使用して、FailSafe がクラスタをモニタし、ノードのリセットおよびグループ・フェイルオーバーの必要性を検出する方法を変更できます。

1. 「クラスタ名(Cluster Name)」: クラスタの名前。この値はデフォルトで入力されます。

2. 「ハートビート周期(Heartbeat Interval)」: ハートビート・メッセージ間の周期(ミリ秒単位)。この周期は、500 ミリ秒より大きく設定しなければなりませんが、ノード・タイムアウト時間の値の 1/10 より大きく設定することはできません。この周期は、デフォルトでは 1 秒に設定されています。デフォルト値は、1000 ミリ秒です。

   ハートビートの数が多いほど(ハートビート周期が小さいほど)、ネットワークの速度が遅くなる可能性が高くなります。逆に、ハートビートの数が小さいほど(ハートビート周期が大きいほど)、リソースの可用性が減少する可能性が高くなります。

3. 「ノード・タイムアウト(Node Timeout)」: ノード・タイムアウト期間内にノードからハートビートを受信しない場合、ノードはダウンしていると見なされ、FailSafe メンバーシップの一部と見なされなくなります。

   値は、ミリ秒単位で入力します。ノード・タイムアウトは、少なくとも 5 秒に設定してください。また、FailSafe を正しく動作させるため、ノード・タイムアウトはハートビート周期の少なくとも 10 倍に設定してください。この値より低く設定すると、フェイルオーバーが誤って実行されることがあります。デフォルト値は、15000 ミリ秒です。

   ノード・タイムアウトは、クラスタ全体のパラメータです。

4. 「ノード待ち時間(Node Wait Time)」: 新しい FailSafe メンバーシップを宣言する前に、ほかのノードがクラスタに設定されるまでノードが待機する間の周期(ミリ秒単位)。クラスタに対して値が設定されていない場合、FailSafe では、「ノード・タイムアウト」の値にノード数を掛けることによってこの値を計算します。

5. 「電源異常モード(Powerfail Mode)」: ボックスをオンにして、このモードをオンにします。電源異常モードは、リセット要求後にシステム・コントローラから応答が受信されない場合に特定の電源異常アルゴリズムを実行するかどうかを示します。電源異常はノード固有パラメータであり、リセット操作を実行するノードに対して定義します。

6. 「タイブレーカー・ノード(Tie-Breaker Node)」: ノード名を選択します。FailSafe タイブレーカー・ノードは、クラスタ内のノードの 50% が互いに通信できる状況において FailSafe メンバーシップを計算するために使用されるノードです。タイブレーカー・ノードを指定しない場合、ノード ID 番号の最も小さいノードが使用されます。

クラスタ内のノードの数が奇数の場合でも、タイブレーカー・ノードを設定することをお薦めします。これは、1 つのノードが停止したために、偶数のノードでメンバーシップを決定しなければならなくなる可能性があるためです。

ノードのサイズや機能が異なるクラスタでは、最も重要なアプリケーションまたは最大リソース・グループ数を持つ、クラスタ内で最大のノードをタイブレーカー・ノードとして設定します。

cmgr を使った FailSafe HA パラメータの設定

FailSafe パラメータは、次のコマンドを使用して変更できます。

```
modify ha_parameters [on node nodename] [in cluster clustername]
    set node_timeout to timeout_value
    set heartbeat to heartbeat_interval
    set run_pwrfail to true|false
    set node_wait to node_wait_time
    set tie_breaker to tie_breaker_nodename
```

上記のコマンド書式について、次の点に注意してください。

- node_timeout は、ノード・タイムアウト期間です。ノード・タイムアウト期間内にノードからハートビートを受信しない場合、ノードはダウンしていると見なされ、FailSafe メンバーシップの一部と見なされなくなります。

  値は、ミリ秒単位で入力します。ノード・タイムアウトは、少なくとも 5 秒に設定してください。また、FailSafe を正しく動作させるため、ノード・タイムアウトはハートビート周期の少なくとも 10 倍に設定してください。この値より低く設定すると、フェイルオーバーが誤って実行されることがあります。デフォルト値は、60000 ミリ秒です。

  node_timeout は、クラスタ全体のパラメータです。

- heartbeat は、ハートビート・メッセージ間のハートビート周期（ミリ秒単位）です。この周期は、500 ミリ秒より大きく設定しなければなりませんが、ノード・タイムアウト時間の値の 1/10 より大きく設定することはできません。この周期は、デフォルトでは 1 秒に設定されています。デフォルト値は、1000 ミリ秒です。

  ハートビートの数が多いほど（ハートビート周期が小さいほど）、ネットワークの速度が遅くなる可能性が高くなります。逆に、ハートビートの数が小さいほど（ハートビート周期が大きいほど）、リソースの可用性が減少する可能性が高くなります。

- run_pwrfail は、リセット要求後にシステム・コントローラから応答が受信されない場合に特定の電源異常アルゴリズムを実行するかどうか (true) を示します。

  電源異常はノード固有パラメータであり、リセット操作を実行するノードに対して定義します。

- node_wait は、新しい FailSafe メンバーシップを宣言する前に、ほかのノードがクラスタに設定されるまでノードが待機する間の周期(ミリ秒単位)です。クラスタに対して値が設定されていない場合、FailSafe では、「ノード・タイムアウト」の値にノード数を掛けることによってこの値を計算します。
- tie_breaker は、FailSafe タイブレーカーとして動作するノードの名前です。

  tie_breaker を ""(クォーテーション・マークの間はスペースなし)に設定すると、tie_breaker 値の設定が解除されます。tie_breaker の設定を解除することは、最初から値を設定しないことと同じです。この場合、FailSafe では、ノード ID の最も小さいノードがタイブレーカー・ノードとして使用されます。

ログの設定

この節では、ログを設定する方法について説明します。

GUI を使ったログの設定

FailSafe では、各 FailSafe デーモンごとにシステム・ログが維持されています。維持するログのレベルに基づいて、システム・ログをカスタマイズできます。変更は、以下のように適用されます。

- cli ログ・グループと crsd ログ・グループに対しては、プール内のすべてのノード
- その他のすべてのログ・グループに対しては、クラスタ内のすべてのノード

また、クラスタまたはプール内の特定のノードに合わせてログ・グループ設定をカスタマイズすることもできます。

デフォルトのログ・ファイル名

FailSafe は、通常の処理と重大なエラーの両方を、 /var/adm/SYSLOG ファイルのほかに各ログ・グループの個々のログ・ファイルにも記録します。

ログを設定するには、以下のフィールドに適切な値を入力します。

1. 「ログ・グループ(Log Group)」: ログ・グループは、同じログ設定に基づいて同じログ・ファイルにログを記録するプロセスのセットです。すべての FailSafe デーモンは、ログ・グループをそれぞれ 1 つ作成します。FailSafe では、以下のログ・グループが維持されます。

cli	コマンドのログ
crsd	クラスタ・リセット・サービス (crsd) のログ
diags	診断のログ
ha_agent	HA モニタ・エージェント (ha_ifmx2) のログ
ha_cmsd	FailSafe メンバーシップ・デーモン (ha_cmsd) のログ

ha_fsd	FailSafe デーモン (ha_fsd) のログ
ha_gcd	グループ通信デーモン (ha_gcd) のログ
ha_ifd	ネットワーク・インタフェース・モニタ・デーモン (ha_ifd) のログ
ha_script	アクション・スクリプトおよびフェイルオーバー・ポリシー・スクリプトのログ
ha_srmd	システム・リソース・マネージャ (ha_srmd) のログ

2. 「ログ・レベル(Log Level)」: GUI を使用する場合は文字列、cmgr を使用する場合は数値(1〜19)で指定されるログ・レベル。ログ・レベルは、ログの詳細を指定し、関連付けられたログ・グループのファイルに FailSafe が書込むログ・メッセージの量を制御します。デバッグ・レベルには、10 段階あります。表5-4 に、GUI および cmgr を使用して指定する場合のログ・レベルを示します。

表5-4 ログ・レベル

GUI レベル	cmgr レベル	意味
「オフ(Off)」	0	ログには記録されません。
「最小(Minimal)」	1	重大なエラーと通常の操作の通知をログに記録します。
「情報(Info)」	2	「最小(Minimal)」通知と警告をログに記録します。
「デフォルト(Default)」	5	すべての「情報(Info)」メッセージとその他の通知をログに記録します。
「デバッグ 0 (Debug0)」	10	
...		「デバッグ 0 (Debug0)」〜「デバッグ 9 (Debug9)」(cmgr の場合は 10〜19)では、数字の大きさに応じて、ログに記録されるデバッグ情報(データ構造を含む)が増えます。ログ設定でデバッグ・レベルを使用すると、サーバで大量のディスク容量が使用される場合があります。
「デバッグ 9 (Debug9)」	19	

重大なエラーと通常の操作の通知は、常に /var/adm/SYSLOG に送信されます。ログ・グループのログ・レベルを変更しても、SYSLOG には影響はありません。

3. 「ログ・ファイル(Log File)」: 特定のログ・グループの FailSafe 通知が含まれるファイル。スラッシュで始まるログ・ファイル名は絶対パス名です。一方、スラッシュで始まらないログ・ファイル名は、/var/cluster/ha/log ディレクトリへの相対パス名です。

   FailSafe ソフトウェアによって、指定したログ・ファイルの名前にノード名が追加されます。たとえば、ログ・グループのログ・ファイル名を /var/cluster/ha/log/cli として指定すると、ファイル名は /var/cluster/ha/log/cli_*nodename* になります。

   以下に、デフォルトのログ・ファイル名を示します。

   システム回復でのログ・グループの使用についての詳細は、第9章「システムの回復とトラブルシューティング」を参照してください。

4. 「OK」をクリックして、タスクを完了します。

表5-5 デフォルトのログ・ファイル名

ファイル名	説明
/var/cluster/ha/log/cmsd_*nodename*	ノード nodename の FailSafe メンバーシップ・サービス・デーモンのログ・ファイル
/var/cluster/ha/log/gcd_*nodename*	ノード *nodename* のグループ通信デーモンのログ・ファイル
/var/cluster/ha/log/srmd_*nodename*	ノード *nodename* のシステム・リソース・マネージャ・デーモンのログ・ファイル
/var/cluster/ha/log/failsafe_*nodename*	リソース・グループにポリシーを実装する、ノード *nodename* の FailSafe デーモンのログ・ファイル
/var/cluster/ha/log/*agent_nodename*	ノード *nodename* の *agent* というモニタ・エージェントのログ・ファイル。たとえば、 ifd_*nodename* は、インタフェースと IP アドレスのモニタ、および IP アドレスのローカル・フェイルオーバーを行うインタフェース・デーモン・モニタ・エージェントのログ・ファイルです。
/var/cluster/ha/log/crsd_*nodename*	ノード nodename のリセット・デーモンのログ・ファイル
/var/cluster/ha/log/script_*nodename*	ノード nodename のスクリプトのログ・ファイル
/var/cluster/ha/log/cli _*nodename*	GUI および cmgr によって呼出されるノード nodename の内部管理コマンドのログ・ファイル

GUI を使ったログ・グループ定義の表示

GUI を使用してログ・グループ定義を表示するには、「ログ・グループ(Log Group)」メニューを選択します。これを選択すると、そのログ・グループの現在のログ・レベルとログ・ファイルがタスク・ウィンドウに表示されます。必要に応じて、このウィンドウでこれらの設定を変更できます。

cmgr を使ったログ・グループの定義

ログ・グループを定義するには、次のコマンドを使用します。

```
define log_group groupname [on node nodename] [in cluster clustername]
    set log_level to level
    add log_file logfile_name
    remove log_file logfile_name
```

上記のコマンド書式について、次の点に注意してください。

- 特定のノードに対してのみログ・グループ設定をカスタマイズする場合は、そのノード名を指定します。正しい値についての詳細は、196 ページの「GUI を使ったログの設定」を参照してください。
- log_level には、以下のいずれかの値が含まれます。
  - 0: ログには記録されません。
  - 1: 重大なエラーと通常の操作の通知をログに記録します(これらのメッセージのログは、SYSLOG ファイルにも記録されます)。
  - 2:「最小(Minimal)」通知と警告をログに記録します。
  - 5〜7: 数字の大きさに応じて、ログに記録される詳細通知が増えます。
  - 10〜19: 数字の大きさに応じて、ログに記録されるデバッグ情報(データ構造を含む)が増えます。
- log_file は、特定のログ・グループの FailSafe 通知が含まれるファイルです。スラッシュで始まるログ・ファイル名は絶対パス名です。一方、スラッシュで始まらないログ・ファイル名は、/var/cluster/ha/log ディレクトリへの相対パス名です。

  FailSafe ソフトウェアによって、指定したログ・ファイルの名前にノード名が追加されます。たとえば、ログ・グループのログ・ファイル名を /var/cluster/ha/log/cli として指定すると、ファイル名は /var/cluster/ha/log/cli_*nodename* になります。

  デフォルトのログ名のリストについては、198 ページの表5-5 を参照してください。

cmgr を使ったログ・グループの設定

ログ・グループは、次のコマンドを使用して設定できます。

```
define log_group log_group on node hostname [in cluster clustername]
```

log_group 変数は、以下のいずれかになります。

```
cli
crsd
diags
ha_agent
ha_cmsd
ha_fsd
ha_gcd
ha_ifd
ha_script
ha_srmd
```

注意: ログ・ファイルの名前は変更しないでください。名前を変更すると、エラーが発生します。

ログ・レベル 5 を使用してログ・グループ cli をノード fs6 で定義する場合:

```
cmgr> define log_group cli on node fs6 in cluster fs6-8

(Enter "cancel" at any time to abort)

Log Level ? (11) 5

CREATE LOG FILE OPTIONS

        1) Add Log File.
        2) Remove Log File.
        3) Show Current Log Files.
        4) Cancel. (Aborts command)
        5) Done. (Exits and runs command)

Enter option:5
Successfully defined log group cli
```

cmgr を使ったログ・グループの変更

ログ・グループを変更するには、次のコマンドを使用します。

```
modify log_group log_group_name on node hostname [in cluster clustername]
```

ログ・グループの変更は、定義に使用するコマンドと同じコマンドを使用して行います。199 ページの「cmgr を使ったログ・グループの定義」を参照してください。

たとえば、10 になるようログ・レベル cli を変更するには、以下を入力します。

```
cmgr> modify log_group cli on node fs6 in cluster fs6-8

(Enter "cancel" at any time to abort)

Log Level ? (2) 10

MODIFY LOG FILE OPTIONS

        1) Add Log File.
        2) Remove Log File.
        3) Show Current Log Files.
        4) Cancel. (Aborts command)
        5) Done. (Exits and runs command)

Enter option:5
Successfully modified log group cli
```

たとえば、ha_script ログ・グループのログ・レベルを 11 に設定するには、以下を入力します。

```
cmgr> modify log_group ha_script

log_group ha_script ? set log_level to 11
log_group ha_script ? done
Successfully modified log group ha_script
```

ログ・グループ定義の表示

この節では、ログ・グループ定義を表示する方法について説明します。

cmgr を使ったログ・グループ定義の表示

ログ・グループ定義を表示するには、次のコマンドを使用します。

```
show log_groups
```

このコマンドを実行すると、現在定義されているすべてのログ・グループと共に、ログ・グループ名、ログ・レベル、およびログ・ファイルが表示されます。

ログ・ファイルの内容の表示についての詳細は、第9章「システムの回復とトラブルシューティング」を参照してください。

設定例

この章では、3 ノード・クラスタを使用する FailSafe の設定例と、その設定のいくつかのバリエーションについて説明します。さらに、FailSafe 設定で CXFS ファイルシステムをエクスポートする手順についても説明します。この章は、以下の節で構成されています。

- 「例: 3 ノード・クラスタの定義」
- 204 ページの「例: 3 ノード・クラスタを定義するためのスクリプト」
- 210 ページの「例: HA IP アドレスのローカル・フェイルオーバー」
- 211 ページの「例: CXFS ファイルシステムを含めるためのクラスタの変更」
- 212 ページの「例: CXFS ファイルシステムのエクスポート」
- 213 ページの「例: リソース・グループの作成」

例: 3 ノード・クラスタの定義

次の図に、3 ノード FailSafe クラスタを示します。この設定は、ノード `N1`、ノード `N2` 、ノード `N3`、およびノード `N4` を含むプールから構成されています。ノード `N1`、ノード `N2`、およびノード `N3` は、FailSafe クラスタを構成しています。このクラスタ内のノードは、ディスクを共有し、プライベート・コントロール・ネットワークにも接続されている El-8 シリアル・ポート・マルチプレクサに接続されています。

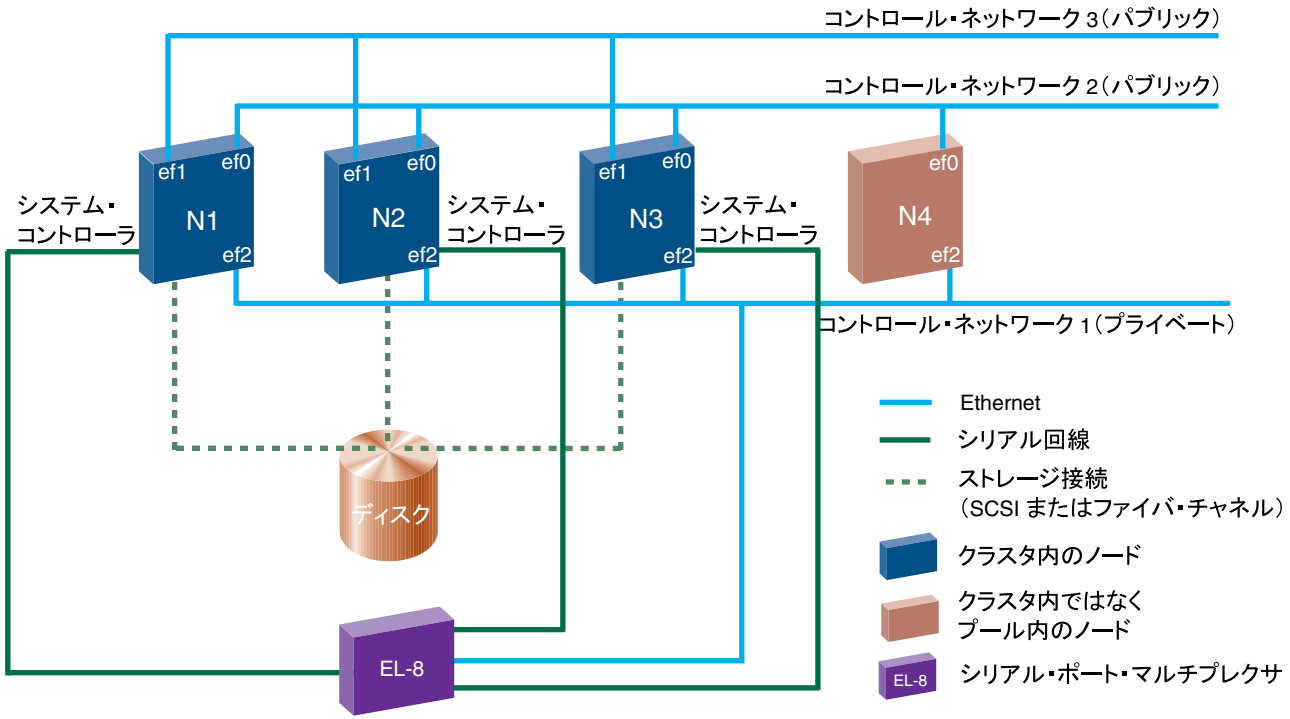


図6-1 FailSafe の設定例

このセットアップを使用する FailSafe の設定例については、以下の節で説明します。

例: 3 ノード・クラスタを定義するためのスクリプト

この節では、図6-1 に示されているような FailSafe の 3 ノード・クラスタを定義する cmgr スクリプトの例について説明します。cmgr スクリプトについての一般情報は、94 ページの「スクリプト・ファイルの使用」を参照してください。独自の設定スクリプトを作成するために使用できるテンプレート・ファイルについての詳細は、96 ページの「テンプレート・スクリプト」を参照してください。

表6-1 に示すように、このクラスタには、RG1 および RG2 という 2 つのリソース・グループがあります。

表6-1 RG1 および RG2 のリソースとフェイルオーバー・ポリシー

リソースとフェイルオーバー・ポリシー	RG1	RG2
リソース		
IP_address	192.26.50.1	192.26.50.2
filesystem	/ha1	/ha2
volume	ha1_vol	ha2_vol
NFS	/ha1/export	/ha2/export
フェイルオーバー・ポリシー		
名前	fp1	fp2
スクリプト	ordered	round-robin
属性	Auto_Failback, Auto_Recovery	Controlled_Failback, InPlace_Recovery
フェイルオーバー・ドメイン	N1, N2, N3	N2, N3

この設定を定義するための cmgr スクリプトは、次のとおりです。

```
#!/usr/cluster/bin/cmgr -f
define node N1
        set hostname to N1
        set is_failsafe to true
        set sysctrl_type to msc
        set sysctrl_status to enabled
        set sysctrl_password to none
        set sysctrl_owner to N4
        set sysctrl_device to /dev/ttydn001
        set sysctrl_owner_type to tty
        add nic ef2-N1
                set heartbeat to true
                set ctrl_msgs to true
                set priority to 1
```

```
        done
        add nic ef0-N1
                set heartbeat to true
                set ctrl_msgs to true
                set priority to 2
        done
        add nic ef1-N1
                set heartbeat to true
                set ctrl_msgs to true
                set priority to 3
        done
done


define node N2
        set hostname to N2
        set is_failsafe to true
        set sysctrl_type to msc
        set sysctrl_status to enabled
        set sysctrl_password to none
        set sysctrl_owner to N4
        set sysctrl_device to /dev/ttydn002
        set sysctrl_owner_type to tty
        add nic ef2-N2
                set heartbeat to true
                set ctrl_msgs to true
                set priority to 1
        done
        add nic ef0-N2
                set heartbeat to true
                set ctrl_msgs to true
                set priority to 2
        done
        add nic ef1-N2
                set heartbeat to true
                set ctrl_msgs to true
                set priority to 3
        done
done

define node N3
        set hostname to N3
```

```
        set is_failsafe to true
        set sysctrl_type to msc
        set sysctrl_status to enabled
        set sysctrl_password to none
        set sysctrl_owner to N4
        set sysctrl_device to /dev/ttydn003
        set sysctrl_owner_type to tty
        add nic ef2-N3
                set heartbeat to true
                set ctrl_msgs to true
                set priority to 1
        done
        add nic ef0-N3
                set heartbeat to true
                set ctrl_msgs to true
                set priority to 2
        done
        add nic ef1-N3
                set heartbeat to true
                set ctrl_msgs to true
                set priority to 3
        done
done


define node N4
        set hostname to N4
        set is_failsafe to true
        add nic ef2-N4
                set heartbeat to true
                set ctrl_msgs to true
                set priority to 1
        done
        add nic ef0-N4
                set heartbeat to true
                set ctrl_msgs to true
                set priority to 2
        done
done
define cluster TEST
        set is_failsafe to true
        set notify_cmd to /usr/bin/mail
```

```
        set notify_addr to failsafe_sysadm@company.com
        add node N1
        add node N2
        add node N3
done

define failover_policy fp1
        set attribute to Auto_Failback
        set attribute to Auto_Recovery
        set script to ordered
        set domain to N1 N2 N3
done


define failover_policy fp2
        set attribute to Controlled_Failback
        set attribute to InPlace_Recovery
        set script to round-robin
        set domain to N2 N3
done

define resource 192.26.50.1 of resource_type IP_address in cluster TEST
        set NetworkMask to 0xffffff00
        set interfaces to ef0,ef1
        set BroadcastAddress to 192.26.50.255
done

define resource ha1_vol of resource_type volume in cluster TEST
        set devname-owner to root
        set devname-group to sys
        set devname-mode to 666
done

define resource /ha1 of resource_type filesystem in cluster TEST
        set volume-name to ha1_vol
        set mount-options to rw,noauto
        set monitor-level to 2
done

modify resource /ha1 of resource_type filesystem in cluster TEST
        add dependency ha1_vol of type volume
done
```

```
define resource /ha1/export of resource_type NFS in cluster TEST
        set export-info to rw,wsync
        set filesystem to /ha1
done

modify resource /ha1/export of resource_type NFS in cluster TEST
        add dependency /ha1 of type filesystem
done
define resource_group RG1 in cluster TEST
        set failover_policy to fp1
        add resource 192.26.50.1 of resource_type IP_address
        add resource ha1_vol of resource_type volume
        add resource /ha1 of resource_type filesystem
        add resource /ha1/export of resource_type NFS
done


define resource 192.26.50.2 of resource_type IP_address in cluster TEST
        set NetworkMask to 0xffffff00
        set interfaces to ef0
        set BroadcastAddress to 192.26.50.255
done

define resource ha2_vol of resource_type volume in cluster TEST
        set devname-owner to root
        set devname-group to sys
        set devname-mode to 666
done

define resource /ha2 of resource_type filesystem in cluster TEST
        set volume-name to ha2_vol
        set mount-options to rw,noauto
        set monitor-level to 2
done

modify resource /ha2 of resource_type filesystem in cluster TEST
        add dependency ha2_vol of type volume
done

define resource /ha2/export of resource_type NFS in cluster TEST
        set export-info to rw,wsync
```

```
        set filesystem to /ha2
done

modify resource /ha2/export of resource_type NFS in cluster TEST
        add dependency /ha2 of type filesystem
done

define resource_group RG2 in cluster TEST
        set failover_policy to fp2
        add resource 192.26.50.2 of resource_type IP_address
        add resource ha2_vol of resource_type volume
        add resource /ha2 of resource_type filesystem
        add resource /ha2/export of resource_type NFS
done


quit
```

例: HA IP アドレスのローカル・フェイルオーバー

FailSafe システムは、同じホスト内の 2 番目のインタフェースに高可用性 (HA) IP アドレスをフェイルオーバーするよう設定できます。これを行うには、IP_address リソース・タイプのリソースに対して複数のインタフェースを指定します。また、異種クラスタをサポートするために異なるインタフェースを指定することもできます。HA IP アドレス・リソースの指定についての詳細は、163 ページの「IP_address 属性」を参照してください。

以下の例では、HA IP アドレスのローカル・フェイルオーバーを設定します。ここでは、図6-1に示されている設定を使用します。

1. 2 つのインタフェースを持つ HA IP アドレス・リソースを定義します。

```
define resource 192.26.50.1 of resource_type IP_address in cluster TEST
          set NetworkMask to 0xffffff00
          set interfaces to ef0,ef1
          set BroadcastAddress to 192.26.50.255
done
```

ef0 インタフェースに異常が発生した場合、HA IP アドレス 192.26.50.1 は、インタフェース ef0 からインタフェース ef1 にローカルでフェイルオーバーされます。

ノード N1、ノード N2、およびノード N3 では、ノードが起動されると ef0 または ef1 のいずれかが自動的に設定されます。 ef0 と ef1 は、両方とも同じサブネット 192.26.50 に物理的に接続さ

れています。同じネットワークに接続されているネットワーク・インタフェースのうち、ノードで設定するものは 1 つだけです。

2. /etc/conf/netif.options ファイルを変更して、ef0 インタフェースおよび ef1 インタフェースを設定します。

```
if1name-ef0
if1addr=192.26.50.10

if2name=ef1
if2addr=192.26.50.11
```

3. etc/init.d/network スクリプトで、ノード N1、ノード N2、およびノード N3 のすべてでネットワーク・インタフェース ef1 を down に設定します。次の行をファイルに追加します。

```
ifconfig ef1 down
```

例: CXFS ファイルシステムを含めるためのクラスタの変更

次の手順の例では、図6-1 に示されているサンプル FailSafe 設定を変更し、CXFS ファイルシステムに高可用性 NFS サービスが含まれるようにします。ただし、CXFS リソース・タイプでは CXFS ファイルシステムはマウントされません。『CXFS Version 2 Software Installation and Administration Guide』で説明されているように、CXFS ファイルシステムは CXFS GUI を使用してマウントする必要があります。CXFS リソース・タイプは、CXFS ファイルシステムのマウント異常をモニタするものです。

メモ: FailSafe では、CXFS ファイルシステムは高可用性であると想定されています。これは、CXFS ファイルシステムをクラスタの別のノードで使用できるようにする場合に、FailSafe フェイルオーバーが必要ないためです。したがって、FailSafe が CXFS ファイルシステムの開始または停止を直接行うことはなく、XVM ボリュームの開始、停止、またはモニタを直接行うこともありません。XVM ボリュームを FailSafe リソース・グループに追加しないでください。

CXFS ファイルシステムを含めるために FailSafe 設定を変更するには、以下の手順を実行します。

1. クラスタ TEST を CXFS で使用するために切替えます。FailSafe クラスタの CXFS への切替えについての詳細は、『CXFS Version 2 Software Installation and Administration Guide』を参照してください。

2. ノード N1 およびノード N2 を CXFS で使用するために切替えます。FailSafe ノードの CXFS への切替えについての詳細は、『CXFS Version 2 Software Installation and Administration Guide』を参照してください。ノードで CXFS サービスを開始します。

3. 新しいリソース・タイプ NFS1 を作成します。これは、リソース・タイプ NFS と同じですが、ファイルシステム依存性を持っていません。このリソース・タイプを作成するには、以下の手順を実行します。

   a. cmgr を使用して、次のコマンドを実行します。

      ```
      cmgr> show resource_type NFS in cluster TEST
      ```

      リソース・タイプ NFS のパラメータが表示されます。

   b. リソース・タイプ NFS に表示されたものと同じ設定情報を使用して、リソース・タイプ NFS1 を定義します。ただし、ファイルシステム依存性はコピーしないでください。

4. 新しいフェイルオーバー・ポリシーの FP3 を、以下の属性を使用して定義します。

   - フェイルオーバー・ドメイン: N1、N2
   - スクリプト: ordered
   - 属性: InPlace_Recovery

5. リソース・タイプ CXFS の /cxfs という名前のリソースを作成します。/cxfs は、CXFS ファイルシステムのマウント・ポイントです。リソース・グループが別のノードに移動されたときに CXFS ファイルシステム /cxfs のメタデータ・サーバを再設定するように決定できます。

6. ファイルオーバー・ポリシー fp3、リソース・タイプ IP_address のリソース ip3、およびリソース・タイプ CXFS のリソース /cxfs を持つ rg3 という名前のリソース・グループを作成します。

7. ノード N1 およびノード N2 に /cxfs をマウントします。XVM ボリュームを持つ CXFS ファイルシステムの定義とマウントについての詳細は、『CXFS Version 2 Software Installation and Administration Guide』を参照してください。

8. リソース・グループ RG3 をクラスタ TEST でオンラインにします。

例: CXFS ファイルシステムのエクスポート

FailSafe 設定で CXFS ファイルシステムをエクスポートするには、以下の手順を実行します。

1. クラスタ内のすべての FailSafe ノードに FailSafe/NFS 2.1 リリースの最新のパッチがインストールされていることを確認します。

2. 211 ページの「例: CXFS ファイルシステムを含めるためのクラスタの変更」で説明されているすべての手順を実行します。

3. /cxfs/share ディレクトリをエクスポートする場合は、/cxfs/share という名前の NFS リソースを作成します。

4. 以下のコマンドを使用して、NFS リソースのほかに HA IP アドレス・リソースと CXFS リソースもリソース・グループ rg3 に追加します。

```
define resource_group rg3 in cluster TEST
set failover_policy to fp3
add resource 99.92.99.99 of resource_type IP_address
add resource /cxfs of resource_type CXFS
add resource /cxfs/share of resource_type NFS
done
```

