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

007–3901–004JP

制作スタッフ

著作 Jenn Byrnes、Steven Levine、Susan Ellis

編集 Rick Thompson

イラスト Dany Galgani

製作 Glen Traefald

協力エンジニア Vidula Iyer、Ashwinee Khaladkar、Tony Kavadias、Linda Lait、Michael Nishimoto、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、および 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。

- 新しい `cluster_status` コマンド。165 ページの「システムのステータス」を参照してください。
- パーティション ID 情報。88 ページの「ノードの定義」を参照してください。
- IRIS FailSafe と CXFS の同時実行環境でサポートされているノードの設定。44 ページの「CXFS との同時実行」を参照してください。
- サポートされているノードの数: 16 CXFS ノード、および同時実行での最高 8 つまでの IRIS FailSafe ノード。10 ページの「IRIS FailSafe クラスタのハードウェア・コンポーネント」を参照してください。
- Node_Failures_Only 属性。133 ページの「フェイルオーバー属性」を参照してください。
- サポートされているディスク・ストレージとして TP9100 および TP9400 を追加。10 ページの「IRIS FailSafe クラスタのハードウェア・コンポーネント」を参照してください。
- ログ・ファイル管理に関するセクションを追加。188 ページの「ログ・ファイル管理」を参照してください。
- マニュアル全体における多数の説明、注記、および更新。
- 最初の日本語版（上記は英語版前版からの改訂履歴です）。

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

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 2.1.1 に関して作成されており、IRIX 6.5.10 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「IRIS FailSafe システムの概要」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「IRIS FailSafe の設定計画」では、IRIS FailSafe クラスタの設定計画の方法を説明します。
- 第3章「IRIS FailSafe ソフトウェアのインストールとシステムの準備」では、ノードを IRIS FailSafe 用に準備するために、IRIS FailSafe 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「IRIS FailSafe 管理ツール」では、IRIS FailSafe システムの管理に使用できるクラスタ・マネージャ・ツールについて説明します。
- 第5章「IRIS FailSafe の設定」では、FailSafe システムを設定するための管理タスクの実行方法を説明します。
- 第6章「IRIS FailSafe の設定例」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ための管理タスクの実行方法を説明します。
- 第8章「IRIS FailSafe 設定のテスト」では、設定した IRIS FailSafe システムのテスト方法を説明します。
- 第9章「IRIS FailSafe の回復」では、FailSafe で使用されるログ・ファイルと、FailSafe システムの問題の評価方法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A「IRIS FailSafe 1.2 から IRIS FailSafe 2.X へのアップグレード」では、システムを IRIS FailSafe 1.X から IRIS FailSafe 2.X にアップグレードするために実行する手順について説明します。
- 付録B「IRIS FailSafe 2.1 ソフトウェア」では、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の概要を説明します。
- 付録C「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

IRIS FailSafe システムには、このガイドのほかに以下のドキュメントも付属します。

- 『IRIS FailSafe Version 2 Programmer's Guide』
- 『Performance Co-Pilot User's and Administrator's Guide』
- 『CXFS Software Installation and Administration 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 のリファレンス・ページは、以下のとおりです。

- `cdbBackup`(1M)
- `cdbRestore`(1M)
- `cluster_mgr`(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
failsafe2	IRIS FailSafe 2.1
failsafe2_nfs	IRIS FailSafe NFS
failsafe2_web	IRIS FailSafe Netscape Web
failsafe2_informix	IRIS FailSafe INFORMIX
failsafe2_oracle	IRIS FailSafe Oracle
failsafe2_samba	IRIS FailSafe Samba
cluster_admin	Cluster administration services
cluster_control	Node control services
cluster_services	Cluster services

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。次のリストでは、セクション ID について説明します。

1	ユーザ・コマンド
1B	BSD からポートされたユーザ・コマンド
2	システム・コール
3	ライブラリ・ルーチン、マクロ、および操作定義
4	デバイス（特殊ファイル）
4P	プロトコル
5	ファイル形式
7	その他のトピック
7D	DWB 関連情報
8	管理者コマンド

内部ルーチン（たとえば、`_assign_asgcmd_info()` ルーチンなど）には、関連付けられたマン・ページがないものもあります。

変数	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。

ご意見とお問い合わせ先

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、弊社までお問い合わせください。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, USA
- 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。

IRIS FailSafe システムの概要

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。この章の主な節は、以下のとおりです。

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（継続的な可用性）。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。

Silicon Graphics® 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)。IRIS FailSafe を使用すると、以下のトポロジーのいずれかで高可用性システムを設定できます。

- 基本 2 ノード設定
- リング設定
- スター設定。複数のノードで実行されている複数のアプリケーションが、1 つのノードによってバックアップされます。
- 対称プール設定

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。

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。

IRIS FailSafe 高可用性環境において、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、IRIS 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。IRIS 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。

IRIS 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。

IRIS FailSafe の高可用性サービスは、高可用性リソースおよび高可用性アプリケーションという 2 つのグループに分類されます。高可用性リソースには、IRIS FailSafe 用に設定されているネットワーク・インタフェース、XLV 論理ボリューム、および XFS ファイルシステムが含まれます。SGI では、一部の高可用性アプリケーション向けに IRIS FailSafe ソフトウェア・オプションを開発しています。このオプションのソフトウェアには、以下が含まれます。

- IRIS FailSafe NFS
- IRIS FailSafe Web（Netscape サーバ用）
- IRIS FailSafe INFORMIX
- IRIS FailSafe Oracle
- IRIS FailSafe Samba

IRIS FailSafe には、その他の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。IRIS FailSafe クラスタに高可用性アプリケーションを追加する場合は、モニタおよびフェイルオーバーの機能を処理するためのスクリプトを作成してください。このようなスクリプトの開発についての情報は、『IRIS FailSafe Version 2 Programmer's Guide』で説明されています。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い合わせください。

IRIS FailSafe システムのコンポーネントおよび概念

IRIS FailSafe システムは、相互関係が定義されたさまざまなコンポーネントで構成されるクラスタ・システムです。IRIS FailSafe を使用して高可用性サービスを設定したりモニタするには、システムのコンポーネントに関する以下の概念と定義を理解している必要があります。これらは、IRIS FailSafe システムを定義する実体および属性であり、すべてのシステム管理タスクはこれらの概念に基づいています。

ノード

ノードは、単一の UNIX カーネル・イメージです。通常、ノードは個々のコンピュータです。

ノードは、多数のディスクから構成されたストレージ・エリア・ネットワーク (SAN: Storage Area Network) に接続されている可能性があります。

この意味でのノードという用語は、Origin 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、ネットワークによって相互に結合され、クラスタ・データベースでノードとして定義されているノードのセット全体です。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設定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。

クラスタ・ソフトウェアでは、ネットワークを使用して、クラスタ・データベース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。SGI では、すべてのノードを同じサブネットに置くことを推奨しています。

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、そのノードをリセットすることがあります。

注意: 不必要なリセットを避けるため、SGI では、クラスタ通信専用のプライベート・ネットワークをお薦めしています。このドキュメントの以降の部分は、プライベート・ネットワークを指しています。

クラスタ

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。

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。

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。これは、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。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 Software Installation and Administration Guide』を参照してください。
- `fs2d` メンバーシップ。これは、ユーザ空間メンバーシップとしても知られており、`fs2d` にアクセス可能で、クラスタ設定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です。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。

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、特定の時点では 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。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・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。

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。`web1` 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、`web1` 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。

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。適切なリソース・タイプがない場合は、このガイドの手順に従って、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 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。

リソース依存リスト

リソース依存リストは、あるリソースが依存するリソースのリストです。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・タイプ依存リスト

リソース・タイプ依存リストは、あるリソース・タイプが依存するリソース・タイプのリストです。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。

たとえば、ファイルシステム・インスタンス fs1 がボリューム vol1 にマウントされているとします。リソース・グループに fs1 を追加するには、vol1 に依存するよう fs1 を定義する必要があります。FailSafe では、ファイルシステム・インスタンスの依存リストに 1 つのボリューム・インスタンスが必要であることだけを識別します。この要求は、リソース・タイプ依存リストから推測されます。

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 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。このプロセス (ha_fsd) 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するスクリプトのセットです。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。

IRIS FailSafe 追加機能

IRIS 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。

- 動的に管理されるアプリケーション・モニタ

  FailSafe では、FailSafe を実行し続けながら、アプリケーションの FailSafe 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。

- 動的に管理される FailSafe リソース

  FailSafe では、FailSystem がオンラインの間にリソースを追加できます。

- 動的に管理される FailSafe アップグレード

  FailSafe では、FailSafe クラスタ全体を停止せずに、一度に 1 つのノードで 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

IRIS FailSafe 環境では、細粒度フェイルオーバーを指定できます。

FailSafe バージョン 2 では、フェイルオーバーの単位はリソース・グループで、ノード全体ではありません。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。

IRIS FailSafe 管理

すべての IRIS FailSafe 管理タスクは、IRIS FailSafe クラスタ・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。FailSafe 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、FailSafe 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。

IRIS FailSafe 管理タスクは、IRIS FailSafe クラスタ・マネージャ CLI を使用して直接実行することもできます。IRIS FailSafe クラスタ・マネージャ CLI 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。

IRIS FailSafe クラスタ・マネージャ・ツールについての詳細は、第4章「IRIS FailSafe 管理ツール」を参照してください。

IRIS FailSafe の設定および管理タスクについての詳細は、第5章「IRIS FailSafe の設定」および第7章「IRIS FailSafe のシステム運用」を参照してください。

IRIS FailSafe クラスタのハードウェア・コンポーネント

図1-1に、IRIS 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。

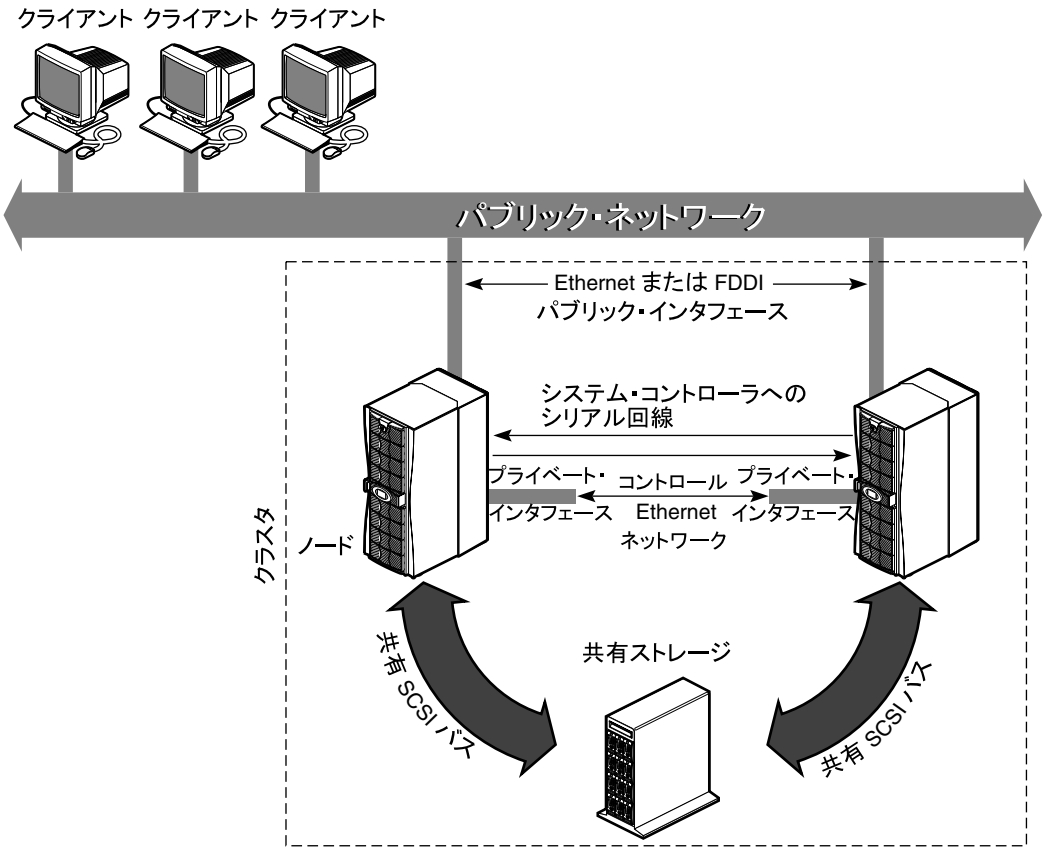

図1-1 IRIS FailSafe システム・コンポーネントのサンプル

IRIS FailSafe システムのハードウェア・コンポーネントは、以下のとおりです。

- 最高 8 つまでの Origin 2000 ノード、Origin 200 ノード、または 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、IRIS 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

  IRIS FailSafe システムのノードは、共有 Fast-Wide SCSI バス上のデュアルホスト・ディスク・ストレージを共有します。このバスは、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。

  – SCSI ディスクが搭載された Origin JBOD 周辺機器の格納装置

  – Origin 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

- 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: IRIS 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。

IRIS FailSafe ディスク接続

IRIS 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。

IRIS FailSafe でサポートされている設定

IRIS 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

IRIS 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。

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。

高可用性リソース

この節では、IRIS FailSafe システムで提供される高可用性リソースについて説明します。

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、IRIS 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。IRIS FailSafe システムがこれらの異常ポイントを処理または取り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、IRIS 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、クラスタ内のすべての高可用性サービスで共有できます。

IRIS 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 IRIS 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。

IRIS 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。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 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 IRIS 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、IRIS FailSafe システムを設定するときに指定します。IRIS 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。IRIS FailSafe では、再 MAC (MAC アドレス偽装)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、IRIS 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。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、「ネットワーク・インタフェースと 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

IRIS FailSafe システムには、1 つまたは複数の Origin FibreVault RAID ストレージ・システムの形式の共有 SCSI ベースのストレージを含めることができます。また、プレックス化されたディスクを使用する SCSI ディスクのある FibreVault 周辺機器の格納装置も含めることができます。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 (FailSafe 2.1 リリースでは) CXFS ファイルシステムを使用する必要があります。高可用性アプリケーション用のすべてのデータは、共有ディスクの XLV 論理ボリュームに格納しなければなりません。また、CXFS ファイルシステムを使用する場合は、XVM 論理ボリュームに格納しなければなりません。

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。

FibreVault のディスク上でプレックス化された XLV 論理ボリュームの場合は、XLV システムによって冗長性が提供されます。ディスク異常に対して IRIS FailSafe システム・ソフトウェアによる処理は必要はあ

りません。ディスク・コントローラに異常が発生した場合は、IRIS FailSafe システム・ソフトウェアによってフェイルオーバー・プロセスが開始されます。

IRIS 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 ファイルシステムまたは XVM ボリュームの開始、停止、またはモニタを直接行うことはありません。また、CXFS ファイルシステムを FailSafe リソース・グループに追加しないでください。IRIS FailSafe および CXFS の同時実行についての詳細は、44 ページの「CXFS との同時実行」を参照してください。

図1-2に、2 ノード・システムにおけるディスク・ストレージの引継ぎを示します。正常ノードは、共有ディスクを引継いで、ディスク上の論理ボリュームとファイルシステムを回復します。このプロセスは、XFS ファイルシステムによって効率的に行われます。XFS ファイルシステムでは、ファイルシステムの整合性チェックに `fsck` コマンドを使用する必要がないジャーナル・テクノロジが使用されるので、高速な回復がサポートされます。

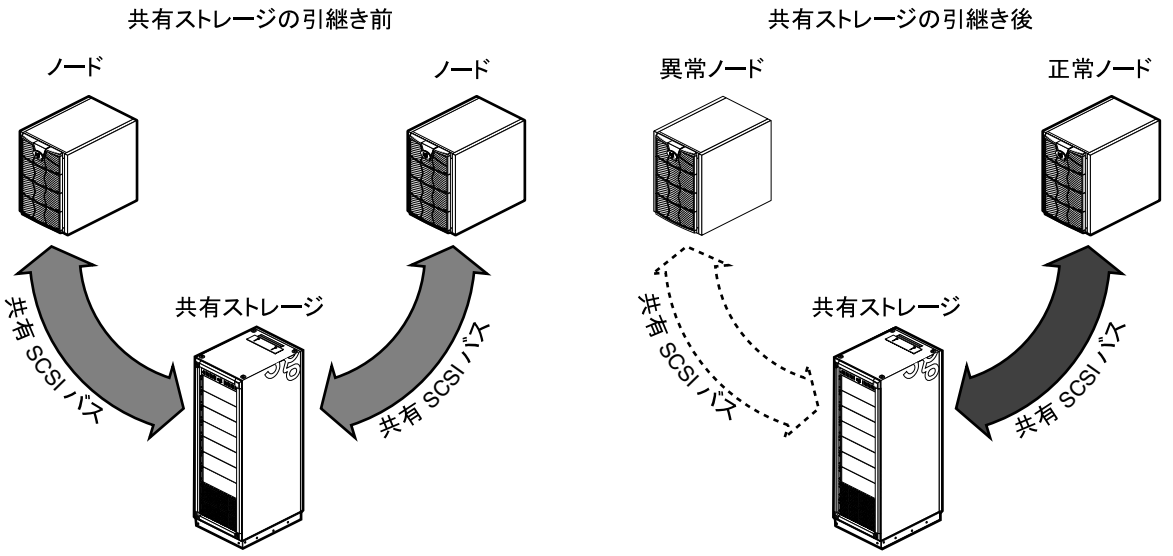

図1-2 2 ノード・システムにおけるディスク・ストレージ・フェイルオーバー

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。IRIS FailSafe ソフトウェアによって

高可用性リソースまたは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高可用性リソースが別のノードにフェイルオーバーされ、その異常ノードの高可用性アプリケーションは停止されます。これらの操作が完了すると、高可用性アプリケーションはバックアップ・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、アプリケーションとモニタのフェイルオーバー・ポリシーなど、高可用性アプリケーションに関するすべての情報は、クラスタ・マネージャ GUI またはクラスタ・マネージャ CLI を使用して IRIS 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、第5章「IRIS FailSafe の設定」を参照してください。高可用性アプリケーションの異常は、モニタ・スクリプトによって検出されます。

IRIS 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

あるノードで異常（ノードがクラッシュ、ハング、またはシャットダウンしたか、あるいは、高可用性サービスが動作していない）が検出されると、別のノードによって、異常が発生したノード（異常ノードと呼びます）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。

高可用性サービスの異常は、別のノードで実行されている IRIS FailSafe プロセスで検出できます。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 IRIS 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。

- 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
- サービスを引継ぐノードにメッセージを送信して、異常が発生する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

メッセージが受信されると、リソース・グループを引継ぐノードで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それ自体に転送する
- 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。

- データの破壊を防ぐために、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
- 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。フェイルオーバー・ポリシーの定義についての詳細は、第5章「IRIS FailSafe の設定」の132 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。

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、24 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 IRIS FailSafe クラスタの設定およびテストの概要

IRIS 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 IRIS 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、IRIS FailSafe の用語について理解します。
2. 第2章「IRIS FailSafe の設定計画」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「IRIS FailSafe ソフトウェアの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、IRIS 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. 第5章「IRIS FailSafe の設定」で説明されているとおりに、IRIS FailSafe 設定を定義します。
5. IRIS FailSafe システムのテストは、IRIS 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、IRIS FailSafe 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という 3 段階で行います。

IRIS FailSafe ソフトウェアの概要

この節では、IRIS FailSafe システムのソフトウェア階層、通信パス、およびクラスタ設定データベースについて説明します。

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。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。
- IRIS 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1-3に、これらの階層のイメージを示します。表1-2 では、`/usr/cluster/bin` ディレクトリ内の FailSafe の階層について説明します。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。CXFS についての詳細は、『CXFS Software Installation and Administration Guide』を参照してください。

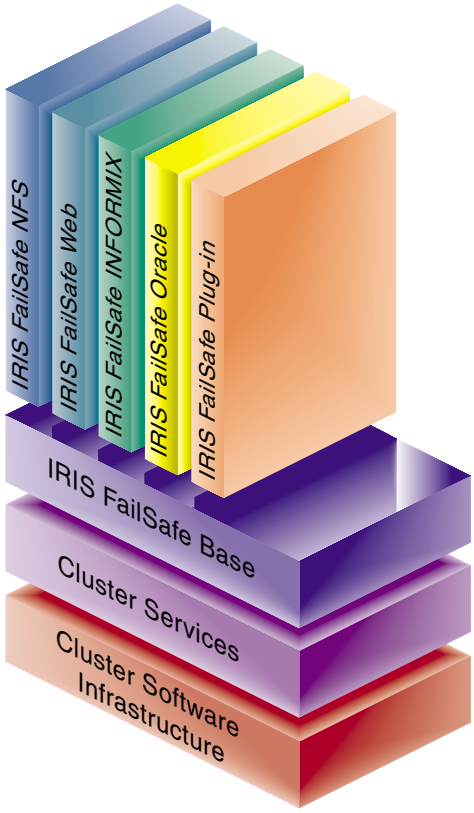

図1-3 ソフトウェア階層

表1-2 /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、22 ページの「インタフェース・エージェント・デーモン (IFD: Interface Agent Daemon)」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。

層	サブシステム	プロセス	説明
	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	設定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: Interface Agent Daemon)

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 HA 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、インタフェースのブロードキャスト・アドレスに対して ping を実行します。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、設定データベースから 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、クラスタ・マネージャ GUI およびクラスタ・マネージャ CLI 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。ただし、ここでは cmond クラスタ・マネージャ・デーモンを表していないことに注意してください。

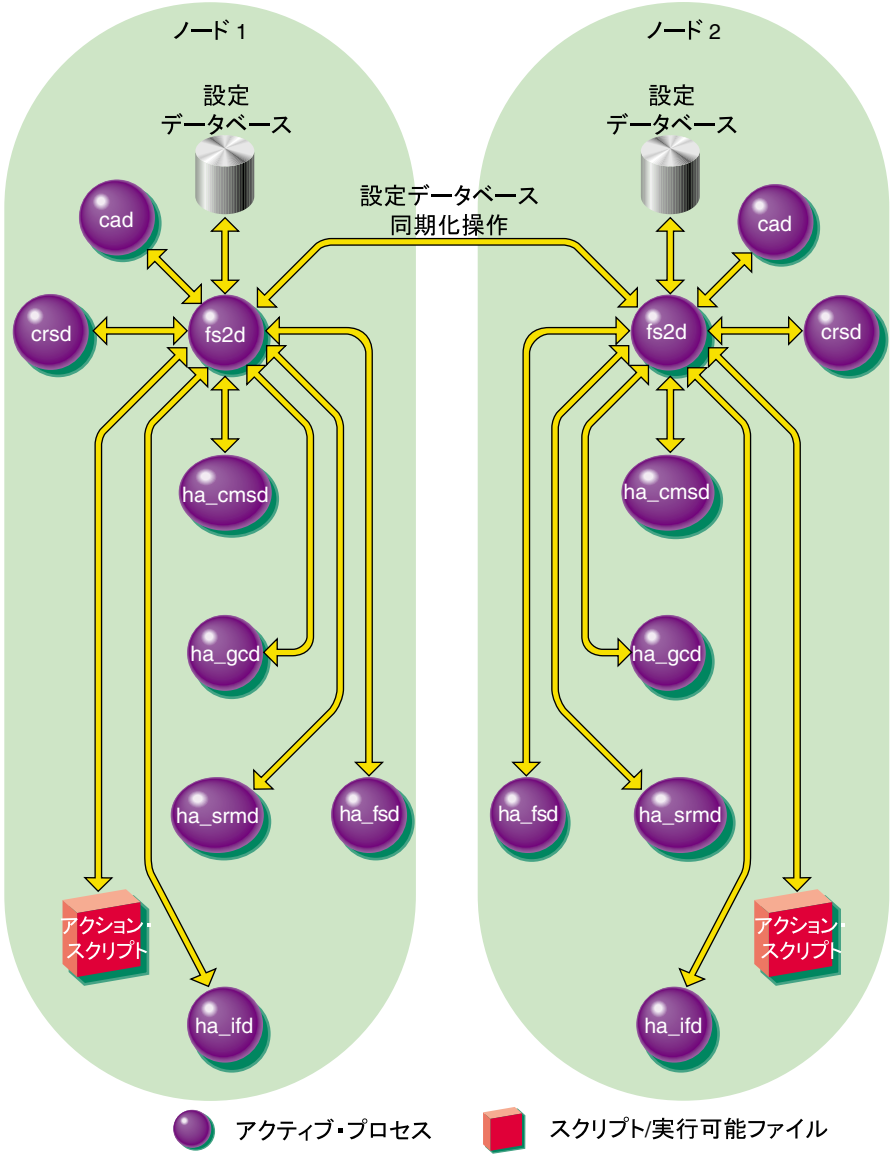

図1-4 クラスタ設定データベースに対する読取り/書込みアクション

図1-5には、プール内にあってクラスタ内にないノードの通信パスを示します。

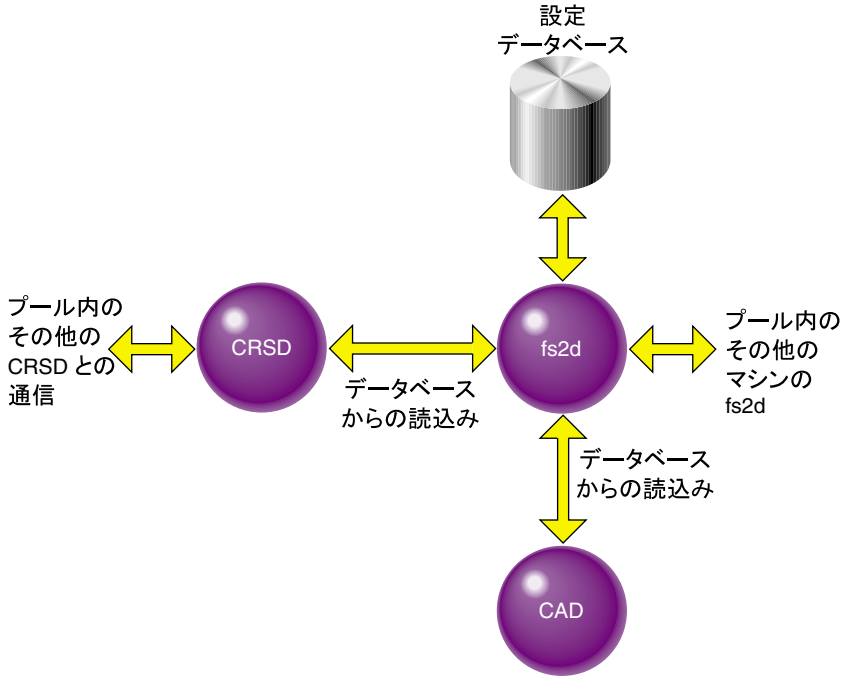

図1-5 クラスタ内にないノードの通信パス

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、クラスタ設定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、 srmd executable error エラーのあるリソース・グループのノードに online というマークが付けられます。

