Sun Cluster データサービスの計画と管理 (Solaris OS 版)

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–0196–10
2004 年 9 月, Revision A

原典: *Sun Cluster Data Services Planning and Administration Guide for Solaris OS*

Part No: 817-6564-10

Revision A

041112@10082

目次

はじめに

『*Sun Cluster* データサービスの計画と管理 *(Solaris OS* 版*)*』は、SPARC® と x86 ベースシステムでの Sun™ Cluster データサービス のインストールと構成について説明します。

注 – このマニュアルでは、「x86」という用語は、Intel 32 ビット系列のマイクロプロセッサチップ、および AMD が提供する互換マイクロプロセッサチップを意味します。

このマニュアルは、Sun のソフトウェアとハードウェアについて幅広い知識を持っている上級システム管理者を対象としています。販売活動のガイドとしては使用しないでください。このマニュアルを読む前に、システムの必要条件を確認し、適切な装置とソフトウェアを購入しておく必要があります。

このマニュアルの説明を理解するためには、Solaris™ オペレーティングシステムの知識と、Sun Cluster とともに使用されるボリューム管理ソフトウェアの知識が必要です。

注 – Sun Cluster ソフトウェアは、SPARC と x86 の 2 つのプラットフォーム上で稼動します。このマニュアル内の情報は、章、節、注、箇条書き項目、図、表、または例などで特に明記されていない限り両方に適用されます。

UNIX コマンド

このマニュアルでは、Sun Cluster データサービスのインストールと構成に固有のコマンドについて説明します。このマニュアルでは、基本的な UNIX® コマンドの包括的な情報や手順 (システムの停止、システムの起動、およびデバイスの構成など) については説明しません。基本的な UNIX コマンドに関する情報および手順については、以下を参照してください。

- Solaris オペレーティングシステムのオンラインドキュメント
- Solaris オペレーティングシステムのマニュアルページ
- システムに付属するその他のソフトウェアマニュアル

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 **P–1** 表記上の規則

字体または記号	意味	例
`AaBbCc123`	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	`.login` ファイルを編集します。 `ls -a` を使用してすべてのファイルを表示します。 `system%`
`AaBbCc123`	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	`system%` **`su`** `password:`
AaBbCc123	変数を示します。実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、`rm` *filename* と入力します。
『 』	参照する書名を示します。	『コードマネージャ・ユーザーズガイド』を参照してください。
「 」	参照する章、節、ボタンやメニュー名、強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。

表 **P–1** 表記上の規則　*(続き)*

字体または記号	意味	例
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	`sun% grep '^#define \ XV_VERSION_STRING'`

コード例は次のように表示されます。

- C シェル

  ```
  machine_name% command y|n [filename]
  ```

- C シェルのスーパーユーザー

  ```
  machine_name# command y|n [filename]
  ```

- Bourne シェルおよび Korn シェル

  ```
  $ command y|n [filename]
  ```

- Bourne シェルおよび Korn シェルのスーパーユーザー

  ```
  # command y|n [filename]
  ```

[ ] は省略可能な項目を示します。上記の例は、*filename* は省略してもよいことを示しています。

| は区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: Shift キーを押します)。ただし、キーボードによっては Enter キーが Return キーの動作をします。

ダッシュ (-) は 2 つのキーを同時に押すことを示します。たとえば、Ctrl-D は Control キーを押したまま D キーを押すことを意味します。

関連マニュアル

関連する Sun Cluster トピックについての情報は、以下の表に示すマニュアルを参照してください。すべての Sun Cluster マニュアルは、http://docs.sun.com で参照できます。

トピック	マニュアル
データサービス管理	『*Sun Cluster* データサービスの計画と管理 *(Solaris OS* 版*)*』
	各データサービスガイド
概念	『*Sun Cluster* の概念 *(Solaris OS* 版*)*』
概要	『*Sun Cluster* の概要 *(Solaris OS* 版*)*』
ソフトウェアのインストール	『*Sun Cluster* ソフトウェアのインストール *(Solaris OS* 版*)*』
システム管理	『*Sun Cluster* のシステム管理 *(Solaris OS* 版*)*』
ハードウェア管理	『*Sun Cluster 3.x Hardware Administration Manual for Solaris OS*』
	各ハードウェア管理ガイド
データサービスの開発	『*Sun Cluster* データサービス開発ガイド *(Solaris OS* 版*)*』
エラーメッセージ	『*Sun Cluster Error Messages Guide for Solaris OS*』
コマンドと関数の参照	『*Sun Cluster Reference Manual for Solaris OS*』

Sun Cluster の完全なマニュアルリストについては、ご使用のリリースの Sun Cluster のリリース情報 (`http://docs.sun.com`) を参照してください。