メモ: この手順を使用して、クラスタ内の複数のノードから同じ CXFS ファイルシステムまたはサブディレクトリをエクスポートすることはできません。

例: リソース・グループの作成

cmgr を使用してリソース・グループを作成するには、以下の手順に従います。

1. 定義するリソース・グループに属するリソースのリストを判別します。同じリソース・グループに属するすべてのリソースは、ノードからノードへまとめて移動されます。

   NFS サービスを提供するリソース・グループには、以下の各タイプのリソースが含まれます。

   - IP_address
   - volume
   - filesystem
   - NFS

     すべてのリソース依存性とリソース・タイプ依存性が満たされている必要があります。たとえば、NFS リソース・タイプは filesystem リソース・タイプに依存するため、NFS リソース・タイプのリソースを含むリソース・グループには、filesystem リソース・タイプのリソースも含まれていなければなりません。

2. リソース・グループによって使用されるフェイルオーバー・ポリシーを決定します。
3. /var/cluster/cmgr-templates/cmgr-create-resource_group ファイル内にあるテンプレート cluster_mgr スクリプトを使用します。

   この例では、以下の特性を持ったリソース・グループを作成するスクリプトを示します。

   - リソース・グループは nfs-group と命名される

- リソース・グループはクラスタ HA-cluster 内にある
- リソース・グループはフェイルオーバー・ポリシーを使用する
- リソース・グループには、IP_Address リソース、 volume リソース、filesystem リソース、および NFS リソースが含まれる

  このリソース・グループは、次のスクリプトを使用して作成できます。

```
define resource_group nfs-group in cluster HA-cluster
        set failover_policy to n1_n2_ordered
        add resource 192.0.2.34 of resource_type IP_address
        add resource havol1 of resource_type volume
        add resource /hafs1 of resource_type filesystem
        add resource /hafs1 of resource_type NFS
done
```

4. cmgr(1M) コマンドの -f オプションを使用して、このスクリプトを実行します。

IRIS FailSafe のシステム運用

この章では、IRIS FailSafe システムを運用したりモニタするために実行する管理タスクについて説明します。また、FailSafe マネージャ GUI と cmgr(1M) コマンドを使用したタスクの実行方法についても説明します。この章の主な節は、以下のとおりです。

- 「Origin 300、Origin 3200C、Onyx 300、および Onyx 3200C のコンソール・サポート」
- 「システム運用時の考慮事項」
- 217 ページの「2 ノード・クラスタ: 単一のノードの使用」
- 223 ページの「システムのステータス」
- 238 ページの「ESP (Embedded Support Partner) による FailSafe イベントのログ」
- 239 ページの「リソース・グループのフェイルオーバー」
- 245 ページの「FailSafe の停止」
- 245 ページの「ノードのリセット」
- 246 ページの「cmgr を使った設定のバックアップと復元」
- 247 ページの「ログ・ファイル管理」

メモ: ネットワーク・パーティションがある場合でもデータベースの最新版コピーが使用可能になるよう、FailSafe 管理はすべてプールの 1 つのノードから行うことをお薦めします。

Origin 300、Origin 3200C、Onyx 300、および Onyx 3200C のコンソール・サポート

Origin 300、Origin 3200C、Onyx 300、および Onyx 3200C システムでは、マシンに対して L1 とコンソールの両方をサポートしているシリアル/USB ポートは 1 つしかありません。FailSafe 設定では、このポート(DB9 接続)はシステム・リセット用に使用され、別のノードのシリアル・ポートまたは Ethernet マルチプレクサに接続されます。

コンソールの入力および出力にアクセスできるようにするには、コンソールをマシンの別のシリアル・ポートにリダイレクトする必要があります。以下の手順に従ってください。

1. 別のシリアル・ポートを使用するよう /etc/inittab ファイルを編集します。

2. `init q`コマンドを発行するか、再起動します。

たとえば、`/etc/inittab` ファイルに以下が記述されているとします(改行して読みやすくしてあります)。

```
# on-board ports or on Challenge/Onyx MP machines, first IO4 board ports
t1:23:respawn:/sbin/suattr -C CAP_FOWNER,CAP_DEVICE_MGT,CAP_DAC_WRITE+ip
-c "exec /sbin/getty ttyd1 console"    # alt console
t2:23:off:/sbin/suattr -C CAP_FOWNER,CAP_DEVICE_MGT,CAP_DAC_WRITE+ip
-c "exec /sbin/getty -N ttyd2 co_9600"     # port 2
```

これは以下のように変更できます。

```
# on-board ports or on Challenge/Onyx MP machines, first IO4 board ports
t1:23:off:/sbin/suattr -C CAP_FOWNER,CAP_DEVICE_MGT,CAP_DAC_WRITE+ip
-c "exec /sbin/getty ttyd1 co_9600"        # port 1
t2:23:respawn:/sbin/suattr -C CAP_FOWNER,CAP_DEVICE_MGT,CAP_DAC_WRITE+ip
-c "exec /sbin/getty -N ttyd2 console" # alt console
```

注意: 上記の手順に従ってコンソールをリダイレクトする方法は、IRIX が実行されている場合にのみ使用できます。IRIX が実行されていない場合 (miniroot) にコンソールにアクセスするには、マシンを物理的に接続する必要があります。つまり、コンソール/L1 ポートからリセット・ケーブルを取外した後、コンソール・ケーブルを接続します。

システム運用時の考慮事項

いったん FailSafe コマンドが開始されると、<Ctrl+c> キーを入力して中断しても、コマンドが部分的に完了してしまう可能性があります。このようにしてコマンドの実行を中止した場合、クラスタは不明な状態になっているので、さまざまな状態コマンドを使用して、クラスタとそのコンポーネントの実際の状態を判断しなければならないことがあります。

2 ノード・クラスタ: 単一のノードの使用

2 ノード・クラスタを使用している場合は、各ノードが単独で動作しなければならなくなった場合に備えて、各ノードに対して非常時用フェイルオーバー・ポリシーを作成しておくことをお薦めします(以下の手順 1)。この状況は、メインテナンスのために他方のノードを停止しなければならない場合や、他方のノードに異常が発生して実行できない場合に発生することがあります。

注意: これらの非常時用フェイルオーバー・ポリシーと適切な手順のセットがないと、正常ノードは「孤立状態(lonely state)」と呼ぶ状態になり、単独でクラスタを形成できなくなります。

単一のノードの使用

以下に、クラスタ内の 1 つのノードだけを使用するときに必要な手順を説明します。

1. 各ノードに対して非常時用フェイルオーバー・ポリシーを作成します。 cmgr コマンドを発行した場合、各ポリシーは以下の例のようになります。*Active_node* はこのポリシーを使用するノードの名前(以下の例では nodeA)で、*Down_node* は機能していないノードの名前(以下の例では nodeB)です。

   ```
   cmgr> show failover_policy emergency-Active_node

   Failover Policy: emergency-Active_node
   Version: 1
   Script: ordered
   Attributes: Controlled_Failback InPlace_Recovery
   Initial AFD: Active_node
   ```

   たとえば、nodeA および nodeB の 2 つのノードがある場合は、以下の 2 つの非常時用フェイルオーバー・ポリシーがあります。

   ```
   cmgr> show failover_policy emergency-nodeA

   Failover Policy: emergency-nodeA
   Version: 1
   Script: ordered
   Attributes: Controlled_Failback InPlace_Recovery
   Initial AFD: nodeA

   cmgr> show failover_policy emergency-nodeB

   Failover Policy: emergency-nodeB
   Version: 1
   Script: ordered
   ```

```
Attributes: Controlled_Failback InPlace_Recovery
Initial AFD: nodeB
```

詳細については、173 ページの「フェイルオーバー・ポリシーの定義」を参照してください。

この手順のこの段階では、実行を試みたものの現在は孤立状態である 1 つのノードがクラスタにあることを想定しています。他方のノードは停止しています。この手順は、この状態からの回復を目的としています。

2. 以下の cmgr コマンドまたは GUI を使用して、適切な単一のノードの非常時用フェイルオーバー・ポリシーが使用されるように各リソース・グループを変更します。

   ```
   modify resource_group RG_name in cluster clustername
   set failover_policy to emergency-Active_node
   ```

   たとえば、nodeA で以下を入力します。

```
cmgr> set cluster test-cluster
cmgr> modify resource_group group1
Enter commands, when finished enter either "done" or "cancel"

resource_group group1 ? set failover_policy to emergency-nodeA
resource_group group1 ? done
Successfully modified resource group group1
```

3. すべてのリソース・グループの状態を「オフライン(offline)」に変更します（マシンが停止する前の、これらのグループの最後の既知の状態は「オンライン(online)」でした。ただし、クラスタはアクティブ・ノードが孤立状態になっている状態で起動されていて、他方のノードが機能していないので、この時点ではリソース・グループは実際にはオンラインではありません。この手順では、後半の手順に備えて、リソース・グループの状態のラベルを適切に付けることだけをデータベースに要求します）。

   次のコマンドを使用します。

   ```
   admin offline resource_group RG_name in cluster clustername
   ```

   例は、次のとおりです。

   ```
   cmgr> set cluster test-cluster
   cmgr> show resource_groups in test-cluster

   cmgr>Resource Groups:
                group1
   ```

```
                group2

cmgr>admin offline resource_group group1
cmgr> admin offline resource_group group2
```

4. 停止しているノードの HA サービスを強制的に停止します。

   stop ha_services on *Down_node* for cluster *clustername* force

   メモ:このタスクの実行には時間がかかり、完了までに数分かかる場合があります。cmgr によってタスクの中間ステータスが表示されます。

   例は、次のとおりです。

   ```
   cmgr> stop ha_services on nodeB for cluster test-cluster force
   ```

5. データベースでリソース・グループに「オンライン(online)」とマークを付けます。これらのサービスは、後の手順で HA サービスが開始されたときに、非常時用フェイルオーバー・ポリシーを使用してオンラインになります。

   admin online resource_group *RG_name* in cluster *clustername*

   例は、次のとおりです。

```
cmgr> set cluster test-cluster
cmgr> admin online resource_group group1
FailSafe daemon (ha_fsd) is not running on this local node or it is not ready to accept admin commands.
Resource Group (group1) is online-ready.

Failed to admin:
        online

admin command failed

cmgr> show status of resource_group group1 in cluster test-cluster

State: Online Ready
Error: No error
Check resource group group1 status in an active node if HA services are active in cluster
```

6. アクティブ・ノードで HA サービスを開始します。

   start ha_services on *Active_node* for *clustername*

これで、この単一のノードのクラスタで HA サービスがアクティブになりました。サービスは、この手順以降の回復手順全体でアクティブで、ダウンタイムは発生しません。

例は、次のとおりです。

```
cmgr> start ha_services on nodeA for cluster test-cluster
```

7. 停止しているノードのデータベースを削除して再初期化します。これにより、データベースがマルチユーザ・モードで起動されます。

```
# cd /var/cluster/cdb
# rm -rf /var/cluster/cdb/cdb*
# /usr/cluster/bin/cdbreinit /var/cluster/cdb/cdb.db
```

数分間待ってから、tail (1) コマンドを使用して、このノードがその識別情報を認識しており、クラスタに設定されたことを示すメッセージを SYSLOG ファイルで監視します。例は、次のとおりです（改行して読みやすくしてあります）。

```
# tail -f /var/adm/SYSLOG
...
Dec 14 18:23:16 6D:Down_node cmond[1074]:  Notification can not be processed, local machine and
  cluster name is not known.
Dec 14 18:23:16 6D:Down_node cmond[1074]:  Local machine belongs to cluster clustername.
Dec 14 18:23:16 6D:Down_node cmond[1074]:  Local machine name is Down_node"
```

2 つのノードの使用の再開

停止しているノードの使用を再開するには、以下を実行します。

1. 停止しているノードをシングルユーザ・モードで起動します。
2. シングルユーザ・モードで、 chkconfig(1M) コマンドを使用して、すべてのクラスタ・サービスを off に設定します。

```
# chkconfig | grep cluster
```

3. *Down_node* をマルチユーザ・モードで起動します。
4. 停止しているノードで HA サービスを開始します。

```
cmgr> start ha_services on node Down_node for cluster clustername
```

5. tail(1) コマンドを使用して、以前に停止していたノードがメンバーシップに含まれ、サービスが「実行中(UP)」状態になっていることを示すメッセージを SYSLOG ファイルで監視します。以下のようなメッセージを確認するまでは、次の手順に進まないでください。

```
# tail -f /var/adm/SYSLOG ... miredb node Active_node [81] : UP  incarnation 7   age
2:0    miredb node Down_node [82] : UP  incarnation 1   age 1:0
```

6. リソース・グループを変更して、異常発生前に使用されていた元のフェイルオーバー・ポリシーを復元します。

```
modify resource_group RG_name in cluster cluster_name
set failover_policy to Original-Failover-Policy
```

メモ: これによって復元されるのは、静的な環境の設定だけです。ランタイム環境では、引続き単一のノードのポリシーが使用されます。

たとえば、通常のフェイルオーバー・ポリシーが normal-fp の場合は、以下のようになります。

```
cmgr> set cluster test-cluster
cmgr> modify resource_group group1
Enter commands, when finished enter either "done" or "cancel"

resource_group group1 ? set failover_policy to normal-fp
resource_group group1 ? done
Successfully modified resource group group1

cmgr modify resource_group group2
Enter commands, when finished enter either "done" or "cancel"

resource_group group2 ? set failover_policy to normal-fp
resource_group group2 ? done
Successfully modified resource group group2
```

7. クラスタ内のリソース・グループに対して offline_detach コマンドを実行します。この操作では、FailSafe でのリソース・グループのモニタが停止されますが、そのグループのプロセスは物理的に停止されません。FailSafe では、このステータスはオフラインとして報告され、グループに対する制御は行われません。リソースは引続きサービスを提供します。

メモ: offline_detach（または offline_detach_force）が実行されると、FailSafe ではグループがモニタされなくなるため、offline_detach が実行されたときに実行していたノード以外で回復することを FailSafe に許可しないでください。

これは、その他のノードを FailSafe メンバーシップに再設定できないことも意味します。このことは、特に Auto_Recovery がリソース・グループのフェイルオーバー・ポリシーで設定されている場合に重要です。メンバーシップに変更があるとフェイルオーバー・ポリシー・スクリプトが常に実行されるため、この制約があります（スクリプトの再実行によって、以前にオフライン分離されていたリソース・グループが、 offline_detach が実行されたノード以外のノードで開始されてしまう場合があります）。

FailSafe ポリシー・スクリプトは、FailSafe が実行されているノード（HA サービスが開始されているノード）でのみ実行されます。たとえば、4 ノードの FailSafe クラスタ（ノード A、B、C、および D）があり、ノード A、B、C の状態は「実行中(UP)」で、ノード D の状態は「停止(DOWN)」であるとします。ノード B で offline_detach または offline_detach_force コマンドを使用してリソース・グループ RG を「オフライン(offline)」にし、同じくノード B で HA サービスをシャットダウンした場合、RG のリソースをノード B で手動で停止する前にノード D をクラスタに再設定しないでください。ノード D をクラスタに再設定すると、リソース・グループ RG はノード A、C、または D で「オンライン(online)」になってしまいます。

次のコマンドを使用します。

```
admin offline_detach resource_group resource_group [in cluster clustername]
```

例は、次のとおりです。

```
cmgr> admin offline_detach resource_group group1 in cluster test-cluster
```

リソース・グループのステータスを表示して、「オフライン(offline)」と表示されることを確認します。

メモ:このコマンドの出力には「オフライン(offline)」と表示されますが、これらのリソースはまだサービスを提供しています。

```
show status of resource_group resource_group in cluster clustername
```

例は、次のとおりです。

```
cmgr> show status of resource_group group1 in cluster test-cluster
```

8. クラスタでリソース・グループをオンラインにします。

```
admin online resource_group RG_name in cluster cluster_name
```

例は、次のとおりです。

```
cmgr> admin online resource_group group1 in cluster test-cluster
cmgr> admin online resource_group group2 in cluster test-cluster
```

9. リソースを元のノードに戻します。元のポリシーに InPlace_Recovery 属性が含まれていたので、すべてのリソースは、この手順全体でアクティブなノードに残っています。

```
admin move resource_group RG_name in cluster cluster_name to node Primary_owner
```

例は、次のとおりです。

```
cmgr> admin move resource_group group1 in cluster test-cluster to node nodeB
```

システムのステータス

FailSafe システムの実行中に、FailSafe コンポーネントのステータスをモニタして、そのコンポーネントの状態を判断できます。FailSafe では、以下の方法でシステムのステータスを表示できます。

- cluster_status コマンドまたは GUI を使用して、クラスタの状態を継続的に監視できます。
- GUI または cmgr のいずれかを使用して、個々のリソース・グループ、ノード、またはクラスタのステータスを照会できます。
- cmgr コマンドに付属する haStatus スクリプトを使用して、設定内のすべてのクラスタ、ノード、リソース、およびリソース・グループを参照できます。

以下の節では、これらのタスクを実行するための手順について説明します。

cluster_status を使ったシステムのステータスのモニタ

curses(3X) インタフェースを使用してクラスタをモニタするには、cluster_status コマンドを使用できます。たとえば、FailSafe 用に設定された 2 ノード・クラスタのステータス表示と、cluster_status ヘルプ・テキストを以下に示します。

```
# /var/cluster/cmgr-scripts/cluster_status
* Cluster=nfs-cluster  FailSafe=ACTIVE  CXFS=Not Configured         08:45:12
   Nodes =   hans2    hans1
FailSafe =      UP       UP
    CXFS =
       ResourceGroup           Owner           State           Error

       bartest-group                          Offline        No error
       footest-group                          Offline        No error
            bar_rg2           hans2            Online        No error
```

```
nfs-group1           hans2          Online        No error
    foo_rg           hans2          Online        No error


      +-------+ cluster_status Help +--------+
      |  on s  - Toggle Sound on event       |
      |  on r  - Toggle Resource Group View  |
      |  on c  - Toggle CXFS View            |
      |     j  - Scroll up the selection     |
      |     k  - Scroll down the selection   |
      |   TAB  - Toggle RG or CXFS selection |
      | ENTER  - View detail on selection    |
      |     h  - Toggle help screen          |
      |     q  - Quit cluster_status         |
      +--- Press 'h' to remove help window --+
```

上記の例では、ノードまたはクラスタのステータスが変わると、サウンドがアクティブになります。s 設定は、-m(ミュート)オプションと共に cluster_status を呼出すことによってオーバーライドできます。また、矢印キーを使用して選択肢をスクロールすることもできます。

メモ: cluster_status コマンドを使用して表示できる CXFS ファイルシステムの数は、最大 128 個までです。

GUI を使ったシステムのステータスのモニタ

クラスタの状態を継続的に監視する簡単な方法は、GUI のツリー表示を使用することです。

問題が発生しているシステム・コンポーネントは、点滅する赤いアイコンとして表示されます。状態が移行中のコンポーネントも、点滅するアイコンとして表示されます。リソース・グループまたはノードに問題がある場合、クラスタのアイコンのほかに、リソース・グループまたはノードのアイコンも赤く点滅します。

クラスタのステータスは、以下のいずれかになります。

- 「アクティブ(ACTIVE)」: クラスタが実行中で、有効な FailSafe メンバーシップがあることを意味します。
- 「非アクティブ(INACTIVE)」:「FailSafe HA サービスの開始(Start FailSafe HA Services)」タスクが実行されておらず、有効な FailSafe メンバーシップがないことを意味します。
- 「エラー(ERROR)」: 一部のノードが「停止(DOWN)」状態であることを意味します(つまり、クラスタは実行されているべきなのに、実行されていません)。

- 「不明(UNKNOWN)」: クエリーを実行しているノードで FailSafe HA サービスが実行されていないために状態を判別できないことを意味します。

GUI ウィンドウを最小化した場合、最小化されたアイコンに、クラスタの現在の状態が表示されます。クラスタがエラー状態になると、アイコンには赤いクラスタが表示されます。

アイコンと状態の説明

以下の表に、FailSafe マネージャ GUI で使用されるアイコンと状態を説明します。

FailSafe クラスタ表示におけるコンポーネントの状態に対する凡例は、以下のとおりです。

表7-1 アイコンの説明

アイコン	実体
	ノード
	クラスタ
	リソース
	リソース・グループ
	リソース・タイプ

アイコン	実体
	フェイルオーバー・ポリシー
	展開されているツリー
	縮小されているツリー

表7-2 状態の説明

アイコン	状態
	「非アクティブ(Inactive)」または「不明(Unknown)」(HA サービスがアクティブでない可能性があります)
	リソース・グループの「オンライン準備済み(Online-ready)」状態
	正常で、「アクティブ(Active)」または「オンライン(Online)」になっている
	(点滅)正常で「アクティブ(Active)」/「オンライン(Online)」に移行中、または「オフライン(Offline)」に移行中

アイコン	状態
	メインテナンス・モードで、リソースは FailSafe によってモニタされていない
	(点滅する赤)コンポーネントに関する問題がある

cmgr を使ったクラスタのステータスの照会

ノードおよびクラスタのステータスを照会するには、次のコマンドを使用します。

```
show status of cluster clustername
```

cmgr を使ったリソースとリセット・シリアル回線のモニタ

リソースのステータスを照会したり、ノードでシステム・コントローラと通信するには、以下の項で説明されているとおりに、 cmgr を使用できます。

cmgr を使ったリソースのステータスの照会

リソースのステータスを照会するには、次のコマンドを使用します。

```
show status of resource resource_name of resource_type RT_name [in cluster clustername]
```

デフォルトのクラスタが指定されている場合は、このコマンドを使用する際にクラスタを指定する必要はなく、デフォルトのクラスタで指定されているリソースのステータスが表示されます。

cmgr を使ったシステム・コントローラへの ping の実行

デバイス名を指定してシステム・コントローラに ping(1M) 操作を実行するには、次のコマンドを使用します。

```
admin ping dev_name devicename of dev_typedevice_type with sysctrl_type
system_controller_type
```

リソース・グループのステータス

リソース・グループのステータスを照会するには、リソース・グループの名前と、そのリソース・グループが含まれているクラスタを指定します。リソース・グループのステータスには、以下のコンポーネントが含まれます。

- リソース・グループの状態
- リソース・グループのエラー状態
- リソース・オーナー

これらのコンポーネントについては、以下の節で説明します。

オンラインのリソース・グループが含まれているノードのステータスが「不明(UNKNOWN)」の場合、そのリソース・グループのステータスは利用可能や「オンライン準備済み(ONLINE-READY)」にはなりません。

リソース・グループの状態

リソース・グループの状態は、以下のいずれかになります。

ONLINE	FailSafe は、ローカル・ノードで実行されています。リソース・グループは、クラスタのノードに割当てられ、FailSafe によってモニタされています。エラーがない場合、リソース・グループは完全に割当てられています。そうでない場合は、一部のリソースが割当てられていないか、エラー状態の可能性があります。
ONLINE-PENDING	FailSafe はローカル・ノードで実行されており、リソース・グループは割当て処理中です。これは一時的な状態です。
OFFLINE	リソース・グループは実行されていないか、または分離されています。FailSafe が実行されているかどうかは関係ありません。このリソース・グループは、FailSafe の開始時に割当てられません。
OFFLINE-PENDING	FailSafe はローカル・ノードで実行されており、リソース・グループは解放処理(オフライン化)中です。これは一時的な状態です。
ONLINE-READY	FailSafe はローカル・ノードで実行されていません。FailSafe は、開始時にこのリソース・グループをオンラインにしようと試みます。この状態が返された場合は、現在のノードで FailSafe プロセスが実行されていません。

ONLINE-MAINTENANCE	リソース・グループはクラスタのノードに割当てられていますが、FailSafe によってモニタされていません。「オンライン・メインテナンス(ONLINE-MAINTENANCE)」状態のリソース・グループがそのノードに存在しているときにノード異常が発生した場合、そのリソース・グループは別のノードに移動され、モニタが再開されます。アップグレードやテストのため、またはそのリソースに対して FailSafe を一定期間動作させないようにしなければならない理由がある場合、管理者は、リソース・グループを「オンライン・メインテナンス(ONLINE-MAINTENANCE)」状態に移動できます。
INTERNAL ERROR	内部 FailSafe エラーが発生しており、リソース・グループの状態は FailSafe によって認識されていません。エラーからの回復が必要です。これは、メモリ・エラー、プログラムのバグ、または通信上の問題が原因の可能性があります。
DISCOVERY (EXCLUSIVITY)	リソース・グループ内の任意のリソースがそのリソース・グループのフェイルオーバー・ドメイン内のすべてのノードにすでに割当てられているかどうかを FailSafe が正しく判断できる場合、リソース・グループはオンライン化処理中です。これは一時的な状態です。
INITIALIZING	ローカル・ノード上の FailSafe は、このリソース・グループに関する情報をまだ取得していません。これは一時的な状態です。

リソース・グループのエラー状態

リソース・グループの状態が「オンライン(ONLINE)」の場合は、そのエラー状態が継続的にモニタされます。リソース・グループのエラー状態は、以下のいずれかになります。

NO ERROR	リソース・グループにエラーはありません。
INTERNAL ERROR - NOT RECOVERABLE	この状態が発生した場合は、日本 SGI まで通知してください。
NODE UNKNOWN	オンラインのリソース・グループが含まれていたノードは、不明な状態になります。これは、ノードがクラスタの一部ではない場合に起こります。最後に認識されたリソース・グループの状態は「オンライン(ONLINE)」ですが、システムはノードと通信できません。
SRMD EXECUTABLE ERROR	リソース・グループのリソースに対して、start アクションまたは stop アクションが失敗しています。
SPLIT RESOURCE GROUP (EXCLUSIVITY)	リソース・グループの一部がクラスタ内の少なくとも 2 つの異なるノードで実行されていたことが FailSafe によって判別されています。

NODE NOT AVAILABLE (EXCLUSIVITY)	リソース・グループのフェイルオーバー・ドメイン内にあるノードの 1 つがメンバーシップに含まれていなかったことが FailSafe によって判別されています。フェイルオーバー・ドメインからノードが削除されるか、HA サービスがノードで開始されるまで、FailSafe はリソース・グループをオンラインにできません。
MONITOR ACTIVITY UNKNOWN	メインテナンス・モードをオンまたはオフにする際にエラーが発生しました。FailSafe は、モニタが有効または無効のどちらになっているかを判断できなくなっています。操作を再試行してください。エラーが解決されない場合は、日本 SGI まで報告してください。
NO AVAILABLE NODES	FailSafe メンバーシップの最後の有効なノードでモニタ・エラーが発生しています。

リソース・オーナー

リソース・オーナーは、現在リソースを所有しているノードの論理ノード名です。

GUI を使ったリソース・グループのステータスのモニタ

FailSafe 設定でリソースのステータスをモニタするには、ツリー表示を使用できます。

- 属しているグループ別にリソースを表示するには、「表示(View)」->「グループ内のリソース(Resources in Groups)」を選択します。
- オンライン・グループが実行されている場所を表示するには、「表示(View)」->「ノードが所有するグループ(Groups owned by Nodes)」を選択します。この表示では、フェイルオーバーの発生時に監視できます。

cmgr を使ったリソース・グループのステータスの照会

リソース・グループのステータスを照会するには、次の cmgr コマンドを使用します。

```
show status of resource_group RG_name [in cluster clustername]
```

デフォルトのクラスタが指定されている場合は、このコマンドを使用する際にクラスタを指定する必要はなく、デフォルトのクラスタで指定されているリソース・グループのステータスが表示されます。

ノードのステータス

ノードのステータスを照会するには、そのノードの論理ノード名を指定します。ノードのステータスは、以下のいずれかになります。

UP	このノードは、FailSafe メンバーシップの一部です。

DOWN	このノードは、FailSafe メンバーシップの一部ではなく(ハートビートなし)、リセットされています。これは一時的な状態です。
UNKNOWN	このノードは、FailSafe メンバーシップの一部ではなく(ハートビートなし)、リセットされていません(リセットの試行は失敗しています)。
INACTIVE	このノードでは HA サービスは開始されていません。

HA サービスを開始すると、ノードの状態は「非アクティブ(INACTIVE)」から「実行中(UP)」に移行します。「非アクティブ(INACTIVE)」から「不明(UNKNOWN)」になった後、「実行中(UP)」に移行する場合もあります。

cluster_status を使ったノードのステータスのモニタ

クラスタ内のノードのステータスをモニタするには、cluster_status コマンドを使用できます。

GUI を使ったクラスタのステータスのモニタ

FailSafe 設定でクラスタのステータスをモニタするには、GUI のツリー表示を使用できます。デフォルトのクラスタ、そのリソース・グループ、およびそのグループのリソースの動作状態をモニタするには、「表示(View)」->「ノードが所有するグループ(Groups owned by Nodes)」を選択します。

cmgr を使ったクラスタのステータスの照会

ノードのステータスを照会するには、次のコマンドを使用します。

```
show status of node nodename
```

cmgr を使ったシステム・コントローラへの ping の実行

FailSafe が実行されているときは、次のコマンドを使用して、ノードのシステム・コントローラが応答しているかどうかを判断できます。

```
admin ping node nodename
```

このコマンドは、FailSafe デーモンを使用して、システム・コントローラが応答しているかどうかをテストします。

FailSafe デーモンが実行されていない場合でも、admin ping コマンドの standalone オプションを使用すると、クラスタ内のノードのリセット接続性を確認できます。

```
admin ping standalone node nodename
```

このコマンドは FailSafe デーモンを経由せず、ping コマンドを直接呼出して、指定したノードでシステム・コントローラが応答しているかどうかをテストします。

haStatus スクリプトを使ったシステムのステータスの表示

haStatus スクリプトを使用すると、設定内のクラスタ、ノード、リソース、およびリソース・グループに関するステータスと設定の情報が表示されます。このスクリプトは、 /var/cluster/cmgr-scripts ディレクトリにインストールされています。このスクリプトは、ニーズに合わせて変更できます。このスクリプトについての詳細は、haStatus(1M) のマン・ページを参照してください。