      - 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・グループが部分的または完全に実行されているノードと同じ場合、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、実行されていないリソース・グループのリソースの 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。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1-6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。

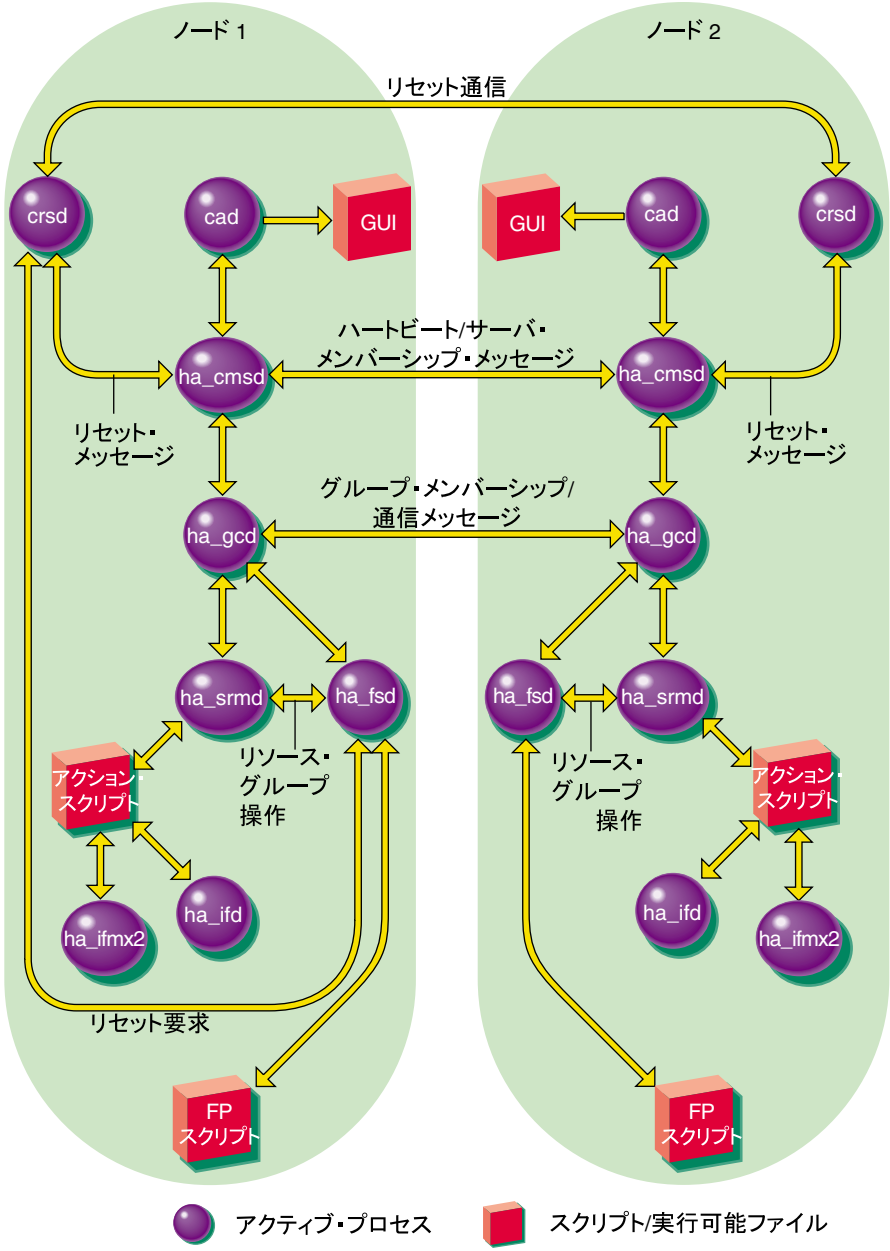

図1-6 アクション・スクリプトおよびフェイルオーバー・ポリシー・スクリプトのメッセージ・パス

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 srmd executable error エラーのマークが付けられます。
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

クラスタ設定データベースは 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

クラスタ設定データベース・デーモン (fs2d) によって、クラスタ内の各ノードで同じデータベースが維持されます。

表1-3 に、/var/cluster/ha ディレクトリの内容を示します。

表1-3 /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。
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 の設定計画

この章では、IRIS FailSafe クラスタでの高可用性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。

設定計画の概要

設定計画には、IRIS FailSafe クラスタの使用方法を決定する作業と、それに基づいて、クラスタで提供する高可用性サービスのニーズに合わせてどのようにディスクとインタフェースをセットアップするかを決定する作業が含まれます。計画プロセスでは、以下の質問に回答する必要があります。

- どのような目的でノードを使用するか

  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 World Wide 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。

- 上記のどの用途を高可用性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以下の IRIS FailSafe ソフトウェア・オプションを開発しています。

  – IRIS FailSafe NFS オプション。このオプションを使用すると、エクスポートされた NFS ファイルシステムを高可用性サービスとして提供できます。

  – IRIS FailSafe Web オプション。Netscape FastTrack および Enterprise Server に使用します。

  – IRIS FailSafe INFORMIX オプション。Informix データベースに使用します。

  – IRIS FailSafe Oracle オプション。Oracle データベースに使用します。

– IRIS FailSafe Samba オプション。Samba for IRIX データベースに使用します。

その他のアプリケーションを高可用性サービスとして提供するには、スイッチ・オーバーおよびスイッチ・バック機能を提供するアプリケーション・モニタ・シェル・スクリプトのセットを開発する必要があります。このようなスクリプトの開発については、『IRIS FailSafe Version 2 Programmer's Guide』で詳しく説明されています。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い合わせください。

- 各高可用性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、ノードが UP 状態のときにサービス(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。

- 各高可用性サービスに対して、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。

- 共有ディスクを raw XLV 論理ボリュームとして使用するか、それとも XFS ファイルシステムが使用されている XLV 論理ボリュームとして使用するか

  IRIS FailSafe では XLV 論理ボリュームが必要です。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。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。

- XVM 論理ボリュームを使用する CXFS ファイルシステムを共有ディスクに含めるかどうか FailSafe および CXFS の使用についての詳細は、44 ページの「CXFS との同時実行」を参照してください。

- 高可用性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

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・インタフェースの高可用性 IP アドレスの数

  高可用性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 つの高可用性 IP アドレスが必要です。

- 高可用性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 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、高可用性サービスのクライアントが使用可能な各高可用性 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

  高可用性サービスが使用するすべての高可用性 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。高可用性 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。

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 IRIS 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高可用性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 4 つあります。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 ef0 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。

以降の節は、上記の設定上の質問に回答したり、IRIS FailSafe に必要なその他の設定を決定したり、第3章「IRIS FailSafe ソフトウェアのインストールとシステムの準備」および第5章「IRIS FailSafe の設定」で説明されている設定タスクの実行に必要な情報を収集する際に役立ちます。

ディスク設定

以下の最初の項では、IRIS FailSafe システムの計画時に考慮が必要なディスク設定上の問題について説明します。また、共有ディスクと非共有ディスクの基本設定や、フェイルオーバー後にこれらのディスクが IRIS FailSafe によってどのように再設定されるのかについても説明します。2 番目の項では、IRIS FailSafe システムを設定する際のディスク設定の指定方法を説明します。

ディスク設定の計画

IRIS 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。

IRIS 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。
- IRIS FailSafe ソフトウェア、特に `/var/ha` ディレクトリは、非共有ディスクに配置する必要があります。通常は、`/var` とそのサブディレクトリは高可用性にしないでください。

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高可用性サービスのニーズに応じて決まります。各高可用性サービスには、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の IRIS FailSafe クラスタの基本ディスク設定を示します。各図には、フェイルオーバー後の設定も示してあります。基本ディスク設定は以下のとおりです。

- 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

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。

IRIS クラスタには、上記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。

図2-1 に、IRIS FailSafe 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。高可用性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。フェイ

ルオーバーが実行されると、IP エイリアスがフェイルオーバーされます。最初に異常ノードにあったデータは、この 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2-1 の設定には、Group1 と Group2 の 2 つのリソース・グループが含まれています。Group1 には IP_address リソース・タイプのリソース 192.26.50.1 が、Group2 には IP_address リソース・タイプのリソース 192.26.50.2 がそれぞれ含まれます。

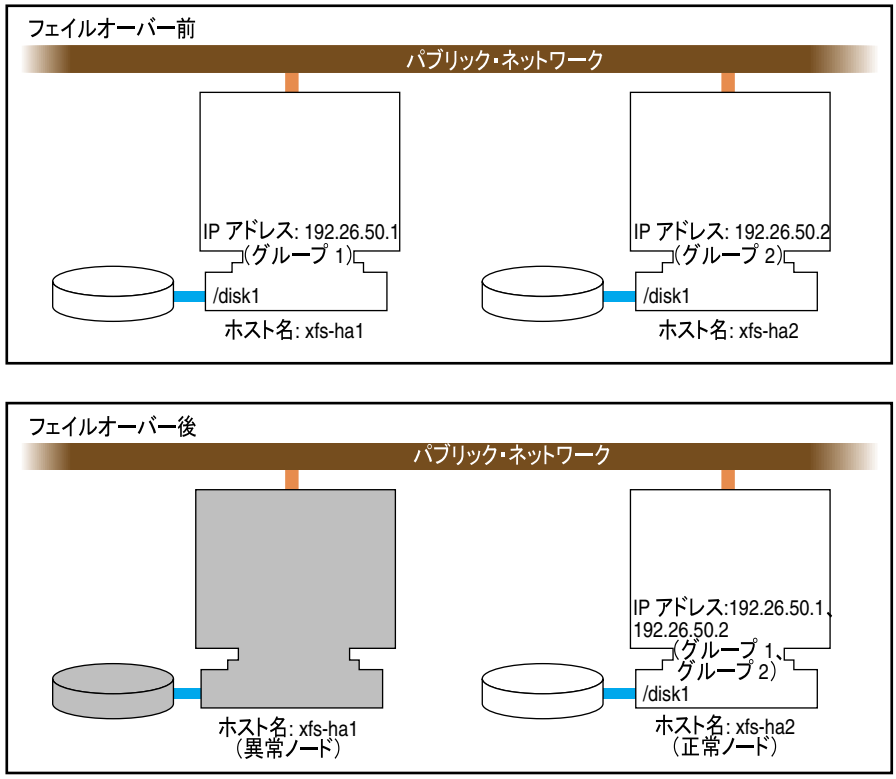

図2-1 非共有ディスク設定とフェイルオーバー

図2-2 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 Group1 には、(xfs-ha1、xfs-ha2)というフェイルオーバー・ドメインがあります。リソース・グループ Group1 には、IP_address リソース・タイプのリソース 192.26.50.1、filesystem リソース・タイプのリソース /shared、および volume リソース・タイプのリソース shared_vol の 3 つのリソースが含まれています。

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高可用性サービスは実行されません。

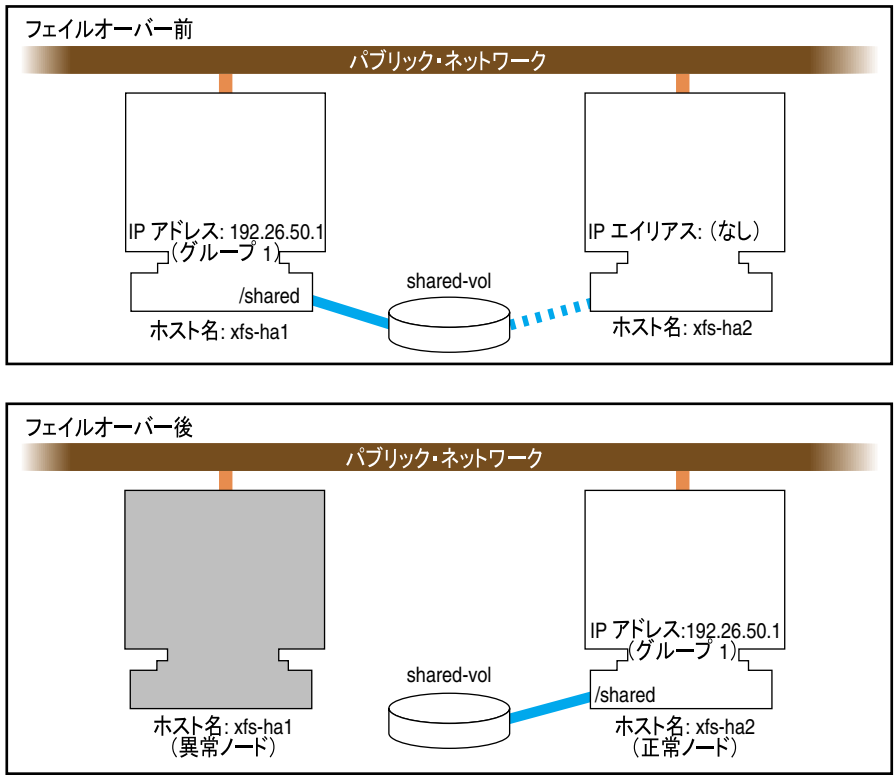

図2-2 アクティブ/バックアップ設定時の共有ディスク設定

図2-3 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 Group1 には、以下のリソースが含まれています。

- IP_address タイプのリソース 192.26.50.1
- ボリューム・タイプのリソース shared1_vol
- ファイルシステム・タイプのリソース /shared1

リソース・グループ Group1 には、(xfs-ha1、xfs-ha2)というフェイルオーバー・ドメインがあります。

リソース・グループ Group2 には、以下のリソースが含まれています。

- IP_address タイプのリソース 192.26.50.2
- ボリューム・タイプのリソース shared2_vol
- ファイルシステム・タイプのリソース /shared2

リソース・グループ Group2 には、(xfs-ha1、xfs-ha2)というフェイルオーバー・ドメインがあります。

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。

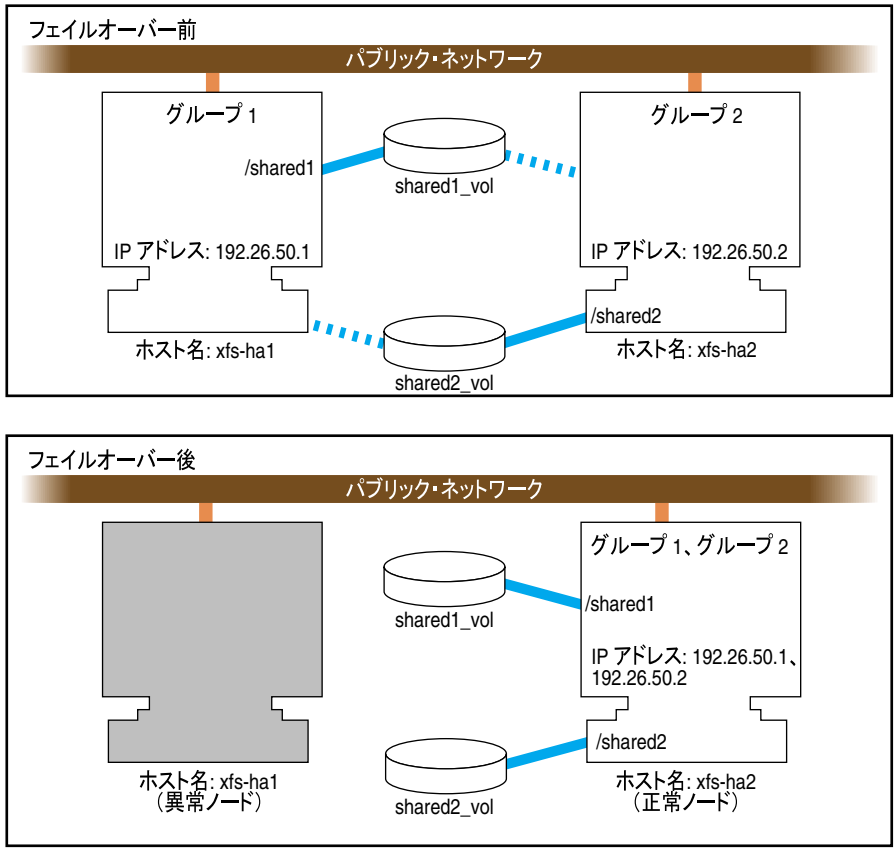

図2-3 デュアルアクティブ設定時の共有ディスク設定

各高可用性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、IRIS 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（実際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、39 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 ファイルシステムを使用する場合についての詳細は、44 ページの「CXFS との同時実行」を参照してください。

論理ボリューム設定

以下の最初の項では、IRIS FailSafe システムの計画時に考慮が必要な論理ボリュームの問題について説明します。2 番目の項では、IRIS FailSafe システムでの XLV 論理ボリュームの設定例を挙げます。3 番目の項では、IRIS FailSafe システムに指定する必要がある設定のさまざまな側面を説明します。

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、44 ページの「CXFS との同時実行」を参照してください。

論理ボリュームの計画

すべての共有ディスクには XLV 論理ボリュームが必要です。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、IRIS FailSafe 設定を正常に動作させるには、以下の規則に従う必要があります。

- 高可用性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。
- XLV では、同じ物理ディスク上に複数のボリュームを作成できます。IRIS FailSafe 環境では、単一のディスク上に複数のボリュームを作成する場合は、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、「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・ボールトまたは RAID LUN の各ディスクは、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、ボールト・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。ボールト・ディスクまたは 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。

IRIS 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 ボリューム名は、*pathname*/*volname* または *pathname*/*nodename.volname* です。以下に各構成要素を説明します。

- *pathname* は、/dev/xlv または /dev/rxlv 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、IRIX 6.5 では /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 IRIX 6.5 で /dev/rxlv/ha1.vol1 と表示されます。この場合、*ha1* がノード名構成要素で、*vol1* がボリューム構成要素です。この例からわかるように、ノード名構成要素がローカル・ホスト名と同じであると、ノード名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。この規則に従わないと、IRIS FailSafe ソフトウェア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、IRIS FailSafe ソフトウェア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。

「ボリューム」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（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、（IRIX 6.5 上の /dev/rxlv および /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（XLV 論理ボリュームのデフォルトのオーナー、グループ、およびモードは、root、sys、および 0600 です）。

論理ボリューム設定の例

XLV 論理ボリューム設定の例として、IRIX 6.5 システム上の 4 つのディスクにディスク 1～ディスク 5 という名前の以下の論理ボリュームがある場合を考えてみます。

- ディスク 1 とディスク 2 の一部が含まれる /dev/xlv/volA（ボリューム A）という名前の論理ボリューム
- ディスク 2 の残りとディスク 3 が含まれる /dev/xlv/volB（ボリューム B）という名前の論理ボリューム
- ディスク 4 とディスク 5 が含まれる /dev/xlv/volC（ボリューム C）という名前の論理ボリューム

ボリューム A と B 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。ボリューム C は、任意のリソース・グループの一部にできます。

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・ファイル名のオーナー（デフォルト値: root）
- デバイス・ファイル名のグループ（デフォルト値: sys）
- デバイス・ファイル名のモード（デフォルト値: 600）

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、「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

以下の最初の項では、IRIS FailSafe システムの計画時に考慮が必要なファイルシステムの問題について説明します。2 番目の項では、IRIS FailSafe システムでの XFS ファイルシステムの設定例を示します。3 番目の項では、IRIS FailSafe システムに指定する必要がある設定のさまざまな側面を説明します。

メモ: 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、44 ページの「CXFS との同時実行」を参照してください。

ファイルシステムの計画

IRIS FailSafe ソフトウェアは、共有ディスク上の XFS ファイルシステムの自動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、IRIS 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・ボールトまたは RAID ストレージ・システムに存在する必要があります。

以下に、IRIS 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、IRIS FailSafe 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 IRIS 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 IRIS FailSafe によってマウントされます。

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

メモ: 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。

ファイルシステム設定の例

この章の「論理ボリューム設定の例」の設定例の続きとして、ボリューム A と B で XFS ファイルシステムが使用されているとします。

- ボリューム A のファイルシステムは、rw および noauto モードで /sharedA にマウントされています。これをファイルシステム A と呼びます。
- ボリューム B のファイルシステムは、rw および noauto モードで /sharedB にマウントされています。「論理ボリューム設定の例」これをファイルシステム B と呼びます。

ファイルシステムの設定パラメータ

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

リソース属性	/sharedA	/sharedB	コメント
monitoring-level	2	2	モニタリングには以下の 2 種類があります。 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	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
モード	rw、noauto	rw、noauto、wsync	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）

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。

IP アドレス設定

以下の最初の項では、IRIS FailSafe システムの計画時に考慮が必要なネットワーク・インタフェースと IP アドレスの問題について説明します。2 番目の項では、IRIS FailSafe システムでのネットワーク・インタフェースと IP アドレスの設定例を示します。3 番目の項では、IRIS FailSafe 設定に指定する必要がある必要のさまざまな側面を説明します。

ネットワーク・インタフェースと IP アドレスの設定計画

ノード間のコントロール・ネットワークとして使用できる、クラスタ内のノード間のプライベート・ネットワークへのインタフェースの設定を計画するときは、以下のガイドラインに従ってください。この情報は、ノードを定義するときに使用されます。

- 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 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 IP アドレスに対して IP 名を指定できます。
- プライベート・ネットワークに関連付けられるこれらの IP アドレスに対して、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、プライベートを表す “ priv-” をホスト名の前に付けます。

クラスタ内にある、1 つまたは複数のパブリック・ネットワークとのノード・インタフェースの設定を計画するときは、以下のガイドラインに従ってください。

- 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(高可用性 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 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高可用性 IP アドレスにできません。
- クライアントが高可用性サービスへのアクセスに使用するすべての 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、すべての高可用性 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。
- 高可用性 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、高可用性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。

- 高可用性サービスのユーザは、hostname コマンドの出力ではなく高可用性 IP アドレスを使用する必要があるので、高可用性 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。
- 高可用性 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 IRIS 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 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 IP アドレスに対して ping 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* で以下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* で以下のコマンドを入力して、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 IP アドレスに対して ping を実行します。

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

   ping 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、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。

IP アドレス設定の例

この例では、192.26.50.1 という IP アドレスを設定します。このアドレスは、ネットワーク・マスクは 0xffffff00、ブロードキャスト・アドレスは 192.26.50.255 で、インタフェース ef0 に設定されています。

また、192.26.50.2 という IP アドレスも設定します。このアドレスも、ネットワーク・マスクは 0xffffff00 で、ブロードキャスト・アドレスは 192.26.50.255 で、インタフェース ef0 に設定されています。

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

表2-3 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: 192.26.50.1
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

IP アドレスのローカル・フェイルオーバー

IP アドレスが同じホスト内の 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、単一のノード上の 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、160 ページの「IP アドレスのローカル・フェイルオーバー」にあります。

CXFS との同時実行

CXFS 6.5.10 と IRIS FailSafe 2.1 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。

メモ: IRIS 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 ファイルシステムまたは XVM ボリュームの開始、停止、またはモニタを直接行うことはありません。また、CXFS ファイルシステムを FailSafe リソース・グループに追加しないでください。

高可用性アプリケーションが CXFS ファイルシステムと IP アドレスを使用する場合は、該当するアプリケーションと IP アドレスをリソース・グループに追加してください。アプリケーションが依存する 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。

ファイルシステムのメタデータ・サーバとして動作しているノードがダウンした場合は、使用可能なメタデータ・サーバのリストに含まれる別のノードが新しいメタデータ・サーバとして選択されます。

以下の点に注意してください。

- 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。
- 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、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。
- 実作業用のクラスタは、重み付けされた 3 つ以上のノードと 16 個以下のノードで設定することをお薦めします (リセット・ケーブルが接続されていて重み付けされた 2–ノード・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)。クラスタ内のすべてのノードで CXFS が実行されていなければなりません。8 つのノードでは IRIS FailSafe も実行できます。
- 使用可能なすべてのメタデータ・サーバ・ノードのタイプは、`CXFS` または `CXFS and FailSafe` でなければなりません。