関連する Sun 以外の Web サイトの参照

このマニュアルで参照されている Sun 以外の URL には、関連する情報が提供されています。

注 – このマニュアルには、サン以外の団体/個人の Web サイトに関する情報が含まれています。サンは、これらのサイトあるいはリソースに関する、あるいはこれらのサイト、リソースから利用可能であるコンテンツ、広告、製品、あるいは資料に関して一切の責任を負いません。こうしたサイトやリソース上で、またはこれらを経由して利用できるコンテンツ、製品、サービスを利用または信頼したことに伴って実際に発生した (あるいは発生したと主張される) いかなる損害や損失についても、Sun は一切の責任を負いません。

Sun のオンラインマニュアル

docs.sun.com では、Sun が提供しているオンラインマニュアルを参照することができます。マニュアルのタイトルや特定の主題などをキーワードとして、検索を行うこともできます。URL は、http://docs.sun.com です。

ヘルプ

Sun Cluster をインストールまたは使用しているときに問題が発生した場合は、ご購入先に連絡し、次の情報をお伝えください。

- 名前と電子メールアドレス (利用している場合)
- 会社名、住所、および電話番号
- システムのモデルとシリアル番号
- Solaris オペレーティングシステムのバージョン番号 (例: Solaris 8)
- Sun Cluster のバージョン番号(例: Sun Cluster 3.0)

サービスプロバイダのために、次のコマンドを使用して、システム上の各ノードに関する情報を収集してください。

コマンド	機能
prtconf -v	システムメモリのサイズと周辺デバイス情報を表示します
psrinfo -v	プロセッサの情報を表示する
showrev -p	インストールされているパッチを報告する
SPARC:prtdiag -v	システム診断情報を表示する
scinstall -pv	Sun Cluster のリリースおよびパッケージのバージョン情報を表示します

上記の情報にあわせて、/var/adm/messages ファイルの内容もご購入先にお知らせください。

第 1 章

Sun Cluster データサービスの計画

この章では、Sun Cluster データサービスのインストールと構成を計画するにあたってのガイドラインを説明します。この章の内容は次のとおりです。

データサービス、リソースタイプ、リソース、およびリソースグループについての概念的な情報については、『 *Sun Cluster* の概念 *(Solaris OS* 版*)*』のマニュアルを参照してください。

Sun Cluster ソフトウェアがサービスを提供できるのは、 Sun Cluster 製品で提供されるデータサービス、または、 Sun Cluster データサービス API (Application Programming Interface) で作成されたデータサービスだけです。

Sun Cluster データサービスとして現在提供されていないアプリケーションについては、『 *Sun Cluster* データサービス開発ガイド *(Solaris OS* 版*)*』を参照してください。アプリケーション用の高可用性データサービスを開発する方法について説明されています。

注 – Sun Cluster ソフトウェアは、`sendmail(1M)` サブシステム用のデータサービスを提供しません。`sendmail` サブシステムを個々のクラスタノードで実行することは認められていますが、`sendmail` の機能 (メールの配信、経路設定、待ち行列化、再試行など) は HA 対応ではありません。

Sun Cluster データサービス構成のガイドライン

この節では、Sun Cluster データサービスを構成するためのガイドラインを説明します。

データサービス固有の要件の確認

Solaris と Sun Cluster のインストールを開始する前に、すべてのデータサービスの要件を確認します。計画に不備があった場合、インストールエラーが発生し、Solaris や Sun Cluster ソフトウェアを完全にインストールし直す必要が生じる可能性もあります。

たとえば、Sun Cluster Support for Oracle Parallel Server/Real Application Clusters の Oracle Parallel Fail Safe/Real Application Clusters Guard オプションには、ユーザーがクラスタ内で使用するホスト名に関する特殊な要件があります。Sun Cluster HA for SAP にも特殊な要件があります。Sun Cluster ソフトウェアをインストールした後にホスト名は変更できないため、このような必要条件は Sun Cluster ソフトウェアをインストールする前に調整しておく必要があります。Sun Cluster Support for Oracle Parallel Server/Real Application Clusters および Sun Cluster HA for SAP は、どちらも x86 ベースのクラスタではサポートされていないので注意してください。

アプリケーションバイナリの格納先の決定

アプリケーションソフトウェアおよびアプリケーション構成ファイルは、次のいずれかの場所にインストールできます。

- 各クラスタノードのローカルディスク – ソフトウェアと構成ファイルを個々のクラスタノードに配置すると、次のようなメリットが得られます。あとでアプリケーションを更新する場合に、サービスを停止することなく実施できます。

  ただし、ソフトウェアや構成ファイルの異なるコピーが存在するため、保守や管理をするファイルが増えるという欠点があります。
- クラスタファイルシステム – アプリケーションバイナリをクラスタファイルシステムに格納した場合、保守や管理をするコピーが 1 つだけになります。しかし、アプリケーションソフトウェアをアップグレードするには、クラスタ全体でデータサービスを停止する必要があります。アップグレード時に多少の時間停止できるようであれば、アプリケーションおよび構成ファイルの 1 つのコピーをクラスタファイルシステムに格納することが可能です。

  クラスタファイルシステムの作成方法については、『*Sun Cluster* ソフトウェアのインストール *(Solaris OS* 版*)*』の計画の章を参照してください。