以下の例に、haStatus スクリプトのさまざまなオプションの出力を示します。

```
# haStatus -help
Usage: haStatus [-a|-i] [-c clustername]
where,
 -a prints detailed cluster configuration information and cluster
status.
 -i prints detailed cluster configuration information only.
 -c can be used to specify a cluster for which status is to be printed.
  “clustername” is the name of the cluster for which status is to be
printed.
# haStatus
Tue Nov 30 14:12:09 PST 1999
Cluster test-cluster:
        Cluster state is ACTIVE.
Node hans2:
        State of machine is UP.
Node hans1:
        State of machine is UP.
Resource_group nfs-group1:
        State: Online
        Error: No error
        Owner: hans1
        Failover Policy: fp_h1_h2_ord_auto_auto
        Resources:
                /hafs1  (type: NFS)
                /hafs1/nfs/statmon      (type: statd)
                150.166.41.95   (type: IP_address)
                /hafs1  (type: filesystem)
                havol1  (type: volume)
# haStatus -i
Tue Nov 30 14:13:52 PST 1999
Cluster test-cluster:
Node hans2:
        Logical Machine Name: hans2
        Hostname: hans2.engr.sgi.com
```

```
        Is FailSafe: true
        Is CXFS: false
        Nodeid: 32418
        Reset type: powerCycle
        System Controller: msc
        System Controller status: enabled
        System Controller owner: hans1
        System Controller owner device: /dev/ttyd2
        System Controller owner type: tty
        ControlNet Ipaddr: 192.26.50.15
        ControlNet HB: true
        ControlNet Control: true
        ControlNet Priority: 1
        ControlNet Ipaddr: 150.166.41.61
        ControlNet HB: true
        ControlNet Control: false
        ControlNet Priority: 2
Node hans1:
        Logical Machine Name: hans1
        Hostname: hans1.engr.sgi.com
        Is FailSafe: true
        Is CXFS: false
        Nodeid: 32645
        Reset type: powerCycle
        System Controller: msc
        System Controller status: enabled
        System Controller owner: hans2
        System Controller owner device: /dev/ttyd2
        System Controller owner type: tty
        ControlNet Ipaddr: 192.26.50.14
        ControlNet HB: true
        ControlNet Control: true
        ControlNet Priority: 1
        ControlNet Ipaddr: 150.166.41.60
        ControlNet HB: true
        ControlNet Control: false
        ControlNet Priority: 2
Resource_group nfs-group1:
        Failover Policy: fp_h1_h2_ord_auto_auto
                Version: 1
                Script: ordered
                Attributes: Auto_Failback Auto_Recovery
```

```
                Initial AFD: hans1 hans2
        Resources:
                /hafs1  (type: NFS)
                /hafs1/nfs/statmon      (type: statd)
                150.166.41.95   (type: IP_address)
                /hafs1  (type: filesystem)
                havol1  (type: volume)
Resource /hafs1 (type NFS):
        export-info: rw,wsync
        filesystem: /hafs1
        Resource dependencies
        statd /hafs1/nfs/statmon
        filesystem /hafs1
Resource /hafs1/nfs/statmon (type statd):
        InterfaceAddress: 150.166.41.95
        Resource dependencies
        IP_address 150.166.41.95
        filesystem /hafs1
Resource 150.166.41.95 (type IP_address):
        NetworkMask: 0xffffff00
        interfaces: ef1
        BroadcastAddress: 150.166.41.255
        No resource dependencies
Resource /hafs1 (type filesystem):
        volume-name: havol1
        mount-options: rw,noauto
        monitor-level: 2
        Resource dependencies
        volume havol1
Resource havol1 (type volume):
        devname-group: sys
        devname-owner: root
        devname-mode: 666
        No resource dependencies
Failover_policy fp_h1_h2_ord_auto_auto:
        Version: 1
        Script: ordered
        Attributes: Auto_Failback Auto_Recovery
        Initial AFD: hans1 hans2
# haStatus -a
Tue Nov 30 14:45:30 PST 1999
Cluster test-cluster:
```

```
        Cluster state is ACTIVE.
Node hans2:
        State of machine is UP.
        Logical Machine Name: hans2
        Hostname: hans2.engr.sgi.com
        Is FailSafe: true
        Is CXFS: false
        Nodeid: 32418
        Reset type: powerCycle
        System Controller: msc
        System Controller status: enabled
        System Controller owner: hans1
        System Controller owner device: /dev/ttyd2
        System Controller owner type: tty
        ControlNet Ipaddr: 192.26.50.15
        ControlNet HB: true
        ControlNet Control: true
        ControlNet Priority: 1
        ControlNet Ipaddr: 150.166.41.61
        ControlNet HB: true
        ControlNet Control: false
        ControlNet Priority: 2
Node hans1:
        State of machine is UP.
        Logical Machine Name: hans1
        Hostname: hans1.engr.sgi.com
        Is FailSafe: true
        Is CXFS: false
        Nodeid: 32645
        Reset type: powerCycle
        System Controller: msc
        System Controller status: enabled
        System Controller owner: hans2
        System Controller owner device: /dev/ttyd2
        System Controller owner type: tty
        ControlNet Ipaddr: 192.26.50.14
        ControlNet HB: true
        ControlNet Control: true
        ControlNet Priority: 1
        ControlNet Ipaddr: 150.166.41.60
        ControlNet HB: true
        ControlNet Control: false
```

```
        ControlNet Priority: 2
Resource_group nfs-group1:
        State: Online
        Error: No error
        Owner: hans1
        Failover Policy: fp_h1_h2_ord_auto_auto
                Version: 1
                Script: ordered
                Attributes: Auto_Failback Auto_Recovery
                Initial AFD: hans1 hans2
        Resources:
                /hafs1  (type: NFS)
                /hafs1/nfs/statmon      (type: statd)
                150.166.41.95   (type: IP_address)
                /hafs1  (type: filesystem)
                havol1  (type: volume)
Resource /hafs1 (type NFS):
        State: Online
        Error: None
        Owner: hans1
        Flags: Resource is monitored locally
        export-info: rw,wsync
        filesystem: /hafs1
        Resource dependencies
        statd /hafs1/nfs/statmon
        filesystem /hafs1
Resource /hafs1/nfs/statmon (type statd):
        State: Online
        Error: None
        Owner: hans1
        Flags: Resource is monitored locally
        InterfaceAddress: 150.166.41.95
        Resource dependencies
        IP_address 150.166.41.95
        filesystem /hafs1
Resource 150.166.41.95 (type IP_address):
        State: Online
        Error: None
        Owner: hans1
        Flags: Resource is monitored locally
        NetworkMask: 0xffffff00
        interfaces: ef1
```

```
        BroadcastAddress: 150.166.41.255
        No resource dependencies
Resource /hafs1 (type filesystem):
        State: Online
        Error: None
        Owner: hans1
        Flags: Resource is monitored locally
        volume-name: havol1
        mount-options: rw,noauto
        monitor-level: 2
        Resource dependencies
        volume havol1
Resource havol1 (type volume):
        State: Online
        Error: None
        Owner: hans1
        Flags: Resource is monitored locally
        devname-group: sys
        devname-owner: root
        devname-mode: 666
        No resource dependencies
# haStatus -c test-cluster
Tue Nov 30 14:42:04 PST 1999
Cluster test-cluster:
        Cluster state is ACTIVE.
Node hans2:
        State of machine is UP.
Node hans1:
        State of machine is UP.
Resource_group nfs-group1:
        State: Online
        Error: No error
        Owner: hans1
        Failover Policy: fp_h1_h2_ord_auto_auto
        Resources:
                /hafs1  (type: NFS)
                /hafs1/nfs/statmon      (type: statd)
                150.166.41.95   (type: IP_address)
                /hafs1  (type: filesystem)
                havol1  (type: volume)
```

ESP (Embedded Support Partner) による FailSafe イベントのログ

ESP は、さまざまなモニタ・アクティビティを実行するデーモンのセットで構成されます。FailSafe イベントがログに記録されるように ESP を設定できます（FailSafe ESP イベント・プロファイルは、デフォルトでは ESP で設定されていません）。

FailSafe にはイベント・クラス ID 77 と記述名 `IRIS FailSafe2` が使用されます。

FailSafe に対して ESP を使用する場合は、次のコマンドを入力して、`failsafe2` イベント・プロファイルを ESP に追加します。

```
# espconfig -add eventprofile failsafe2
```

その後、FailSafe によって、以下に関する ESP イベントがログに記録されます。

- デーモン設定エラー
- フェイルオーバー・ポリシー設定エラー
- リソース・グループ割当て (`start`) 異常
- リソース・グループ異常
  - 割当て (`start`) 異常
  - 開放 (`stop`) 異常
  - モニタ異常
  - 排他異常
  - フェイルオーバー・ポリシー異常
- リソース・グループのステータス
  - `online`
  - `offline`
  - `maintenance_on`
  - `maintenance_off`
- FailSafe のシャットダウン（HA サービスの停止）
- FailSafe の開始（HA サービスの開始）

ログに記録された ESP イベントを表示するには、espreport(1M) または launchESPartner(1) コマンドを使用できます。ESP についての詳細は、esp(5) のマン・ページおよび『Embedded Support Partner User Guide』を参照してください。

リソース・グループのフェイルオーバー

FailSafe システムの実行中に、リソース・グループを特定のノードでオンラインにしたり、リソース・グループをオフラインにできます。さらに、クラスタ内のあるノードから別のノードにリソース・グループを移動することもできます。続く項では、これらのタスクについて説明します。

リソース・グループをオンラインにする

この節では、リソース・グループをオンラインにする方法について説明します。

GUI を使ったリソース・グループのオンライン化

リソース・グループを初めてオンラインにする前に、そのリソース・グループで診断テストを実行します。診断を実行すると、システム設定がチェックされ、リソース・グループをオンラインにしたときには実行されないいくつかの検証が行われます。

以下の状況の場合は、リソース・グループをオンラインにできません。

- リソース・グループにメンバーがいない場合
- リソース・グループが現在クラスタで実行されている場合

リソース・グループを完全にオンラインにするには、HA サービスがアクティブでなければなりません。HA サービスがアクティブになると、クラスタ内でそのリソース・グループの割当てが試みられます。ただし、HA サービスがアクティブでないときは、リソース・グループをオンラインにするコマンドを実行できます。HA サービスがアクティブでない場合、そのリソース・グループは、HA サービスがアクティブになったときにオンラインになるようマークされます。このリソース・グループの状態は、その後「オンライン準備済み(ONLINE-READY)」になります。HA サービスが開始されると、FailSafe は、「オンライン準備済み(ONLINE-READY)」状態のリソース・グループをオンラインにしようとします。

241 ページの「リソース・グループをオフラインにする」で説明されているように、GUI または cmgr を使用してリソース・グループをオフラインにして、HA サービスの開始時にリソース・グループがオンラインにならないように設定できます。

注意: クラスタ内でリソース・グループをオンラインにする前に、そのリソース・グループが、(HA サービスが実行されていない) 無効になっているノードで実行されていないことを確認してください。無効になっているノードで実行されているリソース・グループをオンラインにすると、データが壊れる可能性があります。分離されたリソース・グループについての詳細は、241 ページの「リソース・グループをオフラインにする」を参照してください。

以下を実行します。

1. 「オンラインにするグループ(Group to Bring Online)」: プルダウン・リストを使用して、オンラインにするリソース・グループの名前を選択します。このメニューに表示されるのは、現在オンラインではないリソース・グループだけです。
2. 「OK」をクリックして、タスクを完了します。

cmgr を使ったリソース・グループのオンライン化

リソース・グループをオンラインにするには、次のコマンドを使用します。

```
admin online resource_group RG_name [in cluster clustername]
```

デフォルトのクラスタが指定されている場合は、このコマンドを使用する際にクラスタを指定する必要はありません。

例は、次のとおりです。

```
cmgr> set cluster test-cluster
cmgr> admin online resource_group group1
FailSafe daemon (ha_fsd) is not running on this local node or it is not ready to accept admin commands.
Resource Group (group1) is online-ready.

Failed to admin:
        online

admin command failed

cmgr> show status of resource_group group1 in cluster test-cluster

State: Online Ready
Error: No error
Check resource group group1 status in an active node if HA services are active in cluster
```

リソース・グループをオフラインにする

この節では、リソース・グループをオンラインにする方法について説明します。

GUI を使ったリソース・グループのオフライン化

リソース・グループをオフラインにすると、リソース・グループ内の各リソースも、FailSafe によって定義済みの順序でオフラインにされます。このプロセス中に単一のリソースでエラーが発生すると、プロセスは停止し、残りのリソースはすべて割当てられたままになります。

FailSafe リソース・グループは、以下の方法でオフラインにできます。

- リソース・グループをオフラインにします。この操作では、そのリソース・グループのプロセスが物理的に停止され、エラー状態はリセットされません。この操作に失敗した場合、リソース・グループはエラー状態のままオンラインとなります。
- リソース・グループを強制的にオフラインにします。この操作では、そのリソース・グループのプロセスが物理的に停止されますが、エラー状態はリセットされます。この操作に失敗することはありません。
- リソース・グループを分離します。この操作では、FailSafe でのリソース・グループのモニタが停止されますが、そのグループのプロセスは物理的に停止されません。FailSafe では、このステータスはオフラインとして報告され、グループに対する制御は行われません。この操作に失敗することは、めったにありません。
- リソース・グループを分離して、エラー状態を強制的にクリアします。この操作では、FailSafe でのリソース・グループのモニタが停止されますが、そのグループのプロセスは物理的に停止されません。FailSafe では、このステータスはオフラインとして報告され、グループに対する制御は行われません。さらに、リソース・グループのすべてのエラー状態がリセットされます。この操作に失敗することは、めったにありません。

リソース・グループを停止する必要がなく、変更中は FailSafe にそのリソース・グループをモニタさせたくないが、リソース・グループに対する管理制御は必要な場合は(そのリソース・グループを別のノードに移動する場合など)、243 ページの「リソース・グループのモニタの中断と再開」で説明されているとおりに、GUI の「リソース・グループのモニタの中断(Suspend Monitoring a Resource Group)」タスクまたは `cmgr` の `admin maintenance_on` コマンドを使用して、リソース・グループをメインテナンス・モードにすることができます。

`fsd` デーモンが実行されていない場合や、クライアント要求を受付ける準備ができていない場合は、このタスクを実行することにより、クラスタ・データベースのリソース・グループだけが無効になります。リソース・グループはオンラインのままになり、コマンドは失敗します。

以下を入力します。

1. 「分離のみ(Detach Only)」: リソース・グループのモニタを停止するには、このボックスをオンにします。リソース・グループは停止されませんが、このグループは FailSafe によって制御されなくなります。

2. 「強制分離(Detach Force)」: リソース・グループのモニタを停止するには、このボックスをオンにします。リソース・グループは停止されませんが、このグループは FailSafe によって制御されなくなります。さらに、Failsafe によってすべてのエラーがクリアされます。

注意: 「分離のみ(Detach Only)」および「強制分離(Detach Force)」の設定では、リソース・グループのリソースは、グループがオンラインだったノードで実行されたままになります。ノードの HA サービスを停止してから、クラスタ内の別のノードでそのリソース・グループをオンラインにしないでください。この操作を行うと、データの整合性の問題が発生する可能性があります。代わりに、ノードの HA サービスを停止する前に、そのノードでリソースが実行されていないことを確認してください。

3. 「強制オフライン(Force Offline)」: グループのすべてのリソースを停止して、すべてのエラーをクリアするには、このボックスをオンにします。
4. 「強制オフライン(Force Offline)」: グループのすべてのリソースを停止して、すべてのエラーをクリアするには、このボックスをオンにします。
5. 「オフラインにするグループ(Group to Take Offline)」: オフラインにするリソース・グループの名前を選択します。このメニューに表示されるのは、現在オンラインであるリソース・グループだけです。
6. 「OK」をクリックして、タスクを完了します。

cmgr を使ったリソース・グループのオフライン化

リソース・グループをオフラインにするには、次のコマンドを使用します。

```
admin offline resource_group RG_name [in cluster clustername]
```

force オプションを使用して、エラーがあっても FailSafe に強制的にアクションを完了させるには、次のコマンドを使用します。

```
admin offline_force resource_group RG_name [in cluster clustername]
```

リソース・グループを分離するには、次のコマンドを使用します。

```
admin offline_detach resource_group RG_name [in cluster clustername]
```

リソース・グループを分離して、エラー状態を強制的にクリアするには、次のコマンドを使用します。

```
admin offline_detach_force resource_group RG_name [in
cluster clustername]
```

この操作では、FailSafe でのリソース・グループのモニタが停止されますが、そのグループのプロセスは物理的に停止されません。FailSafe では、このステータスはオフラインとして報告され、グループに対する制御は行われません。さらに、リソース・グループのすべてのエラー状態がリセットされます。この操作に失敗することは、めったにありません。

リソース・グループの移動

この節では、リソース・グループを移動する方法について説明します。

GUI を使ったリソース・グループの移動

FailSafe がアクティブである間に、同じクラスタ内の別のノードにリソース・グループを移動できます。

メモ: アクティブなシステムでリソース・グループを移動すると、コマンドは成功したように見えるにもかかわらず、そのリソース・グループがクラスタ内の同じノードでオンラインのままになっているという予期しない処理となる場合があります。これは、移動先のノードでリソース・グループの開始に失敗した場合に起こることがあります。この場合、FailSafe は、アプリケーション・フェイルオーバー・ドメイン内の次のノードにこのリソース・グループをフェイルオーバーします。このノードは、リソース・グループが初めに実行されていたノードである場合もあります。リソース・グループはオンラインのまま維持されているため、コマンドは成功します。

以下を実行します。

1. 「移動するグループ(Group to Move)」: 移動するリソース・グループの名前を選択します。メニューに表示されるのは、現在オンラインであるリソース・グループだけです。
2. 「フェイルオーバー・ドメイン・ノード(Failover Domain Node)」: (オプション)リソース・グループの移動先のノードの名前を選択します。ノードを指定しなかった場合、リソース・グループは、FailSafe によって、フェイルオーバー・ドメインで次に利用可能なノードに移動されます。
3. 「OK」をクリックして、タスクを完了します。

`cmgr` を使ったリソース・グループの移動

リソース・グループを別のノードに移動するには、次のコマンドを使用します。

```
admin move resource_group RG_name [in cluster clustername]
[to node nodename]
```

リソース・グループのモニタの中断と再開

この節では、リソース・グループのモニタを停止して、メインテナンス・モードにする方法について説明します。

GUI を使ったリソース・グループのモニタの中断

FailSafe による特定のリソース・グループのモニタを一時的に停止できます。これにより、そのリソース・グループはメインテナンス・モードになります。リソース・グループはクラスタ内の同じノードにそのまま残りますが、リソース異常は FailSafe によってモニタされなくなります。

FailSafe によってリソース・グループが一定期間モニタされないようにする場合は、そのグループをメインテナンス・モードにすることができます。これは、アップグレードやテストのため、またはそのリソース・グループに対して FailSafe を動作させないようにしなければならない理由がある場合に行うことができます。メインテナンス・モードのリソース・グループはモニタされず、高可用性ではありません。リソース・グループのオーナー・ノードに異常が発生した場合、FailSafe は、そのリソース・グループを別のノードに移動し、モニタを再開します。

リソース・グループをメインテナンス・モードにすると、リソース・グループ内のリソースは「オンライン・メインテナンス(ONLINE-MAINTENANCE)」状態になります。リソースの「オンライン・メインテナンス(ONLINE-MAINTENANCE)」状態が表示されるのは、オンラインのリソースがあるノードだけです。その他のすべてのノードでは、リソースは「オンライン(ONLINE)」として表示されます。ただし、リソース・グループは、すべてのノードで「オンライン・メインテナンス(ONLINE-MAINTENANCE)」状態として表示されます。

以下を実行します。

1. 「モニタを停止するグループ(Group to Stop Monitoring)」: モニタを開始するグループの名前を選択します。メニューに表示されるのは、現在オンラインでモニタされているリソース・グループだけです。
2. 「OK」をクリックして、タスクを完了します。

GUI を使ったリソース・グループのモニタの再開

このタスクを使用して、現在 FailSafe でモニタされていないリソース・グループのモニタを再開できます（デフォルトでは、エラーがなく「オンライン(Online)」状態のすべてのリソース・グループがモニタされます）。

モニタが再開されると、リソース・グループまたはそのリソースの 1 つに異常が発生した場合、各異常コンポーネントは FailSafe によってフェイルオーバー・ポリシーに基づいて再開されます（再開アクションが有効な場合）。

以下の手順を実行します。

1. 「モニタを開始するグループ(Group to Start Monitoring)」: モニタを開始するグループの名前を選択します。メニューに表示されるのは、現在オンラインでモニタされていないリソース・グループだけです。
2. 「OK」をクリックして、タスクを完了します。

cmgr を使ってリソース・グループをメインテナンス・モードにする

リソース・グループをメインテナンス・モードにするには、次のコマンドを使用します。

```
admin maintenance_on resource_group RG_name [in cluster clustername]
```

デフォルトのクラスタが指定されている場合は、このコマンドを使用する際にクラスタを指定する必要はありません。

cmgr を使ったリソース・グループのモニタの再開

リソース・グループをメインテナンス・モードからオンラインに移動するには、次のコマンドを使用します。

```
admin maintenance_off resource_group RG_name [in cluster clustername]
```

FailSafe の停止

FailSafe は、クラスタ内のすべてのノード、または指定した 1 つのノードのみで停止できます。191 ページの「FailSafe HA サービスの停止」を参照してください。

ノードのリセット

FailSafe を使用してクラスタ内のノードをリセットできます。これにより、指定したノードのシステム・コントローラ・ポートにリセット・コマンドが送信されます。ノードがリセットされると、クラスタ内のその他のノードで、このノードがリセットされたことが検出されます。続いて、運用中のクラスタからノードが削除され、ノードに割当てられていたリソース・グループがバックアップ・ノードに再割当てされます。使用されるバックアップ・ノードは、システムの設定方法によって異なります。

ノードは、再起動するとクラスタに再設定されます。システムの設定方法によっては、特定のリソース・グループが元のノードに戻る場合もあります。

GUI を使ったノードのリセット

GUI を使用してクラスタ内のノードをリセットできます。これにより、指定したノードのシステム・コントローラ・ポートにリセット・コマンドが送信されます。ノードがリセットされると、クラスタ内のほかのノードで変更が検出され、アクティブなクラスタからこのノードが削除されます。ノードは、再起動すると FailSafe メンバーシップに再設定されます。

ノードをリセットするには、以下を実行します。

1. 「リセットするノード(Node to Reset)」: プルダウン・メニューを使用して、リセットするノードを選択します。
2. 「OK」をクリックして、タスクを完了します。

cmgr を使ったノードのリセット

FailSafe が実行されている場合、次のコマンドを使用してノードを再起動できます。

admin reset node *nodename*

このコマンドは、FailSafe デーモンを使用して、指定したノードをリセットします。

FailSafe デーモンが実行されていない場合でも、 admin reset コマンドの standalone オプションを使用すると、クラスタ内のノードをリセットできます。

admin reset standalone node *nodename*

このコマンドは、FailSafe デーモンを経由しません。

cmgr を使った設定のバックアップと復元

cmgr コマンドには、設定をバックアップしたり復元するために使用できる cdbBackup および cdbRestore というスクリプトが付属しています。これらのスクリプトは、/usr/cluster/bin ディレクトリにインストールされており、ニーズに合わせて変更できます。

付属の cdbBackup スクリプトは、/var/cluster/cdb/cdb.db# ディレクトリと /var/cluster/cdb.db ファイルの tar 圧縮ファイルを作成します。

付属の cdbRestore スクリプトは、/var/cluster/cdb/cdb.db# ディレクトリと /var/cluster/cdb.db ファイルの tar 圧縮ファイルを復元します。

cdbBackupスクリプトおよび cdbRestore スクリプトを使用するときは、以下の手順に従います。

- cdbBackupスクリプトおよび cdbRestoreスクリプトは、管理コマンドが実行されていない場合にのみ実行してください。管理コマンドの実行中にこれらのスクリプトを実行すると、バックアップの整合性が失われることがあります。
- クラスタ内の各ノードの設定は、個別にバックアップしてください。設定情報は各ノードによって異なり、すべてのノード固有情報はローカルでしか保存されていません。
- 設定を変更するたびに、バックアップ手順を実行します。
- 同時に作成したプール内のすべてのノードのバックアップは、一緒に復元します。

- クラスタ・プロセスおよび FailSafe プロセスの実行中は、設定の復元を行わないでください。

メモ: 上記の制約に加えて、クラスタ・データベースの情報の変更中も cdbDump は実行しないでください。クラスタ・データベース・アクティビティが行われている場合を判断するには、SYSLOG をチェックします。一般的には、最後のノードがクラスタに設定されてから、または最後の管理コマンドが実行されてから少なくとも 15 分が経過している場合は、cdbDump を実行できます。

ログ・ファイル管理

ディスクがいっぱいにならないよう、ログ・ファイルは、少なくとも毎週ローテーションさせます。

以下の節では、スクリプトの例を示します。このようなスクリプトを定期的に実行するには、root crontab(1) にエントリを加えることができます。

ログ・レベルについての詳細は、196 ページの「ログの設定」を参照してください。

すべてのログ・ファイルのローテーション

すべてのファイルを新しい場所にコピーするには、以下のようなスクリプトを使用できます。

```
#!/bin/sh

DATE=`/sbin/date +'%U-%a'`
LOG_DIR="/var/cluster/ha/log"
HOST=`/usr/bsd/hostname -s`
LOG_FILES="cad_log cmond_log fs2d_log"
LOG_HFILES="cli cmsd crsd failsafe gcd ifd script srmd clconfd"

LOG_ARCH=$LOG_DIR"/Old-Log"

if [ ! -d $LOG_ARCH ] ; then
   mkdir $LOG_ARCH
fi

for file in $LOG_FILES
do

  rm -f ${LOG_ARCH}/${file}-${DATE}
  cp ${LOG_DIR}/${file} ${LOG_ARCH}/${file}-${DATE}
  echo "Log Rotation at `date`" > ${LOG_DIR}/${file}
```

```
done

for file in $LOG_HFILES
do

  rm -f ${LOG_ARCH}/${file}_${HOST}-${DATE}
  cp ${LOG_DIR}/${file}_${HOST} ${LOG_ARCH}/${file}_${HOST}-${DATE}
  echo "Log Rotation at `date`" > ${LOG_DIR}/${file}_${HOST}
done
```

このスクリプトを cron(1) ジョブとして実行して、ログ・ファイルを定期的にクリーンアップできます。このスクリプトでは、FailSafe クラスタで HA サービスがアクティブな場合にログ・ファイルがローテーションされます。デフォルトのログ・レベルでは、大きいログ・ファイルは作成されません。

設定のテスト

この章では、FailSafe マネージャ GUI および cmgr(1M) コマンドを使った IRIS FailSafe システム設定のテスト方法について説明します。これらのツールの使用についての一般的な情報は、83 ページの 第4章「管理ツール」を参照してください。

この章の節は、以下のとおりです。

- 「FailSafe 診断コマンドの概要」
- 250 ページの「GUI を使った診断タスクの実行」
- 251 ページの「cmgr を使った診断タスクの実行」

FailSafe 診断コマンドの概要

表8-1 に、FailSafe 診断コマンドを使用して実行できるテストを示します。

表8-1 FailSafe 診断テストの概要

診断テスト	説明
リソース	以下をチェックします。 • リソース・タイプ・パラメータが設定されているかどうか • パラメータが構文的に正しいかどうか • パラメータが存在するかどうか
リソース・グループ	リソース・グループで定義されたすべてのリソースをテストします。

診断テスト	説明
フェイルオーバー・ポリシー	以下をチェックします。 • フェイルオーバー・ポリシーが存在するかどうか • フェイルオーバー・ドメインにホストの有効なリストが含まれているかどうか
ネットワークの接続性	以下をチェックします。 • コントロール・インタフェースが同じネットワーク上にあるかどうか • ノードが互いに通信できるかどうか
シリアル接続	ノードが互いにリセットできることをチェックします。

すべてのトランザクションのログは、ログ・ディレクトリの診断ファイル `diags_` *nodename* に記録されます。

リソース・グループは、FailSafe HA サービスまたはリソース・グループを開始する前にテストします。これらのテストは、リソース・グループが正常に開始するのを妨げる可能性のあるリソースの不整合性をチェックするよう設計されています。

GUI を使った診断タスクの実行

この節では、GUI を使用して診断テストを実行する方法を説明します。

GUI を使った接続性のテスト

このタスクには、ノード間の `root rsh`(1) アクセスが必要です。接続性をテストするには、「FailSafe マネージャ(FailSafe Manager)」から以下を実行します。

メモ:「ノードの接続性テスト(Test Node Connectivity)」画面では、ホスト間の `rsh`(1) アクセスが必要です。`/.rhosts` ファイルには、接続性をテストするホストとローカル・ホストを含めなければなりません。

- 適切なラジオ・ボタンをクリックして、ネットワーク別またはシリアル接続別のどちらでテストするかを選択します。
- テストするノードをプルダウン・リストから選択し、「追加(Add)」をクリックして、テスト・リストにそのノードを追加します。

テストするノードのリストからノードを削除するには、その論理名をクリックして選択し、「削除(Delete)」をクリックします。

- テストを開始するには、「テストの開始(Start Tests)」をクリックします。テストを停止するには、「テストの停止(Stop Tests)」をクリックします。
- 別のテストを実行するには、「出力をクリア(Clear Output)」をクリックしてステータス画面をクリアし、手順 3 から新たに実行します。
- ウィンドウを閉じるには、「閉じる(Close)」をクリックします。

GUI を使ったリソースのテスト

「リソースのテスト(Test Resources)」タスクでは、要求された入力値を入力することによって、クラスタ内のノードのリソースをテストできます。リソースは、タイプおよびグループ別にテストできます。リソース・タイプまたはリソース・グループのリソースは、クラスタ内のすべてのノードで一度にテストしたり、個々にノードを指定してテストできます。リソース・テストは、リソース・グループのアプリケーション・フェイルオーバー・ドメインのノードでのみ実行されます。

GUI を使ったフェイルオーバー・ポリシーのテスト

「フェイルオーバー・ポリシーのテスト(Test Failover Policy)」タスクでは、フェイルオーバー・ポリシーが正しく定義されているかどうかをテストできます。このテストでは、ポリシー・スクリプトとフェイルオーバー属性、およびクラスタからの有効なノードでアプリケーション・フェイルオーバー・ドメインが構成されているかどうかを検証することによって、フェイルオーバー・ポリシーをチェックします。

cmgr を使った診断タスクの実行

以下の項では、cmgr コマンドを使用したシステムでの診断タスクの実行方法について説明します。

cmgr を使ったシリアル接続のテスト

FailSafe ノード間のシリアル接続をテストするには、cmgr コマンドを使用できます。このテストでは、指定した各ノードに対する ping(1M) がシリアル回線を通して実行され、ping の実行に成功しなかった場合は、エラー・メッセージが生成されます。

メモ: FailSafe が実行されている間は、このコマンドを使用しないでください。

クラスタ内のマシンに対するシリアル接続をテストするには、次のコマンドを使用します。

```
test serial in cluster C_name [on node node1 node node2 ...]
```

たとえば、複数のノードをテストするには、次のコマンドを入力します。

```
cmgr> test serial in cluster test-cluster on node blue node green
```

このシリアル・テストでは、最初のエラーが発生するとエラー・メッセージが表示され、応答しなかったノードが示されます。このテストを実行した後でエラー・メッセージを受取った場合は、指定したノードのシリアル・ポートからリモート電源コントロール装置、またはその他のノードのシステム・コントローラ・ポートへのシリアル・ケーブルの接続を確認し、テストを再び実行します。

例は、次のとおりです。

```
cmgr> test serial in cluster eagan on node cm1
Success: testing serial...
Success: Ensuring Node Can Get IP Addresses For All Specified Hosts
Success: Number of IP addresses obtained for <cm1> = 1
Success:        The first IP address for <cm1> = 128.162.19.34
Success: Checking serial lines via crsd (crsd is running)
Success: Successfully checked serial line
Success: Serial Line OK
Success: overall exit status:success, tests failed:0, total tests executed:1
```

以下に、FailSafe が実行されているときに test serial コマンドの実行を試行した場合の例を示します(コマンドは実行に失敗します)。

```
cmgr> test serial in cluster eagan on node cm1
Error: Cannot run the serial tests, diagnostics has detected FailSafe (ha_cmsd) is running

Failed to execute FailSafe tests/diagnostics ha

test command failed
cmgr>
```

cmgr を使ったネットワークの接続性のテスト

クラスタでのネットワークの接続性をテストするには、cmgr コマンドを使用できます。このテストでは、指定したノードが、ノードで設定されている各インタフェースを通じて互いに通信できるかどうかをチェックします。FailSafe が実行されている場合、このテストは行われません。

クラスタ内のマシンに対するネットワークの接続性をテストするには、次のコマンドを使用します。

```
test connectivity in cluster C_name [on node node1 node node2 ...]
```

以下に、test connectivity コマンドの例を示します。

```
cmgr> test connectivity in cluster eagan on node cm1
Success: testing connectivity...
Success: checking that the control IP_addresses are on the same networks
Success: pinging address cm1-priv interface ef0 from host cm1
Success: pinging address cm1 interface ef1 from host cm1
Success: overall exit status:success, tests failed:0, total tests
executed:1
```

このテストでは、最初のエラーが発生するとエラー・メッセージが表示され、応答しなかったノードが示されます。このテストを実行した後でエラー・メッセージを受取った場合は、次のような ifconfig コマンドを使用して、ネットワーク・インタフェースが設定されていることを確認します。

```
# /usr/etc/ifconfig ec3

ec3: flags=c63<UP,BROADCAST,NOTRAILERS,RUNNING,FILTMULTI,MULTICAST>
        inet 190.0.3.1 netmask 0xffffff00 broadcast 190.0.3.255
```