- cmgr(1m) (cluster_mgr) コマンドは 1 つですが、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。
- CXFS GUI または CLI を使用して、既存の FailSafe クラスタとノードを CXFS または CXFS and FailSafe に切替えることができます。FailSafe GUI を使用すると、並行して操作を実行できます。切替えたノードは、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。
- 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。

- CXFS および FailSafe 用の関連するすべての IRIX パッチをインストールする必要があります。

FailSafe で使用するための CXFS クラスタの切替えについての詳細は、102 ページの「CXFS クラスタの FailSafe への切替え」を参照してください。 FailSafe で使用するための CXFS ノードの切替えについての詳細は、94 ページの「CXFS ノードの FailSafe への切替え」を参照してください。 CXFS ファイルシステムをエクスポートするための FailSafe システムの設定についての詳細は、161 ページの「CXFS ファイルシステムのエクスポート」を参照してください。

CXFS についての詳細は、『CXFS Software Installation and Administration Guide』を参照してください。

IRIS FailSafe ソフトウェアのインストールとシステムの準備

この章では、クラスタのノードを IRIS FailSafe 用に準備および設定する上で、ノードで実行しなければならない複数のシステム管理手順について説明します。これらの手順では、第2章「IRIS FailSafe の設定計画」で説明されている計画を行ったものと仮定します。

この章の主な節は、以下のとおりです。

IRIS FailSafe 用のノードの設定の概要

IRIS FailSafe 用にノードを準備するために必要なシステム管理手順の実行には、以下の手順が含まれます。

1. この章の「必要なソフトウェアのインストール」で説明されているとおりに、必要なソフトウェアをインストールします。
2. 「システム・ファイルの設定」で説明されているとおりに、各ノードでシステム・ファイルを設定します。
3. 「NVRAM 変数の設定」で説明されているとおりに、各ノードで 2 つの重要な NVRAM 変数の設定をチェックします。
4. クラスタで実行する予定の高可用性アプリケーションに必要な論理ボリュームとファイルシステムを作成します。「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。

5. 「ネットワーク・インタフェースの設定」の手順を使用して、ノードでネットワーク・インタフェースを設定します。
6. 「シリアル・ポートの設定」の手順に従って、各ノードでその他のノードとのシリアル接続に使用されるシリアル・ポートを設定します。
7. 準備ができたら、ノードを再起動すると IRIS FailSafe ソフトウェアが開始するようにノードを設定します。

IRIS FailSafe 用にノードの設定を完了するには、第5章「IRIS FailSafe の設定」で説明されているとおりに、IRIS FailSafe システムのコンポーネントを設定しなければなりません。

必要なソフトウェアのインストール

メモ : IRIS FailSafe base CD のインストールには、約 10 MB が必要です。

メモ : base system administration (*sysadm_base*)、cluster administration (*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。

メモ : IRIS FailSafe パッチのインストールについての詳細は、62 ページの「IRIS FailSafe パッチのインストール」を参照してください。

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. クラスタ内のすべてのサーバで、IRIX のサポートされているリリースが実行していることを確認します。
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. プールの各システムに、オペレーティング・システムに適したバージョンの EL-8+ multiplexer driver をインストールします。EL-8+ multiplexer に付属の CD を使用します。インストールの後、システムを再起動してください。
4. プール・ノードにソフトウェアをインストールします。

   プールの一部である各ノードに、以下のソフトウェアをこの順序でインストールします。

   - `sysadm_base.sw.dso`

- sysadm_base.sw.server
- cluster_admin.sw.base
- cluster_control.sw.base
- cluster_services.sw.base
- cluster_services.sw.cli
- cluster_control.sw.cli
- failsafe2.sw.cli
- sysadm_failsafe2.sw.server
- cluster_control.sw

メモ: sysadmdesktop がインストールされていない 6.5 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。

IRIS FailSafe クラスタ・マネージャ GUI の Web ベースのバージョンでプール・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。

- java_eoe.sw、バージョン 3.1.1
- sysadm_base.sw.client
- sysadm_failsafe2.sw.client
- sysadm_failsafe2.sw.web

5. ノードにその他のソフトウェアをインストールします。

クラスタの一部である各ノードに、以下のソフトウェアをこの順序でインストールします。各ノードには、手順 4 で挙げたソフトウェアのほかにこれらのソフトウェアも必要です。

- cluster_services.sw
- failsafe2.sw
- （必要な場合）nfs.ws.nfs（IRIX。すでに存在している可能性があります）。
- failsafe2_nfs.sw
- （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）。
- （必要な場合）ns_fasttrack.sw.server または ns_enterprise.sw.server（Netscape から。すでに存在している可能性があります）。
- failsafe2_web.sw

6. 管理ワークステーション（GUI クライアント）にソフトウェアをインストールします。

   このワークステーションで IRIX デスクトップから GUI クライアントを実行する場合は、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
   - java_eoe.sw、バージョン 3.1.1
   - Java をサポートしている Web ブラウザから GUI クライアントをワークステーションで起動する場合™: IRIS FailSafe CD から java_plugin

   ---

   メモ：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。この競合を解決するには、これらの 3 つのサブシステムをインストールしないようにします。これらのサブシステムは、IRIS FailSafe クラスタ・マネージャ GUI では使用されません。

   Java Plug-in を 1 つもインストールせずにブラウザから IRIS FailSafe マネージャ GUI を実行すると、http://java.sun.com/products/plugin/1.1/plugin-install.html にリダイレクトされます。

   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。

IRIX 以外のワークステーションから IRIS FailSafe マネージャ GUI を実行する場合は、`http://java.sun.com/products/plugin/1.1/plugin-install.html` から Java Plug-in をダウンロードします。

Java Plug-in がインストールされていない場合にブラウザから IRIS FailSafe マネージャ GUI を実行すると、ブラウザはこのサイトにリダイレクトされます。

7. 適切なサーバに、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、カスタマが注文した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 論理ボリュームをカスタマが使用している場合は、以下を実行します。
   - クラスタ内の各サーバの `/var/flex1m/license.dat` 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「IRIS FailSafe の設定計画」を参照してください。
   - 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
     ```

   - `xlv_mgr` を中止します。
9. IRIS FailSafe の推奨パッチをインストールします。

IRIS FailSafe を使用するときは、AutoLoad 変数を Yes に設定してください。これは、56 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、243 ページの付録B「IRIS FailSafe 2.1 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

FailSafe ソフトウェアをインストールする場合、プールの各ノードに対してシステム・ファイルやシステム・パラメータを設定する際の考慮事項がいくつかあります。この節には、以下のトピックに関する情報が含まれています。

- 52 ページの「/etc/services の FailSafe 用の設定」
- 52 ページの「/etc/config/cad.options の FailSafe 用の設定」

- 53 ページの「/etc/config/fs2d.options の FailSafe 用の設定」
- 55 ページの「/etc/config/cmond.options の FailSafe 用の設定」
- 56 ページの「coreplusid システム・パラメータの設定」

/etc/services の FailSafe 用の設定

プールの各ノードに cluster_admin 製品をインストールする前に、/etc/services ファイルに sgi-cad および sgi-crsd のエントリが含まれていなければなりません。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。sgi-cad には TCP ポートが必要であることに注意してください。

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
```

HA サービスをノードで開始する前に、各ノードの /etc/services ファイルに sgi-cmsd および sgi-gcd のエントリが含まれていなければなりません。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 の FailSafe 用の設定

/etc/config/cad.options ファイルには、プロセスの開始時にクラスタ管理デーモン (CAD: Cluster Administration Daemon) 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、FailSafe クラスタ・マネージャ 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* としても指定できます。
-vvvv	詳細レベル。“ 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
```

cad.options ファイルを変更したら、/etc/init.d/cluster restart コマンドで CAD プロセスを再開し、変更を反映してください。

/etc/config/fs2d.options の FailSafe 用の設定

/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設定データベース (CDB: Cluster Configuration Database) の配布を管理する設定データベース・デーモンです。

fs2d.options ファイルでは、以下のオプションを設定できます。

-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。

このオプションのデフォルト値は、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 -logdest stderr ですが、fs2d がデーモンとして実行されるとログは無効になります。これは、fs2d がデーモンとして実行されているときは、stdout および stderr が閉じられるためです。

このオプションのデフォルト値は、logfile です。

-logfile *filename*

ログ・ファイル名を設定します。

デフォルト値は /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、このオプションのデフォルト値は 10000000 になります。

-loglevel *log level*	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。 このオプションのデフォルト値は、`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。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。 このオプションのデフォルト値は、`transactions` です。
-tracefile *filename*	トレース・ファイル名を設定します。
-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すると、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。
-[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。 このオプションのデフォルト値は、-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。 このオプションのデフォルト値は、-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・メッセージがファイル `/var/cluster/ha/log/fs2d_log` に記録されるよう、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。

以下に、すべての fs2d ログ情報を `/var/adm/SYSLOG` に送信し、すべての fs2d トレース情報を `/var/cluster/ha/log/fs2d_ops1` 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・イベントと、rpcs、updates、および transactions というトレース・イベントがログ

に記録されています。トレース・ファイル /var/cluster/ha/log/fs2d_ops1 のサイズが 100000000 を超過した場合は、このファイルの名前は /var/cluster/ha/log/fs2d_ops1.old に変更され、新しいファイル /var/cluster/ha/log/fs2d_ops1 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。

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

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 100000000 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。

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

fs2d.options ファイルを変更したら、/etc/init.d/cluster restart コマンドで FS2D プロセスを再開し、変更を反映してください。

/etc/config/cmond.options の FailSafe 用の設定

/etc/config/cmond.options ファイルには、プロセスの開始時にクラスタ・モニタ・デーモン (cmond) 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。クラスタ・モニタ・デーモンにより、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。クラスタ・モニタ・デーモンについての詳細は、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 *loglevel*	ログ・レベルを *loglevel* に設定します。
-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 *napinterval*	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 [*eventname*]	メッセージのログを stderr 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
```

coreplusid システム・パラメータの設定

FailSafe を実行するときは、systune(1M) コマンドを使用して、システムのすべてのノードで coreplusid フラグを 1 に設定することが推奨されています。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 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

IRIS 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 NVRAM 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。IRIS FailSafe ソフトウェアでは、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 IRIS FailSafe クラスタ内のノードの 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

第2章「IRIS FailSafe の設定計画」では、クラスタ上の高可用性アプリケーションが使用する XLV 論理ボリュームと XFS ファイルシステムを計画しました。これらは、『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、44 ページの「CXFS との同時実行」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 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。

必要な 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します。
- 各 XLV 論理ボリュームは、その論理ボリュームを使用する高可用性アプリケーションのプライマリ・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(「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 *nodename* (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(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` コマンドおよび `chmod` コマンド(`chown`(1) および `chmod`(1) の参照ページを参照してください)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、IRIS 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、IRIS 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 IRIS 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 IRIS 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

この節の手順では、IRIS FailSafe クラスタのノードでネットワーク・インタフェースを設定する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。

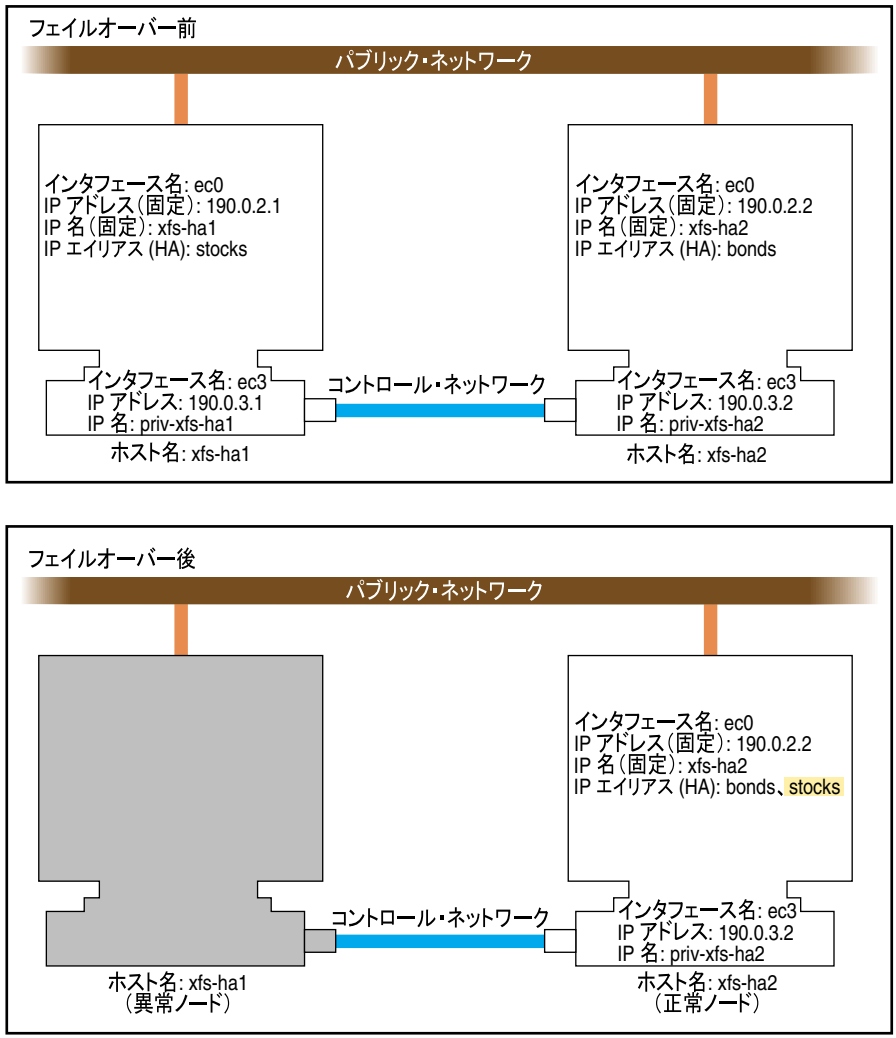

図3-1 インタフェース設定の例

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:

   ```
   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: 高可用性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 エイリアスが高可用性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、クラスタ内のすべてのノードで 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、各アドレスごとに以下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。高可用性 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 8 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・ネットワーク上で表示されないよう(ルーティングがオフになるよう)、各ノードでファイル /etc/config/routed.options を編集して、-q オプションを追加します。以下に、IRIX 6.5 ノードの /etc/config/routed.options の内容の例を示します。

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

   メモ: -q オプションは、IRIS 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。

9. 各ノードで IRIS FailSafe 2.*X* の chkconfig の結果が off であることを確認します。

   ```
   # chkconfig | grep failsafe2
   ...
           failSafe2           off
   ```

```
...
```

いずれかのノードで failsafe2 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 failSafe2 off
```

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します。各ノードに対して、IRIS FailSafe 1.X の chkconfig の結果が off であることを確認します。

```
# 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、IRIS FailSafe が IRIS FailSafe 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(yp の chkconfig の結果は on)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。IRIS 6.5 システムでは、/etc/nsswitch.conf ファイルを以下のように変更する必要があります。

```
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 が使用されている場合は、FDDIXpress リリース・ノートの第 2 章および『FDDIXPress Administration Guide』の第 2 章の説明に従って、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。これを行うには、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 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、IRIS 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可能性があるため、IRISconsole にはリセット・ポートを設定しないでください。

IRIS FailSafe パッチのインストール

この節の手順では、FailSafe 2.*X* システムにソフトウェア・パッチをインストールする方法について説明します。FailSafe パッチは、クラスタ内の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* および 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。

FailSafe 2.*X* およびアップグレード・パッチを各ノードにインストールするには、以下の手順に従います。

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、`cluster_admin` 製品、`cluster_control` 製品、`cluster_services` 製品、`failsafe2` 製品、`sysadm_base` 製品、および `sysadm_failsafe2` 製品が含まれます。
2. FailSafe 2.*X* パッチをノードに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 Manager GUI または FailSafe Manager CLI を実行し、FailSafe 高可用性クラスタを設定して使い始める準備ができました。

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高可用性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、インストールする特定のパッチのパッチ・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 Manager 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。インストールする GUI クライアント・ソフトウェア・サブシステムのパッチは、patchSG*xxxxxxx*.sysadm_base_sw.client、patchSG*xxxxxxx*.sysadm_failsafe2_sw.client、および patchSG*xxxxxxx*.sysadm_failsafe2_sw.desktop です。*xxxxxxx* はパッチ番号です。

2. FailSafe 2.*X* パッチをインストールするノードを選択します。FailSafe マネージャ GUI または FailSafe マネージャ CLI を開始します。

   便宜上、FailSafe マネージャはアップグレードしないノードに接続してください。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 FailSafe マネージャ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、FailSafe マネージャ GUI が切断されてしまいます。

   ノードを選択するには、次の CLI コマンドを使用できます。このコマンドでは、クラスタ名が指定されていることを想定しています。

   ```
   cmgr> set cluster <cluster name>
   ```

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。

   FailSafe 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。

   FailSafe CLI 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 <group name>
   ```

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。

   FailSafe CLI 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 <node name>
   ```

   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。

FailSafe 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。

FailSafe CLI 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 <group name> to node <node name>
```

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
```

FailSafe 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、failsafe2 フラグおよび cluster フラグをオフにします。

```
# chkconfig failsafe2 off
# chkconfig cluster off
```

8. FailSafe 2.*X* パッチをノードにインストールします。

9. chkconfig(1m) を使用して、failsafe2 フラグおよび cluster フラグをオンにします。

```
# chkconfig failsafe2 on
# 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。

FailSafe GUI を使用しており、Web ブラウザで GUI を実行している場合は、ブラウザを終了して、直前にパッチを適用したノードで Web サーバを再起動した後、GUI を再起動して、パッチが適用されているノードに接続します。

「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、以下の CLI 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。

FailSafe CLI 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 <node name>
```

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。

    FailSafe GUI を使用している場合は、「FailSafe クラスタ表示(FailSafe Cluster View)」ウィンドウを開きます。「表示(View)」メニュー ->「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、ステータスがオンラインであることを示す緑になっ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。

    FailSafe CLI 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 <group name>
    ```

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 6.5.6 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3-1 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3-1 PCP for FailSafe コレクタ・サブシステム

サブシステム	サイズ（K バイト単位）
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、次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 6.5.6 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、各モニタ・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ます。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 モニタ・サブシステム

サブシステム	サイズ（K バイト単位）
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
   ```

IRIS FailSafe 管理ツール

この章では、IRIS FailSafe 管理ツールとそれらの操作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。

IRIS FailSafe クラスタ・マネージャのツール

IRIS FailSafe の管理タスクは、以下のいずれかのツールを使用して実行できます。

- IRIS FailSafe クラスタ・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)
- IRIS FailSafe クラスタ・マネージャ・コマンド・ライン・インタフェース (CLI: Command Line Interface)

これらのツールは、同じ下層のソフトウェアを使用して FailSafe システムの設定およびモニタを行いますが、GUI には、実作業用システムで特に重要な以下の機能が備わっています。

- 「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。
- クラスタの状態が視覚的に表示されるため、ステータス、問題、およびフェイルオーバーをすばやく認識できます。
- 状態が動的に更新されるので、継続的にシステムをモニタできます。
- クラスタ・データベース情報に変更を加える前に、構文が正しいかどうかに関してすべての入力値がチェックされます。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。
- タスクおよびタスクセットにより、設定および管理操作を順番に進めることができ、タスクを実行しながらクラスタ設定に実際に変更を加えることができます。
- グラフィカルなツールは、Windows® コンピュータやラップトップを含む、Java 仮想マシンを搭載した任意のコンピュータからリモートで安全に実行できます。

一方、IRIS FailSafe クラスタ・マネージャ CLI の機能は、GUI よりも限られています。CLI では、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 IRIS 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。IRIS FailSafe の経験が豊富な管理者の方にとっては、IRIS FailSafe の基本的な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、クラスタ・マネージャ CLI の方が便利かもしれません。

クラスタ・マネージャ GUI は、下層の FailSafe コマンドを使用して、管理コマンドの実行および設定データベースの更新を行います。クラスタ・マネージャ CLI も、クラスタ・マネージャ GUI と同じ下層の FailSafe 管理コマンドを使用します。

IRIS FailSafe クラスタ・マネージャ GUI の使用

IRIS FailSafe クラスタ・マネージャ GUI では、グラフィカル・ユーザ・インタフェースを使用して、クラスタの設定、管理、およびモニタを行うことができます。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。

クラスタ・マネージャ GUI は、下層の FailSafe コマンドを使用して、管理コマンドの実行および設定データベースの更新を行います。ステータスまたは設定データベースに変更がある場合は、FailSafe CAD から GUI に情報が提供されます。設定データベースの変更がローカル CAD から提供されるのは 10 秒おきなので、GUI の更新は遅い場合があります。

クラスタ・マネージャ GUI は、FailSafe クラスタ表示と FailSafe マネージャおよびそのタスクとタスクセットで構成されます。これらのインタフェースについては、以下の節で説明します。

FailSafe クラスタ表示

「FailSafe クラスタ表示(FailSafe Cluster View)」ウィンドウには、以下の機能が用意されています。

- クラスタ・アイテム(ノードやリソースなど)の間の関係の表示
- すべてのアイテムの設定およびステータスの詳細へのアクセス
- クラスタの動作状態の表示
- FailSafe マネージャおよび SYSLOG へのアクセス
- ヘルプ情報へのアクセス

ユーザは、「FailSafe クラスタ表示(FailSafe Cluster View)」の任意のアイテムをクリックして、そのアイテムのキー情報を表示できます。以下に、この方法で表示できるアイテムを示します。

- 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

FailSafe マネージャ

FailSafe マネージャは、高可用性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。また、IRIS FailSafe 設定ガイド・タスクセットへのアクセスも提供します。

- タスクセットは、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」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タスクを起動できます。
- IRIS FailSafe タスクセットを使用すると、使いやすいグラフィカル・ユーザ・インタフェースから FailSafe クラスタのすべてのコンポーネントを設定およびモニタできます。

IRIS FailSafe マネージャ GUI の起動

FailSafe マネージャ GUI は、FailSafe マネージャまたは FailSafe クラスタ表示のいずれかを起動することで起動できます。

FailSafe マネージャを起動するには、以下のいずれかの方法を使用します。

- FailSafe 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。

- 以下のコマンド・ラインを入力します。

  % **/usr/sbin/fstask**

- 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 FailSafe マネージャを起動できるのは、IRIX 以外のシステムからクラスタ・マネージャ GUI を管理する場合です。IRIX システムでクラスタ・マネージャ GUI を実行する場合は、Toolchest または `/usr/sbin/fstask` コマンドを使用する方法をお薦めします。

FailSafe クラスタ表示を起動するには、以下のいずれかの方法を使用します。

- FailSafe Toolchest から「FailSafe クラスタ表示(FailSafe Cluster View)」を選択します。
- 以下のコマンド・ラインを入力します。

  % **/usr/sbin/fsdetail**

クラスタ・マネージャ GUI を使用すると、単一の管理ポイントからクラスタ全体を管理できます。クラスタ内で FailSafe デーモンがアクティブになっている場合に、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 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。クラスタ内で FailSafe デーモンがアクティブになっていない場合は、プール内の任意のノードに接続できます。

「FailSafe クラスタ表示」ウィンドウを開く

「FailSafe クラスタ表示(FailSafe Cluster View)」ウィンドウは、以下のいずれかの方法で開くことができます。

- 「FaileSafe マネージャ(FailSafe Manager)」ウィンドウの下部にある「FailSafe クラスタ表示(FailSafe Cluster View)」ボタンをクリックします。

  「FailSafe マネージャ」ウィンドウと「FailSafe クラスタ表示」ウィンドウの両方を同時に開く場合は、この方法で「FailSafe クラスタ表示」ウィンドウを開くことをお薦めします。これは、2 番目のウィンドウを開くときに、新しい Java プロセスを起動するのではなく既存の Java プロセスを再利用するので、クライアントのメモリ使用量が抑えられるためです。

- 上記の「IRIS FailSafe マネージャ GUI の起動」で説明されているように、FailSafe マネージャ GUI を起動するときに「FailSafe クラスタ表示」ウィンドウを直接開きます。

クラスタ・アイテムの詳細の表示

クラスタ・アイテムの詳細を表示するには、以下の手順に従います。

1. 「FailSafe クラスタ表示(FailSafe Cluster View)」ウィンドウを開きます。
2. アイテムの名前またはアイコンをクリックします。

設定およびステータスの詳細が別のウィンドウに表示されます。同じウィンドウに詳細を表示するには、「オプション(Options)」を選択します。続いて「詳細を表示(Show Details)」オプションをクリックすると、ウィンドウの右側にステータスの詳細が表示されます。

タスクの実行

FailSafe GUI を使用して個々のタスクを実行するには、以下の手順に従います。

1. 「FailSafe マネージャ(FailSafe Manager)」ウィンドウの左側のカラムでカテゴリの名前をクリックします。

   右側のカラムに個々のタスクセットとタスクセットのトピックのリストが表示されます。

2. 右側のカラムのタスクのタイトルをクリックします。

   「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。

   ---

3. 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ウィンドウ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。

   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、起動可能なその他のタスクが表示されます。

4. 必要に応じて、タスクの起動を続行します。

FailSafe タスクセットの使用

FailSafe マネージャ GUI にはタスクセットも用意されており、複数の異なるタスクが必要な目的を達成するために必要な手順を順番に実行できます。FailSafe タスクセットにアクセスするには、以下の手順に従います。

1. 「FailSafe マネージャ(FailSafe Manager)」ウィンドウの左側のカラムで「設定ガイド(Guided Configuration)」のカテゴリをクリックします。

   右側のカラムにタスクセットのリストが表示されます。

2. 右側のカラムでタスクセットをクリックします。

   目的を達成するために必要な一連のタスクのリストがウィンドウに表示されます。

3. 表示される手順に従って、タスクをクリックして起動します。

   タスクをクリックすると、そのタスクのウィンドウが表示されます。リストに表示されるすべてのタスクを完了したら、ウィンドウの左上角をダブルクリックするか、またはウィンドウに「閉じる(Close)」ボタンがある場合はそのボタンをクリックすると、タスクセットのウィンドウを閉じることができます。

IRIS FailSafe クラスタ・マネージャ CLI の使用

この節では、IRIS FailSafe クラスタ・マネージャ CLI を使用して IRIS FailSafe の管理タスクを実行する方法を説明します。IRIS FailSafe クラスタ・マネージャ CLI でコマンドを実行するには、root としてログインする必要があります。

クラスタ・マネージャ CLI も、クラスタ・マネージャ GUI と同じ下層の FailSafe コマンドを使用します。

クラスタ・マネージャを使用するには、以下のいずれかのコマンドを入力します。

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

または

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

コマンドを入力すると、以下のメッセージとクラスタ・マネージャ CLI のコマンド・プロンプトが表示されます。

```
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 CLI のヘルプを表示できます。

FailSafe システム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。

CLI コマンドの直接入力

クラスタ・マネージャ CLI のコマンドの中には、`cluster_mgr` コマンドの `-c` オプションを使用することで、`cmgr` モードに入らずにコマンド・ラインから直接実行できるものがあります。これらのコマンドは、`show`、`delete`、`admin`、`install`、 `start`、`stop`、`test`、`help`、および `quit` で、以下の形式を使用して直接実行できます。

```
cluster_mgr -c "command"
```

たとえば、`show clusters` CLI コマンドを以下の方法で実行できます。

```
% cluster_mgr -c "show clusters"
1 Cluster(s) defined
        eagan
```

「プロンプト」モードでのクラスタ・マネージャ CLI の呼出し

クラスタ・マネージャ CLI には、FailSafe コンポーネントの定義と変更を行う管理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、CLI をプロンプト・モードで実行できます。

- 以下の例のように、`cluster_mgr`(または `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。

  ```
  # cluster_mgr -p
  ```

- 以下の例のように、CLI を起動した後で `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 CLI コマンドを実行するときにプロンプト・モードとそれ以外のモードを切替えることができます。CLI の実行中にプロンプト・モードを終了するには、以下の CLI コマンドを入力します。

  ```
  cmgr> set prompting
  ```

たとえば、CLI のプロンプト・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 A
Enter commands, when finished enter either "done" or "cancel"

A?
```

このプロンプトで、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、92 ページの「クラスタ・マネージャ CLI を使ったノードの定義」を参照してください）。

```
set hostname to B
 set nodeid to C
 set partition_id to D
 set reset_type to E
 set sysctrl_type to F
 set sysctrl_password to G
 set sysctrl_status to H
 set sysctrl_owner to I
 set sysctrl_device to J
 set sysctrl_owner_type to K
 set is_failsafe to L
 set is_cxfs to M
 set weight to N
 add nic O
         set heartbeat to P
         set ctrl_msgs to Q
         set priority to R
 remove nic S
```

続いて、ネットワーク・インタフェースを追加した後に、ネットワーク・インタフェースのパラメータを入力するよう求めるプロンプトが表示されるので、同じように入力します。

ただし、CLI をプロンプト・モードで実行中の場合は、適切な値を入力したときに以下の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
```

```
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
```

CLI 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 cluster_mgr 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、cluster_mgr の起動時に実行されます。cluster_mgr スタートアップ・ファイルで使用できるのは、cluster_mgr 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 cluster_mgr 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 CLI スタートアップ・スクリプトとして指定するには、IRIX プロンプトで以下のコマンドを実行します。

```
$ setenv CMGR_START_FILE /cmgr_rc
```

クラスタ・マネージャ CLI を起動すると cmgr_rc スクリプトが実行され、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。

CLI コマンドの入力ファイルの使用

cluster_mgr コマンドの -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、クラスタ・マネージャ CLI コマンドを連続して実行できます。

```
cluster_mgr -f "input_file"
```

入力ファイルにはクラスタ・マネージャ CLI コマンドが含まれており、quit コマンドで終了する必要があります。

たとえば、ファイル input.file に以下のコマンドが含まれるとします。

```
show clusters

show nodes in cluster beta3
quit
```

この場合は、以下のコマンドを実行でき、以下の出力が得られます。

```
% cluster_mgr -f input.file

1 Cluster(s) defined
        eagan
Cluster eagan has following 2 machine(s)
        cm1
        cm2
```

cluster_mgr コマンドには、-f オプションと併せて使用する -i オプションが用意されています。これは“ ignore” オプションで、スクリプトの実行中にコマンドが失敗してもクラスタ・マネージャが終了しないように指定します。

CLI コマンド・スクリプト

cluster_mgr コマンドの -f オプションを使用して、直接実行できるクラスタ・マネージャ CLI コマンドのスクリプトを作成できます。このスクリプトには、スクリプトの 1 行目として以下の行が含まれている必要があります。

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

メモ: cluster_mgr コマンドの -i オプションを使用して、スクリプトの実行中にコマンドが失敗してもクラスタ・マネージャが終了しないように指定した場合は、スクリプト・ファイルの 1 行目に #!/usr/cluster/bin/cluster_mgr -if という構文を使用してください。コマンド・ラインから直接 -i オプションを使用するときは、-if の構文を使用する必要はありません。

スクリプトの各行は、hereドキュメントと同様に、有効な cluster_mgr command コマンド・ラインでなければなりません。クラスタ・マネージャ CLI は、インタラクティブにコマンドを入力した場合と同様にコマンドで動作するので、複数のレベルのコマンドを完了してクラスタ・マネージャ CLI を終了できるよう、done および quit の行を含めてください。

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可能なスクリプトの CLI テンプレート・ファイルが用意されています。これらのファイルは、/var/cluster/cmgr-templates ディレクトリにあります。CLI テンプレートについての詳細は、81 ページの「CLI テンプレート・スクリプト」を参照してください。

以下に、CLI コマンド・スクリプト cli.script の例を示します。

```
% more cli.script
#!/usr/cluster/bin/cluster_mgr -f

show clusters
show nodes in cluster beta3
quit

% cli.script
1 Cluster(s) defined
        eagan
Cluster eagan has following 2 machine(s)
        cm1
        cm2

%
```

CLI 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 CLI 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 cluster_mgr 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、テンプレートでは、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。

var/cluster/cmgr-templates ディレクトリには、以下のテンプレートが含まれます。

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 CLI 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 CLI コマンド・スクリプトを実行できます。

メモ: 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。cluster_mgr 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 つのボリューム、1 つのファイルシステム、1 つの IP アドレス、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
- 1 つの cmgr-create-cluster ファイル
- 1 つの cmgr-create-failover_policy ファイル
- 1 つの cmgr-create-resource_group ファイル
- 1 つの cmgr-create-resource-volume ファイル
- 1 つの cmgr-create-resource-filesystem ファイル

- 1 つの cmgr-create-resource-IP_address ファイル
- 1 つの cmgr-create-resource-NFS ファイル

CLI 内部からのシェルの呼出し

クラスタ・マネージャ CLI 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、以下のコマンドを入力します。

```
cmgr> sh
```

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

IRIS FailSafe の設定

この章では、IRIS FailSafe システムのコンポーネントを設定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、IRIS FailSafe クラスタ・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) と IRIS FailSafe クラスタ・マネージャ・コマンド・ライン・インタフェース (CLI: Command Line Interface) 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。

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。

デフォルトの設定

IRIS FailSafe システムのコンポーネントを設定する前に、IRIS FailSafe で使用するコンポーネントの一部に対して、コンポーネントの定義時にデフォルト値を設定できます。

デフォルトのクラスタ	特定のクラスタ・マネージャ・コマンドでは、クラスタを指定する必要があります。デフォルトのクラスタを指定すると、クラスタを明示的に指定しなかった場合にデフォルトとして使用できます。
デフォルトのノード	特定のクラスタ・マネージャ・コマンドでは、ノードを指定する必要があります。クラスタ・マネージャ CLI では、デフォルトのノードを指定すると、ノードを明示的に指定しなかった場合にデフォルトとして使用できます。

デフォルトのリソース・タイプ　特定のクラスタ・マネージャ・コマンドでは、リソース・タイプを指定する必要があります。クラスタ・マネージャ CLI では、デフォルトのリソース・タイプを指定すると、リソース・タイプを明示的に指定しなかった場合にデフォルトとして使用できます。

クラスタ・マネージャ GUI を使ったデフォルトのクラスタの設定

デフォルトのクラスタの名前を指定していない場合は、名前を入力するよう GUI によってプロンプトが表示されます。または、「FailSafe マネージャ (FailSafe Manager)」ウィンドウの下部にある「クラスタの選択...(Select Cluster...)」をクリックして、デフォルトのクラスタを設定できます。

GUI を使用する場合、デフォルトのノードやリソース・タイプを設定する必要はありません。

クラスタ・マネージャ CLI を使ったデフォルトの設定と表示

クラスタ・マネージャ CLI を使用している場合、デフォルト値を指定するために以下のコマンドを使用できます。デフォルト値は、クラスタ・マネージャ CLI の現在のセッションに対してのみ有効となります。

デフォルトのクラスタを指定するには、次のコマンドを使用します。

```
cmgr> set cluster A
```

デフォルトのノードを指定するには、次のコマンドを使用します。

```
cmgr> set node A
```

デフォルトのリソース・タイプを指定するには、次のコマンドを使用します。

```
cmgr> set resource_type A
```

クラスタ・マネージャ CLI の現在のデフォルト値は、次のコマンドを使用して表示できます。

```
cmgr> show set defaults
```

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、システムの設定にクラスタ・マネージャ GUI またはクラスタ・マネージャ CLI 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。

アプリケーション・ダウンタイム = 異常検出 + 異常を処理する時間 + 異常から回復する時間

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、IRIS FailSafe HA パラメータの 1 つで、変更できます。ハートビート異常や OS 異常など、ノード異常と考えられる異常は、すべてこの異常のカテゴリに分類されます。ノード・タイムアウト時間のデフォルト値は、15 秒です。ノード・タイムアウト値の変更についての詳細は、98 ページの「IRIS FailSafe HA パラメータ」を参照してください。
- 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、119 ページの「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、システム・パフォーマンスの点で FailSafe のオーバーヘッドが大幅に増えるほか、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

クラスタ設定

IRIS FailSafe システムを設定するには、高可用性サービスをサポートするクラスタを設定します。これには、以下の手順が必要です。

- ローカル・ホストの定義
- クラスタに含めることができる追加のノードの定義
- クラスタの定義

続く項では、これらのタスクについて説明します。

ノードの定義

ノードは、単一の UNIX イメージです。通常、ノードは個々のコンピュータです。ノードという用語は、このガイド内では略して使用されることもあり、この意味でのノードという用語は、Origin システムのノードとは異なる意味を持ちます。

プールは、クラスタリングに利用できるノードのセット全体です。

定義する最初のノードはローカル・ホストでなければならず、これは、クラスタ管理を実行するためにログインしたホストになります。

複数のノードを定義する場合は、次のノードを定義するまで 1 分程度待つことをお薦めします。設定データベースにノードが追加されると、そのノードの現在の設定データベースの内容は古いものとしてマークされ、現在の上限量のノードの設定データベースが、追加されたノードにコピーされます。ノード定義操

作は、新しいノード設定がデータベースに追加され、データベース設定が同期化されると完了します。2 つのノードを順に定義する場合、2 番目の操作は失敗することがあります。これは、最初のデータベース同期化が完了していないために起こります。

クラスタに含むことができるノードのプールに論理ノード定義を追加するには、ノードに関して次の情報を指定してください。

- ノードの論理名。この名前には文字や数字を使用できますが、スペースやシャープ記号は使用できません。名前は、255 文字以内で指定してください。正しいホスト名は、正しいノード名でもあります。たとえば、ホスト名が `venus.eng.company.com` であるノードに対しては、`venus` や `node1` など、使いやすいノード名を付けることができます。

- ホスト名。これは、`server1.company.com` のような、ホストの完全修飾名です。ホスト名は、アンダースコアで始めたり、空白を含めたり、255 文字より長くすることはできません。このアドレスは、定義しているノードの `hostname` コマンドの出力と同じでなければなりません。このホスト名に関連付けられる IP アドレスは、FailSafe `IP_address` リソースを定義する際に高可用性として定義する IP アドレスと同じにすることはできません。FailSafe は、この入力に対して IP アドレス(`192.0.2.22` など)を受付けません。

- 16 ビットの符号なしの値であるノード ID(ユーザがオプションで指定します)。デフォルト値は、CAD プロセスから取得されます。この数字は、プールの各ノードに対して一意であり、1～32767 の範囲でなければなりません。

- パーティション ID。パーティションが設定された 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（CLI では *rack#.slot#* 形式を使用してください）。

例:

```
# mkpart -n
Partition id = 1
# mkpart -l
partition: 3 = brick: 003c10 003c13 003c16 003c21 003c24 003c29 ...
partition: 1 = brick: 001c10 001c13 001c16 001c21 003c24 001c29 ...
```

「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、このフィールドは空のままにしてください。

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。

- システム・コントローラ情報。ノードにシステム・コントローラがあり、そのコントローラを使用して FailSafe にノードをリセットさせる場合は、システム・コントローラに関する次の情報を指定してください。

  – システム・コントローラのタイプ:

    - msc モジュール・システム・コントローラ
    - mmsc マルチモジュール・システム・コントローラ
    - l2 Origin 3000 用 L2 システム・コントローラ

  – システム・コントローラのリセット・タイプ: powerCycle のみサポートされます。

  – システム・コントローラ・ポートのパスワード（オプション）

  – FailSafe でポートを使用できるかどうかを判断するために設定できる管理ステータス: enabled、disabled。

  – システム・コントローラ・オーナー（つまり、システム・コントローラに物理的に接続されているシステム）の論理ノード名。

  - システム・コントローラに接続されているオーナー・ノードのポートのデバイス名
  - オーナー・デバイスのタイプ: `tty`。

- ノードが FailSafe ノードであるかどうか(クラスタ・マネージャ CLI の場合のみ。FailSafe クラスタ・マネージャ GUI で定義されたノードは、FailSafe ノードです)。FailSafe ノードおよび CXFS ノードについての詳細は、44 ページの「CXFS との同時実行」を参照してください。
- ノードが CXFS ノードであるかどうか(クラスタ・マネージャ CLI の場合のみ。FailSafe クラスタ・マネージャ GUI で定義されたノードは、FailSafe ノードです)。FailSafe ノードおよび CXFS ノードについての詳細は、44 ページの「CXFS との同時実行」を参照してください。
- ハートビート、リセット・メッセージ、およびその他の FailSafe メッセージに対して使用されるネットワークであるコントロール・ネットワークのリスト。各ネットワークに対して、以下の情報を指定してください。
  - ホスト名または IP アドレス。このアドレスは、FailSafe `IP_address` リソースを定義する際に高可用性として定義する IP アドレスと同じにすることはできず、`/etc/hosts` ファイル内で解決される必要があります。
  - フラグ(ハートビート用の `hb`、コントロール・メッセージ用の `ctrl`、`priority`)。少なくとも 2 つのコントロール・ネットワークでハートビートを使用し、1つのコントロールでコントロール・メッセージを使用しなければなりません。

    FailSafe では、複数のハートビート・ネットワークが必要です。通常、ノードがハートビート・メッセージを送信するのは、一度に 1 つのネットワーク上の別のノードに対してのみです。ただし、複数のネットワーク上の別のノードに対してハートビート・メッセージが同時に送信される場合もあります。これは、起動しているネットワークと起動していないネットワークを送信側のノードが識別できない場合に行われます。これは一時的な状態であり、最終的には、ハートビート・ネットワークは、優先度の最も高い、起動しているネットワークに収束されます。これは、ハートビート・ネットワークが一度に 1 つずつ順番に試行される FailSafe 1.2 とは異なります。

    ノードの異なるペアでは、ハートビートに対して異なるネットワークが使用される可能性があることについて、常に注意してください。

    FailSafe クラスタのすべてのノードには 2 つのコントロール・ネットワークがありますが、1 つのコントロール・ネットワークがあるプールに加わるようノードを定義することが可能です。

クラスタ・マネージャ GUI を使ったノードの定義

クラスタ・マネージャ GUI を使用してノードを定義するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「ノードとクラスタ(Nodes & Cluster)」カテゴリをクリックします。
3. 画面の右側で「ノードの定義(Define a Node)」タスク・リンクをクリックし、タスクを起動します。

4. 選択した入力値を画面で入力します。画面の下部にある「次へ(Next)」をクリックし、引続き 2 番目の画面で情報を入力します。

クラスタ・マネージャ CLI を使ったノードの定義

論理ノード定義を追加するには、次のコマンドを使用します。

```
cmgr> define node A
```

このコマンドを入力すると、定義しているノードの名前が指定され、ノードのパラメータを定義できるモードになります。これらのパラメータは、88 ページの「ノードの定義」で定義されるアイテムに対応します。以下のプロンプトが表示されます。

```
Enter commands, you may enter "done" or "cancel" at any time to exit

A?
```

ノード名のこのプロンプトが表示されたら、次の形式でノード・パラメータを入力します。

```
set hostname to B
 set nodeid to C
 set partition_id to D
 set reset_type to E
 set sysctrl_type to F
 set sysctrl_password to G
 set sysctrl_status to H
 set sysctrl_owner to I
 set sysctrl_device to J
 set sysctrl_owner_type to K
 set is_failsafe to L
 set is_cxfs to M
 set weight to N
 add nic O
         set heartbeat to P
         set ctrl_msgs to Q
         set priority to R
 remove nic S
```

ネットワーク・インタフェースを定義するには、add nic *J* コマンドを使用します。このコマンドは、定義する各ネットワーク・インタフェースに対して使用します。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。

```
Enter network interface commands, when finished enter "done" or "cancel"
NIC - J?
```

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 O
set ctrl_msgs to P
set priority to Q
```

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

```
cmgr> remove nic R
```

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
cmgr>
```

-p オプションでクラスタ・マネージャ CLI 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>? (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
```

CXFS ノードの FailSafe への切替え

既存の CXFS ノードは、FailSafe に適用されるよう再設定できます。

クラスタ・マネージャ GUI を使った CXFS ノードの FailSafe への切替え

クラスタ・マネージャ GUI を使用して CXFS ノードを CXFS と FailSafe の両方で使用できるよう再設定するには、以下の手順を実行します。GUI を使用して接続したサーバに CXFS をインストールしていない場合、このタスクは GUI メニューに表示されません。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「ノードとクラスタ(Nodes & Cluster)」カテゴリをクリックします。
3. 画面の右側で「CXFS ノードの FailSafe への切り替え(Convert a CXFS Node to FailSafe)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。

5. 画面の下部にある「OK」をクリックしてタスクを完了するか、「キャンセル(Cancel)」をクリックしてキャンセルします。

ノードは、CXFS と FailSafe の両方に適用されるようになります。

クラスタ・マネージャ CLI を使った CXFS ノードの 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、cmgr の 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。

プロンプト表示のある例:

```
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  ? (false) true
Is this a CXFS node  ? (true)
Node ID[optional] ? (13203)
Reset type  ? (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)
NIC 1 - (use network for control messages)  ? (true)
NIC 1 - Priority <1,2,...> ? (1)
Node Weight ? (0)

Successfully modified node cxfs6
```

プロンプト表示のない例:

```
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
```

ノードの変更

ノードを定義したら、クラスタ・マネージャ GUI またはクラスタ・マネージャ CLI を使用して、そのノードを変更できます。

クラスタ・マネージャ GUI を使ったノードの変更

クラスタ・マネージャ GUI を使用してノードを変更するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「ノードとクラスタ(Nodes & Cluster)」カテゴリをクリックします。
3. 画面の右側で「ノード定義の変更(Modify a Node Definition)」タスク・リンクをクリックし、タスクを起動します。
4. ノード・パラメータを変更します。
5. 画面の下部にある「OK」をクリックしてタスクを完了するか、「キャンセル(Cancel)」をクリックしてキャンセルします。

クラスタ・マネージャ CLI を使ったノードの変更

既存のノードを変更するには、次のコマンドを使用できます。このコマンドの入力後は、ノードの定義に使用するどのコマンドも実行できます。

```
cmgr> modify node A
```

ノードの削除

ノードを定義したら、クラスタ・マネージャ GUI またはクラスタ・マネージャ CLI を使用して、そのノードを削除します。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。

クラスタ・マネージャ GUI を使ったノードの削除

クラスタ・マネージャ GUI を使用してノードを削除するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「ノードとクラスタ(Nodes & Cluster)」カテゴリをクリックします。
3. 画面の右側で「ノードの削除(Delete a Node)」タスク・リンクをクリックし、タスクを起動します。
4. 削除するノードの名前を入力します。

5. 画面の下部にある「OK」をクリックしてタスクを完了するか、「キャンセル(Cancel)」をクリックしてキャンセルします。

クラスタ・マネージャ CLI を使ったノードの削除

ノードを定義した後、次のコマンドを使用してノードを削除できます。

```
cmgr> delete node A
```

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。つまり、ノードを削除するには、そのノードが含まれるクラスタをまず変更して、ノードが含まれないようにする必要があります。

ノードの表示

ノードを定義したら、以下の表示タスクを実行できます。

- ノードの属性の表示
- 特定のクラスタのメンバーであるノードの表示
- 定義されているすべてのノードの表示

これらのどのタスクも、IRIS FailSafe クラスタ・マネージャ GUI または IRIS FailSafe クラスタ・マネージャ CLI を使用して実行できます。

クラスタ・マネージャ GUI を使ったノードの表示

クラスタ・マネージャ GUI を使用した場合は、クラスタの定義済みのノードとそれらのノードの属性を FailSafe クラスタ表示で簡単にグラフィック表示できます。FailSafe クラスタ表示は、直接起動したり、「FailSafe マネージャ(FailSafe Manager)」画面の下部にある「FailSafe クラスタ表示(FailSafe Cluster View)」をクリックして、いつでも起動できます。

FailSafe クラスタ表示の「表示(View)」メニュー ->「プール内のノード(Nodes in Pool)」を選択して、FailSafe プール内で定義したノードをすべて表示できます。また、「クラスタ内のノード(Nodes In Cluster)」を選択して、デフォルトのクラスタに属するノードをすべて表示することもできます。ノードに関する詳細なステータスおよび設定情報を表示するには、任意のノードの名前またはアイコンをクリックします。

クラスタ・マネージャ CLI 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。

```
cmgr> show node A
```

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。

```
cmgr> show nodes in pool
```

指定したクラスタに対して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。

```
cmgr> show nodes [in cluster A]
```

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しなくても、デフォルトのクラスタで定義されているノードが表示されます。

IRIS FailSafe HA パラメータ

IRIS FailSafe システムのクラスタ内のノードの処理を決定するパラメータは、複数あります。

IRIS FailSafe パラメータは、以下のとおりです。

- タイブレーカー・ノード。これは、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マシンの論理名です。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。

  タイブレーカー・ノードは、クラスタ全体のパラメータです。

  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異種クラスタでは、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。

- ノード・タイムアウト。ミリ秒単位でのタイムアウト時間です。この時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。デフォルト値は、15000 ミリ秒です。

  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。

  ノード・タイムアウトは、クラスタ全体のパラメータです。

- ハートビート周期。ハートビート・メッセージ間の周期（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。ハートビート周期は、クラスタ全体のパラメータです。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。

- ノード待ち時間（ミリ秒単位）。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時間です。クラスタに対して値が設定されていない場合、FailSafe は、その値を、ノード・タイムアウトにノード数を掛けた値と仮定します。

- 電源異常モード。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指定します。これは、オンまたはオフに設定できます。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マシンに対して定義します。

クラスタ マネージャ GUI を使った IRIS FailSafe パラメータのリセット

クラスタ・マネージャ GUI を使用して IRIS FailSafe パラメータを設定するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「ノードとクラスタ(Nodes & Cluster)」カテゴリをクリックします。
3. 画面の右側で「FailSafe HA パラメータの設定(Set FailSafe HA Parameters)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックしてタスクを完了するか、「キャンセル(Cancel)」をクリックしてキャンセルします。

クラスタ マネージャ CLI を使った IRIS FailSafe パラメータのリセット

FailSafe パラメータは、次のコマンドを使用して変更できます。

```
cmgr> modify ha_parameters [on node A] [in cluster B]
```

デフォルトのノードまたはデフォルトのクラスタが指定されている場合は、このコマンドでノードやクラスタを指定する必要はありません。FailSafe は、デフォルトを使用します。

```
Enter commands, You may enter "done" or "cancel" at any time to exit
```

A?