出力の最初の行にある UP は、インタフェースが設定されていることを示します。

ネットワーク・インタフェースが設定されている場合は、ネットワーク・ケーブルが正しく接続されていることを確認し、テストを再び実行します。

cmgr を使ったリソースのテスト

設定されたリソースをリソース名またはリソース・タイプ別にテストするには、cmgr を使用できます。

リソースを名前別にテストするには、次のコマンドを使用します。

```
test resource resourcename of resource_type RT_name  in cluster C_name [on node node1  node node2 ...]
```

例は、次のとおりです。

```
cmgr> test resource /disk1 of resource_type filesystem in cluster eagan on machine cm1
Success: *** testing node resources on node cm1 ***
Success: *** testing all filesystem resources on node cm1 ***
Success: testing resource /disk1 of resource type filesystem on node cm1
Success: overall exit status:success, tests failed:0, total tests executed:1
```

リソースをリソース・タイプ別にテストするには、次のコマンドを使用します。

```
test resource_type RT_name in cluster C_name [on node node1 node node2 ...]
```

例は、次のとおりです。

```
cmgr> test resource_type filesystem in cluster eagan on machine cm1
```

```
Success: *** testing node resources on node cm1 ***
Success: *** testing all filesystem resources on node cm1 ***
Success: testing resource /disk4 of resource type filesystem on node cm1
Success: testing resource /disk5 of resource type filesystem on node cm1
Success: testing resource /disk2 of resource type filesystem on node cm1
Success: testing resource /disk3 of resource type filesystem on node cm1
Success: testing resource /disk1 of resource type filesystem on node cm1
Success: overall exit status:success, tests failed:0, total tests executed:5
```

destructive モードでボリュームおよびファイルシステム・リソースをテストするには、cmgr を使用できます。これにより、ファイルシステムとボリュームをさらに詳細にテストできます。FailSafe が実行されている場合、cmgr テストは destructive モードで実行されません。

リソースを destructive モードでテストするには、次のコマンドを使用します。

```
test resource resourcename of resource_type RT_name  in cluster C_name  [on node node1 node node2 ...] destructive
```

以下の節では、リソースに使用可能な診断テストについて説明します。

cmgr を使った論理ボリュームのテスト

クラスタ内の論理ボリュームをテストするには、cmgr コマンドを使用できます。このテストでは、指定したボリュームが正しく設定されているかどうかをチェックします。

論理ボリュームをテストするには、次のコマンドを使用します。

```
test resource resourcename of resource_type volume on cluster  C_name
[on node node1 node node2  ...]
```

例は、次のとおりです。

```
cmgr> test resource alternate of resource_type volume on cluster eagan
Success: *** testing node resources on node cm1 ***
Success: *** testing all volume resources on node cm1 ***
Success: running resource type volume tests on node cm1
Success: *** testing node resources on node cm2 ***
Success: *** testing all volume resources on node cm2 ***
Success: running resource type volume tests on node cm2
Success: overall exit status:success, tests failed:0, total tests executed:2
```

以下の例では、destructive モードで論理ボリュームをテストします。

```
cmgr> test resource alternate of resource_type volume on cluster eagan destructive
Warning: executing the tests in destructive mode
Success: *** testing node resources on node cm1 ***
Success: *** testing all volume resources on node cm1 ***
Success: running resource type volume tests on node cm1
Success: successfully assembled volume: alternate
Success: *** testing node resources on node cm2 ***
Success: *** testing all volume resources on node cm2 ***
Success: running resource type volume tests on node cm2
Success: successfully assembled volume: alternate
Success: overall exit status:success, tests failed:0, total tests executed:2
```

cmgr を使ったファイルシステムのテスト

クラスタで設定されているファイルシステムをテストするには、cmgr を使用できます。このテストでは、指定したファイルシステムが正しく設定されているかどうか、およびファイルシステムを存在させるボリュームが正しく設定されているかどうかをチェックします。

ファイルシステムをテストするには、次のコマンドを使用します。

```
test resource resourcename of resource_type filesystems on cluster C_name [on node node1 node node2 ...]
```

以下の例では、クラスタで定義されているファイルシステムを表示して、それらの 1 つをテストします。

```
cmgr> show resources of resource_type filesystem in cluster eagan
/disk4 type filesystem
/disk5 type filesystem
/disk2 type filesystem
/disk3 type filesystem
/disk1 type filesystem
cmgr> test resource /disk4 of resource_type filesystem in cluster eagan on node cm1
Success: *** testing node resources on node cm1 ***
Success: *** testing all filesystem resources on node cm1 ***
Success: successfully mounted filesystem: /disk4
Success: overall exit status:success, tests failed:0, total tests executed:1
```

以下の例では、destructive モードでファイルシステムをテストします。

```
cmgr> test resource /disk4 of resource_type filesystem in cluster eagan on node cm1 destructive
Warning: executing the tests in destructive mode
Success: *** testing node resources on node cm1 ***
Success: *** testing all filesystem resources on node cm1 ***
```

```
Success: successfully mounted filesystem: /disk4
Success: overall exit status:success, tests failed:0, total tests executed:1
```

cmgr を使ったリソース・グループのテスト

リソース・グループをテストするには、 cmgr を使用できます。このテストでは、リソース・グループに定義されているすべてのリソースに対して、リソース・テストを繰返し実行します。リソース・テストは、リソース・グループのアプリケーション・フェイルオーバー・ドメインのノードでのみ実行されます。

リソース・グループをテストするには、次のコマンドを使用します。

```
test resource_group RG_name in cluster C_name [on node node1 node node2 ...]
```

以下の例では、クラスタで定義されているリソース・グループを表示して、それらの 1 つをテストします。

```
cmgr> show resource_groups in cluster eagan
Resource Groups:
        nfs2
        informix
cmgr> test resource_group nfs2 in cluster eagan on machine cm1
Success: *** testing node resources on node cm1 ***
Success: testing resource /disk4 of resource type NFS on node cm1
Success: testing resource /disk3 of resource type NFS on node cm1
Success: testing resource /disk3/statmon of resource type statd on node cm1
Success: testing resource 128.162.19.45 of resource type IP_address on node cm1
Success: testing resource /disk4 of resource type filesystem on node cm1
Success: testing resource /disk3 of resource type filesystem on node cm1
Success: testing resource dmf1 of resource type volume on node cm1
Success: testing resource dmfjournals of resource type volume on node cm1
Success: overall exit status:success, tests failed:0, total tests executed:16
```

cmgr を使ったフェイルオーバー・ポリシーのテスト

フェイルオーバー・ポリシーが正しく定義されているかどうかをテストするには、cmgr を使用できます。このテストでは、ポリシー・スクリプト、フェイルオーバー属性、およびクラスタからの有効なノードでアプリケーション・フェイルオーバー・ドメインが構成されているかどうかを検証することによって、フェイルオーバー・ポリシーをチェックします。

フェイルオーバー・ポリシーをテストするには、次のコマンドを使用します。

```
test failover_policy FP_name in cluster C_name [on node node1 node node2 ...]
```

以下の例では、show コマンドを使用して、クラスタで定義されているフェイルオーバー・ポリシーを表示し、それらの 1 つをテストします。

```
cmgr> show failover_policies
Failover Policies:
        reverse
        ordered-in-order
cmgr> test failover_policy reverse in cluster eagan
Success: *** testing node resources on node cm1 ***
Success: testing policy reverse on node cm1
Success: *** testing node resources on node cm2 ***
Success: testing policy reverse on node cm2
Success: overall exit status:success, tests failed:0, total tests executed:2
```

システムの回復とトラブルシューティング

この章では、FailSafe システムの回復について説明します。この章には、以下のトピックに関する節が含まれます。

システムの回復の概要

FailSafe システムに問題があるときは、FailSafe の複数の機能とコマンドを使用して、問題のある箇所を判断できます。

FailSafe では、システム異常を評価して異常から回復するために以下のツールが用意されています。

- ログ・ファイル
- システム・コンポーネントのステータスをモニタするためのコマンド
- 高可用性サービスの開始、停止、およびフェイルオーバーを行うためのコマンド

FailSafe ログでは、FailSafe で問題を起こさないシステムの問題は検出されない場合があることに注意してください。たとえば、CPU に異常が発生した場合やハードウェアのメインテナンスが必要な場合などの異常は、FailSafe で検出できないことがあります。

一般的に、FailSafe 設定に関する性質の問題を評価するときは、問題を解決するためにノードをシャットダウンする必要があるかどうかを判断する必要があります。

ノードをシャットダウンするときは、以下の手順に従います。

1. ノードで FailSafe HA サービスを停止します。
2. ノードをシャットダウンして、必要なメインテナンスと修復を実行します。
3. ノードを開始します。
4. ノードで FailSafe HA サービスを開始します。

FailSafe がノードのシャットダウンをノード異常と解釈しないよう、可能であれば、ノードをシャットダウンする前に FailSafe サービスを明示的に停止することが重要です。FailSafe がサービスの中断をノード異常と解釈した場合、リソース・グループとアプリケーション・フェイルオーバー・ドメインの設定によっては、予期しない変更が行われる可能性があります。

ノードをシャットダウンしてメインテナンスを実行するときは、システムの動作を継続させるために、FailSafe 設定を変更しなければならない場合があります。

メインテナンスのためにリソース・グループをオフラインにする

ノードでメインテナンスを実行する場合など、リソースをオフラインにする必要があるときは、以下の手順に従います。

1. `offline_detach` オプションまたは `offline_detach_force` オプション（リソース・グループにエラーがある場合）を使用して、リソース・グループをオフラインにします。詳細については、266 ページの「リソース・グループの回復」および268 ページの「リソース・グループのメインテナンスとエラー回復」を参照してください。
2. 必要なメインテナンスを実行します。
3. ノードを再起動します。
4. リソース・グループをオンラインにします。

FailSafe ログ・ファイル

FailSafe では、各 FailSafe デーモンごとにシステム・ログが維持されています。維持するログのレベルに基づいて、システム・ログをカスタマイズできます。表9-1 に、メッセージのレベルを示します。

`cad`、 `cmond`、および `fs2d` のログの設定についての詳細は、57 ページの「システム・ファイルの設定」を参照してください。ログの設定についての詳細は、99 ページの 第5章「設定」の196 ページの「ログの設定」を参照してください。

表9-1 メッセージ・レベル

メッセージ・レベル	説明
通常	通常メッセージは、タスクが正常に完了したことを報告します。以下に、通常メッセージの例を示します(<N の表記が通常メッセージを示します)。 `Wed Sep 2 11:57:25.284 <N ha_gcd cms 10185:0> Delivering TOTAL membership (S# 1, GS# 1)`
エラー/警告	エラーまたは警告メッセージは、エラーが発生したか、まもなく発生する可能性があることを示します。これらのメッセージは、間違ったコマンドや不正な構文を使用したために発生することがあります。以下に、警告メッセージの例を示します(<W の表記が警告メッセージを示し、<E がエラーを示します)。 `Wed Sep 2 13:45:47.199 <W crsd crs 9908:0 crs_config.c:634> CI_ERR_NOTFOUND, safer - no such node`
SYSLOG	すべての通常メッセージとエラー・メッセージは、syslog にも記録されます。システム・ログ・メッセージには、クラスタ関連のメッセージであることを示す <CI> という記号がヘッダに含まれます。以下に、システム・ログ・メッセージの例を示します。 `Wed Sep 2 12:22:57 6X:safe syslog: <<CI> ha_cmsd misc 10435:0> CI_FAILURE, I am not part of the enabled cluster anymore`
デバッグ	デバッグ・メッセージは、ログ・レベルが debug0 以上(GUI 使用時)または 10 以上(cmgr 使用時)に設定されている場合に、ログ・グループ・ファイルに記録されます。 debug0(メッセージ内の D0 を参照してください)またはログ・レベル 10 では、次のメッセージがログに記録されます。 `Thu Sep 27 14:43:24.233 <D0 ha_fsd fsd 57540:0 fs_failsafe.c:1471> Determine oldest sta`

ログ・ファイルを調べると、システム・エラーの性質を判断することができます。エラーの発生時刻に注意したり、ログ・ファイルを参照してエラー発生直前の複数のデーモンのアクティビティを監視すると、異常の原因となった状況を判断できることがあります。

メモ:ログ設定でデバッグ・レベルを使用すると、サーバで大量のディスク容量が使用される場合があります。

FailSafe メンバーシップとリセット

異常が発生した箇所やプロセスがどのように転送されたかを判断するために、異常発生時の FailSafe システムのアクションを調査するときは、FailSafe メンバーシップの概念を考慮に入れることが重要です。フェイルオーバーが発生した場合にランタイム・フェイルオーバー・ドメインに含めることができるのは、FailSafe メンバーシップに存在するノードだけです。

FailSafe メッセージとタイブレーカー・ノード

ノードが FailSafe メンバーシップに加わることができるのは、ノードが無効になっておらず、既知の状態である場合だけです。共有ストレージには FailSafe メンバーシップ内のノードしかアクセスできないので、これによってデータの整合性が保たれることになります。FailSafe によって制御されていないメンバーシップ外のノードが共有ストレージにアクセスできると、2 つのノードが同じデータに同時にアクセスする可能性があり、結果的にデータが破壊されてしまいます。このような理由から、無効になっているノードはメンバーシップの計算に含まれません。

メモ: FailSafe メンバーシップを確定する前に無効に設定されていたノードがリセットされることはありません。

クラスタの FailSafe メンバーシップは、上限量の過半数に基づきます。クラスタが有効になるには、クラスタ内の 50% を超えるノードが既知の状態で、ハートビート・コントロール・ネットワークを使用して相互に通信できなければなりません。この上限量によって、形成される FailSafe メンバーシップの一部になるノードが決まります。

クラスタ内のノードの数が偶数の場合は、過半数の上限量が存在しない可能性があります。つまり、それぞれがノードの総数の 50% で構成され、他方のノードのセットと通信できない 2 つのセットのノードができてしまうことがあります。この場合、FailSafe は、FailSafe パラメータの設定時にタイブレーカー・ノードとして設定したノードを使用します。タイブレーカー・ノードが設定されていない場合は、HA サービスが開始されている、ノード ID 番号が最も小さいノードを使用します。

上限量に含まれていないノードは、上限量に含まれるノードによってリセットされます。リセットできるノードはメンバーシップ内で「ダウン (DOWN)」と宣言され、リセットできないノードは「不明 (UNKNOWN)」と宣言されます。上限量に含まれるノードは「実行中 (UP)」です。

新しい過半数の上限量が計算された場合は、ノードがリセット可能かどうかに関係なく、新しいメンバーシップが宣言されます。

現在の上限量の少なくとも 1 つのノードが現在のメンバーシップを持っている場合、これらのノードは、少なくとも 1 つのノードをリセットできれば、新しいメンバーシップを宣言します。

新しい同数の上限量に含まれるすべてのノードが初めて開始された場合は、上限量にタイブレーカー・ノードが含まれているときにかぎり、これらのノードはリセットを実行して、新しいメンバーシップで処理を続けます。

クラスタ内のノードの同数のサブセットが以前のメンバーシップを持っていない場合は、タイブレーカー・ノードがある方のクラスタのノードのサブセットが、クラスタ内のノードのもう一方のサブセットに含まれるノードをリセットします。少なくとも 1 つのノードのリセットが成功すると、新しいメンバーシップが確立されます。

クラスタ内のノードの同数のサブセットが以前のメンバーシップを持っている場合は、クラスタ内のノードの一方のサブセットが、クラスタ内のノードのもう一方のサブセットに含まれるノードをリセットします。少なくとも 1 つのノードのリセットが成功すると、新しいメンバーシップが確立されます。タイブレーカー・ノードがある方のクラスタ内のノードのサブセットは直ちにリセットされ、クラスタ内のノードのもう一方のサブセットはしばらくしてからリセットされます。

リセットは、tty ポートに接続されたシステム・コントローラを通じてシリアル回線経由で実行されます。シリアル回線の定期的なモニタが停止されることはありません。予想されるシリアル回線モニタ異常の周期と、予想されるハートビート切断の周期が重なっている場合は、リセット対象のノードの電源異常が原因の可能性があります。

メンバーシップが形成されない

FailSafe メンバーシップが形成されないときは、以下の部分に問題がないかどうかをチェックしてください。

- `ha_cmsd` FailSafe メンバーシップ・デーモンが実行されているかどうか。`fs2d` データベース・デーモンが実行されているかどうか。
- ノードが相互に通信可能かどうか。コントロール・ネットワークがハートビート・ネットワークとして設定されているかどうか。
- ピア・ノードから発行する `ping`(1M) コマンドでコントロール・ネットワーク・アドレスと通信できるかどうか。
- 上限量の過半数または同数ルールが満たされているかどうか。メンバーシップのステータスを判断するには、`cmsd` ログを参照してください。
- リセットが必要な場合は、以下の条件が満たされているかどうか。
  - `crsd` ノード・コントロール・デーモンが実行されているかどうか。
  - リセット・シリアル回線が正常な状態かどうか。

  リセット・シリアル回線の状態は、関係のあるノードの `crsd` ログを参照するか、またはノードに対して `admin ping` および `admin reset` コマンドを実行することでチェックできます。

ステータスのモニタ

FailSafe では、指定したクラスタ、ノード、リソース、およびリソース・グループのステータスをモニタおよびチェックできます。この機能を使用して、システムの問題がある箇所を特定できます。

GUI ツリー表示では、コンポーネントの表示機能を使用して、FailSafe コンポーネントのステータスを継続的にモニタできます。 `cmgr` コマンドでは、`show` コマンドを使用して、個々のコンポーネントのステータスを表示できます。

ステータスのモニタと FailSafe コンポーネントの状態の意味についての詳細は、第7章「IRIS FailSafe のシステム運用」の223 ページの「システムのステータス」を参照してください。

FailSafe サービスの動的な制御

FailSafe では、問題のあるシステムのトラブルシューティングに役立つさまざまな管理タスクを、システム全体を停止することなく実行できます。これらのタスクには、以下の機能が含まれます。

- クラスタで実行されている FailSafe サービスやアプリケーションに影響を与えずに、クラスタに対してノードの追加や削除を行うことができます。
- その他のオンライン・リソース・グループには影響を与えずに、リソース・グループを追加または削除できます。
- リソース・グループをオンラインにしたまま、リソース・グループに対してリソースの追加や削除を行うことができます。
- サービスを実行したままハートビート周期やノード・タイムアウトなどの FailSafe パラメータを変更して、変更した値を直ちに反映させることができます。
- 指定したノードで FailSafe サービスを開始または停止できます。
- リソース・グループをオンラインまたはオフラインにできます。
- リソース・グループをメインテナンス・モードにすることで、リソース・グループのモニタを停止できます。この操作はリソース・グループを停止または開始するのではなく、単に FailSafe で使用できない状態にするだけなので、影響は大きくありません。
- 個々のノードをリセットできます。

これらのタスクの実行方法についての詳細は、99 ページの 第5章「設定」および第7章「IRIS FailSafe のシステム運用」を参照してください。

回復手順

以下の節では、さまざまな Failsafe コンポーネントに異常が発生した場合に実行できる回復手順を説明します。ここでは、以下のような場合の手順が説明されています。

単一のノードの回復

2 ノード・クラスタの一方のノードがメインテナンスのために意図的に停止されていたり、実行できない場合は、そのノードが停止していることが正常ノードのデータベースによって認識され、そのノードがフェイルオーバー・ドメインにあると見なされないように、一定の手順に従う必要があります。これらの手順がないと、正常ノードは「孤立状態 (lonely state)」と呼ぶ状態になり、単独でクラスタを形成できなくなります。

217 ページの「2 ノード・クラスタ: 単一のノードの使用」の手順を参照してください。

クラスタ・エラーの回復

クラスタ内のすべてのノードでクラスタのステータスが「不明 (UNKNOWN)」の場合は、以下の手順に従います。

1. 異常が発生したコントロール・ネットワークがあるかどうかを確認します（271 ページの「コントロール・ネットワークの異常の回復」を参照してください）。
2. コントロール・ネットワークを使用して相互に通信し、上限量を形成できる十分なノードがクラスタ内にあるかどうかを判断します（少なくともノードの 50% が相互に通信できる必要があります）。十分な数のノードがある場合は、force オプションを使用して、通信できないノードの HA サービスを停止します。これにより、上限量の計算に使用されるノードの数が変更されます。
3. ハードウェア設定に問題がない場合は、以下を実行します。
   - クラスタ内でオンラインのリソース・グループがある場合は、すべて分離します。
   - クラスタの HA サービスを停止します。
   - クラスタの HA サービスを再開します。

   266 ページの「リソース・グループの回復」を参照してください。

たとえば、次の cmgr コマンドを実行すると、クラスタ web-cluster のリソース・グループ web-rg が分離されます。

```
cmgr> admin detach resource_group web-rg in cluster web-cluster
```

クラスタ web-cluster の HA サービスを停止して、エラーを無視する（force オプション）場合は、次のコマンドを使用します。

```
cmgr> stop ha_services for cluster web-cluster force
```

クラスタ web-cluster の HA サービスを開始するには、次のコマンドを使用します。

```
cmgr> start ha_services for cluster web-cluster
```

リソース・グループの回復

リソース・グループがエラー状態であっても、そのリソース・グループのすべてのリソースに異常があるわけではありません。ただし、リソースを「オンライン (online)」状態に戻すには、まずそれらのリソースを「オフライン (offline)」状態に設定する必要があります。次の cmgr コマンドを使用すると、実際にリソースをオフラインにせずに操作できます。

```
admin offline_detach_force resource_group [in cluster clustername]
```

例は、次のとおりです。

```
cmgr> admin offline_detach_force RG1 in cluster test-cluster
```

注意: このコマンドを使用する場合は、InPlace_Recovery フェイルオーバー・ポリシー属性を使用してください。この属性により、リソースは、offline_detach_force の実行時に実行されていたノードと同じノードにとどまるように指定されます。

ノード・エラーの回復

ノードがクラスタ内の過半数のノードと通信できないときは、CMSD が「孤立状態 (lonely state)」であるというメッセージが SYSLOG に表示されます。また、ノードがリセットされたり不明な状態になるなどの別の問題が発生する場合もあります。

ノード・エラーを解決するには、以下の手順に従います。

1. ノードのコントロール・ネットワークが動作しているかどうかをチェックします（271 ページの「コントロール・ネットワークの異常の回復」を参照してください）。
2. ノードをリセットするためのシリアル・リセット・ケーブルが動作しているかどうかをチェックします（272 ページの「シリアル・ケーブルの異常の回復」を参照してください）。
3. sgi-cmsd のポートがクラスタ内のすべてのノードで同じであることを確認します。
4. ノード設定をチェックします。ノード設定は、統一されていて正確でなければなりません。
5. syslog および cmsd ログでエラーをチェックします。ノードがクラスタに設定されない場合は、クラスタの一部であるノードのログをチェックしてください。
6. ハードウェア設定に問題がない場合は、ノードの HA サービスを停止して再開します。

   たとえば、クラスタ web-cluster のノード web-node3 の HA サービスを停止して、エラーを無視する（force オプション）には、次のコマンドを使用します。

```
cmgr> stop ha_services in node web-node3 for cluster web-cluster force
```

   たとえば、クラスタ web-cluster のノード web-node3 のHA サービスを開始するには、次のコマンドを使用します。

```
cmgr> start ha_services in node web-node3 for cluster web-cluster
```

リソース・グループのメインテナンスとエラー回復

リソース・グループの一部であるアプリケーションに対して簡単なメインテナンスを実行するには、以下の手順に従います。この手順では、メインテナンス・モードをオンにしたときに、リソース・グループのリソースのモニタが停止されます。アプリケーションのメインテナンスが終了したら、メインテナンス・モードをオフにする必要があります。

注意: リソース・グループのメインテナンスを実行中のノードでノード異常が発生した場合、リソース・グループはフェイルオーバー・ポリシー・ドメインの別のノードに移動されます。

例は、次のとおりです。

1. リソース・グループ web-rg をメインテナンス・モードにするには、次の cmgr コマンドを使用します。

```
cmgr> admin maintenance_on resource_group web-rg in cluster web-cluster
```

2. リソース・グループの状態が「オンライン (ONLINE_MAINTENANCE)」に戻ります。必要なアプリケーション・メインテナンスを行います (簡単なアプリケーション・メインテナンスの例としては、アプリケーション・ログのローテーションなどが挙げられます)。

3. リソース・グループ web-rg のメインテナンス・モードを解除するには、次のコマンドを使用します。

```
cmgr> admin maintenance_off resource_group web-rg in cluster web-cluster
```

リソース・グループの状態が「オンライン (ONLINE)」に戻ります。

リソース・グループの状態が「オンライン (ONLINE)」で「SRMD 実行可能フェイル・エラー (SRMD EXECUTABLE ERROR)」がある場合は、以下の手順を実行します。

1. SRM ログ (デフォルトの場所: /var/cluster/ha/logs/srmd_*nodename*) を参照して、異常の原因と異常が発生したリソースを判断します。SRMD ログ・ファイルで ERROR の文字列を検索します。

```
Wed Nov 3 04:20:10.135
<E ha_srmd srm 12127:1 sa_process_tasks.c:627>
CI_FAILURE, ERROR: Action (start) for resource (192.0.2.45) of type
(IP_address) failed with status (failed)
```

2. 同じノードのスクリプト・ログで、IP_address 開始スクリプト・エラーがないかどうかをチェックします。

3. 異常の原因を解決します。この作業では、リソース設定の変更や、リソース・タイプの停止/開始/フェイルオーバー・アクション・タイムアウトの変更が必要になる場合があります。

4. 問題を解決したら、force オプションを使用してリソース・グループをオフラインにしてから、クラスタでオンラインにします。

たとえば、次のコマンドを実行すると、クラスタ web-cluster のリソース・グループ web-rg がオフラインになり、エラーは無視されます。

```
cmgr> admin offline resource_group web-rg in cluster web-cluster force
```

次のコマンドを実行すると、クラスタ web-cluster のリソース・グループ web-rg がオンラインになります。

```
cmgr> admin online resource_group web-rg in cluster web-cluster
```

リソース・グループ web-rg は「オンライン (ONLINE)」状態になり、エラーは発生しません。

リソース・グループがオンラインではなくエラー状態の場合は、以下の手順に従います。このようなエラーの大部分は、排他プロセスが原因で発生します。このプロセスは、リソース・グループをオンラインにするときに実行され、リソースがリソース・グループの異常ドメインのどこかにすでに割当てられているかどうかを判断します。排他スクリプトが失敗した場合は、リソースがノードに割当てられているという結果が返されるので注意してください。つまり、リソースが存在しないことが確認できる場合以外は、リソースが割当てられていることを示す値が返されます。

異常ノードで発生する可能性のあるエラー状態には、「分割リソース・グループ (排他) (SPLIT RESOURCE GROUP (EXCLUSIVITY))」、「ノード使用不可 (排他) (NODE NOT AVAILABLE (EXCLUSIVITY))」、「使用可能なノードなし (NO AVAILABLE NODES)」などがあります。リソース・グループのエラー・コードについては、228 ページの「リソース・グループのステータス」を参照してください。

1. failsafe ログおよび SRMD ログ (デフォルトのディレクトリ: /var/cluster/ha/logs、ファイル: failsafe_*nodename*、 srmd_*nodename*) を参照して、異常の原因と異常が発生したリソースを判断します。

   たとえば、リソース・グループをオンラインにするタスクを実行した結果、リソース・グループのエラー状態が「分割リソース・グループ (排他) (SPLIT RESOURCE GROUP (EXCLUSIVITY))」になったとします。これは、リソース・グループの一部が少なくとも 2 つの異なるノードに割当てられていることを意味します。以下の例に示すように、リソース・グループが部分的に割当てられていると考えられるノードは、failsafe ログの 1 つに記録されています。

```
[Resource Group:RG_name]:Exclusivity failed -- RUNNING on node1 and node2

[Resource Group:RG_name]:Exclusivity failed -- PARTIALLY RUNNING on node1 and PARTIALLY RUNNING on node2
```

   ここで、該当する各ノードの srmd ログで排他スクリプト・エラーを検索して、割当てられていると考えられるリソースを確認します。場合によっては、誤って設定されたリソースが、割当てられているリソースであることが判明することもあります。これは、特に Netscape_web リソースの場合に当てはまります。

2. 異常の原因を解決します。この作業では、リソース設定の変更や、リソース・タイプの開始/停止/排他タイムアウトの変更が必要になる場合があります。

3. 問題を解決したら、force オプションを使用してリソース・グループをオフラインにしてからオンラインにします。

リソース・グループに「AFD のノード不足 (no more nodes in AFD)」エラーがあると表示される場合は、以下のチェックを実行します。

1. フェイルオーバー・ドメインの一部のノードがメンバーシップに含まれていません。CMSD ログでエラーをチェックします。
2. フェイルオーバー・ドメインのすべてのノードの SRMC/スクリプト・ログで、開始／モニタ・スクリプト・エラーがないかどうかをチェックします。

クラスタでは複数の二重の異常が発生する可能性があります。この場合、リソース・グループは非高可用性の状態のままになります。場合によっては、リソース・グループがオフライン状態のままになってしまうことがあります。また、新しいノードがクラスタに設定され、そのノードにリソース・グループを移行してエラーをクリアできる場合でも、特定のノードでリソース・グループがエラー状態のままになることがあります。これらの状況が発生した場合は、以下を実行します。

1. リソース・グループがオフラインの場合は、オンラインにします。
2. 特定のノードでリソース・グループの状態が変わらない場合は、そのリソース・グループを分離してから、再度オンラインにします。多くのエラーはこの方法でクリアできます。
3. リソース・グループを分離しても効果がない場合は、リソース・グループを強制的にオフラインにしてからオンラインに戻します。
4. コマンドがハングしたり、正常に動作していないように見える場合は、すべてのリソース・グループを分離してクラスタをシャットダウンしてから、オンラインに戻します。

リソース・グループを分離したり、リソース・グループを強制的にオフラインにする場合についての詳細は、241 ページの「リソース・グループをオフラインにする」を参照してください。

リソース・エラー状態のクリア

この手順は、リソース・グループの一部ではないリソースが、「オンライン (ONLINE)」状態でエラーがある場合に使用します。この状態は、リソース・グループへのリソースの追加や削除に失敗した場合に発生することがあります。

以下を実行します。

1. SRM ログを参照して、異常の原因と異常が発生したリソースを判断します。以下に、デフォルトの場所を示します。

   /var/cluster/ha/logs/srmd_*nodename*

2. 異常の原因である問題を解決します。この作業では、リソース設定の変更や、リソース・タイプの停止/開始/フェイルオーバー・アクション・タイムアウトの変更が必要になる場合があります。

3. GUI または cmgr コマンドを使用して、エラー状態をクリアします。

   - 「リソース・エラー状態のクリア(Clear Resource Error State)」GUI タスクを使用します。次の情報を入力します。

     – 「リソース・タイプ (Resource Type)」: リソースのタイプを選択します。

     – 「エラー状態のリソース(Resource in Error State)」:エラー状態からクリアする必要があるリソースの名前を選択します。

     「OK」をクリックして、タスクを完了します。

   - cmgr admin offline_force コマンドを使用して、リソースをオフラインにします。たとえば、Netscape_Web タイプのリソース web-srvr のエラー状態を解消して、リソース・グループに追加できるようにするには、次を入力します。

```
cmgr> admin offline_force resource web-srvr of resource_type Netscape_Web in cluster web-cluster
```