ノード名またはクラスタ名のこのプロンプトが表示されたら、変更する FailSafe パラメータを次の形式で入力します。

```
set node_timeout to A
set heartbeat to B
set run_pwrfail to C
set node_wait to D
set tie_breaker to E
```

tie_breaker を"" 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。

クラスタの定義

クラスタは、ネットワークや、ネットワークと同様の相互接続によって互いに連結された 1 つまたは複数のノードの集合です。IRIS FailSafe では、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。

クラスタの定義を完了させるには、時間がかかります。クラスタを定義すると、デフォルトのログ設定が作成され、CDB にデフォルトの HA パラメータが作成されます。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 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、後から resource type install コマンドを使用して追加できます。

クラスタを定義するには、次の情報を指定してください。

- クラスタの論理名（最大 255 文字まで）
- ノードが FailSafe クラスタであるかどうか（クラスタ・マネージャ CLI の場合のみ。FailSafe クラスタ・マネージャ GUI で定義されたクラスタは、FailSafe クラスタです）。FailSafe クラスタおよび CXFS クラスタについての詳細は、44 ページの「CXFS との同時実行」を参照してください。

- クラスタが CXFS クラスタであるかどうか(クラスタ・マネージャ CLI の場合のみ。FailSafe クラスタ・マネージャ GUI で定義されたクラスタは、FailSafe クラスタです)。FailSafe クラスタおよび CXFS クラスタについての詳細は、44 ページの「CXFS との同時実行」を参照してください。
- 操作のモード: 通常(デフォルト)または試験中。試験中モードでは、ノード異常が検出されてもリソース・グループがフェイルオーバーしない FailSafe クラスタを設定できます。このモードは、ノード・タイムアウトやハートビート値を調整する場合に便利です。通常モードでクラスタを設定した場合は、ノードまたはリソース・グループで異常が検出されると、リソース・グループがフェイルオーバーされます。
- (オプション)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(たとえば、`root@system`)。
- (オプション)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・プログラム(たとえば、`/usr/sbin/Mail`)。

  電子メール・プログラムの指定はオプションであり、メールで通知を受信するためには、通知アドレスを指定するだけです。アドレスが指定されていない場合、通知は送信されません。

クラスタへのノードの追加

プールにノードを追加してクラスタを定義したら、そのクラスタに含めるためにノードの名前を指定してください。

クラスタ・マネージャ GUI を使ったクラスタの定義

クラスタ・マネージャ GUI を使用してクラスタを定義するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「設定ガイド(Guided Configuration)」をクリックします。
3. 画面の右側で「新規クラスタの設定(Set Up a New Cluster)」をクリックし、タスク・リンクを起動します。
4. 結果として表示されるウィンドウで各タスク・リンクが使用可能になったら、順にクリックします。各タスクに対して、選択した入力値を入力します。
5. 終了したら「OK」をクリックし、タスクセット・ウィンドウを閉じます。

クラスタ・マネージャ CLI を使ったクラスタの定義

CLI 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そのクラスタへのノードの追加を行います。

クラスタを定義するには、次のクラスタ・マネージャ CLI コマンドを使用します。

```
cmgr> define cluster A
```

このコマンドを入力すると、定義しているノードの名前が指定され、クラスタにノードを追加できるモードになります。次のプロンプトが表示されます。

```
cluster A?
```

クラスタ作成中にこのプロンプトが表示されたら、ノードを指定してクラスタに含め、このクラスタからのメッセージを送信する電子メール・アドレスを指定できます。

次のコマンドを使用してノードを指定し、クラスタに含めます。

```
cluster A? add node C
cluster A?
```

クラスタには、いくつでもノードを追加できます。

次のコマンドを使用して、メッセージの送信に使用する電子メール・プログラムを指定します。

```
cluster A? set notify_cmd to B
cluster A?
```

次のコマンドを使用して、メッセージを送信する電子メール・アドレスを指定します。

```
cluster A? set notify_addr to B
cluster A?
```

次のコマンドを使用して、FailSafe クラスタのモード（通常または試験中）を指定します。

```
cluster A? set ha_mode to D
cluster A?
```

次のコマンドを使用して、クラスタが FailSafe クラスタかどうかを指定します。

```
cluster A? set is_failsafe to E
cluster A?
```

次のコマンドを使用して、クラスタが CXFS クラスタかどうかを指定します。

```
cluster A? set is_cxfs to E
cluster A?
```

クラスタ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。

CXFS クラスタの FailSafe への切替え

既存の CXFS クラスタを、FailSafe でも使用できるよう再設定できます。

GUI を使用していて、既存の CXFS クラスタを FailSafe だけに適用させる場合は、その CXFS クラスタを削除し、FailSafe クラスタとして最初から再定義しなければなりません。CLI を使用している場合は、FailSafe クラスタ、CXFS クラスタ、または CXFS および FailSafe クラスタになるようクラスタを再設定できるので、この作業は必要ありません。

クラスタ・マネージャ GUI を使った CXFS クラスタの FailSafe への切替え

クラスタ・マネージャ GUI を使用して CXFS クラスタを CXFS と FailSafe の両方で使用できるよう再設定するには、以下の手順を実行します。GUI を使用して接続したサーバに CXFS をインストールしていない場合、このタスクは GUI メニューに表示されません。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「ノードとクラスタ(Nodes & Cluster)」カテゴリをクリックします。
3. 画面の右側で「CXFS クラスタの FailSafe への切り替え(Convert a CXFS Cluster to FailSafe)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックしてタスクを完了するか、「キャンセル(Cancel)」をクリックしてキャンセルします。

クラスタは、CXFS と FailSafe の両方に適用されるようになります。

クラスタ・マネージャ CLI 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 node` コマンドに続けて以下のコマンドを使用します。

```
set is_failsafe to true|false
set is_cxfs to true|false
```

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、次のコマンドを入力します。

```
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。

クラスタの変更

クラスタを定義したら、そのクラスタの属性を変更できます。クラスタの変更プロセスは、クラスタの作成プロセスと類似しています。

クラスタ・マネージャ GUI を使ったクラスタの変更

クラスタ・マネージャ GUI を使用してクラスタを変更するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「ノードとクラスタ(Nodes & Cluster)」カテゴリをクリックします。
3. 画面の右側で「クラスタ定義の変更(Modify a Cluster Definition)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックしてタスクを完了するか、「キャンセル(Cancel)」をクリックしてキャンセルします。

クラスタ・マネージャ CLI を使ったクラスタの変更

既存のクラスタを変更するには、次のコマンドを入力します。

```
cmgr> modify cluster A
```

このコマンドを入力すると、変更しているクラスタの名前が指定され、クラスタを変更できるモードになります。次のプロンプトが表示されます。

```
cluster A?
```

このプロンプトが表示されたら、次のコマンドを使用してクラスタ定義を変更できます。

```
cluster A? set notify_addr to B
```

```
cluster A? set notify_cmd to C
cluster A? add node D
cluster A? remove node D
cluster A?
```

クラスタを変更し終わったら、done と入力して、cmgr プロンプトに戻ります。

クラスタの削除

クラスタを定義したら、そのクラスタを削除できます。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。

クラスタ・マネージャ GUI を使ったクラスタの削除

クラスタ・マネージャ GUI を使用してクラスタを削除するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「ノードとクラスタ(Nodes & Cluster)」カテゴリをクリックします。
3. 画面の右側で「クラスタの削除(Delete a Cluster)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックしてタスクを完了するか、「キャンセル(Cancel)」をクリックしてキャンセルします。

クラスタ・マネージャ CLI を使ったクラスタの削除

定義済みのクラスタは、次のコマンドを使用して削除できます。

```
cmgr> delete cluster A
```

クラスタの表示

定義済みのクラスタは、クラスタ・マネージャ GUI またはクラスタ・マネージャ CLI を使用して表示できます。

クラスタ・マネージャ GUI を使ったクラスタの表示

クラスタ・マネージャ GUI を使用すると、クラスタとそのコンポーネントを FailSafe クラスタ表示で簡単に表示できます。FailSafe クラスタ表示は、直接起動したり、「FailSafe マネージャ(FailSafe Manager)」画面の下部にある「FailSafe クラスタ表示(FailSafe Cluster View)」プロンプトをクリックして、いつでも起動できます。

FailSafe クラスタ表示の「表示(View)」メニューから、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。ステータスおよび設定の情報が、新しいウィンドウに表示されます。「FailSafe クラスタ表示(FailSafe Cluster View)」ウィンドウ内でこの情報を表示するには、「オプション(Options)」を選択します。続いて「詳細を表示(Show Details)」オプションをクリックすると、ウィンドウの右側にステータスの詳細が表示されます。

クラスタ・マネージャ CLI を使ったクラスタの表示

クラスタを定義したら、次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。

```
cmgr> show cluster A
```

定義されているクラスタのリストは、次のコマンドを使用して表示できます。

```
cmgr> show clusters
```

リソース設定

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。

リソースの定義

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・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。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。

IRIS 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。

リソースを定義するには、次の情報を指定します。

- 定義するリソースの名前（最大 255 文字まで）
- 定義するリソースのタイプ。IRIS FailSafe システムには、NFS、Netscape_web、statd、MAC_Address、IP_Address、Oracle_DB、INFORMIX_DB、volume、filesystem などの定義済みリソース・タイプが含まれています。また、独自のリソース・タイプを定義することもできます。
- リソースが含まれているクラスタの名前

- リソースが含まれているノードの論理名（オプション）。ノードを指定した場合、リソースのローカルの定義がそのノードで定義されます。
- リソースのリソース・タイプ固有属性。各リソース・タイプでは、続く項で説明されているとおり、リソースに対して定義する特定のパラメータが必要な場合があります。

FailSafe 設定では、リソースを 100 個まで定義できます。

ボリューム・リソース属性

ボリューム・リソースは、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。

メモ：IRIS 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 ファイルシステムまたは XVM ボリュームの開始、停止、またはモニタを直接行うことはありません。また、CXFS ファイルシステムおよび 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。

ボリューム・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` にできますが、`/dev/xlv/xlv_vol` や `/dev/dsk/xlv/xlv_vol` にはできません。

XLV ボリュームがノードでまとめられると、`/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。

- 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。
- 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。
- 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。

ファイルシステム・リソース属性

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。

メモ: IRIS 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 ファイルシステムまたは XVM ボリュームの開始、停止、またはモニタを直接行うことはありません。また、CXFS ファイルシステムおよび 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。

高可用性でなければならない XFS ファイルシステムは、ファイルシステム・リソースとして設定する必要があります。ファイルシステム・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。ファイルシステム・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は、以下のパラメータをすべて指定してください。

- 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。
- 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。
- 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・チェックで問題があることが知られているものもあります。

IP_address リソース属性

IP_address リソースは、リソース・グループ内の高可用性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 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 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。

- IP アドレスのブロードキャスト・アドレス
- IP アドレスのネットワーク・マスク
- 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 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、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 IP アドレスを決定する優先順序が決まります。

MAC アドレス・リソース属性

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 (MAC) 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。

インタフェースに対して MAC アドレスを定義するときは、再 MAC されなければならないインタフェースを指定してください。

現在、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。

NFS リソース属性

NFS リソースは、高可用性として設定するすべての NFS ファイルシステムです。このリソース定義は、ファイルシステムおよび statd リソース・タイプに対して依存性を持ちます。

NFS リソースのリソース名は、NFS エクスポート・マウント・ポイントです。名前は有効なファイルシステム名でなければならないため、/disk1 の例のように、“ /” で開始する必要があります。

NFS リソースを定義するときは、以下のパラメータを指定してください。

- mount(1M) コマンドへの入力として使用されるファイルシステム。これは、既存のファイルシステム・リソースでなければなりません。
- exportfs(1M) コマンドで使用されるファイルシステムのエクスポート・オプション

- NFS リソースのファイルシステム・リソース依存性。これは、定義済みの filesystem リソースの名前でなければなりません。

statd_unlimited リソース属性

statd_unlimited リソースは、NFS リソースが含まれるリソース・グループに定義する場合のみ適用可能です。statd_unlimited リソースは、高可用性ファイルにロックと回復（lockf(3C)、fcntl(2)、および flock(3B)）を提供するために使用されます。statd_unlimited リソース・タイプを使用すると、クラスタ内の無制限のリソース・グループに対して NFS ロック・フェイルオーバーを提供できます。

IRIS FailSafe 用の statd_unlimited（NFS ロック）ディレクトリは、statd_unlimited リソースのリソース名で定義されます。これは、リソース・グループの一部である既存の高可用性ファイルシステムのディレクトリです。同じリソース・グループで定義されているすべてのファイルシステムに対して NFS フェイルオーバー・サポートを提供するには、リソース・グループに 1 つの statd_unlimited リソースだけを追加する必要があります。ディレクトリは、/disk1/statmon の例のように、通常は *filesystem*/statmon という形式です。

statd_unlimited リソース・タイプを使用すると、必要な数だけ statmon ディレクトリを持つことができます。このため、エクスポートされてロックが必要となるファイルシステムを FailSafe 設定に追加したり、リソース・グループにわたって多数のエクスポートを必要とする場合は、statd リソース・タイプではなく statd_unlimited リソース・タイプを使用してファイルシステムを設定します。

statd_unlimited リソースは、NFS リソース・タイプに対して依存性を持ちます。

statd_unlimited リソースを設定するときは、以下のパラメータを指定します。

- statmon ディレクトリに関連付けられた NFS エクスポート・ポイント。これは、statmon ディレクトリを置いたディレクトリまたはファイルシステムである場合があります。
- statd リソースの NFS リソース依存性

statd リソース属性

statd リソースは、IRIS FailSafe 2.*X* の以前のリリースとの互換性のために提供されています。

statd リソースは、NFS リソースが含まれるリソース・グループに定義する場合のみ適用可能です。statd リソースは、高可用性ファイルにロックと回復（lockf(3C)、fcntl(2)、および flock(3B)）を提供するために使用されます。

IRIS FailSafe 用の statd（NFS ロック）ディレクトリは、statd_unlimited リソースのリソース名で定義されます。これは、リソース・グループの一部である既存の高可用性ファイルシステムのディレクトリです。同じリソース・グループで定義されているすべてのファイルシステムに対して NFS フェイルオーバー・サポートを提供するには、リソース・グループに 1 つの statd リソースだけを追加する必要があります。ディレクトリは、/disk1/statmon の例のように、通常は *filesystem*/statmon という形式です。

statdリソースは、IP_addressリソース・タイプおよびfilesystemリソース・タイプに対して依存性を持ちます。

statdリソースを設定するときは、以下のパラメータを指定します。

- NFSクライアントの高可用性インタフェース・アドレス
- statdリソースのリソース依存性
  - IP_address依存性
  - filesystem依存性

Netscape_web リソース属性

高可用性でなければならないNetscape Webサーバは、Netscape_webリソースとして設定します。サーバは、Netscape FastTrackサーバまたはEnterpriseサーバにすることができます。このリソース定義は、IP_addressリソース・タイプに対して依存性を持ちます。ここでは、独自のファイルシステム依存性を追加することが推奨されています。Webサーバの内容は複数のノード間で複製できるため、これは必要な依存性ではありません。

クラスタのコンテキストの中でこのリソースを一意に定義する任意の文字列として、Netscape_webリソースのリソース名を指定します。

Netscape_webリソースを定義する際、以下のパラメータを指定してください。

- Webサーバがlistenするポートのポート番号
- Webサーバの開始コマンドと停止コマンドの場所
- モニタ・レベル。モニタ・スクリプトによって実行されるモニタ・アクションのタイプを定義します。モニタ・レベル1は、Webサーバのプロセスをモニタし、モニタ・レベル2は、サーバの応答を要求することによってWebサーバをモニタします。
- ホーム・ページ・ディレクトリ。Webサーバのホーム・ページ・ディレクトリの場所を定義します。
- Web IPアドレス。サーバ・ホストのIPアドレスです。これは、既存のIP_addressリソースでなければなりません。
- Netscape_webリソースのリソース依存性。IP_addressリソース・タイプです。

リソースへの依存性の追加

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。

リソースを定義する際、どのリソースに対してどのリソースが依存するかを定義できます。たとえば、Web サーバは、IP アドレスとファイルシステムの両方に対して依存できます。そして、ファイルシステムはボリュームに依存できます。

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 に対して依存している場合、リソース B をリソース A に対して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。

リソース定義に依存性を追加するときは、次の情報を指定します。

- 依存性を追加する先の既存のリソースの名前
- 依存性を追加する先の既存のリソースのリソース・タイプ
- リソースが含まれているクラスタの名前
- リソースが含まれているクラスタのノードの論理ノード名（オプション）。論理ノード名を指定すると、リソース依存性はリソースのノードの定義に追加されます。指定しないと、リソース依存性はクラスタ全体のリソース定義に追加されます。
- リソース依存性のリソース名
- リソース依存性のリソース・タイプ

クラスタ・マネージャ GUI を使ったリソースの定義

クラスタ・マネージャ GUI を使用してリソースを定義するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「リソースとリソース・タイプ(Resources & Resource Types)」カテゴリをクリックします。

3. 画面の右側で「新規リソースの定義(Define a New Resource)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックして、タスクを完了します。
6. 画面の右側で「リソース定義の依存性の追加と削除(Add/Remove Dependencies for a Resource Definition)」をクリックし、タスクを起動します。
7. 選択した入力値を入力します。
8. 画面の下部にある「OK」をクリックして、タスクを完了します。

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。ノード固有リソースについての詳細は、116 ページの「ノード固有リソースの定義」を参照してください。

クラスタ・マネージャ CLI を使ったリソースの定義

クラスタ全体のリソースを定義するには、次の CLI コマンドを使用します。

```
cmgr> define resource A [of resource_type B] [in cluster C]
```

このコマンドを入力すると、指定したクラスタ内に定義するリソースの名前とリソース・タイプが指定されます。デフォルトのクラスタまたはデフォルトのリソース・タイプが指定されている場合は、このコマンドでリソース・タイプやクラスタを指定する必要はなく、CLI によってデフォルトが使用されます。

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。ノード固有リソースについての詳細は、116 ページの「ノード固有リソースの定義」を参照してください。

次のプロンプトが表示されます。

```
resource A?
```

リソース作成中にこのプロンプトが表示されたら、以下のコマンドを入力して、定義するリソースの属性を指定し、リソースに対して依存性を追加または削除できます。

```
resource A? set key to value
resource A? add dependency E of type F
resource A? remove dependency E of type F
```

set *key* to *value* コマンドで定義する属性は、106 ページの「リソースの定義」で説明されているとおり、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、114 ページの「クラスタ・マネージャ CLI 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。

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
```

クラスタ・マネージャ CLI 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。

```
cmgr> show resource_type A in cluster B
```

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 *key* 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 chaos
```

結果画面の下部に、以下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 キーは XLV デバイス・ファイルのグループ ID、devname_owner キーは XLV デバイス・ファイルのオーナー、devname_mode キーはデバイス・ファイルのアクセス権をそれぞれ指定します。

たとえば、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
```

この節の以降の部分では、クラスタ・マネージャ CLI の *set key to value* コマンドを使用して、定義済みの IRIS FailSafe リソース・タイプに対して指定する属性の概要を説明します。

volume リソースを定義するときは、以下の属性をキーとして指定します。

devname-group	XLV デバイス・ファイルのグループ ID
devname_owner	XLV デバイス・ファイルのオーナー
devname_mode	デバイス・ファイルのアクセス権

filesystem リソースを定義するときは、以下の属性をキーとして指定します。

volume-name	ファイルシステムに関連付けられた XLV ボリュームの名前
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

IP_address リソースを定義するときは、以下の属性を指定します。

NetworkMask	IP アドレスのサブネット・マスク
interfaces	IP アドレスを設定できるインタフェースをコンマで区切ったリスト
BroadcastAddress	IP アドレスのブロードキャスト・アドレス

MAC アドレス・リソースを定義するときは、次の属性をキーとして指定します。

interface-name	再 MAC が必要なインタフェースの名前

NFS リソースを定義するときは、以下の属性をキーとして指定します。

export-info	exportfs(1M) コマンドで使用されるファイルシステムのエクスポート・オプション
filesystem	mount(1M) コマンドへの入力として使用されるファイルシステム

statd_unlimited リソースを定義するときは、次の属性をキーとして指定します。

ExportPoint	statmon ディレクトリに関連付けられている NFS エクスポート・ポイント

statd リソースを定義するときは、次の属性をキーとして指定します。

InterfaceAddress	NFS クライアントが使用するインタフェースの名前

Netscape_web リソースを定義するときは、以下の属性をキーとして指定します。

monitor-level	モニタ・レベル。モニタ・スクリプトによって実行されるモニタ・アクションのタイプを定義します。
port-number	Web サーバが listen するポートのポート番号
admin-scripts	Web サーバの開始コマンドと停止コマンドの場所
default-page-location	サーバのデフォルトの Web ページの場所
web-ipaddr	Web サーバ用の高可用性インタフェースの IP アドレス

ノード固有リソースの定義

特定のノードだけに適用されるリソース定義を持つ既存のリソースを再定義できます。再定義できるのは、既存のクラスタ全体のリソースだけです。特定のノードに対してすでに定義されているリソースは再定義できません。

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ます。たとえば、あるOrigin ノードの Gigabit Ethernet インタフェース eg0 と、別の Origin ノードの 100baseT インタフェース ef0 に、IP_address 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。

クラスタ・マネージャ GUI を使ったノード固有リソースの定義

クラスタ・マネージャ GUI を使用すると、既存のクラスタ全体のリソース定義を取得して、そのクラスタ内の特定のノードで使用できるように再定義できます。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「リソースとリソース・タイプ(Resources & Resource Types)」カテゴリをクリックします。
3. 画面の右側で「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・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックして、タスクを完了します。

クラスタ・マネージャ CLI を使ったノード固有リソースの定義

クラスタ・マネージャ CLI 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 CLI コマンドを使用します。

```
cmgr> define resource A of resource_type B on node C [in cluster D]
```

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、CLI によってデフォルトが使用されます。

リソースの変更

リソースを定義したら、そのリソースを変更したり削除できます。

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。

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。

クラスタ・マネージャ GUI を使ったリソースの変更

クラスタ・マネージャ GUI を使用してリソースを変更するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「リソースとリソース・タイプ(Resources & Resource Types)」カテゴリをクリックします。
3. 画面の右側で「リソース定義の変更(Modify a Resource Definition)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックしてタスクを完了するか、「キャンセル(Cancel)」をクリックしてキャンセルします。

クラスタ・マネージャ CLI を使ったリソースの変更

リソースを変更するには、次の CLI コマンドを使用します。

```
cmgr> modify resource A of resource_type B [in cluster C]
```

このコマンドを入力すると、指定したクラスタ内の変更中のリソースの名前とリソース・タイプが指定されます。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、CLI によってデフォルトが使用されます。

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。

リソースの削除

リソースを定義したら、そのリソースを削除できます。

クラスタ・マネージャ GUI を使ったリソースの削除

クラスタ・マネージャ GUI を使用してリソースを削除するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「リソースとリソース・タイプ(Resources & Resource Types)」カテゴリをクリックします。
3. 画面の右側で「リソースの削除(Delete a Resource)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックしてタスクを完了するか、「キャンセル(Cancel)」をクリックしてキャンセルします。

クラスタ・マネージャ CLI 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できます。

```
cmgr> delete resource A of resource_type B [in cluster D]
```

リソースの表示

リソースはさまざまな方法で表示できます。特定の定義済みリソースの属性を表示したり、指定したリソース・グループ内のすべての定義済みリソースを表示したり、指定したリソース・タイプのすべての定義済みリソースを表示できます。

クラスタ・マネージャ GUI を使ったリソースの表示

クラスタ・マネージャ GUI を使用すると、FailSafe クラスタ表示でリソースを簡単に表示できます。FailSafe クラスタ表示は、直接起動したり、「FailSafe マネージャ(FailSafe Manager)」画面の下部にある「FailSafe クラスタ表示(FailSafe Cluster View)」プロンプトをクリックして、いつでも起動できます。

すべての定義済みリソースを表示するには、FailSafe クラスタ表示の「表示(View)」メニュー ->「リソース(Resources)」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（緑はオンライン、グレーはオフラインを示します）。また、「表示(View)」メニュー ->「タイプごとのリソース(Resources by Type)」を選択して、リソース・タイプ別にリソースを表示したり、「グループ内のリソース(Resources in Groups)」を選択して、リソース・グループ別にリソースを表示することもできます。

クラスタ・マネージャ CLI を使ったリソースの表示

定義済みリソースのパラメータを表示するには、次のコマンドを使用します。

```
cmgr> show resource A of resource_type B
```

リソース・グループ内の定義済みリソースをすべて表示するには、次のコマンドを使用します。

```
cmgr> show resources in resource_group A [in cluster B]
```

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、CLI によってデフォルトが使用されます。

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には、次のコマンドを使用します。

```
cmgr> show resources of resource_type A [in cluster B]
```

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、CLI によってデフォルトが使用されます。

リソース・タイプの定義

IRIS 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 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、130 ページの「リソース・タイプのクラスタへのインストール（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。

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、次の情報が必要です。

- リソース・タイプの名前（最大 255 文字まで）
- リソース・タイプの適用先のクラスタの名前
- リソース・タイプが適用されるノード（ソース・タイプを特定のノードに制限する場合）
- 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る
  - リソースは、この値が大きい順に停止される

    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。
- 再開モード。以下のいずれかの値を指定できます。
  - 0 = モニタ異常時に再開しない

  - 1 = 固定数回再開する

- ローカル再開の数（再開モードが 1 の場合）

  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 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。クラスタ・マネージャ GUI またはクラスタ・マネージャ CLI 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、183 ページの「リソース・グループのモニタリングの停止（メインテナンス・モード）」を参照してください。

- 実行可能スクリプトの場所。これは、常に `/var/cluster/ha/resource_types/`*rtname* です。*rtname* は、リソース・タイプ名です。
- モニタ周期。`monitor` アクション・スクリプトの連続実行間隔（ミリ秒単位）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。
- モニタの開始時刻。ノードでリソース・グループがオンラインになると、IRIS FailSafe は、指定した時間（ミリ秒単位）後にリソースをモニタし始めます。
- このリソース・タイプに対して定義するアクション・スクリプト。`start`、`stop`、`exclusive`、および `monitor` 用のスクリプトを指定する必要があります。ただし、`monitor` スクリプトには return-success 関数しか含めることができません。再開モードに対して 1 を指定した場合は、`restart` スクリプトを指定してください。
- このリソース・タイプに対して定義するタイプ固有属性。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。たとえば、NFS には以下のリソース・キーが必要です。

  - `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
  ```

新しいリソース・タイプを定義するには、クラスタ・マネージャ GUI またはクラスタ・マネージャ CLI を使用します。

クラスタ・マネージャ GUI を使ったリソース・タイプの定義

クラスタ・マネージャ GUI を使用してリソース・タイプを定義するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「リソースとリソース・タイプ(Resources & Resource Types)」カテゴリをクリックします。
3. 画面の右側で「リソース・タイプの定義(Define a Resource Type)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックして、タスクを完了します。

クラスタ・マネージャ CLI を使ったリソース・タイプの定義

以下の手順に、newresourcetype というリソース・タイプをインタラクティブに定義するための cmgr（cluster_mgr(1m) と同じコマンド）の使用方法を示します。

1. root としてログインします。
2. -p オプションを使用して cmgr コマンドを実行し（コマンド名は cmgr に省略できます）、情報を入力するプロンプトを表示します。

   ```
   # /usr/cluster/bin/cmgr -p
   Welcome to SGI Cluster Manager Command-Line Interface

   cmgr>
   ```

3. set サブコマンドを使用して、cmgr 操作に使用するデフォルトのクラスタを指定します。この例では、TEST というクラスタを使用しています。

```
cmgr> set cluster TEST
```

メモ:各サブコマンドの必要に応じてクラスタ名を指定できます。

4. define resource_type サブコマンドを使用します。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。

メモ:以下の例に、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
#
```

ノード固有リソース・タイプの定義

特定のノードだけに適用されるリソース定義を持つ既存のリソース・タイプは、再定義できます。再定義できるのは、既存のクラスタ全体のリソース・タイプだけです。特定のノードに対してすでに定義されているリソース・タイプは再定義できません。

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。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。

クラスタ・マネージャ GUI を使ったノード固有リソース・タイプの定義

クラスタ・マネージャ GUI を使用すると、既存のクラスタ全体のリソース・タイプ定義を取得して、そのクラスタ内の特定のノードで使用できるよう再定義できます。以下のタスクを実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「リソースとリソース・タイプ(Resources & Resource Types)」カテゴリをクリックします。
3. 画面の右側で「特定ノードへのリソース・タイプの再定義(Redefine a Resource Type For a Specific Node)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックして、タスクを完了します。

クラスタ・マネージャ CLI を使ったノード固有リソース・タイプの定義

クラスタ・マネージャ CLI 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同じ方法で、ノード固有リソース・タイプを再定義できます。ただし、`define resource_type` 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・タイプを定義するには、次の CLI コマンドを使用します。

```
cmgr> define resource_type A on node B [in cluster C]
```

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、CLI によってデフォルトが使用されます。

リソース・タイプへの依存性の追加

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。

クラスタ・マネージャ GUI を使用している場合、「リソースとリソース・タイプ(Resources & Resource Types)」画面から「リソース・タイプの依存性の追加と削除(Add/Remove Dependencies for a Resource Type)」を選択して、示された入力を提供することにより、リソース・タイプに対して依存性を追加したり削除します。クラスタ・マネージャ CLI を使用している場合は、リソース・タイプを削除または変更する際に依存性を追加したり削除します。

リソース・タイプの変更

リソース・タイプを定義したら、そのリソース・タイプを変更できます。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。

クラスタ・マネージャ GUI を使ったリソース・タイプの変更

クラスタ・マネージャ GUI を使用してリソース・タイプを変更するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「リソースとリソース・タイプ(Resources & Resource Types)」カテゴリをクリックします。
3. 画面の右側で「リソース・タイプ定義の変更(Modify a Resource Type Definition)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックしてタスクを完了するか、「キャンセル(Cancel)」をクリックしてキャンセルします。

クラスタ・マネージャ CLI を使ったリソース・タイプの変更

リソースを変更するには、次の CLI コマンドを使用します。

```
cmgr> modify resource_type A [in cluster B]
```

このコマンドを入力すると、指定したクラスタ内で変更中のリソース・タイプが指定されます。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、CLI によってデフォルトが使用されます。

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。

CLI では、リソース・タイプ・タイムアウトの現在の値を表示でき、どのアクション・タイムアウトでも変更できます。

以下に、CLI を使用して 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
Welcome to SGI Cluster Manager Command-Line Interface

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
```

以下の例に、CLI を使用してプロンプト・モードでリソース・タイプ・タイムアウトを変更する方法を示します。

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

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

リソース・タイプを定義したら、そのリソース・タイプを削除できます。

クラスタ・マネージャ GUI を使ったリソース・タイプの削除

クラスタ・マネージャ GUI を使用してリソース・タイプを削除するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「リソースとリソース・タイプ(Resources & Resource Types)」カテゴリをクリックします。
3. 画面の右側で「リソース・タイプの削除(Delete a Resource Type)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックしてタスクを完了するか、「キャンセル(Cancel)」をクリックしてキャンセルします。

クラスタ・マネージャ CLI 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できます。

```
cmgr> delete resource_type A [in cluster B]
```

リソース・タイプのクラスタへのインストール(ロード)

クラスタを定義すると、デフォルト値が含まれている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。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。

インストールするリソース・タイプ定義は、クラスタに存在できません。

クラスタ・マネージャ GUI を使ったリソース・タイプのインストール

GUI を使用してリソース・タイプをインストールするには、「リソースとリソース・タイプ(Resources & Resource Types)」タスク・ページから「リソース・タイプのロード(Load a Resource Type)」タスクを選択して、ロードするリソース・タイプを入力します。

クラスタ・マネージャ CLI を使ったリソース・タイプのインストール

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 CLI コマンドを使用します。

```
cmgr> install resource_type A [in cluster B]
```

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、CLI によってデフォルトが使用されます。

リソース・タイプの表示

リソース・タイプを定義したら、そのリソース・タイプを表示できます。

クラスタ・マネージャ GUI を使ったリソース・タイプの表示

クラスタ・マネージャ GUI を使用すると、リソース・タイプを FailSafe クラスタ表示で簡単に表示できます。FailSafe クラスタ表示は、直接起動したり、「FailSafe マネージャ(FailSafe Manager)」画面の下部にある「FailSafe クラスタ表示(FailSafe Cluster View)」プロンプトをクリックして、いつでも起動できます。

FailSafe クラスタ表示の「表示(View)」メニュー ->「タイプ(Types)」を選択して、すべての定義済みリソース・タイプを表示します。その後、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表示できます。

クラスタ・マネージャ CLI を使ったリソース・タイプの表示

指定したクラスタ内の定義済みリソース・タイプのパラメータを表示するには、次のコマンドを使用します。

```
cmgr> show resource_type A [in cluster B]
```

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、CLI によってデフォルトが使用されます。

クラスタ内の定義済みリソース・タイプをすべて表示するには、次のコマンドを使用します。

```
cmgr> show resource_types [in cluster A]
```

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、CLI によってデフォルトが使用されます。

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には、次のコマンドを使用します。

```
cmgr> show resource_types installed
```

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。

- フェイルオーバー・ポリシーの名前(最大 63 文字まで)。プール内で一意でなければなりません。
- 既存のフェイルオーバー・スクリプトの名前
- 初期フェイルオーバー・ドメイン。リソース・グループが実行される可能性のあるノードの順序付きリストです。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指定します。これはフェイルオーバー・スクリプトに入力され、ランタイム・フェイルオーバー・ドメインが生成されます。
- フェイルオーバー属性。フェイルオーバー・スクリプトの処理を変更します。

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。

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。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 に対しては、mercury、venus、pluto
  - RG2 に対しては、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。負荷などの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 N3 N1
  N2 N1 N3
  N3 N2 N1
  N3 N1 N2
  ```

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。

フェイルオーバー属性

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、IRIS 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。

- 必須の属性の開始条件は、クラスタがすでに高可用性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開始条件は、高可用性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5-1 に、各属性について説明します。

表5-1 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。

クラス	名前	説明
	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。
	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リース・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。

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 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 フェイルオーバー・スクリプトは、IRIS 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 フェイルオーバー・スクリプトも、IRIS 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。

クラスタ・マネージャ GUI を使ったフェイルオーバー・ポリシーの定義

GUI を使用してフェイルオーバー・ポリシーを定義するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「フェイルオーバー・ポリシーとリソース・グループ(Failover Policies & Resource Groups)」カテゴリをクリックします。
3. 画面の右側で「フェイルオーバー・ポリシーの定義(Define a Failover Policy)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックして、タスクを完了します。

クラスタ・マネージャ CLI 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・ポリシーを定義するには、cmgr プロンプトで次のコマンドを入力し、そのフェイルオーバー・ポリシーの名前を指定します。

```
cmgr> define failover_policy A
```

次のプロンプトが表示されます。

```
failover_policy A?
```

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、フェイルオーバー・ポリシーのコンポーネントを指定できます。

```
failover_policy A? set attribute to B
failover policy A? set script to C
failover policy A? set domain to D
failover_policy A?
```

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コマンドとadd domainコマンドで異なる値を設定することはできません。また、CLIでは、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。

```
failover_policy A? set domain to A B C ...
```

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、132ページの「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。

フェイルオーバー・ポリシー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。

フェイルオーバー・ポリシーの変更

フェイルオーバー・ポリシーを定義したら、そのフェイルオーバー・ポリシーを変更したり削除できます。

クラスタ・マネージャGUIを使ったフェイルオーバー・ポリシーの変更

クラスタ・マネージャGUIを使用してフェイルオーバー・ポリシーを変更するには、以下の手順を実行します。

1. FailSafe Toolchestで「FailSafeマネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「フェイルオーバー・ポリシーとリソース・グループ(Failover Policies & Resource Groups)」カテゴリをクリックします。
3. 画面の右側で「フェイルオーバー・ポリシー定義の変更(Modify a Failover Policy Definition)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックしてタスクを完了するか、「キャンセル(Cancel)」をクリックしてキャンセルします。

クラスタ・マネージャCLIを使ったフェイルオーバー・ポリシーの変更

フェイルオーバー・ポリシーを変更するには、次のCLIコマンドを使用します。

```
cmgr> modify failover_policy A
```

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。

フェイルオーバー・ポリシーの削除

フェイルオーバー・ポリシーを定義したら、そのフェイルオーバー・ポリシーを削除できます。

クラスタ・マネージャ GUI を使ったフェイルオーバー・ポリシーの削除

クラスタ・マネージャ GUI を使用してフェイルオーバー・ポリシーを削除するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「フェイルオーバー・ポリシーとリソース・グループ(Failover Policies & Resource Groups)」カテゴリをクリックします。
3. 画面の右側で「フェイルオーバー・ポリシーの削除(Delete a Failover Policy)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックしてタスクを完了するか、「キャンセル(Cancel)」をクリックしてキャンセルします。

クラスタ・マネージャ CLI 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・ポリシー定義を削除するには、次のコマンドを使用できます。

```
cmgr> delete failover_policy A
```

フェイルオーバー・ポリシーの表示

IRIS 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
- 定義されているすべてのフェイルオーバー・ポリシー
- 定義されているすべてのフェイルオーバー・ポリシー属性
- 定義されているすべてのフェイルオーバー・ポリシー・スクリプト

クラスタ・マネージャ GUI を使ったフェイルオーバー・ポリシーの表示

クラスタ・マネージャ GUI を使用すると、FailSafe クラスタ表示で簡単にフェイルオーバー・ポリシーを表示できます。FailSafe クラスタ表示は、直接起動したり、「FailSafe マネージャ(FailSafe Manager)」画面の下部にある「FailSafe クラスタ表示(FailSafe Cluster View)」プロンプトをクリックして、いつでも起動できます。

FailSafe クラスタ表示の「表示(View)」メニュー ->「フェイルオーバー・ポリシー(Failover Policies)」を選択して、すべての定義済みフェイルオーバー・ポリシーを表示します。

クラスタ・マネージャ CLI を使ったフェイルオーバー・ポリシーの表示

定義済みフェイルオーバー・ポリシーのパラメータを表示するには、次のコマンドを使用します。

```
cmgr> show failover_policy A
```

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、次のコマンドを使用します。

```
cmgr> show failover policies
```

定義済みフェイルオーバー・ポリシー属性をすべて表示するには、次のコマンドを使用します。

```
cmgr> show failover_policy attributes
```

定義済みフェイルオーバー・ポリシー・スクリプトをすべて表示するには、次のコマンドを使用します。

```
cmgr> show failover_policy scripts
```

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、IRIS 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 IP アドレス、内容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、異常時のリソース・グループの処理が制御されます。

リソース・グループを定義するには、次の情報を指定します。

- リソース・グループの名前(最大 63 文字まで)
- リソース・グループが使用可能なクラスタの名前

- リソース・グループに含めるリソースとそのリソース・タイプ
- フェイルオーバー・ポリシーの名前。異常時にリソース・グループのサービスを引継ぐノードを決定します。

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・グループに設定されたリソースを最大 100 個定義できます。

クラスタ・マネージャ GUI を使ったリソース・グループの定義

クラスタ・マネージャ GUI を使用してリソース・グループを定義するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「設定ガイド(Guided Configuration)」をクリックします。
3. 画面の右側で「高可用性リソース・グループの設定(Set Up a Highly Available Resource Groups)」をクリックし、タスク・リンクを起動します。
4. 結果として表示されるウィンドウで各タスク・リンクが使用可能になったら、順にクリックします。各タスクに対して、選択した入力値を入力します。
5. 終了したら「OK」をクリックし、タスクセット・ウィンドウを閉じます。

クラスタ・マネージャ CLI を使ったリソース・グループの定義

リソース・グループを設定するには、cmgr プロンプトで次のコマンドを入力し、リソース・グループの名前と、そのリソース・グループが使用可能なクラスタを指定します。

```
cmgr> define resource_group A [in cluster B]
```

このコマンドを入力すると、指定したクラスタ内に定義するリソース・グループの名前が指定されます。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、CLI によってデフォルトが使用されます。

次のプロンプトが表示されます。

```
Enter commands, you may enter "done" or "cancel" at any time to exit
resource_group A?
```

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、リソース・グループに対してリソースを追加または削除するよう指定したり、リソース・グループに適用するフェイルオーバー・ポリシーを指定できます。

```
resource_group A? add resource B of resource_type C
resource_group A? remove resource D of resource_type E
```

```
resource_group A? set failover_policy to F
```

フェイルオーバー・ポリシーを設定し、リソース・グループにリソースを追加し終わったら、done と入力して、cmgr プロンプトに戻ります。

クラスタ・マネージャ CLI を使ったリソース・グループ作成の例については、148 ページの「リソース・グループの作成例」を参照してください。

リソース・グループの変更

リソース・グループを定義したら、そのリソース・グループを変更できます。リソース・グループのフェイルオーバー・ポリシーは、そのリソース・グループに関連付けられている新しいフェイルオーバー・ポリシーを指定することによって変更できます。また、既存のリソース・グループにリソースを追加したり削除することもできます。ただし、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ないことに注意してください。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。また、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。

クラスタ・マネージャ GUI を使ったリソース・グループの変更

クラスタ・マネージャ GUI を使用してリソース・グループを変更するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「フェイルオーバー・ポリシーとリソース・グループ(Failover Policies & Resource Groups)」カテゴリをクリックします。
3. 画面の右側で「リソース・グループ定義の変更(Modify a Resource Group Definition)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックしてタスクを完了するか、「キャンセル(Cancel)」をクリックしてキャンセルします。

クラスタ・マネージャ GUI を使用してリソース・グループ定義に対してリソースを追加または削除するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「フェイルオーバー・ポリシーとリソース・グループ(Failover Policies & Resource Groups)」カテゴリをクリックします。

3. 画面の右側で「グループ内のリソースの追加と削除(Add/Remove Resources in Resource Group)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックしてタスクを完了するか、「キャンセル(Cancel)」をクリックしてキャンセルします。

クラスタ・マネージャ CLI を使ったリソース・グループの変更

リソース・グループを変更するには、次の CLI コマンドを使用します。

```
cmgr> modify resource_group A [in cluster B]
```

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、CLI によってデフォルトが使用されます。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。

```
resource_group A? add resource B of resource_type C
resource_group A? remove resource D of resource_type E
resource_group A? set failover_policy to F
```

リソース・グループの削除

リソース・グループを定義したら、そのリソース・グループを削除できます。

クラスタ・マネージャ GUI を使ったリソース・グループの削除

クラスタ・マネージャ GUI を使用してリソース・グループを削除するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「フェイルオーバー・ポリシーとリソース・グループ(Failover Policies & Resource Groups)」カテゴリをクリックします。
3. 画面の右側で「リソース・グループの削除(Delete a Resource Group)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックしてタスクを完了するか、「キャンセル(Cancel)」をクリックしてキャンセルします。

クラスタ・マネージャ CLI を使ったリソース・グループの削除

リソース・グループ定義を削除するには、次のコマンドを使用できます。

```
cmgr> delete resource_group A [in cluster B]
```

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、CLI によってデフォルトが使用されます。

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。

クラスタ・マネージャ GUI を使ったリソース・グループの表示

クラスタ・マネージャ GUI を使用すると、FailSafe クラスタ表示で簡単にリソース・グループを表示できます。FailSafe クラスタ表示は、直接起動したり、「FailSafe マネージャ(FailSafe Manager)」画面の下部にある「FailSafe クラスタ表示(FailSafe Cluster View)」プロンプトをクリックして、いつでも起動できます。

FailSafe クラスタ表示の「表示(View)」メニュー ->「グループ(Groups)」を選択して、すべての定義済みリソース・グループを表示します。

どのノードがどのグループを現在実行しているかを表示するには、「ノードが所有するグループ(Groups owned by Nodes)」を選択します。どのグループがどのフェイルオーバー・ポリシーを実行しているかを表示するには、「フェイルオーバー・ポリシーごとのグループ(Groups by Failover Policies)」を選択します。

クラスタ・マネージャ CLI 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を表示するには、次のコマンドを使用します。

```
cmgr> show resource_group A [in cluster B]
```

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、CLI によってデフォルトが使用されます。

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、次のコマンドを使用します。

```
cmgr> show resource_groups [in cluster A]
```

FailSafe システム・ログ設定

IRIS 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。

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`) 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

ログ・グループ設定情報は、`cli` ログ・グループおよび `crsd` ログ・グループの場合はプール内のすべてのノードについて維持され、その他のすべてのログ・グループの場合はクラスタ内のすべてのノードについて維持されます。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。

ログ・グループを設定するときは、次の情報を指定します。

- ログ・レベル。下に説明されているとおり、GUI の場合は文字列、CLI の場合は数値（1～19）で指定します。
- ログの記録先のログファイル
- カスタマイズする指定したログ・グループがあるノード（オプション）

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5-2 に、GUI および CLI を使用して指定する場合のログ・レベルを示します。

表5-2 ログ・レベル

GUI レベル	CLI 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。
「Debug0」	10	
...		「Debug0」～「Debug9」(CLI の場合は 11～19)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。
「Debug9」	19	

メモ: 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。

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・ファイル名は、以下のとおりです。

/var/cluster/ha/log/cmsd_*nodename*

ノード nodename の FailSafe メンバーシップ・サービス・デーモンに対するログ・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*

クラスタ・マネージャ GUI およびクラスタ・マネージャ CLI によって呼出されるノード nodename のログ・ファイルまたは内部管理コマンド

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「IRIS FailSafe の回復」を参照してください。

クラスタ・マネージャ GUI を使ったログ・グループの設定

クラスタ・マネージャ GUI を使用してログ・グループを設定するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「ノードとクラスタ(Nodes & Cluster)」カテゴリをクリックします。
3. 画面の右側で「ログの設定(Set Log Configuration)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。

5. 画面の下部にある「OK」をクリックして、タスクを完了します。

クラスタ・マネージャ CLI を使ったログ・グループの設定

ログ・グループは、次の CIl コマンドを使用して設定できます。

```
cmgr> define log_group A [on node B] [in cluster C]
```

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、そのノードを指定します。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、FailSafe によってデフォルトが使用されます。

次のプロンプトが表示されます。

```
Enter commands, you may enter "done" or "cancel" at any time to exit
log_group A?
```

ノード名のこのプロンプトが表示されたら、変更するログ・グループ・パラメータを次の形式で入力します。

```
log_group A? set log_level to A
log_group A? add log_file A
log_group A? remove log_file A
```

ログ・グループを設定し終わったら、done と入力して、cmgr プロンプトに戻ります。

クラスタ・マネージャ CLI 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 CLI コマンドを使用します。

```
cmgr> modify log_group A on [node B] [in cluster C]
```

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。

クラスタ・マネージャ GUI を使ったログ・グループ定義の表示

クラスタ・マネージャ GUI を使用してログ・グループ定義を表示するには、「ログの設定(Set Log Configuration)」を実行し、表示するログ・グループをロールオーバー・メニューから選択します。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。

クラスタ・マネージャ CLI を使ったログ・グループ定義の表示

定義済みリソースのパラメータを表示するには、次のコマンドを使用します。

```
cmgr> 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「IRIS FailSafe の回復」を参照してください。

リソース・グループの作成例

クラスタ・マネージャ CLI 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

     リソース・グループ内のリソースについて、リソース依存関係とリソース・タイプ依存性がすべて満足されていなければなりません。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ます。

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表示します。

   - 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. cluster_mgr(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。

3 ノード・クラスタに関する FailSafe の例

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。

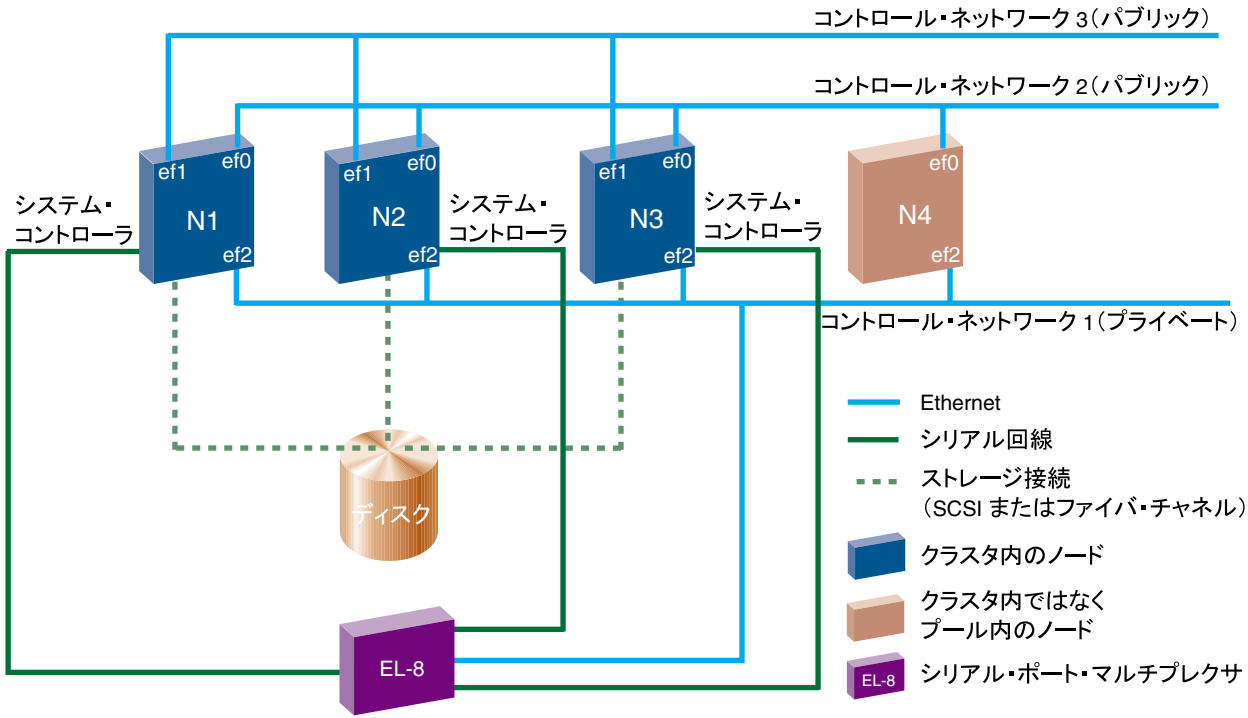

図6-1 FailSafe の設定例

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。

設定する FailSafe cmgr スクリプトの例

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。CLI スクリプトについての一般情報は、80 ページの「CLI コマンド・スクリプト」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 CLI テンプレート・ファイルについての詳細は、81 ページの「CLI テンプレート・スクリプト」を参照してください。

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。

リソース・グループ `RG1` には、以下のリソースが含まれています。

IP アドレス　　192.26.50.1

ファイルシステム	/ha1
ボリューム	ha1_vol
NFS	/ha1/export

リソース・グループ RG1 には、FP1 というフェイルオーバー・ポリシーがあります。FP1 には、以下のコンポーネントが含まれています。

スクリプト	ordered
属性	Auto_Failback
	Auto_Recovery
フェイルオーバー・ドメイン	N1, N2, N3

リソース・グループ RG2 には、以下のリソースが含まれています。

IP アドレス	192.26.50.2
ファイルシステム	/ha2
ボリューム	ha2_vol
NFS	/ha2/export

リソース・グループ RG2 には、FP2 というフェイルオーバー・ポリシーがあります。FP2 には、以下のコンポーネントが含まれています。

スクリプト	round-robin
属性	Controlled_Failback
	Inplace_Recovery
フェイルオーバー・ドメイン	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/cluster_mgr -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 monitoring-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 monitoring-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
```