コントロール・ネットワークの異常の回復

コントロール・ネットワークの異常は、cmsd ログに報告されます。 cmsd ログのデフォルトの場所は、/var/cluster/ha/logs/cmsd_*nodename* です。コントロール・ネットワークに異常が発生したときは、以下の手順に従います。

1. ping(1M) コマンドを使用して、ノードにコントロール・ネットワークの IP アドレスが設定されているかどうかをチェックします。

2. ノード設定をチェックして、コントロール・ネットワークの IP アドレスが正しく指定されているかどうかを確認します。

   次の cluster_mgr コマンドを実行すると、web-node3: のノード設定が表示されます。

   ```
   cmgr> show node web-node3
   ```

3. コントロール・ネットワークに対して、*XX.XX.XX.XX* という表記法の IP アドレスではなく IP 名が指定されている場合は、DNS を使用して IP 名を解決できるかどうかを確認します。IP 名ではなく IP アドレスを使用することをお薦めします。

4. クラスタに対してハートビート周期とノード・タイムアウトが正しく設定されているかどうかをチェックします。これらの HA パラメータは、cluster_mgr show ha_parameters コマンドを使用して参照できます。

シリアル・ケーブルの異常の回復

シリアル・ケーブルは、ノード異常が発生したときにノードをリセットするために使用されます。シリアル・ケーブルの異常は、crsd ログに報告されます。crsd ログのデフォルトの場所は、/var/cluster/ha/log/crsd_*nodename* です。

ノード設定をチェックして、シリアル・ケーブル接続が正しく設定されているかどうかを確認します。

次の cmgr コマンドを実行すると、web-node3 のノード設定が表示されます。

```
cmgr> show node web-node3
```

admin ping コマンドを使用して、シリアル・ケーブルを確認します。次のコマンドを実行すると、ノード web-node3 のシリアル・ケーブルの問題が報告されます。

```
cmgr> admin ping node web-node3
```

クラスタ・データベース Sync 異常

クラスタ・データベースの同期で異常が発生する場合は、以下の手順に従います。

1. ターゲット・ノードの /var/adm/SYSLOG ファイルで次のメッセージをチェックします。

   ```
   Starting to receive CDB sync series from machine <node1_node_ID>
   ...
   Finished receiving CDB sync series from machine <node1_node_ID>
   ```

2. コントロール・ネットワークまたは portmapper/rpcbind に問題がないかどうかをチェックします。
3. クラスタ・データベースのノード定義をチェックします。
4. ソース・ノードの SYSLOG および fs2d ログをチェックします。

クラスタ・データベースのメインテナンスと回復

クラスタ・データベース全体を初期化する必要があるときは、次のコマンドを実行します。

```
# /usr/cluster/bin/cdbreinit /var/cluster/cdb/cdb.db
```

このコマンドを実行すると、すべてのクラスタ・プロセスが再起動されます。プール内にほかのノードがある場合、クラスタ・データベースの内容はその他のノードと自動的に同期されます。

プール内にほかのノードがない場合は、ここでクラスタ・データベースをバックアップから復元する必要があります。クラスタ・データベースのバックアップと復元についての詳細は、246 ページの「cmgr を使った設定のバックアップと復元」を参照してください。

GUI が実行されない

GUI が実行されない場合は、以下をチェックします。

- クラスタ・デーモンが実行されているかどうか。

  ソフトウェアを初めてインストールすると、以下のデーモンが実行されます。

  - fs2d
  - cmond
  - cad
  - crsd

  実行されているデーモンを判断するには、次を入力します。

  # **ps -ef | grep cluster**

  以下に、初期デーモンだけが実行されている場合の出力の例を示します。ここでは、見やすいように空白を削除してデーモンを強調表示してあります。

```
fs6 # ps -ef | grep cluster
root 31431     1 0 12:51:36 ?     0:14 /usr/lib32/cluster/cbe/fs2d /var/cluster/cdb/cdb.db #
root 31456 31478 0 12:53:01 ?     0:03 /usr/cluster/bin/crsd -l
root 31475 31478 0 12:53:00 ?     0:08 /usr/cluster/bin/cad -l -lf /var/cluster/ha/log/cad_log --append_log
root 31478     1 0 12:53:00 ?     0:00 /usr/cluster/bin/cmond -L info -f /var/cluster/ha/log/cmond_log
root 31570 31408 0 14:01:52 pts/0 0:00 grep cluster
```

  これらのプロセスが表示されない場合は、ログを参照して、考えられる問題を確認してください。デーモンを再開する必要がある場合は、次を入力します。

  # **/etc/init.d/cluster start**

- tcpmux サービスおよび tcpmux/sgi_sysadm サービスが /etc/inetd.conf ファイル内で有効になっているかどうか。

  sysamd_base がインストールされると、次の行が /etc/inetd.conf に追加されます。

```
tcpmux/sgi_sysadm stream tcp nowait root   ?/usr/sysadm/bin/sysadmd sysadmd
```

  tcpmux 行がコメント・アウトされている場合は、コメントを外した後、次を実行しなければなりません。

  # **kill -HUP inetd**

- inetd ラッパまたは tcp ラッパが干渉しているかどうか。これは、connection refused メッセージまたは login failed メッセージによって示される場合があります。

- IRIX ノードに接続しているかどうか。fstask(1M) コマンドは、IRIX ノードでのみ実行できます。GUI または IRIX ノードに接続している場合、IRIX 以外のオペレーティング・システムが実行されているノードから Web を介して GUI を実行できます。

GUI と cmgr の不整合

GUI に FailSafe cmgr コマンドと異なる情報が表示される場合は、次のコマンドを実行して、GUI の接続先のノードで cad プロセスを再開します。

```
# killall cad
```

クラスタ管理デーモンは、cmond プロセスによって自動的に再開されます。

GUI で情報が報告されない

GUI で設定情報とステータスが報告されない場合は、以下の手順を実行します。

1. cmgr(1M) コマンドを使用して情報をチェックします。cmgr で正しい情報が報告される場合は、GUI の更新に問題があります。
2. GUI の更新に問題がある場合は、該当するノードの cad デーモンを強制終了します。数分間待ってから、cad で正しい情報が収集されるかどうかを確認します。ノードの cad ログにエラーがないかどうかをチェックします。
3. ノードの CLI ログにエラーがないかどうかをチェックします。
4. ステータス情報が間違っている場合は、ノードの cmsd または fsd ログをチェックします。

cdbreinit コマンドの使用

クラスタ内の一部のノードのクラスタ・データベースが同期されていない場合は、cdbreinit コマンドを実行して回復できます。cdbreinit コマンドは、同期されていないノードで実行してください。

以下の手順を実行します。

メモ: この回復手順では、すべてのノードで各手順を実行してから、次の手順に進んでください。

---

1. GUI または cmgr コマンドを使用して、クラスタの FailSafe サービスを停止します。

2. プール内のすべてのノードでクラスタ・プロセスを停止します。

```
# /etc/init.d/cluster stop
# killall fs2d
```

3. クラスタ・データベースが同期されていないノードで cdbreinit を実行します。

4. プール内のすべてのノードでクラスタ・プロセスを開始します。

```
# /etc/init.d/cluster start
```

5. クラスタ・データベースが同期されるまで数分間待ちます。クラスタ・データベース同期ログ・メッセージは、ノードの SYSLOG にあります。

6. クラスタの FailSafe サービスを開始します。

CXFS メタデータ・サーバの再設定

FailSafe では、-k オプションと併せて umount(1M) コマンドを使用し、CXFS リソース定義の relocate-mds 属性が true に設定されているときに CXFS メタデータ・サーバを再設定する場合に、リソースを移動します。umount -k コマンドは、CXFS ファイルシステムを使用してすべてのサーバのプロセスを強制終了します。

CXFS 同時実行に関するその他の問題

CXFS との同時実行に関する問題の解決方法についての詳細は、『CXFS Version 2 Software Installation and Administration Guide』のトラブルシューティングの章を参照してください。

運用中のクラスタのアップグレードとメインテナンス

IRIS FailSafe システムの実行中に、クラスタ全体をシャットダウンせずにさまざまな管理手順を実行しなければならないことがあります。この章では、運用中のクラスタでのアップグレードおよびメインテナンスの方法について説明します。この章で説明する手順は以下のとおりです。

運用中のクラスタへのノードの追加

運用中のクラスタにノードを追加するには、以下の手順に従います。この手順では、`cluster_admin`、`cluster_control`、`cluster_ha`、および `failsafe2` 製品がすでにこのノードにインストールされていることが前提となります。

1. `ping`(1M) コマンドを使用して、ノードからクラスタのその他の部分へのコントロール・ネットワークの接続を確認します。コントロール・ネットワークの IP アドレスのリストを記録します。
2. シリアル接続を確認して、このノードをリセットします。このノードをリセットできるノードの名前を記録します。
3. ノードの診断を実行します。FailSafe の診断コマンドについての詳細は、249 ページの 第8章「設定のテスト」を参照してください。
4. `/etc/services` ファイルに、`sgi-cad`、 `sgi-crsd`、`sgi-cmsd`、および `sgi-gcd` のエントリが存在することを確認します。これらのプロセスのポート番号は、クラスタ内のほかのノードのポート番号と一致している必要があります。

エントリの例

```
sgi-cad         7200/tcp      # SGI cluster admin daemon
sgi-crsd        7500/udp      # SGI cluster reset services daemon
sgi-cmsd        7000/udp      # SGI FailSafe membership Daemon
sgi-gcd         8000/udp      # SGI group communication Daemon
```

5. クラスタ・プロセス(cad、cmond、および crsd)が実行されているかどうかを確認します。

   # **ps -ef | grep cad**

   クラスタ・プロセスが実行されていない場合は、cdbreinit コマンドを実行します。

```
# /usr/cluster/bin/cdbreinit /var/cluster/cdb/cdb.db
 Killing fs2d...
 Removing database header file /var/cluster/cdb/cdb.db...
 Preparing to delete database directory /var/cluster/cdb/cdb.db# !!
 Continue[y/n]y
 Removing database directory /var/cluster/cdb/cdb.db#...
 Deleted CDB database at /var/cluster/cdb/cdb.db
 Recreating new CDB database at /var/cluster/cdb/cdb.db with cdb-exitop...
  fs2d
  Created standard CDB database in /var/cluster/cdb/cdb.db

  Please make sure that "sgi-cad" service is added to /etc/services file
  If not, add the entry and restart cluster processes.
  Please refer to IRIS FailSafe administration manual for more
  information.

  Modifying CDB database at /var/cluster/cdb/cdb.db with cluster_ha-exitop...
  Modified standard CDB database in /var/cluster/cdb/cdb.db

  Please make sure that "sgi-cmsd" and "sgi-gcd" services are added
  to /etc/services file before starting HA services.
  Please refer to IRIS FailSafe administration manual for more
  information.

  Starting cluster control processes with cluster_control-exitop...

  Please make sure that "sgi-crsd" service is added to /etc/services file
  If not, add the entry and restart cluster processes.
  Please refer to IRIS FailSafe administration manual for more
  information.
```

```
Started cluster control processes
Restarting cluster admin processes with failsafe-exitop...
```

6. GUI 、 cmgr コ マ ン ド 、 ま た は cmgr テ ン プ レ ー ト (/var/cluster/cmgr-templates/cmgr-create-node)を使用して、ノードを定義します。

   メモ:このノードは、すでにクラスタ内に存在するノードの 1 つから定義してください。

7. cmgr コマンドまたは GUI を使用して、ノードをクラスタに追加します。

   例は、次のとおりです。以下の cmgr コマンドを実行して、ノード web-node3 をクラスタ web-cluster に追加します。

   ```
   cmgr> modify cluster web-cluster
   Enter commands, when finished enter either "done" or "cancel"

   web-cluster ? add node web-node3
   web-cluster ? done
   ```

8. cmgr コマンドまたは GUI を使用して、このノードで高可用性 (HA) サービスを開始します。

   たとえば、次の cmgr コマンドを実行すると、クラスタ web-cluster 内のノード web-node3 で HA サービスが開始されます。

   ```
   cmgr> start ha_services on node web-node3 in cluster web-cluster
   ```

9. このノードを、関連のあるフェイルオーバー・ポリシーの異常ドメインに追加します。このためには、異常ドメイン内の追加ノードを含むフェイルオーバー・ポリシー全体を再定義する必要があります。

運用中のクラスタからのノードの削除

運用中のクラスタからノードを削除するには、以下の手順に従います。この手順では、ノード・ステータスが UP であることが前提となります。

1. ノードでリソース・グループがオンラインである場合は、cmgr コマンドまたは GUI を使用して、リソース・グループをクラスタ内の別のノードに移動します。

   リソース・グループをクラスタ内の別のノードに移動するには、そのリソース・グループのフェイルオーバー・ポリシー・ドメイン内に、使用可能な別のノードが存在する必要があります。リソース・グループを同じノードで実行したままにする場合は、cmgr コマンドまたは GUI を使用して、リソース・グループを分離します。

たとえば、次のコマンドを実行すると、リソース・グループ web-rg は、クラスタ web-cluster 内の同じノードで実行されたままになります。

```
cmgr> admin detach resource_group web-rg in cluster web-cluster
```

2. このノードを使用するフェイルオーバー・ポリシーの異常ドメインのノードを削除します。このためには、フェイルオーバー・ポリシー全体を再定義して、影響を受けるノードを異常ドメインから削除してください。

3. ノードで HA サービスを停止します。

   たとえば、ノード web-node3 上の HA サービスを停止するには、次の cmgr コマンドを使用します。このコマンドを実行すると、可能であれば、このノード上のオンラインのリソース・グループがすべてクラスタ内の別のノードに移動されます。

   ```
   cmgr> stop ha_services on node web-node3 for cluster web-cluster
   ```

   このコマンドは、ノード web-node3 上のオンラインのリソース・グループを移動できない場合は失敗します。 force オプションを使用すると、エラーが発生した場合でもノードの HA サービスを停止できます。オフラインにできないリソースや正常に割当て解除できないリソースがある場合に、強制オフライン・コマンドを実行すると、副次的な効果として、このようなリソースはノード上に割当てられたままになります。

クラスタ・データベースからノードを削除しなければならない場合は、以下の手順 4、5、6、および 7 を実行してください。

4. クラスタからノードを削除します。

   たとえば、ノード web-node3 を web-cluster 設定から削除するには、次の cmgr コマンドを使用します。

   ```
   cmgr> modify cluster web-cluster
   Enter commands, when finished enter either "done" or "cancel"

   web-cluster ? remove node web-node3
   web-cluster ? done
   ```

5. クラスタ・データベースからノード設定を削除します。

   次の cmgr コマンドを実行すると、クラスタ・データベースから web-node3 ノード定義が削除されます。

   ```
   cmgr> delete node web-node3
   ```

6. すべてのクラスタ・プロセスを停止して、クラスタ・データベースを削除します。

以下のコマンドを実行すると、ノード上のクラスタ・プロセスが停止されて、クラスタ・データベースが削除されます。

```
# /etc/init.d/cluster stop
# killall fs2d
# cdbdelete /var/cluster/cdb/cdb.db
```

7. ノードの起動時に最初からクラスタおよび HA プロセスを無効にします。これらのタスクには以下のコマンドを使用します。

```
# chkconfig cluster off
# chkconfig failsafe2 off
```

クラスタ内のコントロール・ネットワークの変更

現在運用中のクラスタ内のコントロール・ネットワークを変更するには、以下の手順に従います。この手順を使用できるのは、ノード node1 および node2 で構成される 2 ノード・クラスタの場合だけです。ここでは、各手順を完了してから次の手順に進んでください。

メモ: この手順の実行中は、その他の管理操作を行わないでください。

1. 任意のノードからクラスタの HA サービスを停止します。必ず両方のノードのすべての HA プロセスを終了してください。
2. 以下のコマンドを実行して、node2 から node2 上のクラスタ・プロセスを停止します。

```
# /etc/init.d/cluster stop
# killall fs2d
```

node2 上の fs2d が強制終了されていることを確認します。

3. node1 から、node1 および node2 の定義を変更します。GUI または以下の cmgr コマンドを使用します。

```
cmgr> modify node node1
Enter commands, when finished enter either "done" or "cancel"
node1?> remove nic old_nic _address
node1> add nic new_nic_address
NIC - new_nic_address set heartbeat to ...
NIC - new_nic_address set ctrl_msgs to ...
NIC - new_nic_address set priority to ...
NIC - new_nic_address done
node1? done
```

同じ手順を繰返して、node2 を変更します。

4. node1 から、node1 および node2 の定義が正しいかどうかを確認します。ノード定義を表示するには、node1 で cmgr を使用して以下のコマンドを実行します。

   ```
   cmgr> show node node1
   cmgr> show node node2
   ```

5. node1 と node2 の両方で、 /etc/config/netif.options のネットワーク・インタフェース IP アドレスを変更します。続いて、ifconfig を実行して、node1 および node2 で新規 IP アドレスを設定します。設定した IP アドレスがクラスタ・データベースのノード定義と一致することを確認します。

6. 以下のコマンドを実行して、node1 から node1 上のクラスタ・プロセスを停止します。

   ```
   # /etc/init.d/cluster stop
   # killall fs2d
   ```

   node1 上の fs2d が強制終了されていることを確認します。

7. node2 から、次のコマンドを実行して node2 上でクラスタ・プロセスを開始します。

   ```
   # /usr/cluster/bin/cdbreinit /var/cluster/cdb/cdb.db
   ```

   プロンプトが表示されたら、**y** と答えます。

8. 次のコマンドを実行して、node1 から node1 上のクラスタ・プロセスを開始します。

   ```
   # /etc/init.d/cluster start
   ```

   node2 上の SYSLOG に、以下のメッセージが記録されているはずです。

   ```
   Starting to receive CDB sync series from machine <node1 node id>
   ...
   Finished receiving CDB sync series from machine <node1 node id>
   ```

   同期が完了するまで約 60 秒待ちます。

9. 任意のノードから、クラスタの HA サービスを開始します。

運用中のクラスタでの OS ソフトウェアのアップグレード

運用中のクラスタで OS ソフトウェアをアップグレードする場合は、一度に 1 つずつノードをアップグレードしていきます。

OS ソフトウェアをアップグレードしても再起動が必要ない場合や、FailSafe ソフトウェアが影響を受けない場合は、この OS アップグレード手順に従う必要はありません。OS のアップグレードによって FailSafe ソフトウェアが影響を受けるかどうかや、マシンの再起動が必要になるかどうかがわからない場合は、以下の手順に従ってください。

以下の手順では、ノード web-node3 の OS ソフトウェアをアップグレードします。

1. ノードでリソース・グループがオンラインである場合は、cmgr コマンドまたは GUI を使用して、リソース・グループをクラスタ内の別のノードに移動します。リソース・グループをクラスタ内の別のノードに移動するには、そのリソース・グループのフェイルオーバー・ポリシー・ドメイン内に、使用可能な別のノードが存在する必要があります。

   次の cmgr コマンドを実行して、リソース・グループ web-rg をクラスタ web-cluster 内の別のノードに追加します。

   ```
   cmgr> admin move resource_group web-rg in cluster web-cluster
   ```

2. 次の cmgr コマンドまたは GUI を使用して、ノード web-node3 上の HA サービスを停止します。このコマンドを実行すると、可能であれば、このノード上のオンラインのリソース・グループがすべてクラスタ内の別のノードに移動されます。

   ```
   cmgr> stop ha_services on node web-node3 for cluster web-cluster
   ```

   このコマンドは、ノード web-node3 上のオンラインのリソース・グループを移動できない場合は失敗します。force オプションを使用すると、エラーが発生した場合でもノードの HA サービスを停止できます。

3. ノード web-node3 で OS のアップグレードを実行します。

4. OS のアップグレードが完了したら、クラスタ・プロセス（cmond、cad、crsd）が実行されていることを確認します。

5. このノードの HA サービスを再開します。たとえば、次の cmgr コマンドを実行して、このノードの HA サービスを再開します。

   ```
   cmgr> start ha_services on node web-node3 for cluster web-cluster
   ```

   HA サービスの再開後、リソース・グループが最適なノードで実行されていることを確認してください。

運用中のクラスタでの FailSafe ソフトウェアのアップグレード

運用中のクラスタで FailSafe ソフトウェアをアップグレードする場合は、クラスタ内のノードを一度に 1 つずつアップグレードします。

以下の手順では、ノード web-node3 の FailSafe ソフトウェアをアップグレードします。

1. ノードでリソース・グループがオンラインである場合は、cmgr コマンドまたは GUI を使用して、リソース・グループをクラスタ内の別のノードに移動します。リソース・グループをクラスタ内の別のノードに移動するには、そのリソース・グループのフェイルオーバー・ポリシー・ドメイン内に、使用可能な別のノードが存在する必要があります。

   たとえば、次の cmgr コマンドを実行して、リソース・グループ web-rg をクラスタ web-cluster 内の別のノードに追加します。

   ```
   cmgr> admin move resource_group web-rg in cluster web-cluster
   ```

2. 次の cmgr コマンドまたは GUI を使用して、ノード web-node3 上の HA サービスを停止します。このコマンドを実行すると、可能であれば、このノード上のオンラインのリソース・グループがすべてクラスタ内の別のノードに移動されます。

   ```
   cmgr> stop ha_services on node web-node3 for cluster web-cluster
   ```

   このコマンドは、ノード web-node3 上のオンラインのリソース・グループを移動できない場合は失敗します。force オプションを使用すると、エラーが発生した場合でもノードの HA サービスを停止できます。

3. このノード上で実行されているすべてのクラスタ・プロセスを停止します。

   ```
   # /etc/init.d/cluster stop
   ```

4. ノード web-node3 で FailSafe のアップグレードを実行します。

5. FailSafe のアップグレードが完了したら、クラスタ・プロセス（cmond、cad、 crsd）が実行されているかどうかを確認します。実行されていない場合は、次のコマンドを実行してクラスタ・プロセスを再開します。

   ```
   # chkconfig cluster on; /etc/init.d/cluster start
   ```

6. このノードの HA サービスを再開します。たとえば、次の cmgr コマンドを実行して、このノードの HA サービスを再開します。

   ```
   cmgr> start ha_services on node web-node3 for cluster web-cluster
   ```

   HA サービスの再開後、リソース・グループが最適なノードで実行されていることを確認してください。

運用中のクラスタへの新規リソース・グループまたは新規リソースの追加

以下の手順では、運用中のクラスタにリソース・グループとリソースを追加する方法を説明します。既存のリソース・グループにリソースを追加するには、リソースの設定（手順 4）、リソースの診断の実行（手順 5）、およびリソース・グループへのリソースの追加（手順 6）を行います。

1. 同時に移動する必要があるリソースをすべて識別します。ノードで実行されているこれらのリソースは、クライアントにサービスを提供できる必要があります。また、リソース・グループに配置されている必要もあります。たとえば、Netscape Web サーバ mfg-web の高可用性 (HA) IP アドレス 192.26.50.40、および Web 設定とドキュメント・ページが含まれるファイルシステム /shared/mfg-web は、同じリソース・グループ（mfg-web-rg など）に配置する必要があります。

2. リソース・グループがオンラインになると予想される、クラスタ内のすべてのノードでリソースを設定します。たとえば、クラスタ内のノード web-node1 および web-node2 上の Netscape Web サーバ mfg-web の設定が必要になる場合があります。

3. フェイルオーバー・ポリシーを作成します。リソース・グループに必要なフェイルオーバー属性のタイプを決定します。フェイルオーバー・ポリシーを作成するには、次の cmgr テンプレートを使用できます。

   /var/cluster/cmgr-templates/cmgr-create-failover_policy

4. クラスタ・データベースにリソースを設定します。/var/cluster/cmgr-templates ディレクトリにある cmgr テンプレートを使用して、さまざまなリソース・タイプのリソースを作成できます。たとえば、ボリューム・リソース、/shared/mfg-web ファイルシステム、192.26.50.40 IP_address リソース、および mfg-web Netscape_web リソースをクラスタ・データベースに作成する必要があります。これらのリソースに対してリソースの依存性を作成します。

5. リソースの診断を実行します。診断コマンドについての詳細は、249 ページの 第8章「設定のテスト」を参照してください。

6. リソース・グループを作成して、そのリソース・グループにリソースを追加します。リソース・グループを作成して、リソース・グループにリソースを追加するには、次の cmgr テンプレートを使用できます。

   /var/cluster/cmgr-templates/cmgr-create-resource_group

   相互に依存するリソースは、すべて同時にリソース・グループに追加してください。クラスタ内のノード上にあるオンラインの既存のリソース・グループにリソースを追加した場合は、追加したリソースも同じノード上でオンラインになります。

運用中のクラスタへの新規ハードウェア・デバイスの追加

運用中のクラスタに新規ハードウェア・デバイスを追加する場合は、一度に 1 つずつノードに追加します。

運用中のクラスタ内のノードにハードウェア・デバイスを追加するには、運用中のクラスタで OS ソフトウェアをアップグレードする場合と同じ手順に従います。この手順については、282 ページの「運用中のクラスタでの OS ソフトウェアのアップグレード」で説明されています。以下に概要を示します。

- ハードウェア・デバイスを追加する前に、リソース・グループをオフラインにしてノードの HA サービスを停止してください。

- ハードウェア・デバイスの追加後は、クラスタ・プロセスが実行されていることを確認し、該当するノードで HA サービスを開始してください。

クラスタ・データベースに新規ハードウェア・デバイスを含めるには、リソース設定およびノード設定の該当部分を変更する必要があります。

Performance Co-Pilot for FailSafe

この章では、PCP (Performance Co-Pilot) for FailSafe を使用して IRIX FailSafe クラスタの可用性をモニタする方法を説明します。PCP for FailSafe のインストールについての詳細は、76 ページの「PCP (Performance Co-Pilot) ソフトウェアのインストール」を参照してください。

PCP は、以下の機能を備えています。

- FailSafe ハートビートとリソース・モニタ統計を PCP フレームワークにエクスポートするためのエージェント
- これらの統計をわかりやすく表示するための 3-D 表示ツール

表示された統計を参照すると、FailSafe によってモニタされるノードとリソースの可用性について有益な情報が得られます。たとえば、モニタ応答時間の低下を強調表示できます。これは、クラスタによって提供されるサービスの可用性に問題があることを示している場合があります。

PCP for FailSafe は PCP フレームワークの拡張機能なので、ほかの PCP ツールを使用して、FailSafe モニタ統計を解析または提示したり、PCP for FailSafe メトリックをアーカイブとして記録しておいて後から解析できます。また、PCP を使用して、クラスタの各ノードについて、CPU やメモリの利用率、ネットワークとディスクのアクティビティ、およびその他のパフォーマンス・メトリックに関する統計を収集することもできます。

表示ツールの使用

FailSafe クラスタに関する統計を表示するには、`rmvis`(1) および `hbvis`(1) コマンドを使用します。

`hbvis`(1) コマンドでは、クラスタの各ノードのハートビート応答時間の分布を示す画面が表示されます。図11-1 に、画面の例を示します。

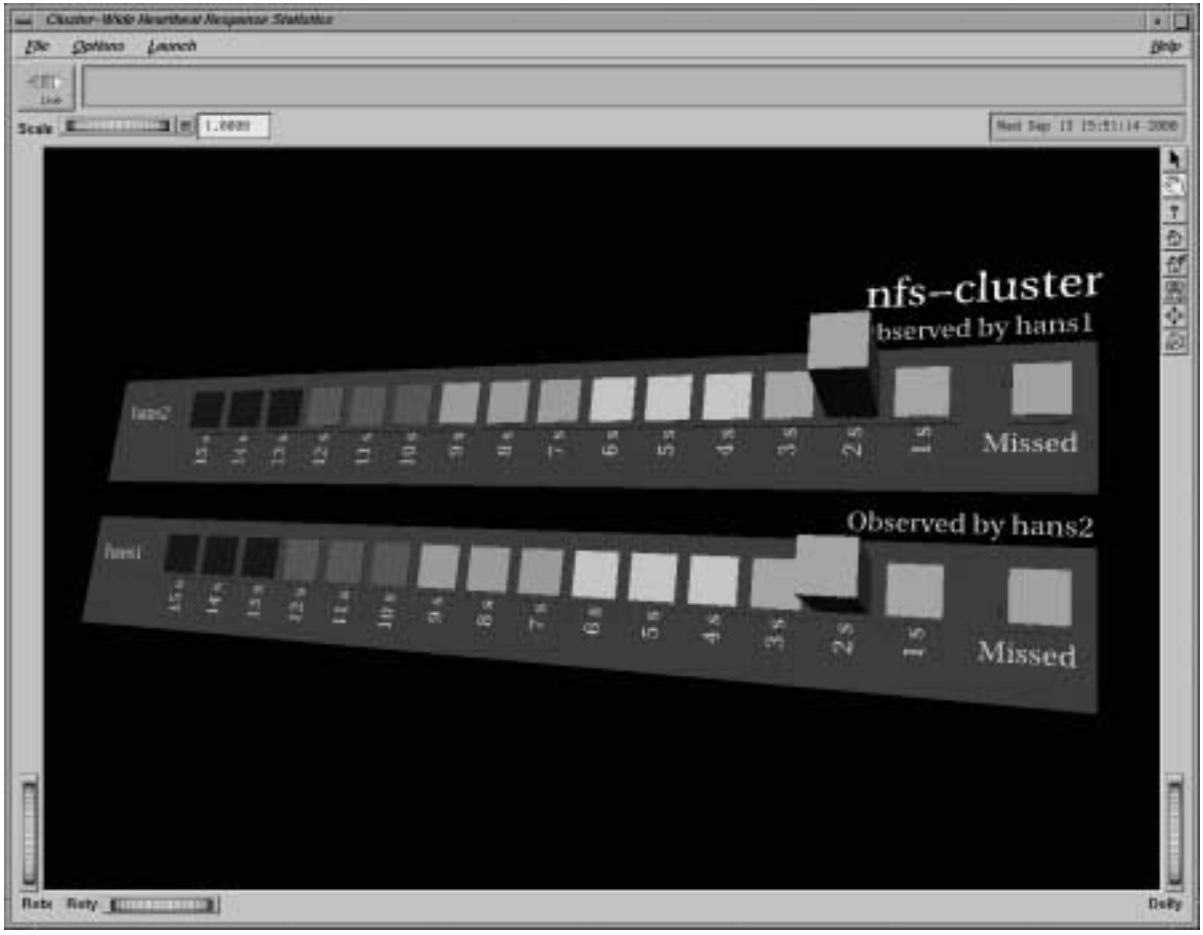


図11-1 ハートビート応答統計

この画面の主な機能としては、タイムアウト周期内に特定の周期で受信されたハートビート応答の頻度、失われた(受信されていないと判断された)ハートビート応答の頻度などがあります。失われたハートビート応答の頻度を表すバーの色が変わり、ノードの可用性に関する問題の緊急度を示します。

`rmvis`(1) コマンドでは、クラスタのすべてのノードでモニタされているリソースのリソース・モニタ応答時間を示す画面が表示されます。図11-2 に、画面の例を示します。

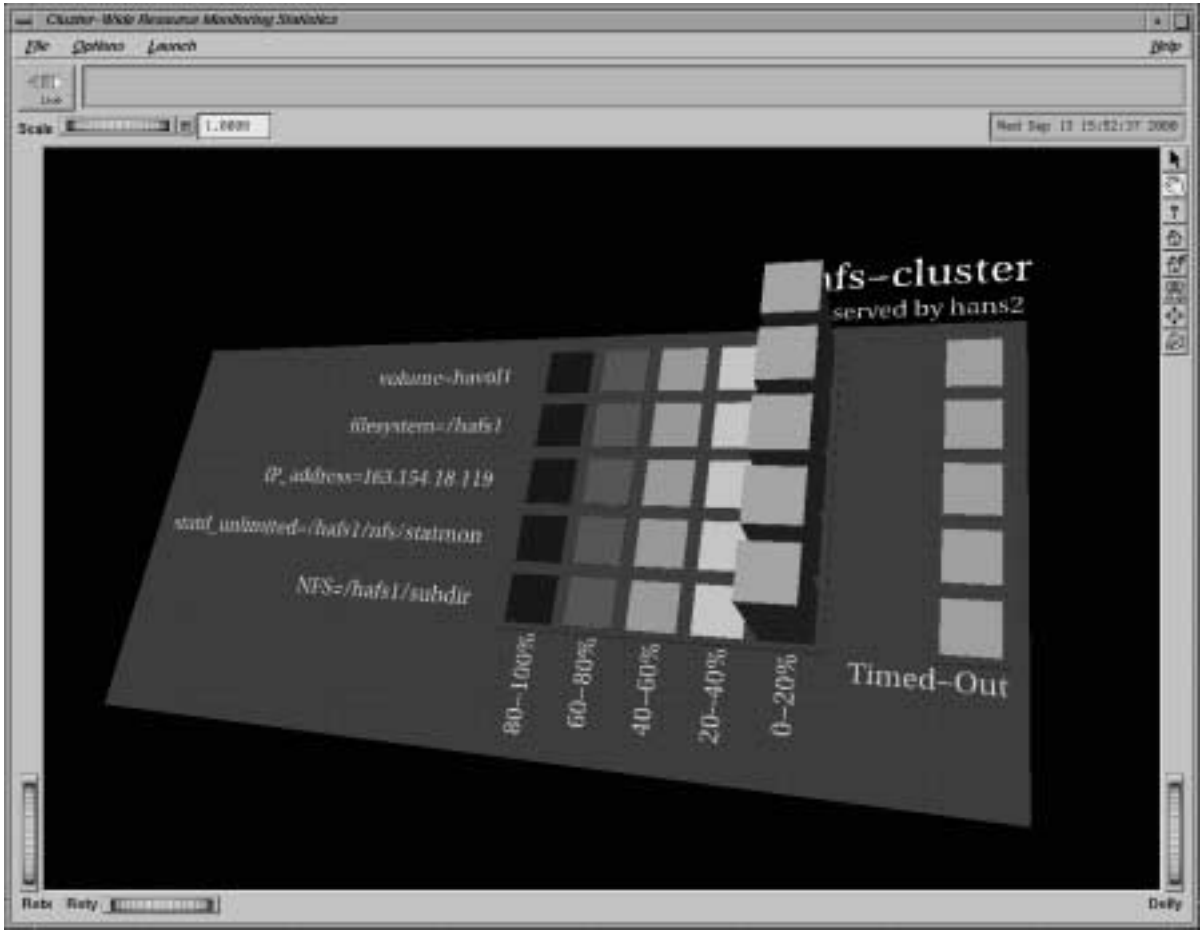


図11-2 リソース・モニタ統計

この画面の概念は hbvis(1) と同様で、タイムアウト周期内に受信されたリソース・モニタ応答の頻度と、タイムアウトした応答の頻度を表示します。また、タイムアウトしたリソース応答の頻度を表すバーの色が変わり、特定のリソースの可用性に関する問題の緊急度を示します。

ノードに異常が発生したり、リソースがフェイルオーバーされた場合、その統計は画面から消えます。

モニタ・ホストで表示ツールを実行するには、 -h オプションを使用して、クラスタで使用可能なコレクタ・ホスト (*host*) を指定します。

```
% hbvis -h host
```

または

```
% rmvis -h host
```

コレクタ・ホストには、統計を表示するクラスタのメンバーである任意のコレクタ・ホストを指定できます。

これらのツールには、以下の FailSafe GUI メニューからアクセスすることもできます。

「ファイル(File)」
>「リソース・モニタの起動(Launch Resource Monitoring)」

「ファイル」
>「ハートビート・モニタの起動(Launch Heartbeat Monitoring)」

コマンド・ラインから起動した場合、hbvis(1) および rmvis(1) で表示される画面を変更するには、以下のようなさまざまオプションを使用できます。

`-H hostfile`	画面に表示するノードのリストのファイルを指定します。クラスタのノードが多いほど表示に時間がかかるので、表示するノードの数を制限する場合に便利です。
`-t interval`	画面のサンプリング時間を割当てます。特に多くのノードがあるクラスタでは、サンプリング時間の周期を延ばすと、アプリケーションの反応が良くなる場合があります。統計は FailSafe によって保持されているので、 hbvis(1) および rmvis(1) を使用すると、選択したサンプリング時間で使用できる最新の統計が常に表示されます。*interval* オプションについての詳細は、pmview(1) および PCPIntro(1) のマン・ページを参照してください。
`-r`	最後に統計をリセットした時刻以降に収集された統計のサンプリングを表示する FailSafe メトリックを選択します。これにより、hbvis(1) および rmvis(1) は、FailSafe モニタ統計に急激な変化が現れた場合に表示感度を向上させることができます。 -r オプションを指定しない場合、表示される統計は、ha_cmsd(1m) または ha_srmd (1m)、あるいはその両方が最後に再開された時刻以降に収集された FailSafe メトリックのサンプリングの統計になります。
`-R`	新しい統計サンプリングを開始します。
`-v`	(hbvis(1) のみ) クラスタの各ノードのハートビート統計を、選択したコレクタ・ホストだけについて表示します (コレクタ・ホストは -h オプションで選択します)。選択したコレクタ・ホストによって収集されたクラスタの各ノードのハートビート統計が、ノードごとにグラフィカルな画面に表示されます。

-w	(hbvis(1) のみ)クラスタのすべてのノードのハートビート統計の合計を、選択したコレクタ・ホストだけについて表示します (コレクタ・ホストは -h オプションで選択します)。選択したコレクタ・ホストによって収集されたクラスタ全体のハートビート統計が、1 つのグラフィカルな画面に表示されます。

オプションについての詳細は、 hbvis(1) および rmvis(1) のマン・ページを参照してください。

hbvis(1) および rmvis(1) コマンドは、pmview (1) コマンドを使用して、FailSafe パフォーマンス・メトリックの 3-D 画面を表示します。表示ウィンドウのさまざまなメニュー・コマンドやコントロールについての詳細は、pmview(1) のマン・ページを参照してください。

PCP for FailSafe のパフォーマンス・メトリック

pmlogger(1)、pmchart(1)、pminfo(1) などの PCP ツールでは、PCP for FailSafe によってエクスポートされたメトリックを使用できます。

PCP for FailSafe のメトリックは、331 ページの付録D「PCP (Peformance Co-Pilot) for FailSafe によってエクスポートされるメトリック」で説明されています。また、次のコマンドを使用してメトリックの説明を表示することもできます。

```
% pminfo -tT -h host
```

(コレクタ・ホストにログインしている場合、 -h オプションは省略できます。)

PCP のグレーの画面

hbvis(1) または rmvis(1) を使用したときにグレーの画面が表示される場合(つまり、ノードのグレーの基本プレーンに、色の付いた長方形のバーが表示されない場合)は、以下のいずれかの状態を示している場合があります。

- ノードが停止している

  動作しているノードだけを表示したい場合は、表示するノードのリストが含まれるファイルを作成し、-H オプション(または環境変数 PCP_FSAFE_NODES)を使用してこのファイルを hbvis(1)/rmvis(1) にオプションとして渡し、クラスタの新しい画面を生成できるようにします。 -H オプションについての詳細は、hbvis(1)/rmvis(1) のマン・ページを参照してください。

- ノードでコレクタ・デーモンが強制終了されている

  この問題を解決するには、以下のいずれかの方法で pmdafsafe (1) を再開します。

– pmcd(1) がまだ実行されている場合は、以下を入力して、pmcd(1) に SIGHUP シグナルを送信します。

```
# killall -HUP pmcd
```

– pmcd(1) が実行されていない場合は、以下を入力して PCP を再開します。

```
# /etc/init.d/pcp start
```

- タイムアウトおよびサンプリングの時間の設定が短すぎる

  サンプリング時間を変更するには、pmview(1) のウィンドウに用意されている時間コントロールを使用します。デフォルトでは、サンプリング時間は 2 秒です。表示される内容が不十分な場合は、サンプリング周期を長くしなければならない場合があります。

  あるいは、 pmdafsafe(1) と pmcd(1)、または pmcd(1) と pmview(1) の間にタイムアウトの問題が存在する可能性があります。さまざまな PCP ツールのタイムアウト設定の変更方法についての詳細は、pmcd(1) および PCPIntro(1) のマン・ページを参照してください。

- リソースがフェイルオーバーされている(rmvis(1) の場合)

  この場合は、クラスタの新しい画面を生成できるように、rmvis(1) を再開します。

ソフトウェアの概要

この節では、以下の内容について説明します。

ソフトウェア階層

FailSafe システムのソフトウェア階層は、以下のとおりです。

- プラグイン。高可用性サービスを作成します。表 A-1 に、付属の FailSafe プラグインとオプションの FailSafe プラグイン、およびそれらに関連付けられているリソース・タイプを示します。

表A-1 付属のプラグインおよびオプションのプラグイン

付属のプラグイン	リソース・タイプ	オプションのプラグイン	リソース・タイプ
CXFS ファイルシステム	`CXFS`	FailSafe/DMF	`DMF`
IP アドレス	`IP_address`	FailSafe/NFS	`NFS` および `statd_unlimited`
MAC アドレス	`MAC_address`	FailSafe/Informix	`INFORMIX_DB`
XFS ファイルシステム	`filesystem`	FailSafe/Oracle	`Oracle_DB`
XLV 論理ボリューム	`volume`	FailSafe/Samba	`Samba`

付属のプラグイン	リソース・タイプ	オプションのプラグイン	リソース・タイプ
		FailSafe/TMF	TMF
		FailSafe/Web (Netscape)	Netscape_web

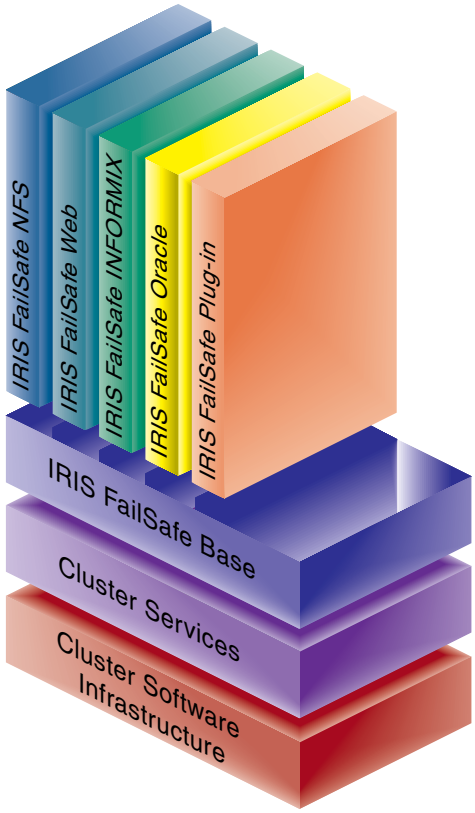
これらの製品のサポートされている特定のリリースについての詳細は、リリース・ノートを参照してください。

必要なアプリケーションがない場合は、必要なソフトウェアの開発を日本 SGI に依頼するか、『IRIS FailSafe Version 2 Programmer's Guide』を参照して独自のソフトウェアを作成できます。

- FailSafe base。リソース・グループおよびフェイルオーバー・ポリシーを定義する機能が含まれます。
- クラスタ・サービス。クラスタ、リソース、およびリソース・タイプを定義できます(これは、cluster_services インストール・パッケージで構成されています)。
- クラスタ・ソフトウェア・インストラクチャ。以下を実行できます。
  - ノードでのログの実行
  - クラスタの管理
  - ノードの定義

  クラスタ・ソフトウェア・インフラストラクチャは、cluster_admin サブシステムおよび cluster_control サブシステムから構成されます。

図A-1 に、これらの階層のイメージを示します。クラスタ・サービスおよびクラスタ・ソフトウェア・インフラストラクチャの階層は、CXFS と共有されます。296 ページの表A-2 に、/usr/cluster/bin ディレクトリの内容を示します。CXFS についての詳細は、『CXFS Version 2 Software Installation and Administration Guide』を参照してください。



図A-1 ソフトウェア階層

表A-2 /usr/cluster/bin の内容

層	サブシステム	プロセス	説明
プラグイン	failsafe_informix failsafe2_oracle	ha_ifmx2	IRIS FailSafe データベース・エージェント。各データベース・エージェントは、1 つのタイプのデータベースのすべてのインスタンスをモニタします。
IRIS FailSafe Base	failsafe2	ha_fsd	IRIS FailSafe デーモン。IRIS FailSafe ソフトウェアの基本コンポーネントを提供します。
クラスタ・サービス(高可用性プロセス)	cluster_services	ha_cmsd	FailSafe メンバーシップ・ドメイン。FailSafe メンバーシップと呼ばれる、クラスタで使用可能なノードのリストを提供します。
		ha_gcd	グループ・メンバーシップ・デーモン。IRIS FailSafe プロセスに異常が発生した場合に、グループ・メンバーシップおよび信頼性の高い通信サービスを提供します。
		ha_srmd	システム・リソース・マネージャ・デーモン。リソース、リソース・グループ、およびリソース・タイプを管理します。リソースのアクション・スクリプトを実行します。
		ha_ifd	インタフェース・エージェント・デーモン。ローカル・ノードのネットワーク・インタフェースをモニタします。このデーモンについては、297 ページの「インタフェース・エージェント・デーモン (IFD)」で詳細に説明されています。
クラスタ・ソフトウェア・インフラストラクチャ(クラスタ管理プロセス)	cluster_admin	cad	クラスタ管理デーモン。管理サービスを提供します。
	cluster_control	crsd	ノード・コントロール・デーモン。その他のノードとのシリアル接続をモニタします。その他のノードをリセットする機能があります。
		cmond	その他のすべてのデーモンを管理するデーモン。このプロセスは、クラスタ内のすべてのノードでほかのプロセスを開始し、異常時にはこれらのプロセスを再開します。
		fs2d	クラスタ・データベースを管理し、プール内のすべてのノードで各コピーの同期を保ちます。

インタフェース・エージェント・デーモン (IFD)

IFD は、ネットワーク・インタフェースおよび IP アドレスをモニタするエージェントです。IFD は、ノードに高可用性 IP アドレスがない場合でも、そのノードで定義されているすべてのネットワーク・インタフェースと IP アドレスをモニタします。

IFD は、各インタフェースの入力パケットの数をチェックします。10 秒間に入力パケットの数が増えない場合、IFD は、`ping`(1M) コマンドを使用して、インタフェースのブロードキャスト・アドレスと通信します。次の 10 秒間でも入力パケット・カウントが増えない場合は、ネットワーク・インタフェースとそのインタフェースのすべての IP アドレスが不正としてマークされます。

IFD は、クラスタ・データベースから IP アドレスの設定を読取ります。

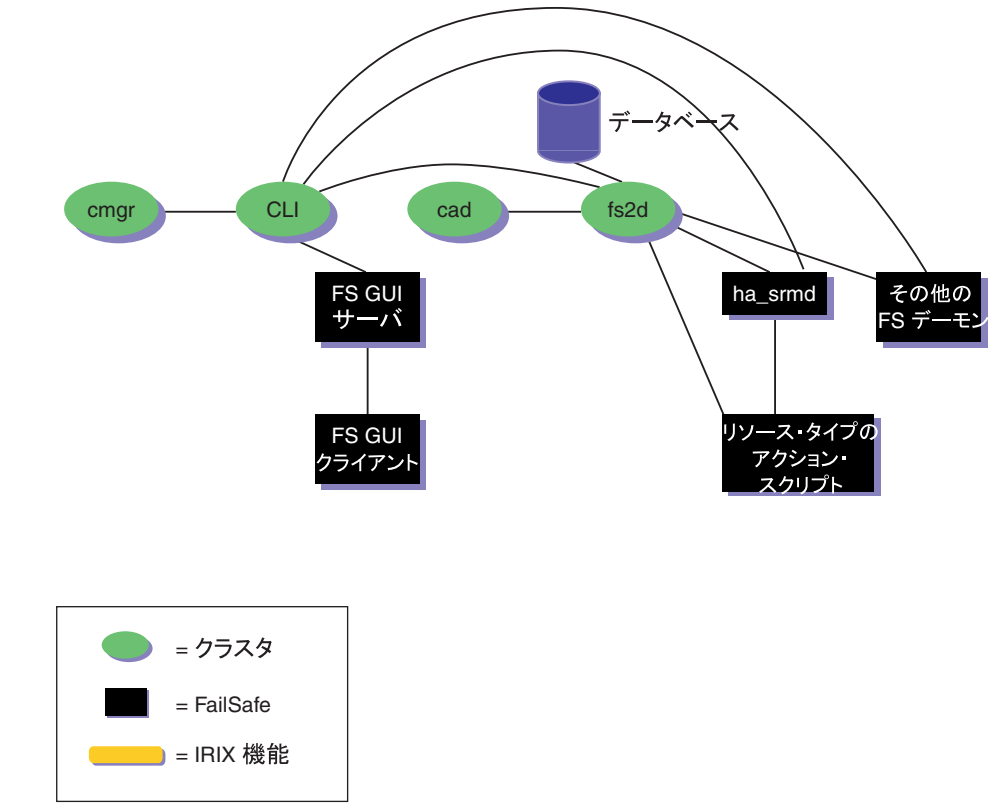
`IP_address` リソース・タイプのアクション・スクリプトは、 `ha_ifdadmin` コマンドを使用して IFD と通信します。アクション・スクリプトは、IFD からステータスおよび設定 IP アドレスを取得します。

IFD ログは、GUI および `cmgr` コマンドを使用して制御できます。

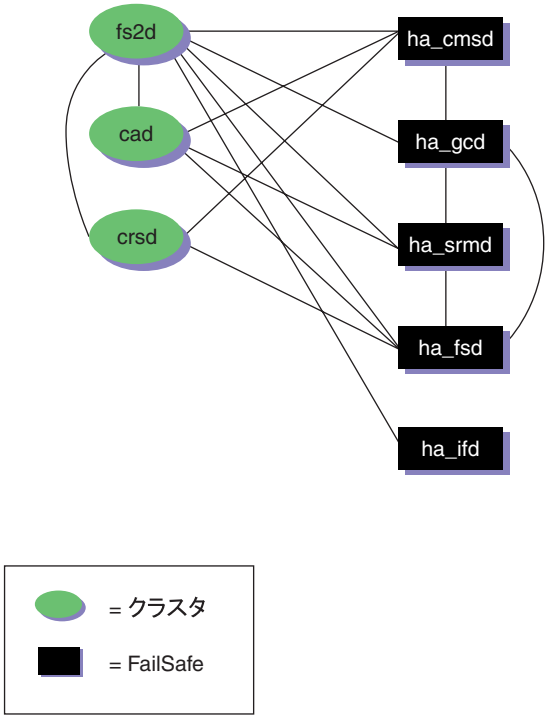
通信パス

以下の図に、FailSafe の通信パスを示します。

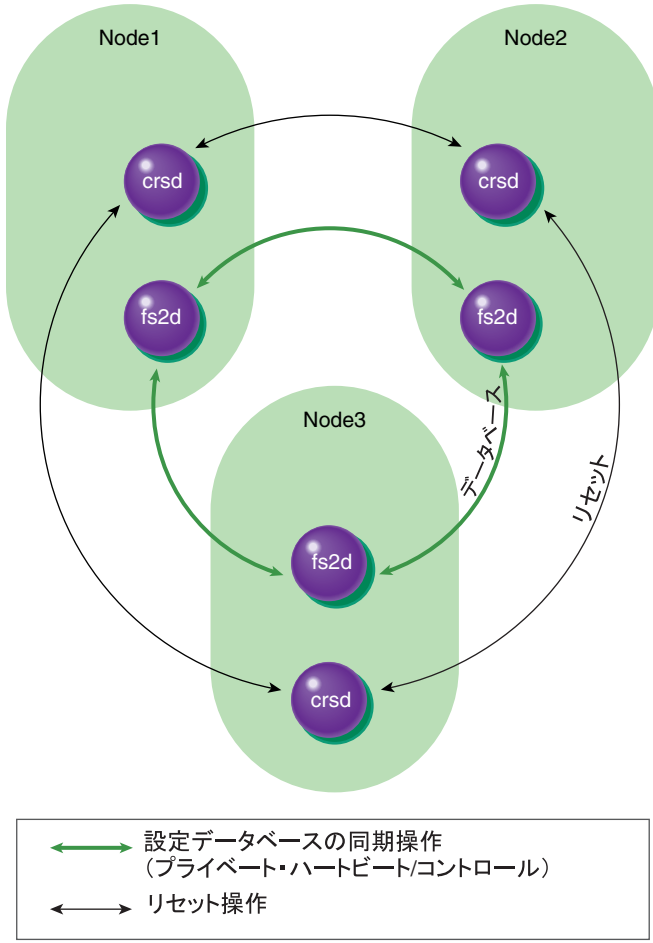
メモ: 以下の図には、`cmond` クラスタ・マネージャ・デーモンは表されていません。このデーモンの目的は、その他のデーモンを実行し続けることです。



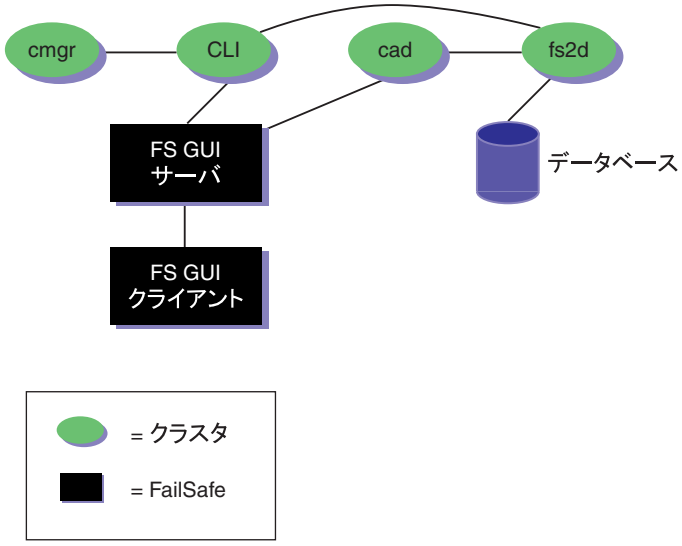
図A-2 ノード内の管理通信



図A-3 ノード内のデーモン通信



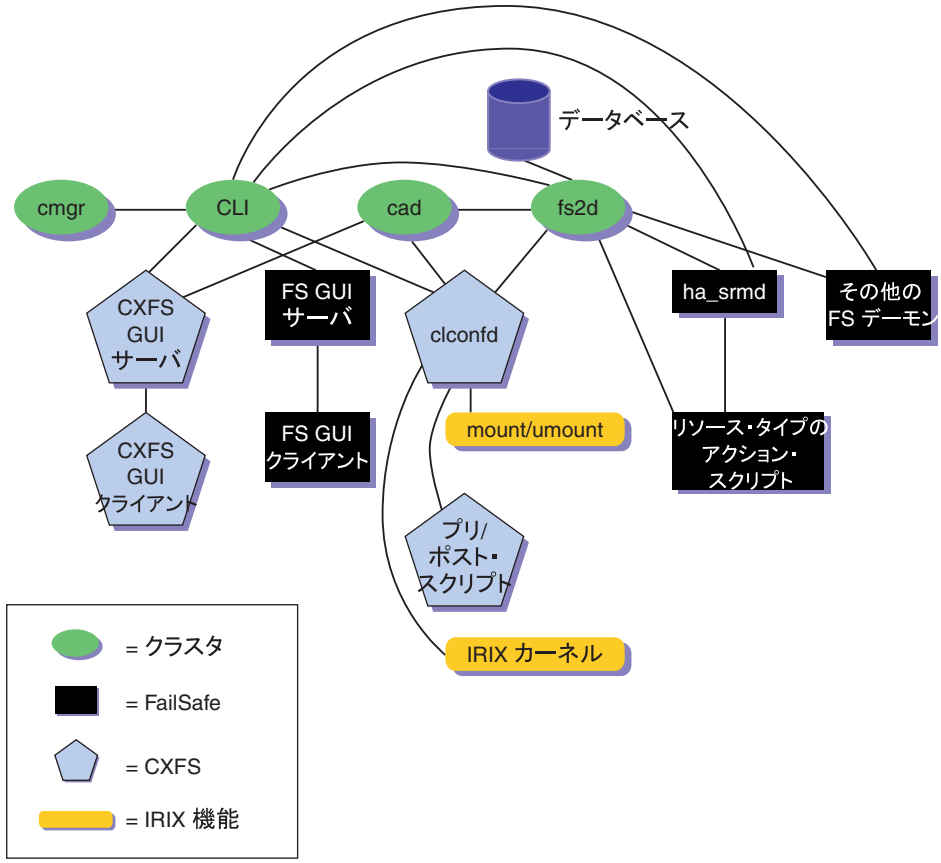
図A-4 プール内のノード間の通信



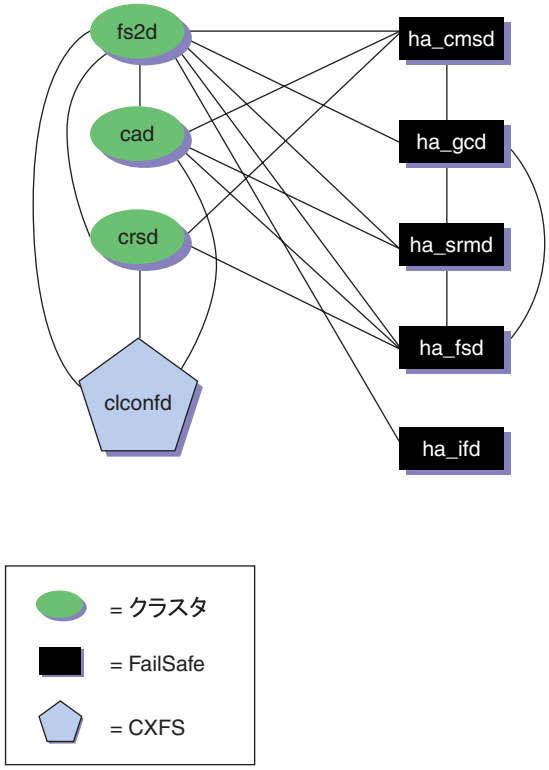
図A-5 クラスタ内にないノードの通信

同時実行クラスタ内の通信パス

以下の図に、同時実行クラスタ内の 1 つのノードの中の通信パスを示します。



図A-6 同時実行時のノード内の管理通信



図A-7 同時実行時のノード内のデーモン通信

FailSafe のアクション・スクリプトおよびフェイルオーバー・スクリプトの実行

実行の順序は、以下のとおりです。

1. FailSafe は、`cmgr` の `start ha_services` コマンドによって開始されるか、またはノード起動手順の一部として開始されます。その後、FailSafe は、クラスタ・データベースからリソース・グループ情報を読込みます。

2. FailSafe は、状態が `Online ready` であるすべてのリソース・グループの `exclusive` スクリプトを実行するようシステム・リソース・マネージャ (SRM: System Resource Manager) に要求します。

3. SRM により、各リソース・グループについて以下のいずれかの状態が返されます。

   - `running`
   - `partially running`
   - `not running`

4. HA サービスが開始されたノードに `not running` の状態のリソース・グループがある場合は、以下が行われます。

   a. FailSafe によって、リソース・グループに関連付けられているフェイルオーバー・ポリシー・スクリプトが実行されます。フェイルオーバー・ポリシー・スクリプトは、リソース・グループを実行できるノードのリスト(フェイルオーバー・ドメイン)をパラメータとして使用します。

   b. フェイルオーバー・ポリシーによって、優先度に基づいて降順に表示されたノードの順序付きリスト(ラインタイム・フェイルオーバー・ドメイン)が返されます。これらのノードにリソース・グループを配置できます。

   c. FailSafe によって、ランタイム・フェイルオーバー・ドメインの最初のノードにリソース・グループを移動するよう SRM に要求が送信されます。

   d. SRM によって、リソース・グループの全リソースの `start` アクション・スクリプトが実行されます。

      - `start` スクリプトが失敗した場合は、リソース・グループのノードに、以下のエラーとともに `online` というマークが付けられます。

        ```
        srmd executable error
        ```

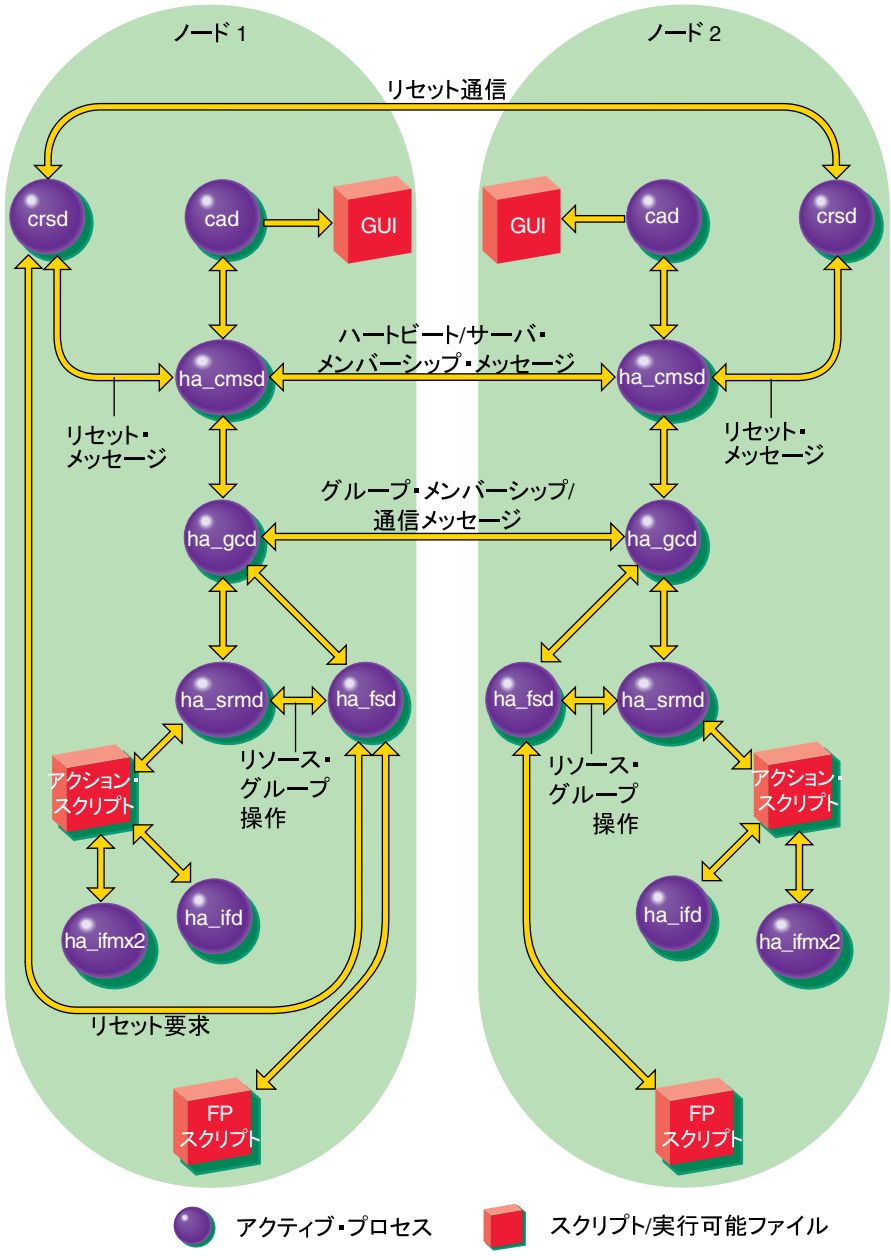
      - `start` スクリプトが成功した場合は、SRM によって、それらのリソースのモニタが自動的に開始されます。指定したモニタ開始時刻を過ぎると、SRM により、リソース・グループのリソースに対して `monitor` アクション・スクリプトが実行されます。