CXFS ファイルシステムを含めるための FailSafe 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。

メモ: IRIS 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 ファイルシステムまたは XVM ボリュームの開始、停止、またはモニタを直接行うことはありません。また、CXFS ファイルシステムおよび 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 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 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。

      ```
      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。

AFD	`N1, N2`
スクリプト	`ordered`
属性	`Inplace_Recovery`

5. 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 `NFS1` のリソース `/cxfs/exports` を持つリソース・グループ `RG3` を作成します。
6. 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、および CXFS ファイルシステムのマウントについての詳細は、『CXFS Software Installation and Administration Guide』を参照してください。

7. 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。

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 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。IP アドレス・リソースの指定についての詳細は、108 ページの「IP_address リソース属性」を参照してください。

以下の例では、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 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、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
```

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、CXFS ファイルシステムで使用する特別な NFS リソース・タイプを作成しなければなりません。これは、エクスポートされる CXFS ファイルシステムは、FailSafe によって操作またはモニタされないので必要となります。このため、新しいリソース・タイプにはファイルシステム依存性が含まれません。

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. NFS エージェント・ソフトウェアとして、NFS 2.1 リリースがロードされていることを確認します。
2. CXFS ファイルシステムに対して新しい NFS リソース・タイプを作成します。この手順では、新しいリソース・タイプを NFS_CXFS と呼びます。

   以下の手順を使用して、既存の NFS リソース・タイプに類似した新しいリソース・タイプを作成できます。

   a. var/cluster/ha/resource_types/NFS ディレクトリを、新しいリソース・タイプの名前を持つディレクトリにコピーします。この場合、コピー先のディレクトリは var/cluster/ha/resource_types/NFS_CXFS です。

   b. 新しい NFS_CXFS ディレクトリを作成したら、ディレクトリ内にあるすべてのスクリプトを変更し、LOCAL_TEST_KEY に対して NFS_CXFS を指定します。NFS_CXFS という新しいリソース・タイプ名を反映するために、スクリプトで NFS を参照している部分（変数名など）を変更することが推奨されていますが、ログ・メッセージの NFS の参照部分は変更しないでください。
3. 新しい NFS_CXFS リソース・タイプを作成したら、以下のとおりに GUI または cmgr のいずれかを使用して、このリソース・タイプからファイルシステム依存性を削除します。

```
perf34 40# cmgr
Welcome to SGI Cluster Manager Command-Line Interface

cmgr> modify resource_type NFS_CXFS in cluster "testcluster"
Enter commands, when finished enter either "done" or "cancel"

resource_type NFS_CXFS ? remove dependency filesystem
resource_type NFS_CXFS ? done
Successfully modified resource_type NFS_CXFS
```

4. 新しいリソース・タイプ /var/cluster/ha/resource_types/NFS_CXFS/monitor のモニタ・スクリプトを変更します。新しいリソース・タイプのモニタ・スクリプトが標準の NFS モニタ・スクリプトと違うのは、CXFS ファイルシステムをエクスポートしたときは、ファイルシステムがマウントされているかどうかを FailSafe がチェックしないようにし、マウントされていなくても HA_CMD_FAILED で FailSafe が終了しないようにしている点です。ファイルシステムがアンマウントされた場合に必要なアクションは、CXFS モニタ・スクリプト自体で決定します。

モニタ・スクリプトの次のセクションでは、exit_script で始まる行がコメント・アウトされています。コマンドのステータスはログに書込まれますが、スクリプトは終了しません。

```
# Check to see if the filesystem is mounted
        HA_CMD="/sbin/mount | grep $fs  >> /dev/null 2>&1"
        ha_execute_cmd "check to see if $fs is mounted"
        if [ $? -ne 0 ]; then
            ${HA_LOG} "NFS: $fs not mounted";
            ha_write_status_for_resource ${resource} ${HA_CMD_FAILED};
#           exit_script $HA_CMD_FAILED;
```

メモ: SGI では、NFS エクスポートされた CXFS ファイルシステムのロックのフェイルオーバーは現在サポートしていません。ロックを使用する場合は、NFS ではなく NFS_CXFS の依存性を持つ新しい statd_unlimited リソース・タイプを作成しなければなりません。元の statd リソース・タイプは、エクスポートされた CXFS ファイルシステムに対しては使用できません。

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、IRIS FailSafe クラスタ・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) と IRIS FailSafe クラスタ・マネージャ・コマンド・ライン・インタフェース (CLI: Command Line Interface) 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。

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。

システム運用のデフォルトの設定

実行中のシステムで実行するいくつかのコマンドでは、ノードまたはクラスタを指定するオプションを使用できます。ノードまたはクラスタを指定すると、ノードやクラスタを明示的に指定しなかった場合にデフォルトとして使用できます。

クラスタ・マネージャ GUI を使ったデフォルトのクラスタの設定

デフォルトのクラスタの名前を指定していない場合は、名前を入力するようクラスタ・マネージャ GUI によってプロンプトが表示されます。または、「FailSafe マネージャ (FailSafe Manager)」ウィンドウの下部にある「クラスタの選択...(Select Cluster...)」をクリックして、デフォルトのクラスタを設定できます。

クラスタ・マネージャ GUI を使用する場合、デフォルトのノードを設定する必要はありません。

クラスタ・マネージャ CLI を使ったデフォルトの設定

クラスタ・マネージャ CLI を使用している場合、デフォルトの値を指定するために以下のコマンドを使用できます。デフォルトのクラスタを指定するには、以下のコマンドのいずれかを使用します。

```
cmgr> set cluster A
cmgr> set node A
```

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。

IRIS FailSafe のアクティブ化（開始）

IRIS 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、FailSafe を開始して高可用性サービスをアクティブにすることができます。FailSafe は、システム全体ベース、クラスタ内のすべてのノード、または指定したノードでのみ開始できます。

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、クラスタ内のほかのノードでリソース・グループが実行されていない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。

1. CDB のクラスタ内のすべてのノードが有効になります。

2. CDB 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。
3. ローカル CMOND が fs2d/CDBD から通知を受信します。
4. 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。
5. CMOND によって `failsafe2 chkconfig` フラグが `on` に設定されます。

クラスタ マネージャ GUI を使った IRIS FailSafe のアクティブ化

クラスタ・マネージャ GUI を使用して FailSafe サービスを開始するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「ノードとクラスタ(Nodes & Cluster)」カテゴリをクリックします。
3. 画面の右側で「FailSafe HA サービスの開始(Start FailSafe HA Services)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックして、タスクを完了します。

クラスタ マネージャ CLI を使った IRIS FailSafe のアクティブ化

クラスタで IRIS FailSafe をアクティブにするには、次のコマンドを使用します。

```
cmgr> start ha_services [on node A] [for cluster B]
```

システムのステータス

IRIS FailSafe システムの実行中に、IRIS 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、またはクラスタ・マネージャ GUI の FailSafe クラスタ表示を使用して、クラスタの状態を継続的に監視できます。
- クラスタ・マネージャ GUI またはクラスタ・マネージャ CLI 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。
- クラスタ・マネージャ CLI 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、これらの各タスクを実行するための手順について説明します。

cluster_status コマンド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、NFS リソース・グループと cluster_status ヘルプ・テキストのみが表示された、FailSafe 用に設定された 2 ノード・クラスタを以下に示します。

```
# /var/cluster/cmgr-scripts/cluster_status
+ Cluster=nfs-cluster  FailSafe=ACTIVE CXFS=Not Configured              10:57:57
   Nodes =   hans2    hans1
FailSafe =      UP       UP
    CXFS =

ID ]             ResourceGroup           Owner           State            Error
 0 ]                nfs-group1           hans1          Online         No error

                      +--------+ cluster_status Help +---------+
                      |  on s  - Toggle Sound on event         |
                      |  on r  - Toggle Resource Group View    |
                      |  on c  - Toggle CXFS View              |
                      |     h  - Toggle help screen            |
                      |     i  - View Resource Group detail    |
                      |     q  - Quit cluster_status           |
                      +--- Press 'h' to remove help window --+

--------------------------------------------------------------------------------
cmd('h' for help) >
```

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。

クラスタ・マネージャ 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、クラスタ・マネージャ GUI の FailSafe クラスタ表示を使用することです。FailSafe クラスタ表示は、FailSafe Toolchest から直接起動できます。

「FailSafe クラスタ表示(FailSafe Cluster View)」ウィンドウでは、システム・コンポーネントで発生している問題が、点滅する赤いアイコンとして表示されます。状態が移行しているコンポーネントも、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「FailSafe ク

ラスタ表示(FailSafe Cluster View)」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なって点滅します。

FailSafe クラスタ表示におけるコンポーネントの状態のすべてのカラーの凡例は、以下のとおりです。

グレー	正常だが、オンラインまたはアクティブになっていない
緑	正常で、アクティブまたはオンラインになっている
点滅する緑	緑に移行中
点滅する赤	コンポーネントに関する問題がある
黒と白のアウトライン	リソース・タイプ
グレーに黄色のレンチ	メインテナンス・モードで、FailSafe によって現在モニタされている場合とモニタされていない場合がある

「FailSafe クラスタ表示(FailSafe Cluster View)」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。クラスタにアクティブな FailSafe HA サービスがあり、エラーがない場合は、アイコンに緑のクラスタが表示されます。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。アクティブではない FailSafe HA サービスがクラスタにある場合は、アイコンにグレーのクラスタが表示されます。

クラスタ・マネージャ CLI 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・コントローラに対して ping を実行するには、以下の項で説明されているとおりに、CLI を使用できます。

クラスタ・マネージャ CLI 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 CLI コマンドを使用します。

```
cmgr> show status of resource A of resource_type B [in cluster C]
```

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。

クラスタ・マネージャ CLI 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 操作を実行するには、次の CLI コマンドを使用します。

```
cmgr> admin ping dev_name A of dev_type B with sysctrl_type C
```

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。

「オンライン(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、IRIS 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。
「オンライン・ペンディング(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。
「オフライン(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。
「オフライン・ペンディング(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。
「オンライン準備済み(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。

「オンライン・メインテナンス (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、IRIS 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 IRIS 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。
「内部エラー (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。
「発見(排他)(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・グループのアプリケーション異常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。
「初期化中 (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。

「エラーなし(NO ERROR)」	リソース・グループにエラーはありません。
「内部エラー - 回復できません(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。
「ノード不明 (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 実行可能エラー (SRMD EXECUTABLE ERROR)」	リソース・グループのリソースに対して、開始アクションまたは停止アクションが失敗しています。
「分割リソース・グループ(排他)(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。

「ノード使用不可(排他)(NODE NOT AVAILABLE (EXCLUSIVITY))」	リソース・グループのアプリケーション異常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。アプリケーション異常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。
「モニタ・アクティビティ不明 (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。
「使用可能なノードなし(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。

クラスタ・マネージャ 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、FailSafe クラスタ表示を使用できます。FailSafe クラスタ表示は、直接起動したり、「FailSafe マネージャ(FailSafe Manager)」画面の下部にある「FailSafe クラスタ表示(FailSafe Cluster View)」をクリックして、いつでも起動できます。

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)」メニュー ->「グループ内のリソース(Resources in Groups)」を選択し、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、「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。

クラスタ・マネージャ CLI 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 CLI コマンドを使用します。

```
cmgr> show status of resource_group A [in cluster B]
```

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。

「実行中(UP)」	このノードは、FailSafe メンバーシップの一部です。

「停止(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。
「不明(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)。
「非アクティブ(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 コマンド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。

クラスタ・マネージャ 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、FailSafe クラスタ表示を使用できます。FailSafe クラスタ表示は、直接起動したり、「FailSafe マネージャ(FailSafe Manager)」画面の下部にある「FailSafe クラスタ表示(FailSafe Cluster View)」をクリックして、いつでも起動できます。

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)」メニュー ->「ノードが所有するグループ(Groups owned by Nodes)」を選択します。

クラスタ・マネージャ CLI を使ったノード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 CLI コマンドを使用します。

```
cmgr> show status of node A
```

クラスタ・マネージャ CLI 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、次のクラスタ・マネージャ CLI 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。

```
cmgr> admin ping node A
```

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、CLI の 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。

```
cmgr> admin ping standalone node A
```

このコマンドは FailSafe デーモンを経由しませんが、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。

クラスタのステータス

クラスタのステータスを照会するには、そのクラスタの名前を指定します。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)」	有効な FailSafe メンバーシップがあり、このノードはクラスタの一部です。
「非アクティブ(INACTIVE)」	FailSafe メンバーシップはありません。このクラスタでは HA サービスは開始されていません。
「不明(UNKNOWN)」	このクラスタでは HA サービスが開始されています。このノードは、メンバーシップの一部ではありません。有効な FailSafe メンバーシップがある可能性があります。

クラスタで HA サービスを開始すると、クラスタの状態は「非アクティブ(INACTIVE)」から「アクティブ(ACTIVE)」に移行します。

クラスタ・マネージャ GUI を使ったクラスタのステータスの照会

FailSafe システムでクラスタのステータスを照会するには、クラスタ・マネージャ GUI のクラスタ表示を使用できます。

クラスタ・マネージャ CLI 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 CLI コマンドを使用します。

```
cmgr> show status of cluster A
```

haStatus CLI スクリプトを使用し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
        monitoring-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
        monitoring-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)
```

リソース・グループのフェイルオーバー

IRIS FailSafe システムの実行中に、オンラインのリソース・グループを特定のノードに移動したり、リソース・グループをオフにできます。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。

リソース・グループをオンラインにするには、リソースの名前と、ノードが含まれているクラスタの名前を指定します。

リソース・グループにメンバーがいない場合は、そのリソース・グループをオンラインにすることはできません。また、クラスタ内で現在そのリソース・グループが実行されている場合も、オンラインにできません。

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 ONLINE-READY になります。HA サービスが開始されると、FailSafe は、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。

180 ページの「リソース・グループのオフライン化」で説明されているように、FailSafe GUI または FailSafe CLI 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、180 ページの「リソース・グループのオフライン化」を参照してください。

クラスタ・マネージャ GUI を使ったリソース・グループのオンライン化

クラスタ・マネージャ GUI を使用してリソース・グループをオンラインにするには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「フェイルオーバー・ポリシーとリソース・グループ(Failover Policies & Resource Groups)」カテゴリをクリックします。
3. 画面の右側で「リソース・グループをオンラインにする(Bring a Resource Group Online)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。

5. 画面の下部にある「OK」をクリックして、タスクを完了します。

クラスタ・マネージャ CLI 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 CLI コマンドを使用します。

```
cmgr> admin online resource_group A [in cluster B]
```

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。

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 3 つ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）、183 ページの「リソース・グループのモニタリングの停止（メインテナンス・モード）」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 CLI 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、このコマンドを実行することにより、設定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。

注意: リソース・グループを分離すると、リソース・グループ内のリソースは、そのリソース・グループがオンラインになってい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。データが壊れる可能性があります。

クラスタ・マネージャ GUI を使ったリソース・グループのオフライン化

クラスタ・マネージャ GUI を使用してリソース・グループをオフラインにするには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「フェイルオーバー・ポリシーとリソース・グループ(Failover Policies & Resource Groups)」カテゴリをクリックします。
3. 画面の右側で「リソース・グループをオフラインにする(Take a Resource Group Offline)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックして、タスクを完了します。

クラスタ・マネージャ CLI 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 CLI コマンドを使用します。

```
cmgr> admin offline resource_group A [in cluster B]
```

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、CLI によってデフォルトが使用されます。

force オプションを使用して、実際にリソース・グループをオフラインにするには、次の CLI コマンドを使用します。

```
cmgr> admin offline_force resource_group A [in cluster B]
```

リソース・グループを分離するには、次の CLI コマンドを使用します。

```
cmgr> admin offline_detach resource_group A [in cluster B]
```

リソース・グループの移動

IRIS 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。リソース・グループを移動するときは、以下を指定します。

- リソース・グループの名前
- 移動先のノードの論理名（オプション）。移動先のノードの論理名を指定しないと、FailSafe によってフェイルオーバー・ポリシーに基づいて移動先が選択されます。
- ノードが含まれているクラスタの名前

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。

クラスタ・マネージャ GUI を使ったリソース・グループの移動

クラスタ・マネージャ GUI を使用してリソース・グループを移動するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「フェイルオーバー・ポリシーとリソース・グループ(Failover Policies & Resource Groups)」カテゴリをクリックします。
3. 画面の右側で「リソース・グループの移動(Move a Resource Group)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックして、タスクを完了します。

クラスタ・マネージャ CLI 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 CLI コマンドを使用します。

```
cmgr> admin move resource_group A [in cluster B] [to node C]
```

リソース・グループのモニタリングの停止（メインテナンス・モード）

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 IRIS 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 IRIS 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」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」状態として表示されます。

クラスタ・マネージャ GUI 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

クラスタ・マネージャ GUI を使用してリソース・グループをメインテナンス・モードにするには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「フェイルオーバー・ポリシーとリソース・グループ(Failover Policies & Resource Groups)」カテゴリをクリックします。
3. 画面の右側で「リソース・グループのモニタの中断(Suspend Monitoring a Resource Group)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。

クラスタ・マネージャ GUI を使ったリソース・グループのモニタの再開

クラスタ・マネージャ GUI を使用してリソース・グループのモニタを再開するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「フェイルオーバー・ポリシーとリソース・グループ(Failover Policies & Resource Groups)」カテゴリをクリックします。

3. 画面の右側で「リソース・グループのモニタの再開(Resume Monitoring a Resource Group)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。

クラスタ・マネージャ CLI 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 CLI コマンドを使用します。

```
cmgr> admin maintenance_on resource_group A [in cluster B]
```

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。

クラスタ・マネージャ CLI 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 CLI コマンドを使用します。

```
cmgr> admin maintenance_off resource_group A [in cluster B]
```

IRIS FailSafe の停止(非アクティブ化)

IRIS FailSafe の実行は、システム全体ベース、クラスタ内のすべてのノード、または指定したノードでのみ停止できます。

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を目的の状態のままにしておくことができます。

リソース・グループが安定した正常な状態でない場合、ノードまたはクライアントの 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した後は、ノードまたはクラスタのリソース・グループはなくなり、HA サービスもすべてなくなります。

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。

1. シャットダウン要求が FailSafe (FSD) に送信されます。

2. 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。
3. 設定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）。
4. 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。
5. 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。
6. ノードが無効になると、CMOND は設定データベースから通知を受信します。
7. 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。
8. 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。
9. 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。

ノードでの HA サービスの停止

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・グループを高可用性の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。

クラスタ内での HA サービスの停止

クラスタの停止操作では、すべてのリソース・グループを解放してからクラスタ内のすべて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。

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。

クラスタ マネージャ GUI を使った FailSafe の停止

クラスタ・マネージャ GUI を使用して FailSafe サービスを停止するには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「ノードとクラスタ(Nodes & Cluster)」カテゴリをクリックします。
3. 画面の右側で「FailSafe HA サービスの停止(Stop FailSafe HA Services)」タスク・リンクをクリックし、タスクを起動します。
4. 選択した入力値を入力します。
5. 画面の下部にある「OK」をクリックして、タスクを完了します。

クラスタ マネージャ CLI を使った FailSafe の停止

クラスタで IRIS FailSafe を停止し、FailSafe での処理を中止するには、次のコマンドを使用します。

```
cmgr> stop ha_services [on node A] [for cluster B][force]
```

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。

クラスタ・マネージャ GUI を使ったノードのリセット

クラスタ・マネージャ GUI を使用して FailSafe ノードをリセットするには、以下の手順を実行します。

1. FailSafe Toolchest で「FailSafe マネージャ(FailSafe Manager)」を選択します。
2. 画面の左側で、「ノードとクラスタ(Nodes & Cluster)」カテゴリをクリックします。
3. 画面の右側で「ノードのリセット(Reset a Node)」タスク・リンクをクリックし、タスクを起動します。
4. リセットするノードを入力します。
5. 画面の下部にある「OK」をクリックして、タスクを完了します。

クラスタ・マネージャ CLI を使ったノードのリセット

FailSafe が実行されているときに、次のクラスタ・マネージャ CLI コマンドを使用して、ノードを再起動できます。

```
cmgr> admin reset node A
```

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、CLI の `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。

```
cmgr> admin reset standalone node A
```

このコマンドは、FailSafe デーモンを経由しません。

クラスタ・マネージャ CLI を使った設定のバックアップと復元

クラスタ・マネージャ CLI 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 `cdbDump` 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、`/var/cluster/cmgr-scripts` 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 `cdbDump` 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。

`cdbDump` 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。

- cdbDump 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、CDB の情報の変更中も cdbDump は実行しないでください。CDB 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 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、144 ページの「FailSafe システム・ログ設定」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、次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 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。

IRIS FailSafe 設定のテスト

この章では、クラスタ・マネージャ GUI およびクラスタ・マネージャ CLI 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。クラスタ・マネージャ GUI およびクラスタ・マネージャ CLI の使用についての一般情報は、第4章「IRIS FailSafe 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

表8-1に、IRIS 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	チェックされる実行内容
リソース	リソース・タイプ・パラメータが設定されていることをチェックします。
	パラメータが構文的に正しいことをチェックします。
	パラメータが存在することを検証します。
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。
フェイルオーバー・ポリシー	フェイルオーバー・ポリシーが存在することをチェックします。
	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ことをチェックします。

診断テスト	チェックされる実行内容
ネットワークの接続性	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ことをチェックします。 ノードが互いに通信できることをチェックします。
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。

クラスタ・マネージャ GUI を使った診断タスクの実行

クラスタ・マネージャ GUI を使用して FailSafe システムのコンポーネントをテストするには、以下の手順を実行します。

1. FailSafe Toolchest で「タスク・マネージャ(Task Manager)」を選択します。
2. 画面の左側で、「診断(Diagnostics)」カテゴリをクリックします。
3. 画面の右側に表示される診断タスク(「接続性のテスト(Test Connectivity)」、「リソースのテスト(Test Resources)」、または「フェイルオーバー・ポリシーのテスト(Test Failover Policy)」)の中から、1 つ選択します。

クラスタ・マネージャ GUI を使った接続性のテスト

「診断(Diagnostics)」画面 ->「接続性のテスト(Test Connectivity)」タスクを選択すると、要求された入力値を入力することによって、ネットワークおよびクラスタ内のノードのシリアル接続をテストできます。クラスタ内のノードは、すべて一度にテストしたり、個々に指定してテストできます。

クラスタ・マネージャ GUI を使ったリソースのテスト

「診断(Diagnostics)」画面 ->「リソースのテスト(Test Resources)」タスクを選択すると、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。

クラスタ・マネージャ GUI を使ったフェイルオーバー・ポリシーのテスト

「診断(Diagnostics)」画面 ->「フェイルオーバー・ポリシーのテスト(Test Failover Policy)」タスクを選択すると、フェイルオーバー・ポリシーが正しく定義されている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、フェイルオーバー属性、およびクラスタ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。

クラスタ・マネージャ CLI を使った診断タスクの実行

以下の項では、クラスタ・マネージャ CLI 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。

クラスタ・マネージャ CLI を使ったシリアル接続のテスト

IRIS FailSafe ノード間のシリアル接続をテストするには、クラスタ・マネージャ CLI 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 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。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・マネージャ CLI を使用している場合は、次のコマンドを使用して、クラスタ内のマシンのシリアル接続をテストします。

```
cmgr> test serial in cluster A [on node B node C ...]
```

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、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。

以下に、test serial CLI コマンドの例を示します。

```
# cluster_mgr
Welcome to IRIS FailSafe Cluster Manager Command-Line Interface

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` CLI 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>
```