5. クラスタ内の 1 つのノードだけでリソース・グループの状態が `running` または `partially running` の場合は、FailSafe によって、関連付けられているフェイルオーバー・ポリシー・スクリプトが実行されます。

   - 優先度の最も高いノードが、リソース・グループが `partially running` または `running` のノードと同じ場合、このリソース・グループは同じノード上でオンラインになります。 `partially running` の場合、FailSafe は、リソース・グループのすべてのリソースの `start` スクリプトを実行するよう SRM に要求します。
   - 優先度の最も高いノードがクラスタ内の別のノードである場合、FailSafe は、その他のノードでそのリソース・グループのリソースの `stop` アクション・スクリプトを実行するよう SRM に要求します。次に、このリソース・グループは、FailSafe によって、クラスタ内で優先度が最も高いノードでオンラインにされます。

6. クラスタ内の複数のノードでリソース・グループの状態が `running` または `partially running` の場合は、リソース・グループに `error exclusivity` エラーのマークが付けられます。これらのリソース・グループをクラスタでオンラインにするには、オペレータによる操作が必要になります。

図A-8に、アクション・スクリプトおよびフェイルオーバー・ポリシー・スクリプトのメッセージ・パスを示します。



図A-8 アクション・スクリプトおよびフェイルオーバー・ポリシー・スクリプトのメッセージ・パス

start スクリプトが失敗した場合

start アクション・スクリプトが失敗した場合、実行の順序は以下のようになります。

1. SRM によって、start アクション・スクリプト異常がリソース・グループ異常として FailSafe に通知されます。
2. FailSafe は、フェイルオーバー・ポリシー・スクリプトを実行して、リソース・グループの次のノードを決定します。
3. FailSafe は、リソース・グループを解放して、クラスタ内の次のノードにリソース・グループを割当てるよう SRM に要求を送信します。

stop スクリプトが失敗した場合

stop アクション・スクリプトが失敗した場合、実行の順序は以下のようになります。

1. SRM によって、stop アクション・スクリプト異常がリソース・グループ異常として FailSafe に通知されます。
2. FailSafe によって、リソース・グループに次のエラーのマークが付けられます。

   ```
   srmd executable error
   ```

3. システム管理者は、ノード内のリソース・グループを停止した後で、 offline force コマンドを使用してエラー状態をクリアする必要があります。

コンポーネント

クラスタ・データベースは FailSafe ソフトウェアのキー・コンポーネントで、以下に関するすべての情報が含まれています。

- リソース
- リソース・タイプ
- リソース・グループ
- フェイルオーバー・ポリシー
- ノード
- クラスタ

クラスタ・データベース・デーモン (fs2d) によって、クラスタ内の各ノードで同一内容のデータベースが維持されます。

以下の図に、/var/cluster/ha ディレクトリの内容を示します。

表A-3 /var/cluster/ha ディレクトリの内容

ディレクトリまたはファイル	用途
comm/	さまざまなデーモンの間で送受信されるファイルが含まれるディレクトリ。FailSafe プロセスは、このディレクトリに一時ファイルを作成します。FailSafe クラスタの各ノードの root ファイルシステムにこのディレクトリ用の十分なディスク容量(約 2〜3 MB)がないと、FailSafe プロセス間通信が失敗します。
common_scripts/	スクリプト・ライブラリ(アクション・スクリプトで使用されることがある共通関数)が含まれるディレクトリ。
log/	IRIS FailSafe によって実行されたすべてのスクリプトとデーモンのログが含まれるディレクトリ。スクリプト内のコマンドからの出力やエラーのログは、 script_*nodename* ファイルに記録されます。
policies/	リソース・グループに使用されるフェイルオーバー・スクリプトが含まれるディレクトリ。
resource_types/template	テンプレート・アクション・スクリプトが含まれるディレクトリ。
resource_types/*rt_name*	*rt_name* リソース・タイプのアクション・スクリプトが含まれるディレクトリ。たとえば、/var/cluster/ha/resource_types/filesystem などがあります。
resource_types/*rt_name*/exclusive	このリソース・タイプのリソースがすでに実行中ではないことを確認するスクリプト。
resource_types/*rt_name*/monitor	このリソース・タイプのリソースをモニタするスクリプト。
resource_types/*rt_name*/restart	モニタ異常の発生後、このリソース・タイプのリソースを同じノードで再開するスクリプト。

ディレクトリまたはファイル	用途
resource_types/*rt_name*/start	このリソース・タイプのリソースを開始するスクリプト。
resource_types/*rt_name*/stop	このリソース・タイプのリソースを停止するスクリプト。

IRIS FailSafe 1.2 から IRIS FailSafe 2.1.*x* へのアップグレード

IRIS FailSafe 2.1.*x* は IRIS FailSafe 1.2 製品の新しいリリースではなく、高可用性システムのサイズや複雑さに合った多くの追加機能を提供するファイルやスクリプトの新しいセットです。これらの機能を活用するために FailSafe 1.2 システムを FailSafe 2.1.*x* システムに移行する場合は、システム設定をアップグレードする必要があります。FailSafe 1.2 を自動的に FailSafe 2.1.*x* にアップグレードするためのアップグレード・インストール・オプションはありません。

この付録では、システムを FailSafe 1.2 から FailSafe 2.1.*x* にアップグレードする手順を説明します。この付録は、以下の節で構成されています。

ハードウェアの変更

システムを FailSafe 2.1.*x* にアップグレードするためにハードウェアを変更する必要はありません。FailSafe 1.2 システムは、FailSafe 2.1.*x* では、リセット・リングの 2 ノード設定のデュアルホスト・ストレージになります。

FailSafe 2.1.*x* では、FailSafe 診断コマンドを使用してハードウェア設定をテストできます。FailSafe を使用して接続をテストする方法については、249 ページの 第8章「設定のテスト」を参照してください。これらの診断は、FailSafe 2.1.*x* の起動時に自動的に実行されることはないので、手動で実行する必要があります。

また、`admin ping` コマンドを使用して、FailSafe 2.1.*x* でシリアル・リセット回線をテストすることもできます。このコマンドは、FailSafe 1.2 で使用されていた `ha_spng` コマンドに代わるものです。

以下に、シリアル・リセット回線をテストするための FailSafe 1.2 のコマンドを示します。

```
# /usr/etc/ha_spng -i 1 -d msc -f /dev/ttyd2
# echo $status
```

以下に、シリアル・リセット回線をテストするための FailSafe 2.1.*x* の cmgr コマンドを示します。

```
cmgr> admin ping dev_name /dev/ttyd2 of dev_typetty with sysctrl_type msc
```

cmgr コマンドの使用についての詳細は、83 ページの 第4章「管理ツール」を参照してください。

ソフトウェアの変更

FailSafe 2.1.*x* は、FailSafe 1.2 とは異なるファイルのセットで構成されます。FailSafe 1.2 と FailSafe 2.1.*x* のソフトウェアを同じノード上に共存させることはできますが、両方のバージョンの FailSafe を同時に実行することはできません。

FailSafe 1.2 では、設定ファイル ha.conf が使用されます。FailSafe 2.1.*x* では、設定情報は /var/cluster/cdb/cdb.db にあるクラスタ・データベースに収められ、プール内のすべてのノードで保持されます。クラスタ・データベースは、cmgr コマンドまたはグラフィカル・ユーザ・インタフェース (GUI: Graphical User Interface) を使用して作成します。

FailSafe 2.1.*x* のクラスタ・データベースは自動的にプール内のすべてのノードにコピーされ、プール内のすべてのノードで FailSafe 2.1.*x* の設定が保持されます。

設定の変更

FailSafe 1.2 システムを FailSafe 2.1.*x* システムとして設定するには、FailSafe 2.1.*x* GUI または FailSafe 2.1.*x* の cmgr コマンドを使用して FailSafe 1.2 システムを再設定してください。これらの管理ツールの使用についての詳細は、83 ページの 第4章「管理ツール」を参照してください。

FailSafe 1.2 の設定を更新するには、以下を参照して、FailSafe 1.2 の設定とリソース・グループの概念がどのように対応しているかを考慮してください。

- FailSafe 1.2 のデュアルアクティブ設定では、各ノードごとに 1 つずつ、2 つのリソース・グループが含まれます。
- FailSafe 1.2 のアクティブ／スタンバイ設定では、ノード全体 (アクティブ・ノード) で構成される 1 つのリソース・グループが含まれます。

各リソース・グループには、各ノードでプライマリになっていて、他方のノードでバックアップされていたすべてのアプリケーションが含まれます。

FailSafe 2.1.*x* システムを設定する場合は、以下の手順に従ってください。

1. ノードをプールに追加します。
2. クラスタを作成します。
3. 作成したクラスタにノードを追加します。
4. HA パラメータを設定します（必要であれば、この時点で FailSafe 2.1.*x* を起動できます）。
5. リソースを作成します。
6. フェイルオーバー・ポリシーを作成します。
7. リソース・グループを作成します。
8. リソース・グループにリソースを追加します。
9. リソース・グループをオンラインにします。

上記の手順は、GUI の設定ガイド・タスクセットに記載されています。これらのタスクセットを使用して、上記の設定を手順を実行できます。

FailSafe 1.2 設定と FailSafe 2.1.*x* 設定を比較した設定例については、315 ページの「アップグレードの例」を参照してください。

スクリプト

FailSafe 1.2 のスクリプトは、すべて FailSafe 2.1.*x* 用に作成し直す必要があります。『IRIS FailSafe Version 2 Programmer's Guide』には、FailSafe 2.1.*x* のスクリプトの詳しい説明に加え、FailSafe 1.2 のスクリプトを、FailSafe 2.1.*x* で同等の機能を持つスクリプトに移行するための詳しい方法が記載されています。

動作の比較

フェイルオーバーの単位は、FailSafe 1.2 ではノードで、FailSafe 2.1.*x* ではリソース・グループです。このため、ノード・フェイルオーバーやノード・フェイルバックの概念だけでなく、ノード状態の概念さえも、FailSafe 2.1.*x* には当てはまりません。さらに、これらの 2 つのリリースでは、FailSafe スクリプトもすべて異なります。

以下の表に、これらのリリースの違いの概要を示します。

表B-1 IRIS FailSafe 1.2 と 2.1.*x* の相違点

FailSafe 1.2	FailSafe 2.1.*x*
`ha.conf` 設定ファイル	`/var/cluster/cdb/cdb/db` にあるクラスタ・データベース。このデータベースは、プール内のすべてのノードに自動的にコピーされます。FailSafe 1.2 の `ha.conf` ファイルに含まれるデータの大部分は 2.1.*x* データベースで使用されますが、形式はまったく異なります。データベースは、クラスタ・マネージャのグラフィカル・ユーザ・インタフェースまたは `cmgr` コマンドを使用して設定します。
ノードの状態 (スタンバイ、通常、低下、起動中、または動作中)	リソース・グループの状態 (オンライン、オフライン、ペンディング、メインテナンス、エラー)
スクリプト	スクリプト
`giveaway, giveback`	`stop`
`takeover, takeback`	`start`
`check`	`monitor`
(該当なし)	`exclusive, probe, restart`
	フェイルオーバー・スクリプト
	フェイルオーバー属性

FailSafe 1.2	FailSafe 2.1.*x*
すべての共通関数と変数は、/var/ha/actions/common.vars ファイルに保持されています。	すべての共通関数と変数は、/var/cluster/ha/common_scripts/scriptlib ファイルに保持されています。
設定情報は、ha_cfginfo コマンドを使用して読込みます。	設定情報は、ha_get_info() および ha_get_field() シェル関数を使用して読込みます。
アプリケーションの順序はソフトウェア・リンクで指定します。	順序の指定にソフトウェア・リンクを使用しません。
スクリプトでは /sbin/shが使用されます。	スクリプトでは /sbin/kshが使用されます。
スクリプトでは、設定のチェックサムの確認が必要です。	スクリプトでは設定のチェックサムの確認はありません。
スクリプトにはリソースの所有権が必要です。	アクション・スクリプトでは、リソースの所有権は認識されません。
複数のスクリプトを並行して実行することはできません。	アクション・スクリプトの複数のインスタンスを同時に実行できます。
各サービスごとに、 /var/ha/logs に専用のログがあります。	アクション・スクリプトはクラスタのログ機能を使用します。すべてのスクリプトは、ha_cilog コマンドを使用して、同じファイルにログを記録します。
クラスタ内の各ノードごとに 1 つずつ、2 つのフェイルオーバーの単位があります。	各高可用性サービスごとに 1 つのフェイルオーバーの単位 (リソース・グループ) があります。

アップグレードの例

FailSafe 1.2 システムを FailSafe 2.1.*x* システムにアップグレードするには、ha.conf ファイルを調べて、FailSafe 2.1.*x* クラスタ・データベースで該当するパラメータを定義する方法を判断してください。

続く節では、以下のタスクに関するアップグレードの例を説明します。

- ノードの定義
- クラスタの定義
- HA パラメータの設定

- リソースの定義: XLV ボリューム
- リソースの定義: XFS ファイルシステム
- リソースの定義: IP アドレス

以下のタスクのアップグレードの例については、『IRIS FailSafe Version 2 Programmer's Guide』を参照してください。このガイドは、カスタマイズしたリソースとスクリプトについて説明しています。

- リソース・タイプの定義
- フェイルオーバー・ポリシーの定義
- FailSafe スクリプトの作成

ノードの定義

以下に、FailSafe 1.2 の `ha.conf` ファイルのノード定義の例を示します。FailSafe 2.1.*x* システムの設定の際に必要なパラメータは、太字で示されています。

```
Node node1
{
interface node1-fxd
{
name = rns0
ip-address = 54.3.252.6
netmask = 255.255.255.0
broadcast-addr = 54.3.252.6
}
heartbeat
{
hb-private-ipname = 192.0.2.3
hb-public-ipname = 54.3.252.6
hb-probe-time = 6
hb-timeout = 6
hb-lost-count = 4
}
reset-tty = /dev/ttyd2

sys-ctlr-type = MSC
}
```

この設定例の場合、FailSafe 2.1.*x* で同じノードを定義するときは、以下の値を使用します。

- ノード名: node1
- プライマリ・ネットワーク・インタフェース: node1
- システム・コントローラのタイプ: msc
- システム・コントロール・デバイス名: /dev/ttyd2
- コントロール・ネットワーク: 192.0.2.3, 54.3.252.6

これらの値を使用して FailSafe 2.1.*x* でノードを定義するには、次の cmgr コマンドを使用します。なお、このノードを定義するときは、その他のパラメータも指定する必要があります。

```
cmgr> define node node1
Enter commands, you may enter "done" or "cancel" at any time to exit

Hostname[optional]? node1
Is this a FailSafe node <true|false> ? true
Is this a CXFS node <true|false> ? false
Node ID ? 10
Reset type <powerCycle> ? (powerCycle)
Do you wish to define system controller info[y/n]:y
Sysctrl Type <msc|mmsc>? (msc) msc
Sysctrl Password [optional]? ( )
Sysctrl Status <enabled|disabled>? enabled
Sysctrl Owner? node2
Sysctrl Device? /dev/ttyd2
Sysctrl Owner Type <tty> [tty]?
Number of Network interfaces [2]? 2
NIC 1 - IP Address? 192.0.2.3
NIC 1 - Heartbeat HB (use network for heartbeats) <true|false>? true
NIC 1 - (use network for control messages) <true|false>? true
NIC 1 - Priority <1,2,...>? 1
...
```

この ha.conf のノード定義からわかるように、FailSafe 1.2 では、ノードを定義する際に、モニタ・メッセージの送信頻度を指定する値や、応答のない時間がどの程度続いたときに異常と見なすかを指定する値を設定するためのパラメータを定義していました。FailSafe 2.1.*x* でのモニタ値の設定についての詳細は、318 ページの「HAパラメータの設定」を参照してください。

クラスタの定義

FailSafe 1.2 ではクラスタの定義は必要ありませんが、FailSafe 2.1.*x* がクラスタ定義で使用する `ha.conf` ファイルでは、クラスタ内で問題が発生した場合にシステム管理者に通知するために使用する電子メール・アドレスを指定します。

`ha.conf` ファイルには、以下の指定が含まれます。

```
system configuration
{
mail-dest-addr = root@localhost
...
}
```

FailSafe 2.1.*x* でクラスタを定義する場合は、この定義を、問題の通知に使用する電子メール・アドレスとして使用できます。

FailSafe 2.1.*x* クラスタを定義する場合は、通知に使用する電子メール・プログラムやクラスタに含めるノードなど、このパラメータ以外にも指定が必要な項目があります。ノードを定義するには、次の `cmgr` コマンドを使用します。

```
cmgr> define cluster apache-cluster
Enter commands, you may enter "done" or "cancel" at any time to exit

cluster apache-cluster? set notify_addr to root@localhost
cluster A? done
```

クラスタにノードを追加するには、次の `cmgr` コマンドを使用します。

```
cmgr> modify cluster apache-cluster
Enter commands, you may enter "done" or "cancel" at any time to exit

cluster apache-cluster? add node node1
cluster A? done
```

HAパラメータの設定

以下に、モニタおよびタイムアウトの値の設定に使用する FailSafe 1.2 の `ha.conf` ファイルのセクションの例を示します。FailSafe 2.1.*x* システムの設定の際に必要なパラメータは、太字で示されています。

```
system-configuration
{
      pwrfail = true
      ...
```

```
}


Node node1
{
...
heartbeat
{
        hb-private-ipname = 192.0.2.3
        hb-public-ipname = 54.3.252.6
        hb-probe-time = 6
        hb-timeout = 6
        hb-lost-count = 4
}
...
}
```

この ha.conf ノード定義からわかるように、FailSafe 1.2 では、パラメータ hb-probe-time、hb-timeout 、および hb-lost-count を定義して、モニタ・メッセージの送信頻度を指定する値や、応答のない時間がどの程度続いたときに異常と見なすかを判断する値を設定していました。FailSafe 2.1.*x* では、クラスタ内のノードのモニタに FailSafe 1.2 とは異なる方法が採用されており、クラスタ内の他方のノードに継続的にメッセージを送信すると同時に、そのノードから送信されるメッセージを継続的にモニタします。

このように 2 つのシステムではモニタ方法が異なるため、 ha.conf ファイルで設定した値と、FailSafe HA パラメータの設定時に FailSafe 2.1.*x* で設定するタイムアウトおよびハートビート周期は、1 対 1 では対応しません。ただし、異常が発生したとシステムが判断するまでの時間周期をほぼ同じ値で維持したい場合は、次の数式を使用して、ノードのタイムアウト周期に設定する値を決定できます。

node_timeout = (*probetime* + *timeout*) * *lostcount*

この数式を使用すると、ノード間の総通信時間が同じになるはずです。

FailSafe 2.1.*x* のタイムアウトはすべてミリ秒単位で、FailSafe 2.1.*x* の実行中に変更できます。また、クラスタ内の特定のノードのクラスタに対してタイムアウトを指定できます。

FailSafe 2.1.*x* には、長いタイムアウト値はありません。長いタイムアウト値に相当する値は、リソース・タイプの開始および停止アクション・モニタ・タイムアウトで設定します。リソース・タイプの開始、モニタ、および停止アクション・タイムアウトは、GUI または cmgr を使用して変更できます。

FailSafe 2.1.*x* で node1 の HA パラメータを変更するには、次の cmgr コマンドを使用します。

```
cmgr> modify ha_parameters on node node1 in cluster apache-cluster
Enter commands, when finished enter either "done" or "cancel"

node1 ? set node_timeout to 24000
node1 ? set heartbeat to 6000
node1 ? set run_pwrfail to true
node1 ? done
```

リソースの定義 : XLV ボリューム

以下に、FailSafe 1.2 の `ha.conf` ファイルでのボリューム定義の例を示します。同じボリュームを FailSafe 2.1.*x* システムのボリューム・リソースとして設定する際に必要なパラメータは、太字で示されています。

```
volume apache-vol
{
   server-node = node1
   backup-node = node2
   devname = apache-vol
   devname-owner = root
   devname-group = sys
   devname-mode = 600
}
```

この設定例の場合、FailSafe 2.1.*x* で同じボリュームを定義するときは、以下の値を使用します。

- ボリューム名: `apache-vol`
- デバイス・ファイルのオーナー・ユーザ名: `root`
- デバイス・ファイルのグループ名: `sys`
- デバイス・ファイルのアクセス権: `600`

XLV ボリューム・リソースを作成するには、以下の `cmgr` コマンドを使用します。

```
cmgr> define resource apache-vol of resource_type volume in cluster apache-cluster
Enter commands, when finished enter either "done" or "cancel"

resource apache-vol? set devname-owner to root
resource apache-vol? set devname-group to sys
resource apache-vol? set devname-mode to 600
resource apache-vol? done
```

リソースの定義 : XFS ファイルシステム

以下に、FailSafe 1.2 の ha.conf ファイルでの XFS ファイルシステム定義の例を示します。同じファイルシステムを FailSafe 2.1.*x* システムのファイルシステム・リソースとして設定する際に必要なパラメータは、太字で示されています。

```
filesystem apache-fs
{
   mount-point = /apache-fs
   mount-info
{
   fs-type = xfs
   volume-name = apache-vol
   mode = rw, noauto
}
}
```

この設定例の場合、FailSafe 2.1.*x* で同じファイルシステムを定義するときは、以下の値を使用します。

- リソース名 (マウント・ポイント): /apache-vol
- XLV ボリューム: apache-vol
- マウント・オプション: rw, noauto

ファイルシステム・リソースを作成するには、以下の cmgr コマンドを使用します。

```
cmgr> define resource /apache-fs of resource_type filesystem in cluster apache-cluster
Enter commands, when finished enter either "done" or "cancel"

resource /apache-fs? set volme-name to apache-vol
resource /apache-fs? set mount-options to "rw,noauto"
resource /apache-fs? done
```

リソースの定義 : IP アドレス

以下に、FailSafe 1.2 の ha.conf ファイルでの IP アドレス定義の例を示します。同じ IP アドレスを FailSafe 2.1.*x* システムの高可用性リソースとして設定する際に必要なパラメータは、太字で示されています。

```
interface-pair FDDI_1
{
   primary-interface = node-fxd
```

```
    secondary-interface = node2-fxd
    re-mac = false
    netmask = 0xffffff00
    broadcast-addr = 54.3.252.255

    ip-aliases = ( 54.3.252.7 )
}
```

この設定例の場合、FailSafe 2.1.*x* で同じ IP アドレスを定義するときは、以下の値を使用します。

- リソース名: `54.3.252.7`
- ブロードキャスト・アドレス: `54.3.252.255`
- ネットワーク・マスク: `0xffffff00`

IP アドレス・リソースを作成するには、以下の `cmgr` コマンドを使用します。

```
cmgr> define resource 54.3.252.7 of resource_type IP_address in cluster apache-cluster
Enter commands, when finished enter either "done" or "cancel"

resource 54.3.252.7? set interfaces to rns0
resource 54.3.252.7? set NetworkMask to 0xffffff00
resource 54.3.252.7? set BroadcasetAddress to 54.3.252.255
resource 54.3.252.7? done
```

FailSafe 2.1.*x* のその他のタスク

ノード、クラスタ、およびリソースを定義したら、リソース・グループを定義します。リソース・グループは、FailSafe 1.2 には該当するものがないタスクです。リソース・グループを定義するときは、リソース・グループに含まれるリソースと、異常発生時にリソース・グループのサービスを引継ぐノードを指定したフェイルオーバー・ポリシーを指定します。

リソース・グループの定義についての詳細は、184 ページの「GUI を使ったリソース・グループの定義」を参照してください。

システムの設定を完了したら、190 ページの「FailSafe HA サービスの開始」の手順に従って FailSafe サービスを開始できます。

ステータス

FailSafe 1.2 では、ha_admin -a コマンドを使用してシステムのステータスを表示します。FailSafe 2.1.*x* では、以下の方法でシステムのステータスを表示できます。

- GUI を使用して、クラスタの状態を継続的に監視できます。
- GUI または cmgr のいずれかを使用して、個々のリソース・グループ、ノード、またはクラスタのステータスを照会できます。
- cmgr に付属する /var/cluster/cmgr-scripts/ha Status スクリプトを使用して、設定内のすべてのクラスタ、ノード、リソース、およびリソース・グループを参照できます。

これらのタスクの実行についての詳細は、223 ページの「システムのステータス」を参照してください。

IRIS FailSafe 2.1.*x* ソフトウェア

この付録では、IRIS FailSafe 2.1.*x* 用に使用するシステムにインストールされるソフトウェアの概要を説明します。この付録は、以下の節で構成されています。

メモ: ソフトウェアのインストール手順は、53 ページの「ソフトウェアのインストール」で順を追って説明されています。

CD に含まれるサブシステム

IRIS FailSafe 2.1.*x* base CD のインストールには、約 10 MB が必要です。

326 ページの表C-1 に、IRIS FailSafe 2.1.*x* CD に含まれる FailSafe 2.1.*x* サブシステムを示します。

表C-1 IRIS FailSafe 2.1.*x* CD

用途	システム
IRIS FailSafe 2.1.*x*	`failsafe2 failsafe2.idb failsafe2.man failsafe2.sw failsafe2.books`（InSight バージョンのカスタマ・マニュアル）
FailSafe システム管理	`sysadm_failsafe2 sysadm_failsafe2.idb sysadm_failsafe2.man sysadm_failsafe2.sw`

base system administration (`sysadm_base`、cluster administration (`sysadm_cluster.sw` および `cluster_admin`)、cluster control (`cluster_control`)、cluster services (`cluster_services`)、java (`java_eoe`)、および Java Plug-in (`java_plugin`) は、ユーザが IRIX CD セットからインストールする必要があります。

EL-8+ multiplexer driver サブシステムは `el_serial`、 `el_serial.man`、および `el_serial.sw` で、EL-8+ multiplexer に付属の CD に収録されています。

プール内のサーバとワークステーション用のサブシステム

以下の表に、プール内のサーバとワークステーションに必要なサブシステムを示します。プールは、クラスタ設定に利用できるサーバ(ノード)のセット全体です。プールには、クラスタの管理に使用するサーバとワークステーションが含まれます。

表C-2 プール内のノード(サーバおよび GUI クライアント)に必要なサブシステム

製品	イメージおよびサブシステム	条件
Base system administration	sysadm_base.sw.dso	なし
Base system administration server	sysadm_base.sw.server	sysadm_base.sw.dso
Cluster administration GUI	sysadm_cluster.sw.server	sysadm_base.sw.server
IRIS FailSafe 2.1.*x* administration server	sysadm_failsafe2.sw.server	sysadm_base.sw.server sysadm_cluster.sw.server cluster_admin.sw.base cluster_services.sw.cli cluster_control.sw.cli failsafe2.sw.cli
Cluster administration	cluster_admin.sw cluster_control.sw	sysadm_base.sw.dso
Web-based administration	sysadm_failsafe2.sw.web	sysadm_failsafe2.sw.client sysadm_failsafe2.sw.server sysadmbase.sw.client java_eoe.sw、バージョン 3.1.1Web サーバ
EL-8+ multiplexer driver (multiplexer に付属の CD に収録)	el_serial el_serial.man el_serial.sw	

FailSafe クラスタ内のノード用のその他のサブシステム

以下の表に、クラスタ内のノードである各サーバに必要なその他のサブシステムを示します。クラスタとは、ネットワークにより相互接続された 1 つまたは複数のノードです。ノードは単一の UNIX イメージで、通常は個々のサーバです。ノードがメンバーになることができるクラスタは 1 つだけです。

表C-3 クラスタ内のノードに必要なその他のサブシステム

製品	イメージおよびサブシステム	条件
Highly available clustering software >	`cluster_services.sw`	`cluster_admin.sw` `cluster_control.sw`
IRIS FailSafe 2.1.*x* ソフトウェア	`failsafe2.sw`	`cluster_services.sw`

管理ワークステーション用のその他のサブシステム

GUI クライアントの実行に使用するワークステーションには、ワークステーションのタイプに応じたサブシステムをインストールする必要があります。以降の節に、以下のワークステーションにインストールするサブシステムを示します。

- IRIX 管理ワークステーション
- IRIX 以外の管理ワークステーション

IRIX 管理ワークステーション用のサブシステム

IRISconsoleなど、IRIX デスクトップから GUI クライアントを実行する際に使用するワークステーションには、以下の表に示すサブシステムをインストールします。

表C-4 IRIX 管理ワークステーションに必要なサブシステム

製品	サブシステム	条件
Cluster administration GUI	`sysadm_cluster.sw.client`	`sysadm_base.sw.client`

FailSafe GUI	`sysadm_failsafe2.sw.client`
	`sysadm_failsafe2.sw.desktop`
		`sysadm_base.sw.client`
		`sysadm_cluster.sw.client`
		`java_eoe.sw`、バージョン 3.1.1

Java Plug-in (Java をサポートする Web ブラウザから GUI クライアントを起動する目的でワークステーションを使用する場合にのみ必要)	`java_plugin.sw` `java_plugin.sw32`	Java をサポートする Web ブラウザ

IRIX 以外の管理ワークステーション用のサブシステム

IRIX 以外のワークステーションから、Java をサポートする Web ブラウザを使用して GUI を起動できます。

PCP (Peformance Co-Pilot) for FailSafe によってエクスポートされるメトリック

この付録では、pmdafsafe(1) によって実装されるメトリックを示します。

fsafe.srm.all.* メトリックは fsafe.srm.* メトリックと同じですが、 ha_srmd(1M) などのリソース自体が使用できない場合でも、すべてのリソースの最新の取得値を使用できる点が異なります。

表D-1 PCP のメトリック

メトリック	説明
fsafe.srm.status fsafe.srm.all.status	リソースに対して実行されたモニタ・イベントの最新のステータス。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.timeout fsafe.srm.all.timeout	リソースをモニタするために指定されているタイムアウト(ミリ秒単位)。
fsafe.srm.probes fsafe.srm.all.probes	ha_srmd(1M) の起動以降にリソースがモニタされた回数。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.recent.probes	fsafe.control.reset_srm によるデータ・コレクションのリセット以降にリソースがモニタされた回数。このノードでモニタ対象として設定されているすべてのリソースが対象となります。
fsafe.srm.timeouts fsafe.srm.all.timeouts	リソースが最後に使用可能だった時刻以降に、リソースが異常として宣言される前にタイムアウトしたリソース・モニタ・イベントの数。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.recent.timeouts	fsafe.control.reset_srm によるデータ・コレクションのリセット以降に、リソースが異常として宣言される前にタイムアウトしたリソース・モニタ・イベントの数。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.min_resp fsafe.srm.all.min_resp	特定のリソースに対するモニタ・イベントの完了に要したおおよその最短時間(ミリ秒単位)。モニタ対象に設定されているすべてのリソースが対象となります。