クラスタ・マネージャ CLI 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、クラスタ・マネージャ CLI を使用できます。このテストでは、指定したノードが、ノード内の設定された各インタフェースを通し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・マネージャ CLI を使用している場合は、次のコマンドを使用して、クラスタ内のマシンのネットワークの接続性をテストします。

```
cmgr> test connectivity in cluster A [on node B node C ...]
```

以下に、`test connectivity` CLI 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。

クラスタ・マネージャ CLI 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、クラスタ・マネージャ CLI を使用できます。

クラスタ・マネージャ CLI では、次の構文を使用して、名前別にリソースをテストします。

```
cmgr> test resource A of resource_type B in cluster C [on node D node E ...]
```

以下に、名前別のリソースのテストの例を示します。

```
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
```

クラスタ・マネージャ CLI では、次の構文を使用して、リソース・タイプ別にリソースをテストします。

```
cmgr> test resource_type A in cluster B [on node C node D...]
```

以下に、リソース・タイプ別のリソースのテストの例を示します。

```
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、CLI を使用できます。CLI を使用すると、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、CLI テストは destructive モードで実行されません。

クラスタ・マネージャ CLI では、destructive モードでリソースをテストするコマンドに対して次の構文を使用します。

```
cmgr> test resource A of resource_type B in cluster C [on node D node C ...] 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。

論理ボリュームのテスト

クラスタの論理ボリュームをテストするには、クラスタ・マネージャ CLI 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。

クラスタ・マネージャ CLI を使用している場合は、次のコマンドを使用して、論理ボリュームをテストします。

```
cmgr> test resource A of resource_type volume on cluster B [on node C node D ...]
```

以下の例では、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
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
cmgr>
```

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>
```

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、クラスタ・マネージャ CLI 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に加え、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もチェックします。

クラスタ・マネージャ CLI を使用している場合は、次のコマンドを使用して、ファイルシステムをテストします。

```
cmgr> test resource A of resource_type filesystems on cluster B [on node C node D ...]
```

以下の例では、ファイルシステムをテストします。この例では、まず CLI show コマンドを使用して、クラスタで定義されているファイルシステムを表示します。

```
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
cmgr>
```

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>
```

NFS ファイルシステムのテスト

クラスタで設定されている NFS ファイルシステムをテストするには、クラスタ・マネージャ CLI を使用できます。このテストでは、指定した NFS ファイルシステムが正しく設定されているかどうかに加え、NFS 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もチェックします。

クラスタ・マネージャ CLI を使用している場合は、次のコマンドを使用して、NFS ファイルシステムをテストします。

```
cmgr> test resource A of resource_type NFS on cluster B [on node C node D ...]
```

以下の例では、NFS ファイルシステムをテストします。

```
cmgr> test resource /disk4 of resource_type NFS in cluster eagan
Success: *** testing node resources on node cm1 ***
Success: *** testing all NFS resources on node cm1 ***
Success: *** testing node resources on node cm2 ***
Success: *** testing all NFS resources on node cm2 ***
Success: overall exit status:success, tests failed:0, total tests executed:2
cmgr>
```

statd リソースのテスト

クラスタで設定されている statd リソースをテストするには、クラスタ・マネージャ CLI を使用できます。クラスタ・マネージャ CLI を使用している場合は、次のコマンドを使用して、statd リソースをテストします。

```
cmgr> test resource A of resource_type statd on cluster B [on node C node D ...]
```

以下の例では、statd リソースをテストします。

```
cmgr> test resource /disk1/statmon of resource_type statd in cluster eagan
Success: *** testing node resources on node cm1 ***
Success: *** testing all statd resources on node cm1 ***
Success: *** testing node resources on node cm2 ***
Success: *** testing all statd resources on node cm2 ***
Success: overall exit status:success, tests failed:0, total tests executed:2
cmgr>
```

Netscape Web リソースのテスト

クラスタで設定されている Netscape Web リソースをテストするには、クラスタ・マネージャ CLI を使用できます。

クラスタ・マネージャ CLI を使用している場合は、次のコマンドを使用して、Netscape Web リソースをテストします。

```
cmgr> test resource A of resource_type Netscape_web on cluster B [on node C  node D ...]
```

以下の例では、Netscape Web リソースをテストします。この例では、ノード cm2 の Netscape Web リソースでの診断テストに失敗しています。

```
cmgr> test resource nss-enterprise of resource_type Netscape_web in cluster eagan
```

```
Success: *** testing node resources on node cm1 ***
Success: *** testing all Netscape_web resources on node cm1 ***
Success: *** testing node resources on node cm2 ***
Success: *** testing all Netscape_web resources on node cm2 ***
Warning: resource nss-enterprise has invalid script /var/netscape/suitespot/https-ha85 location
Warning: /var/netscape/suitespot/https-ha85/config/magnus.conf must contain the
"Port" parameter
Warning: /var/netscape/suitespot/https-ha85/config/magnus.conf must contain the
"Address" parameter
Warning: resource nss-enterprise of type Netscape_web failed
Success: overall exit status:failed, tests failed:1, total tests executed:2
Failed to execute FailSafe tests/diagnostics ha
test command failed
cmgr>
```

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、クラスタ・マネージャ CLI 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・テストを繰り返し実行します。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。

クラスタ・マネージャ CLI では、リソース・グループをテストするコマンドに対して次の構文を使用します。

cmgr> **test resource_group** *A* **in cluster** *B* [**on node** *C* **node** *D* ...]

以下の例では、リソース・グループをテストします。この例では、まず CLI show コマンドを使用して、クラスタで定義されているリソース・グループを表示します。

```
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>
```

クラスタ・マネージャ CLI 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、クラスタ・マネージャ CLI 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。

クラスタ・マネージャ CLI では、フェイルオーバー・ポリシーをテストするコマンドに対して次の構文を使用します。

```
cmgr> test failover_policy A in cluster B [on node C node D ...]
```

以下の例では、フェイルオーバー・ポリシーをテストします。この例では、まず CLI 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
cmgr>
```

IRIS FailSafe の回復

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。

FailSafe 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 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 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。

FailSafe ログ・ファイル

IRIS 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。

CAD、CMOND、および CDBD/fs2d 用のログの設定についての詳細は、51 ページの「システム・ファイルの設定」を参照してください。ログ設定の設定についての詳細は、第5章「IRIS FailSafe の設定」の144 ページの「FailSafe システム・ログ設定」を参照してください。

ログ・メッセージには、以下のタイプを設定できます。

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。

```
Wed Sep 2 11:57:25.284
<N ha_gcd cms 10185:0>
Delivering TOTAL membership (S# 1, GS# 1)
```

<N の表記が通常メッセージを示します。

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。

```
Wed Sep 2 13:45:47.199
<W crsd crs 9908:0 crs_config.c:634>
CI_ERR_NOTFOUND, safer - no such node
```

<W の表記が警告を表し、<E がエラーを示します。

システム・ログ・メッセージ

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・レベルがデバッグ0 以上（GUI 使用時）または 10 以上（CLI 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。

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。

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に注意すると、異常の原因となった状況を判断できることがあります。

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。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ないので注意してください。

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、ノード ID 番号が最も小さい有効なノードを使用します。

タイブレーカー・ノードの設定についての詳細は、第5章「IRIS FailSafe の設定」の 98 ページの「IRIS FailSafe HA パラメータ」を参照してください。

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」と宣言され、リセットできないノードは「不明」と宣言されます。上限量に含まれるノードは「実行中」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。

- FailSafe メンバーシップ・デーモン `ha_cmsd` が実行されているかどうか。データベース・デーモン `fs2d` 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・ノードからコントロール・ネットワーク・アドレスに ping を実行できるかどうか
- 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
  - ノード・コントロール・デーモンである `crsd`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、システムに問題がある箇所を特定できます。

FailSafe クラスタ・マネージャ GUI のクラスタ表示では、そ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。FailSafe クラスタ・マネージャ CLI 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 165 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、第5章「IRIS FailSafe の設定」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。

- 207 ページの「クラスタ・エラーの回復」
- 207 ページの「ノード・エラーの回復」
- 208 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」
- 211 ページの「リソース・エラーの回復」
- 212 ページの「コントロール・ネットワークの異常の回復」
- 212 ページの「シリアル・ケーブルの異常の回復」
- 212 ページの「CDB sync 異常」
- 213 ページの「CDB のメインテナンスと回復」

- 213 ページの「IRIS FailSafe クラスタ・マネージャ GUI と CLI の不整合」
- 213 ページの「GUI で情報が報告されない」
- 214 ページの「cdbreinit コマンドの使用」

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」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（212 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. アクティブなクラスタが存在するには、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（上限量の条件）。コントロール・ネットワークを使用して相互に通信できる十分な数のノードがクラスタにない場合は、一部のノードで HA サービスを停止して上限量の条件を満たします。
3. ハードウェア設定に問題がない場合は、クラスタでオンラインになっているすべてのリソース・グループを分離します（リソース・グループがある場合）。続いて、クラスタの HA サービスを停止して再開します。

以下の `cluster_mgr` 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 サービスを分離して、エラーを無視する（`force` オプション）場合は、以下の `cluster_mgr` 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、以下の `cluster_mgr` コマンドを使用します。

```
cmgr> start ha_services for cluster web-cluster
```

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（212 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（212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。

   クラスタ web-cluster のノード web-node3 の HA サービスを分離して、エラーを無視する（force オプション）には、以下の cluster_mgr コマンドを使用します。

   ```
   cmgr> stop ha_services in node web-node3 for cluster web-cluster
   force
   ```

   クラスタ web-node3 のノード web-node3 のHA サービスを停止するには、以下の cluster_mgr 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。

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、以下の cluster_mgr 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・メインテナンス」に変わります。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、以下の cluster_mgr 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」に戻ります。

リソース・グループの状態が「オンライン」で「SRMD 実行可能フェイル・エラー」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。

   以下の cluster_mgr 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
   ```

   以下の cluster_mgr 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」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。

異常ドメイン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）」、「ノード使用不可（排他）」、「使用可能なノードなし」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、第7章「IRIS FailSafe のシステム運用」の 168 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）」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:name]:Exclusivity failed -- RUNNING on nodename and nodename
```

```
[Resource Group:name]:Exclusivity failed -- PARTIALLY RUNNING on nodename and
PARTIALLY RUNNING on nodename
```

   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」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. AFD 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. AFD 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。

以下に、このような状況が発生した場合の正しいアクションを示します。

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、180 ページの「リソース・グループのオフライン化」を参照してください。

リソース・エラーの回復

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」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。

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。
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. 問題を解決したら、クラスタ・マネージャ CLI の `admin offline` コマンドの `force` オプションを使用して、リソースをオフラインにします。

   ```
   cmgr> admin offline_force resource web-srvr of resource_type
   Netscape_Web in cluster web-cluster
   ```

   このコマンドを実行すると、`Netscape_Web` タイプのリソース `web-srvr` のエラー状態がクリアされ、リソース・グループに追加できるようになります。

   また、クラスタ・マネージャ GUI を使用してリソースのエラー状態をクリアすることもできます。この操作を行うには、FailSafe マネージャの「リソースとリソース・タイプ(Resources and Resource Types)」カテゴリから「リソース・エラー状態のクリア(Clear Resource Error State)」タスクを選択します。

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。

   以下の 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。

1. 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。

   以下の 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
   ```

   cluster_mgr 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。

   ```
   cmgr> admin ping node web-node3
   ```

上記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。

CDB sync 異常

CDB sync に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 SYSLOG で以下のメッセージをチェックします。

   ```
   Starting to receive CDB sync series from machine <node1's node id>
   ...
   Finished receiving CDB sync series from machine <node1's 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. CDB 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 CDBD/fs2d ログをチェックします。

CDB のメインテナンスと回復

設定データベース (CDB) 全体を初期化する必要があるときは、以下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・プロセスが再開されます。プール内にほかのノードがある場合、設定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 CDB をバックアップから復元する必要があります。CDB のバックアップと復元についての詳細は、第7章「IRIS FailSafe のシステム運用」の 187 ページの「クラスタ・マネージャ CLI 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。

IRIS FailSafe クラスタ・マネージャ GUI と CLI の不整合

FailSafe クラスタ・マネージャ GUI で FailSafe `cluster_mgr` コマンドと異なる情報が表示される場合は、以下のコマンドを実行して、クラスタ・マネージャ 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

FailSafe クラスタ・マネージャ GUI で設定情報とステータスが報告されない場合は、以下のチェックを実行します。

1. クラスタ・マネージャ CLI 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 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. GUI の更新に問題がない場合は、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設定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 cluster_mgr 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. CDB 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. CDB が同期されるまで数分間待ちます。CDB 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。

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、第8章「IRIS FailSafe 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. cluster_mgr テンプレート (/var/cluster/cmgr-templates/cmgr-create-node) または cluster_mgr コマンドを使用して、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. cluster_mgr コマンドを使用して、ノードをクラスタに追加します。

   たとえば、以下の cluster_mgr 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. cluster_mgr コマンドを使用して、このノードで HA サービスを開始できます。たとえば、以下の cluster_mgr 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. このノードは、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するようにしてください。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、cluster_mgr コマンドを使用して、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、cluster_mgr コマンドを使用して、リソース・グループを分離します。たとえば、以下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. 以下の cluster_mgr コマンドを使用して、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。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。

設定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。ノード web-node3 を web-cluster 設定から削除するには、以下の cluster_mgr 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. 設定データベースからノード設定を削除します。

   以下の cluster_mgr コマンドを実行すると、設定データベースから 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、設定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、設定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. 以下のように、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 が強制終了したことを確認します。

3. node1 から、node1 および node2 の定義を変更します。変更するには、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 nnew 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 CDB 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 が強制終了したことを確認します。

7. node2 から、以下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. 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. このノード上のリソース・グループがオンラインの場合は、cluster_mgr コマンドを使用して、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。

   以下の cluster_mgr コマンドを実行して、ノード 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. 以下の cluster_mgr コマンドを使用して、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。以下の cluster_mgr 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. このノード上のリソース・グループがオンラインの場合は、cluster_mgr コマンドを使用して、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。

以下の cluster_mgr コマンドを実行して、ノード 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. 以下の cluster_mgr コマンドを使用して、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、以下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。以下の cluster_mgr 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（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 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。フェイルオーバー属性は、cluster_mgr テンプレート (/var/cluster/cmgr-templates/cmgr-create-failover_policy) を使用して作成できます。

4. 設定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 cluster_mgr 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 リソースを設定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、第8章「IRIS FailSafe 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、cluster_mgr テンプレート (/var/cluster/cmgr-templates/cmgr-create-resource_group) を使用できます。

   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ハードウェア・デバイスを追加するときは、一度に 1 つのノードに追加していきます。

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、220 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内の 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。

設定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、66 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、`hbvis`(1) および `rmvis`(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。

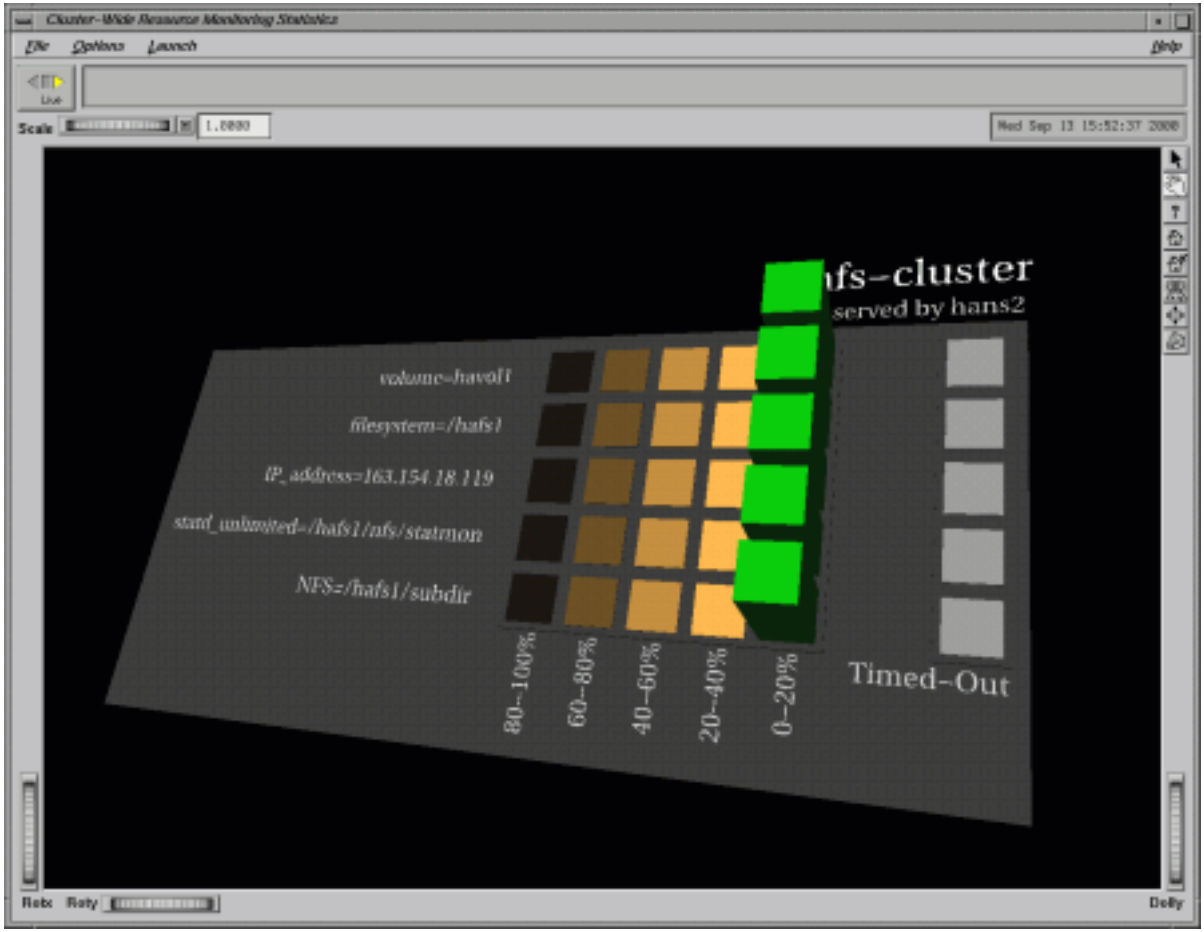

図11-2 リソース・モニタ統計

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。

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) をすると、常に、選択したサンプリング時間で使用できる最新の統計が表示されます。*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) 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、249 ページの付録C「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、以下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。）

トラブルシューティング

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、以下のコマンドを入力して、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、以下のコマンドを入力して 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。

IRIS FailSafe 1.2 から IRIS FailSafe 2.X へのアップグレード

IRIS FailSafe 2.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 IRIS FailSafe 1.2 システムを IRIS FailSafe 2.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.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。

この章では、システムを IRIS FailSafe 1.2 から IRIS FailSafe 2.X にアップグレードする手順を説明します。この章は、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.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.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.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、第8章「IRIS FailSafe 設定のテスト」を参照してください。これらの診断は、FailSafe 2.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` CLI コマンドを使用して、FailSafe 2.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.X CLI のコマンドを示します。

```
cmgr> admin ping dev_name /dev/ttyd2 of dev_typetty with sysctrl_type msc
```

CLI コマンドの使用についての詳細は、第4章「IRIS FailSafe 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.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.X 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.X では、設定情報は `/var/cluster/cdb/cdb.db` にある設定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。設定データベースは、クラスタ・マネージャ・コマンド・ライン・インタフェース (CLI: Command Line Interface) またはクラスタ・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) を使用して作成します。

FailSafe 2.X の設定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.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.X システムとして設定するには、FailSafe 2.X クラスタ・マネージャ GUI または FailSafe クラスタ・マネージャ CLI 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、第4章「IRIS FailSafe 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.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.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、FailSafe クラスタ・マネージャ GUI の FailSafe マネージャの「設定ガイド」ページのタスク・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.X 設定を比較した設定例については、235 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.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.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.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.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.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。

234 ページの表A-1 に、これらのリリースの違いの概要を示します。

表A-1 IRIS FailSafe 1.2 と 2.X の相違点

IRIS FailSafe 1.2	IRIS FailSafe 2.X
ha.conf 設定ファイル	/var/cluster/cdb/cdb/db にある設定データベース。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 に含まれるデータの大部分は 2.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 cluster_mgr 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 giveaway, giveback takeover, takeback check (該当なし)	スクリプト stop start monitor 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
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。

IRIS FailSafe 1.2	IRIS FailSafe 2.X
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.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.X 設定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。

以下の節では、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.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.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:	node 1
プライマリ・ネットワーク・インタフェース:	node 1
システム・コントローラの種類:	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 FaiSafe 2.X でノードを定義するには、以下の `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.Xでのモニタ値の設定についての詳細は、238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.Xがクラスタ定義で使用するha.confファイルでパラメータを指定します。つまり、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.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.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、以下の 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、以下の 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.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.X では、クラスタ内のノードの監視に 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持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持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.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、以下の数式を使用して、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。

ノードのタイムアウト = (検査時間 + タイムアウト) * 切断カウント

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.X のタイムアウトはすべてミリ秒単位で、FailSafe 2.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 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、クラスタ・マネージャ GUI またはクラスタ・マネージャ CLI を使用して変更できます。

FailSafe 2.X で node1 の HA パラメータを変更するには、以下の 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.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.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.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.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。

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.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.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.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、第5章「IRIS FailSafe の設定」の139 ページの「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、第7章「IRIS FailSafe のシステム運用」の164 ページの「IRIS FailSafe のアクティブ化（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.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。

- クラスタ・マネージャ GUI のクラスタ表示を使用して、クラスタの状態を継続的に監視できます。
- クラスタ・マネージャ GUI またはクラスタ・マネージャ CLI 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。
- クラスタ・マネージャ CLI 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。

これらのタスクを実行する場合についての詳細は、第7章「IRIS FailSafe のシステム運用」の165 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 ソフトウェア

この付録では、IRIS FailSafe 2.1 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。

メモ: ソフトウェアのインストール手順は、48 ページの「必要な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。

この付録は以下の節で構成されます。

IRIS FailSafe 2.1 CD に含まれるサブシステム

IRIS FailSafe 2.1 base CD のインストールには、約 10 MB が必要です。

244 ページの表B-1 に、IRIS FailSafe 2.1 CD に含まれる IRIS FailSafe 2.1 サブシステムを示します。

表B-1 IRIS FailSafe 2.1 CD

用途	システム
IRIS FailSafe 2.1	*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

メモ: base system administration (*sysadm_base*)、cluster administration (*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。

IRIS FailSafe 2.1 プール内のサーバおよびワークステーションにインストールするサブシステム

245 ページの表B-2 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B-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*
IRIS FailSafe 2.1 administration server	*sysadm_failsafe2.sw.server*	*sysadm_base.sw.server*
		cluster_admin.sw.base
		cluster_services.sw.cli
		cluster_control.sw.cli
		failsafe2.sw.cli
Cluster administration	*cluster_admin.sw*	*sysadm_base.sw.dso*
	cluster_control.sw	
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 3.1.1
		Web サーバ
EL-8+ multiplexer driver	*el_serial*	
（multiplexer に付属の CD に収録）	*el_serial.man*	
	el_serial.sw	

IRIS FailSafe 2.1 クラスタ内のノード用のその他のサブシステム

246 ページの表B-3 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B-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 software	*failsafe2.sw*	*cluster_services.sw*

管理ワークステーションにインストールする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、247 ページの表B-4 に示すサブシステムをインストールします。

表B-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
IRIS FailSafe 2.1 Cluster Manager GUI	*sysadm_failsafe2.sw.client*	sysadm_base.sw.client
	sysadm_failsafe2.sw.desktop	*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

Java Plug-in がインストールされていない場合にブラウザから IRIS FailSafe 2.1 クラスタ・マネージャ GUI を実行すると、ブラウザは *http://java.sun.com/products/plugin/1.1/plugin-install.html* にリダイレクトされます。

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。IRIX 以外のワークステーションから 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、248 ページの表B-5 に示すサブシステムをインストールします。

表B-5 IRIX 以外の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
Java Plug-in	*http://java.sun.com/products/plugin/1.1/plugin-install.html* から Java Plug-in をダウンロードします。	Java をサポートする Web ブラウザ

Java Plug-in がインストールされていない場合にブラウザから IRIS FailSafe 2.1 クラスタ・マネージャ GUI を実行すると、ブラウザは、248 ページの表B-5に示す Web サイトにリダイレクトされます。

PCP (Peformance Co-Pilot) for FailSafe によってエクスポートされるメトリック

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C-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_smrd(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。

メトリック	説明
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。
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_smrd(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_smrd(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% 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.histo_40 fsafe.srm.all.histo_40	ha_smrd(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_smrd(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。
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_smrd(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_smrd(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。

メトリック	説明
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_smrd(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。
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。

メトリック	説明
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。

メトリック	説明
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。 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。

メトリック	説明
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。
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。

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.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。
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
クラスタ設定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 Software Installation and Administration Guide』を参照してください。

fs2d 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、クラスタ設定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。

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。

依存リスト
リソースの依存性またはリソース・タイプの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。

クラスタ設定データベース

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。IRIS 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。

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。クラスタ設定データベース (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・ドメインは同じこともあります。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

クラスタ設定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、Origin 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。

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設定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。

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` 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、特定の時点では 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、特定のリソース・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。注記: 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