メトリック	説明
fsafe.srm.max_resp fsafe.srm.all.max_resp	特定のリソースに対するモニタ・イベントの完了に要したおおよその最長時間(ミリ秒単位)。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.last_resp fsafe.srm.all.last_resp	特定のリソースに対する最後のモニタ・イベントの完了に要したおおよその時間(ミリ秒単位)。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.cumm_timeouts fsafe.srm.all.cumm_timeouts	ha_srmd(1M) の起動以降にタイムアウトしたリソース・モニタ・イベントの累計数。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.recent.cumm_timeouts	fsafe.control.reset_srm によるデータ・コレクションのリセット以降にタイムアウトしたリソース・モニタ・イベントの累計数。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.histo_20 fsafe.srm.all.histo_20	ha_srmd(1M) の起動以降に、0 ミリ秒～fsafe.srm.timeout の応答時間の 0～20% の範囲内で受信されたモニタ・イベントの割合。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.recent.histo_20	fsafe.control.reset_srm によるデータ・コレクションのリセット以降に、0 ミリ秒～fsafe.srm.timeout の応答時間の 0～20% の範囲内で受信されたモニタ・イベントの割合。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.histo_40 fsafe.srm.all.histo_40	ha_srmd(1M) の起動以降に、0 ミリ秒～fsafe.srm.timeout の応答時間の 20～40% の範囲内で受信されたモニタ・イベントの割合。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.recent.histo_40	fsafe.control.reset_srm によるデータ・コレクションのリセット以降に、0 ミリ秒～fsafe.srm.timeout の応答時間の 20～40% の範囲内で受信されたモニタ・イベントの割合。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.histo_60 fsafe.srm.all.histo_60	ha_srmd(1M) の起動以降に、0 ミリ秒～fsafe.srm.timeout の応答時間の 40～60% の範囲内で受信されたモニタ・イベントの割合。このノードでモニタ対象に設定されているすべてのリソースが対象となります。

メトリック	説明
fsafe.srm.recent.histo_60	fsafe.control.reset_srm によるデータ・コレクションのリセット以降に、0 ミリ秒～fsafe.srm.timeout の応答時間の 40～60% の範囲内で受信されたモニタ・イベントの割合。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.histo_80 fsafe.srm.all.histo_80	ha_srmd(1M) の起動以降に、0 ミリ秒～fsafe.srm.timeout の応答時間の 60～80% の範囲内で受信されたモニタ・イベントの割合。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.recent.histo_80	fsafe.control.reset_srm によるデータ・コレクションのリセット以降に、0 ミリ秒～fsafe.srm.timeout の応答時間の 60～80% の範囲内で受信されたモニタ・イベントの割合。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.histo_100 fsafe.srm.all.histo_100	ha_srmd(1M) の起動以降に、0 ミリ秒～fsafe.srm.timeout の応答時間の 80～100% の範囲内で受信されたモニタ・イベントの割合。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.recent.histo_100	fsafe.control.reset_srm によるデータ・コレクションのリセット以降に、0 ミリ秒～fsafe.srm.timeout の応答時間の 80～100% の範囲内で受信されたモニタ・イベントの割合。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.frac_timeouts fsafe.srm.all.frac_timeouts	リソースが最後に使用可能だった時刻以降に、リソースが異常として宣言される前にタイムアウトしたモニタ・イベントの割合。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.recent.frac_timeouts	fsafe.control.reset_srm によるデータ・コレクションのリセット以降に、リソースが異常として宣言される前にタイムアウトしたモニタ・イベントの割合。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.frac_cumm_timeouts fsafe.srm.all.frac_cumm_timeouts	ha_srmd (1M) の起動以降にタイムアウトしたモニタ・イベントの累計数の割合。このノードでモニタ対象に設定されているすべてのリソースが対象となります。
fsafe.srm.recent.frac_cumm_timeouts	fsafe.control.reset_srm によるデータ・コレクションのリセット以降にタイムアウトしたモニタ・イベントの累計数の割合。このノードでモニタ対象に設定されているすべてのリソースが対象となります。

メトリック	説明
fsafe.srm.recent.timestamp	fsafe.control.reset_srm メトリックにストアを発行した後で、fsafe.srm.recent.* メトリックに対して統計の新しいコレクションが開始された時刻。
fsafe.config.clustername	このクラスタの名前。
fsafe.config.hostname	fsafe.config.clustername で指定されているクラスタ内のすべてのホストの名前。
fsafe.config.nnodes	fsafe.config.clustername で指定されているクラスタ内のノードの数。
fsafe.config.cms.interval	クラスタ・ハートビート・イベント周期(ミリ秒単位)。
fsafe.config.cms.timeout	クラスタ内のすべてのノードに対するハートビート・イベント・タイムアウト(ミリ秒)。
fsafe.config.cms.nbuckets	ノードあたりのハートビート・イベント応答周期の数。各周期は、ハートビート・イベント・タイムアウト (fsafe.config.cms.timeout) までの時間のハートビート・イベント周期 (fsafe.config.cms.interval) に等しい時間になります。
fsafe.control.debug>	このメトリックに 10 進の整数値が格納されている場合の fsafe PMDA 用のデバッグ・フラグ。これは、最終的には、fsafe PMDA のログ(通常は /var/adm/pcplog/fsafe.log)に記録される情報に影響を与えます。 このメトリックを読取ると、現在割当てられているデバッグ・フラグが 10 進の整数として戻されます。
fsafe.control.reset_cms	ha_cmsd(1M) から収集されたすべてのメトリックのデータ・コレクション統計をリセットします。このメトリックを格納すると、提供されたデータは無視されます。リセットが実行されるのは、このメトリックに格納を行った場合です。 このメトリックを読取ると、ゼロ (0) が戻されます。
fsafe.control.reset_srm	ha_srmd(1M) から収集されたすべてのメトリックのデータ・コレクション統計をリセットします。このメトリックを格納すると、提供されたデータは無視されます。リセットが実行されるのは、このメトリックに格納を行った場合です。 このメトリックを読取ると、ゼロ (0) が戻されます。

メトリック	説明
fsafe.control.retry	ha_cmsd(1M) または ha_srmd(1M) と通信した結果、これらが使用中であることが示された場合に、再試行できる回数を設定します。 読取っているメトリックや、必要なメトリックの値を取得するために必要なデーモンによっては、一部のデーモンの値が使用できないため、"Try again.Information not currently available." というメッセージが表示される可能性があります。このメトリックを調整することで、収集が中断されてこのメッセージが表示される前に、メトリックの収集時に実行できる再試行の回数を増やすことができます。再試行は、約 100 ms に 1 回実行されます。 このメトリックを設定しても、ha_cmsd(1M) または ha_srmd (1M) からのより深刻なエラーの fsafe PMDA での処理方法が変更されることはありません。 このメトリックを読取ると、現在の再試行カウントが戻されます。
fsafe.cms.expected	ha_cmsd(1M) の起動以降に、クラスタ内の各ノード(コレクタ・ホストを除く)で受信されると予想されるハートビート・イベントの数。
fsafe.cms.recent.expected	fsafe.control.reset_cms によるデータ・コレクションのリセット以降に、クラスタ内の各ノード(コレクタ・ホストを除く)で受信されると予想されるハートビート・イベントの数。
fsafe.cms.received	ha_cmsd(1M) の起動以降に、クラスタ内の各ノード(コレクタ・ホストを除く)で実際に受信されたハートビート・イベントの数。
fsafe.cms.recent.received	fsafe.control.reset_cms によるデータ・コレクションのリセット以降に、クラスタ内の各ノード(コレクタ・ホストを除く)で実際に受信されたハートビート・イベントの数。
fsafe.cms.missed	ha_cmsd(1M) の起動以降に、クラスタ内の各ノード(コレクタ・ホストを除く)で受信されていないと判断されたハートビート・イベントの数。
fsafe.cms.recent.missed	fsafe.control.reset_cms によるデータ・コレクションのリセット以降に、クラスタ内の各ノード(コレクタ・ホストを除く)で受信されていないと判断されたハートビート・イベントの数。
fsafe.cms.histo	ha_cmsd(1M) の起動以降に、クラスタ内の各ノード(コレクタ・ホストを除く)の別々のハートビート応答周期内に発生したイベントのハートビート・イベント応答時間のヒストグラム。

メトリック	説明
	ハートビート応答周期は、設定されているハートビート・イベント・タイムアウト (fsafe.config.cms.timeout) までの多くの周期に対して、設定されているハートビート・イベント周期 (fsafe.config.cms.interval) と等しくなるように定義します。
fsafe.cms.recent.histo	fsafe.control.reset_cms によるデータ・コレクションのリセット以降に、クラスタ内の各ノード (コレクタ・ホストを除く) の別々のハートビート応答周期内に発生したイベントのハートビート・イベント応答時間のヒストグラム。 ハートビート応答周期は、設定されているハートビート・イベント・タイムアウト (fsafe.config.cms.timeout) までの多くの周期に対して、設定されているハートビート・イベント周期 (fsafe.config.cms.interval) と等しくなるように定義します。
fsafe.cms.frac_received	ha_cmsd(1M) の起動以降に、クラスタ内の各ノードで、予想されたすべてのイベントに対して受信されたハートビート・イベントの割合。
fsafe.cms.recent.frac_received	fsafe.control.reset_cms によるデータ・コレクションのリセット以降に、クラスタ内の各ノードで、予想されたすべてのイベントに対して受信されたハートビート・イベントの割合。
fsafe.cms.frac_missed	ha_cmsd(1M) の起動以降に、クラスタ内の各ノードで、予想されたすべてのイベントに対して受信されていないと判断されたハートビート・イベントの割合。
fsafe.cms.recent.frac_missed	fsafe.control.reset_cms によるデータ・コレクションのリセット以降に、クラスタ内の各ノードで、予想されたすべてのイベントに対して受信されていないと判断されたハートビート・イベントの割合。
fsafe.cms.recent.timestamp	fsafe.control.reset_cms メトリックにストアを発行した後で、fsafe.cms.recent.* メトリックに対して統計の新しいコレクションが開始された時刻。
fsafe.cms.pernode.expected	ha_cmsd(1M) の起動以降に、クラスタ内の特定のノードに関して受信されると予想されるハートビート・イベントの数。
fsafe.cms.recent.pernode.expected	fsafe.control.reset_cms によるデータ・コレクションのリセット以降に、クラスタ内の特定のノードに関して受信されると予想されるハートビート・イベントの数。
fsafe.cms.pernode.received	ha_cmsd(1M) の起動以降に、クラスタ内の特定のノードに関して実際に受信されたハートビート・イベントの数。

メトリック	説明
fsafe.cms.recent.pernode.received	fsafe.control.reset_cms によるデータ・コレクションのリセット以降に、クラスタ内の特定のノードに関して実際に受信されたハートビート・イベントの数。
fsafe.cms.pernode.missed	ha_cmsd(1M) の起動以降に、クラスタ内の特定のノードに関して受信されていないと判断されたハートビート・イベントの数。
fsafe.cms.recent.pernode.missed	fsafe.control.reset_cms によるデータ・コレクションのリセット以降に、クラスタ内の特定のノードに関して受信されていないと判断されたハートビート・イベントの数。
fsafe.cms.pernode.histo	ha_cmsd(1M) の起動以降に、クラスタ内の特定のノードごとに別々のハートビート応答周期内に発生したイベントのハートビート・イベント応答時間のヒストグラム。 ハートビート応答周期は、設定されているハートビート・イベント・タイムアウト (fsafe.config.cms.timeout) までの多くの周期に対して、設定されているハートビート・イベント周期 (fsafe.config.cms.interval) と等しくなるように定義します。
fsafe.cms.recent.pernode.histo	fsafe.control.reset_cms によるデータ・コレクションのリセット以降に、クラスタ内の特定のノードごとに別々のハートビート応答周期内に発生したイベントのハートビート・イベント応答時間のヒストグラム。 ハートビート応答周期は、設定されているハートビート・イベント・タイムアウト (fsafe.config.cms.timeout) までの多くの周期に対して、設定されているハートビート・イベント周期 (fsafe.config.cms.interval) と等しくなるように定義します。
fsafe.cms.pernode.frac_received	ha_cmsd(1M) の起動以降に、クラスタ内の特定ノードについて、すべての予想イベントに対して受信されたハートビート・イベントの割合。
fsafe.cms.recent.pernode.frac_received	fsafe.control.reset_cms によるデータ・コレクションのリセット以降に、クラスタ内の特定のノードについて、すべての予想イベントに対して受信されたハートビート・イベントの割合。
fsafe.cms.pernode.frac_missed	ha_cmsd(1M) の起動以降に、クラスタ内の特定のノードについて、すべての予想イベントに対して受信されていないと判断されたハートビート・イベントの割合。

メトリック	説明
fsafe.cms.recent.pernode.frac_missed	fsafe.control.reset_cms によるデータ・コレクションのリセット以降に、クラスタ内の特定のノードについて、すべての予想イベントに対して受信されていないと判断されたハートビート・イベントの割合。

用語集

FailSafe データベース

クラスタ・データベースを参照してください。

FailSafe メンバーシップ

FailSafe がリソース・グループをオンラインにできるクラスタ内の FailSafe ノードのリスト。FailSafe メンバーシップは、CXFS メンバーシップとは異なります。CXFS についての詳細は、『CXFS Version 2 Software Installation and Administration Guide』を参照してください。

`fs2d` **データベース・メンバーシップ**

ユーザ空間メンバーシップとも呼びます。`fs2d` にアクセス可能で、クラスタ・データベースの更新を受信できるプール内のノードのグループ。プール内で定義されたノードのサブセットの場合があります。

LUN

論理ユニット番号。

アクション・スクリプト

リソースの開始、モニタ、および停止の方法を決定するスクリプトのセット。各リソース・タイプに対して 1 つのセットのアクションを指定する必要があります。指定できるアクション・スクリプトのセットは、 `exclusive`、`start`、`stop`、 `monitor`、および `restart` です。

アクティブ／バックアップ設定

すべてのリソース・グループが同じプライマリ・ノードを使用する設定。バックアップ・ノードでは、フェイルオーバーが実行されるまで高可用性サービスは実行されません。

依存リスト

リソース依存性または**リソース・タイプ依存性**を参照してください。

オーナー TTY 名

システム・コントローラのシリアル・ケーブルが接続されている**オーナー・ホスト**の端末ポート (TTY) のデバイス・ファイル名。オーナー・ホストからノードをリモートで制御できるよう、このケーブルのもう一方はシステム・コントローラ・ポートを持つノードに接続します。

オーナー・ホスト

ノードの電源サイクルなどのノードの制御をリモートで行うことができるシステム。ランタイム時には、オーナー・ホストがプール内のノードとして定義されていなければなりません。

オフライン・リソース・グループ

クラスタ内の高可用性ではないリソース・グループ。リソース・グループをオフライン状態にするには、FailSafe はそのグループを停止し（必要な場合）、グループのモニタを停止します。オフライン・リソース・グループはノードで実行できますが、FailSafe はこのグループに対して制御を行いません。グループをオフラインにするときにクラスタ管理者が「**分離のみ**(detach only)」オプションを指定した場合、グループは停止されませんが、グループのモニタは停止されます。

オンライン・リソース・グループ

クラスタ内の高可用性であるリソース・グループ。リソース・グループの可用性を低下させる異常を検出すると、FailSafe は、そのリソース・グループをクラスタ内の別のノードに移動します。リソース・グループをオンライン状態にするには、FailSafe はそのグループを開始し（必要な場合）、グループのモニタを開始します。グループをオンラインにするときにクラスタ管理者が「**接続のみ**(attach only)」オプションを指定した場合、グループは開始されませんが、グループのモニタは開始されます。

開始／停止順位

各リソース・タイプには開始／停止順序があり、正の整数で指定します。リソース・グループでは、リソース・タイプの開始／停止順位によって、FailSafe がグループをオンラインにするときのリソースの開始順序と、グループをオフラインにするときの停止順序が決まります。グループのリソースは、値が小さい順に開始され、値が大きい順に停止されます。同じタイプのリソースの開始および停止順序は決まっていません。たとえば、 `volume` リソース・タイプの順序が 10 で、`filesystem` リソース・タイプの順序が 20 の場合、FailSafe がリソースをオンラインにするときは、グループのすべてのボリューム・リソースが開始された後、グループのすべてのファイルシステムが開始されます。

キー／値属性

特定のリソース・タイプに対して定義する必要がある情報のセット。たとえば、リソース・タイプ `filesystem` の 1 つのキー／値のペアが *mount_point=/fs1* であるとします。この場合は *mount_point* がキーで、*fs1* が、定義する特定のリソースに固有の値です。値によっては、`string` または `integer` のいずれかのデータ型を指定します。前の例では、値 *fs1* のデータ型として `string` を指定します。

クラスタ

クラスタとして定義されているプール内のノードのセット。クラスタは単純な名前で識別されます。この名前は、プール内で一意でなければなりません。クラスタ内のすべてのノードはプールにも存在しますが、

プール内のすべてのノードがクラスタに存在するとはかぎりません。つまり、クラスタは、プール内のノードのサブセットで構成されている場合があります。各プールのクラスタは 1 つだけです。

クラスタ管理者

クラスタの管理とメインテナンスの担当者。

クラスタ・データベース

すべてのリソース、リソース・タイプ、リソース・グループ、フェイルオーバー・ポリシー、ノード、およびクラスタに関する設定情報が含まれます。

クラスタ・プロセス・グループ

分散型アプリケーションのアプリケーション・インスタンスのグループで、連携してサービスを提供します。

たとえば、各ノードの分散ロック・マネージャのインスタンスはプロセス・グループを形成します。プロセス・グループを形成することで、メンバーシップに加え、信頼性の高い順序付きの原始的な通信サービスを実現できます。UNIX のプロセス・グループとクラスタ・プロセス・グループの間に関係はありません。

コレクタ・ホスト

統計を収集したり、PCP (Peformance Co-Pilot) for FailSafe がコレクタ・エージェントをインストールする、FailSafe クラスタ自体の中にあるノード。

コントロール・ネットワーク

ネットワーク・インタフェース（通常は Ethernet）を通じてノードを接続するネットワーク。接続されているノードにネットワークを通じてハートビート・メッセージとコントロール・メッセージを送信することによって、FailSafe がクラスタの高可用性を維持できるようにします。FailSafe では、コントロール・ネットワーク上で最も優先度が高いネットワーク・インタフェースが使用されます。コントロール・ネットワーク上にある優先度の高いすべてのネットワーク・インタフェースに異常が発生した場合は、優先度の低いネットワーク・インタフェースが使用されます。

ノードには、ハートビート・メッセージ用とコントロール・メッセージ用にそれぞれ少なくとも 1 つのコントロール・ネットワークが必要です（ハートビートとコントロール・メッセージの両方に同じインタフェースを使用するように設定できます）。1 つのノードで 8 つを超えるコントロール・ネットワーク・インタフェースを使用することはできません。

コントロール・メッセージ

ノードやリソース・グループに対する操作を要求したり、これらの情報を配布するためにクラスタ・ソフトウェアがノード間で送信するメッセージ。FailSafe では、コントロール・メッセージは、ノードとグループの高可用性を確保する目的で送信されます。コントロール・メッセージおよびハートビート・メッセージは、

コントロール・ネットワークに接続されているノードのネットワーク・インタフェースを通じて送信されます。ノードは複数のコントロール・ネットワークに接続できます。

再 MAC

あるネットワーク・インタフェースの物理 MAC (Medium Access Control) アドレスを別のインタフェースに移動するプロセス。これは、`macconfig` コマンドを使用して行われます。

システム・コントローラ・ポート

リモートでノードの電源サイクルを行う方法を提供するノード上のポート。クラスタ・データベース (CDB) でシステム・コントローラ・ポートが有効または無効のどちらに設定されているかによって、システム・コントローラ・ポートで FailSafe が処理を実行できるかどうかが決まります。(ポートが有効な場合は、シリアル・ケーブルでそのポートをオーナー・ノードである別のノードに接続する必要があります)。システム・コントローラ・ポートの情報はプール内のノードの場合はオプションですが、そのノードがクラスタに追加される場合は必須です。システム・コントローラ・ポートの情報がないと、そのノードで実行されているリソースは高可用性になりません。

初期フェイルオーバー・ドメイン

フェイルオーバー・ポリシーを最初に作成するときに管理者が定義するノードの順序付きリスト。クラスタを初めて起動するときに使用されます。初期フェイルオーバー・ドメインによって指定される順序付きリストは、**フェイルオーバー・スクリプト**によって**ランタイム・フェイルオーバー・ドメイン**に変換されます。ランタイム・フェイルオーバー・ドメインはフェイルオーバー属性と共に使用され、リソース・グループを存在させるノードを決定します。異常が発生するたびに、フェイルオーバー・スクリプトは現在のランタイム・フェイルオーバー・ドメインを利用し、場合によっては変更します。初期フェイルオーバー・ドメインが再度使用されることはありません。ランタイムの状態やフェイルオーバー・スクリプトの内容によっては、初期フェイルオーバー・ドメインとランタイム・フェイルオーバー・ドメインは同一であることもあります。**ランタイム・フェイルオーバー・ドメイン**も参照してください。

タイプ固有属性

特定のリソース・タイプのリソースを定義するために使用される必須の情報。たとえば、タイプが `filesystem` のリソースの場合は、リソースのボリューム名 (ファイルシステムの場所) の属性を入力して、そのファイルシステムのマウント方法のオプション (たとえば、読取り可能および書込み可能) を指定する必要があります。

タイブレーカー・ノード

FailSafe のタイブレーカーとして識別されるノード。クラスタ内のちょうど半数のノードが動作していて相互に通信できる場合にクラスタの FailSafe クラスタ・メンバーシップを計算する処理で使用されます。タ

イブレーカー・ノードが指定されていない場合は、クラスタの中でノード ID が最も小さいノードがタイブレーカー・ノードとして使用されます。

通知コマンド

クラスタ、ノード、およびリソース・グループの変更や異常をクラスタ管理者に通知するために使用されるコマンド。このコマンドは、クラスタ内のすべてのノードに存在する必要があります。

データベース

クラスタ・データベースを参照してください。

電源異常モード

電源異常モードが `on` の場合、FailSafe は、ノードのシステム・コントローラからの応答を追跡して、ノードにリセット要求を送信します。これらの要求でノードを正常にリセットできなかった場合、FailSafe は、発見的アルゴリズムを使用して、マシンの電源が切れているかどうかを確認します。発見的アルゴリズムが成功した場合は、リモート・マシンが正常にリセットされたと想定します。電源異常モードが `off` の場合、発見的アルゴリズムは使用されず、FailSafe はノードの電源異常を検出できません。

ノード

単一の IRIX カーネル・イメージ。通常、ノードは個々のコンピュータです。この意味でのノードという用語は、SGI 2000 または SGI Origin 3000 システムのノードとは異なる意味を持ちます。

ノード ID

ノードを一意に識別する 16 ビットの正の整数。クラスタ管理者がノード ID を割当てていない場合は、ノードを定義する際に FailSafe によってノード ID が割当てられます。いったん割当てられたノード ID は変更できません。

ノード・タイムアウト

この時間内にノードからハートビートが受信されない場合、ノードはダウンしていると見なされます。FailSafe を正しく動作させるには、ノードのタイムアウト値をハートビート周期の 10 倍以上に設定してください。この値より低く設定すると、フェイルオーバーが誤って実行されることがあります。

ハートビート周期

ハートビート・メッセージの周期。FailSafe を正しく動作させるには、ノードのタイムアウト値をハートビート周期の 10 倍以上に設定してください。この値より低く設定すると、フェイルオーバーが誤って実行されることがあります。ハートビートの数が多いほど（ハートビート周期が小さいほど）、ネットワークの速度が遅

くなる可能性が高くなります。逆に、ハートビートの数が小さいほど（ハートビート周期が大きいほど）、リソースの可用性が減少する可能性が高くなります。

ハートビート・メッセージ

クラスタ・ソフトウェアがノード間で送信する、ノードが動作中であることを示すメッセージ。コントロール・メッセージおよび**ハートビート・メッセージ**は、コントロール・ネットワークに接続されているノードのネットワーク・インタフェースを通じて送信されます。ノードは複数のコントロール・ネットワークに接続できます。

プール

ネットワークによって相互に結合され、FailSafe のノードとして定義されているノードのセット全体。通常、これらのノードは互いに近い位置にあり、同じ目的に使用されます。プール内の各ノードには、複製されたクラスタ・データベースが格納されています。

クラスタに追加できるノードはすべてプールの一部ですが、プール内のすべてのノードをクラスタの一部にする必要はありません。プールは 1 つだけです。ほかのプールを存在させることはできますが、各プールはお互いから分離した状態になり、ノードやクラスタ定義は共有されません。

フェイルオーバー

フェイルオーバー・ポリシーに従って**リソース・グループ**を**ノード**に割当てるプロセス。フェイルオーバーは、リソース異常が発生したとき、FailSafe メンバーシップが変更されたとき（ノードに異常が発生した場合やノードが開始した場合など）、または管理者が手動で要求したときに開始できます。

フェイルオーバー・スクリプト

フェイルオーバー・ポリシーのコンポーネント。**ランタイム・フェイルオーバー・ドメイン**を生成して FailSafe プロセスに返します。このプロセスによりフェイルオーバー属性が適用された後、返されたフェイルオーバー・ドメインの中で、現在の FailSafe メンバーシップにも存在する最初のノードが選択されます。

フェイルオーバー属性

クラスタ内のリソース・グループの割当てを制御する文字列。管理者は、システム定義属性（`Auto_Failback` や `Controlled_Failback` など）を指定する必要があります。また、オプションでサイト固有属性を指定できます。

フェイルオーバー・ドメイン

特定の**リソース・グループ**を割当てることができるノードの順序付きリスト。フェイルオーバー・ドメインにリストされているノードは同じクラスタ内に存在しなければなりませんが、フェイルオーバー・ドメインにクラスタのすべてのノードを含める必要はありません。**初期フェイルオーバー・ドメイン**は、フェイルオーバー・ポリシーを作成するときに管理者が定義します。このリストは、**フェイルオーバー・スクリプト**によって**ラン**

タイム・フェイルオーバー・ドメインに変換されます。ランタイム・フェイルオーバー・ドメインとは、フェイルオーバー・ノードの選択に実際に使用されるドメインです。FailSafe は、ランタイム・フェイルオーバー・ドメインを格納し、次のフェイルオーバー・スクリプト呼出しへの入力として使用します。フェイルオーバー・スクリプトの内容によっては、初期フェイルオーバー・ドメインとランタイム・フェイルオーバー・ドメインは同一であることもあります。一般的に FailSafe では、特定のリソース・グループは、ランタイム・フェイルオーバー・ドメインにリストされていて、FailSafe メンバーシップにも含まれる最初のノードに割当てられます。この割当てがいつ行われるかは、**フェイルオーバー属性**によって変わります。

フェイルオーバー・ポリシー

フェイルオーバー先のノードを決定するときに FailSafe で使用される方法。フェイルオーバー・ポリシーは、**フェイルオーバー・ドメイン**、**フェイルオーバー属性**、および**フェイルオーバー・スクリプト**で構成されます。フェイルオーバー・ポリシーの名前は、**プール**内で一意でなければなりません。

プラグイン

リソース・タイプおよびアクション・スクリプトを含む、アプリケーションを高可用性にするために必要なソフトウェアのセット。プラグインには、base FailSafe リリースに付属するプラグイン、SGI から購入できるオプションのプラグイン、『IRIS FailSafe Version 2 Programmer's Guide』の手順に従って作成できるカスタム・プラグインがあります。

プロセス・メンバーシップ

プロセス・グループを形成する、クラスタ内のプロセス・インスタンスのリスト。各ノードで複数のプロセス・グループを使用できます。

ポート・パスワード

システム・コントローラ・ポートのパスワード。通常は、ファームウェアまたはジャンパ・ワイヤで一度だけ設定します。ポート・パスワードは、ノードの root パスワードとは異なります。

モニタ・ホスト

ディスプレイが接続されていて IRIS Desktop が実行されているワークステーション。このワークステーションには、PCP for FailSafe によってモニタ・クライアントがインストールされています。

ユーザ空間メンバーシップ

`fs2d` **データベース・メンバーシップ**を参照してください。

ランタイム・フェイルオーバー・ドメイン

フェイルオーバー・スクリプトによる変更に従って、異常発生時にリソース・グループを実行できるノードの順序付きセット。ランタイム・フェイルオーバー・ドメインは、フェイルオーバー属性と共に使用され、リソース・グループを存在させるノードを決定します。**初期フェイルオーバー・ドメイン**も参照してください。

リソース

クライアントまたはその他のリソースにサービスを提供する単一の物理的または論理的な実体。たとえば、単一のディスク・ボリューム、特定のネットワーク・アドレス、Web サーバなどのアプリケーションがリソースの場合があります。通常、リソースは、長い間隔で見るとクラスタ内の複数のノードで使用できますが、同時に 1 つのノードにしか割当てることはできません。リソースは、リソース名およびリソース・タイプによって識別されます。依存リソースは、同じリソース・グループの一部でなければなりません。また、依存リソースは、リソース依存リストで識別されます。

リソース依存性

あるリソースにとって別のリソースの存在が必要であるような状態。

リソース依存リスト

あるリソースが依存するリソースのリスト。リソース・インスタンスをリソース・グループに追加するには、各リソース・インスタンスに、リソース・タイプ依存性を満足するリソース依存性が必要です。

リソース・キー

特定のリソース・タイプのリソースを定義する変数。アクション・スクリプトは、この情報を使用して、このリソース・タイプのリソースを開始、停止、およびモニタします。

リソース・グループ

リソースの集合。リソース・グループは単純な名前で識別されます。この名前は、クラスタ内で一意でなければなりません。リソース・グループを重複させることはできません。つまり、2 つのリソース・グループに同じリソースを含めることはできません。すべての相互依存リソースは、同じリソース・グループの一部でなければなりません。リソース・グループのいずれかのリソースが目的の用途に使用できなくなった場合は、リソース・グループ全体が利用不可と見なされます。したがって、リソース・グループがフェイルオーバーの単位になります。

リソース・タイプ

リソースの特定のクラス。フェイルオーバーの目的では、特定のリソース・タイプのリソースはすべて同じ方法で処理できます。各リソースは、常に 1 つのリソース・タイプのインスタンスです。リソース・タイプは単純な名前で識別されます。この名前は、クラスタ内で一意でなければなりません。リソース・タイプは、

特定のノードに対して定義したり、クラスタ全体に対して定義できます。ノードに対して定義されているリソース・タイプは、同じ名前のクラスタ全体のリソース・タイプ定義をオーバーライドします。このため、個々のノードは、クラスタ全体のリソース・タイプ定義のグローバル設定をオーバーライドできます。

リソース・タイプ依存性

あるリソース・タイプが依存するリソース・タイプのセット。たとえば、filesystemリソース・タイプはvolumeリソース・タイプに依存し、Netscape_web リソース・タイプは、filesystem および IP_address リソース・タイプに依存します。

リソース・タイプ依存リスト

あるリソース・タイプが依存するリソース・タイプのリスト。

リソース名

リソース・タイプの特定のインスタンスを識別する単純な名前。リソース名は、リソース・タイプ内で一意でなければなりません。

ログ・グループ

同じログ設定を使用する 1 つまたは複数の FailSafe プロセスのセット。通常、ログ・グループは gcd などの 1 つのデーモンに対応します。

ログ設定

ログ設定は、**ログ・レベル**と**ログ・ファイル**の 2 つの部分で構成され、どちらも**ログ・グループ**に関連付けられます。クラスタ管理者は、ログ出力の場所と量をカスタマイズしたり、すべてのノードまたは 1 つのノードだけを対象にしたログ設定を指定できます。たとえば、 crsd ログ・グループを設定して、ノード foo だけについては詳細レベル 10 のメッセージを /var/cluster/ha/log/crsd-foo ログに記録し、その他のすべてのノードについては最小レベル 1 のメッセージだけを crsd ログに書込むことができます。

ログ・ファイル

特定の**ログ・グループ**に関する通知が含まれるファイル。ログ・ファイルは、ログ・グループの**ログ設定**の一部です。デフォルトでは、ログ・ファイルは /var/cluster/ha/log ディレクトリにありますが、この場所はクラスタ管理者がカスタマイズできます。メモ: FailSafe は、通常の処理と重大なエラーの両方を /var/adm/SYSLOG のほかに特定のログ・グループの個々のログにも記録します。

ログ・レベル

関連付けられているログ・グループのログ・ファイルに FailSafe が書込むログ・メッセージの数を制御する数字。ログ・レベルは、ログ・グループのログ設定の一部です。

索引

こ

さ

し

す

せ

れ

ろ