- **HA** ローカルファイルシステム - `HAStoragePlus` を使用すると、ローカルファイルシステムを Sun Cluster 環境に統合して、ローカルファイルシステムの可用性を高めることができます。`HAStoragePlus` は、Sun Cluster でローカルファイルシステムのフェイルオーバーを行うための付加的なファイルシステム機能 (チェック、マウント、強制的なマウント解除など) も提供します。フェイルオーバーを行うには、アフィニティスイッチオーバーが有効な広域ディスクグループ上にローカルファイルシステムが存在していなければなりません。

  `HAStoragePlus` リソースタイプを使用する方法については、各データサービスガイド、または 86 ページの「高可用性ローカルファイルシステムの有効化」を参照してください。

`nsswitch.conf` ファイルの内容の確認

`nsswitch.conf` ファイルは、ネームサービスの検索用の構成ファイルです。このファイルは次の情報を指定します。

- ネームサービスの検索に使用する Solaris 環境内のデータベース
- データベースの検索順序

一部のデータサービスについては、「`group`」検索の対象の先頭を「`files`」に変更してください。具体的には、`nsswitch.conf` ファイル内の「`group`」行を変更し、「`files`」エントリが最初にリストされるようにします。「`group`」行を変更するかどうかを判断するには、構成するデータサービスに関するマニュアルを参照してください。

Sun Cluster 環境で `nsswitch.conf` ファイルを構成する方法の追加情報については、『*Sun Cluster* ソフトウェアのインストール *(Solaris OS* 版*)* 』の計画の章を参照してください。

クラスタファイルシステムの構成の計画

データサービスによっては、Sun Cluster の要件を満たす必要があります。特別な検討事項が必要かどうかを判断するには、そのデータサービスに関するマニュアルを参照してください。

クラスタファイルシステムの作成方法については、『*Sun Cluster* ソフトウェアのインストール *(Solaris OS* 版*)*』の計画の章を参照してください。

リソースタイプ `HAStoragePlus` を使用すると、フェイルオーバー用に構成された Sun Cluster 環境で HA ローカルファイルシステムを使用できます。`HAStoragePlus` リソースタイプの設定方法については、86 ページの「高可用性ローカルファイルシステムの有効化」を参照してください。

リソースグループとディスクデバイスグループの関係

Sun Cluster は、ディスクデバイスグループとリソースグループに関し、ノードリストという概念を持っています。ノードリストには、ディスクデバイスグループまたはリソースグループの潜在的マスターであるノードが順にリストされています。ダウンしていたノードがクラスタに再結合し、そのノードがノードリストで現在の稼働系より前に設定されているときの挙動を決定するために、Sun Cluster は「フェイルバックポリシー」を使用します。フェイルバックが `True` に設定されていると、デバイスグループまたはリソースグループが現在の稼働系から、再結合したノードに切り替えられ、このノードが新しい稼働系になります。

フェイルオーバーリソースグループの高可用性を保証するには、そのグループのノードリストと関連するディスクデバイスグループのノードリストとを一致させます。スケーラブルリソースグループの場合、そのリソースグループのノードリストは必ずしもデバイスグループのノードリストと一致するとは限りません。これは、現段階では、デバイスグループのノードリストには 2 つのノードしか含むことができないためです。2 ノードを超えるクラスタの場合は、スケーラブルリソースグループのノードリストに、3 ノード以上を含むことができます。

たとえば、ノード `phys-schost-1` と `phys-schost-2` が含まれるノードリストを持つディスクデバイスグループ `disk-group-1` があり、フェイルバックポリシーが `Enabled` に設定されているとします。さらに、 アプリケーションデータの保持に `disk-group-1` を使用する `resource-group-1` というフェイルオーバーリソースグループも持っているとします。このような場合は、`resource-group-1` を設定するときに、リソースグループのノードリストに `phys-schost-1` と `phys-schost-1` も指定し、フェイルバックポリシーを `True` に設定します。

スケーラブルリソースグループの高可用性を保証するためには、そのスケーラブルサービスグループのノードリストをディスクデバイスグループのノードリストのスーパーセットにします。スーパーセットにすることで、ディスクに直接接続されるノードは、スケーラブルリソースグループを実行するノードになります。この利点は、データに接続されている少なくとも 1 つのクラスタノードがクラスタで起動されているときに、スケーラブルリソースグループがこれらと同じノード上で実行されても、スケーラブルサービスは利用できることです。

ディスクデバイスグループの設定方法の詳細は、『 *Sun Cluster* ソフトウェアのインストール *(Solaris OS* 版*)*』を参照してください。ディスクデバイスグループとリソースグループの関係の詳細については、『*Sun Cluster* の概念 *(Solaris OS* 版*)*』のマニュアルを参照してください。

HAStorage と HAStoragePlus の概要

リソースタイプ HAStorage と HAStoragePlus は、次のオプションの設定に使用できます。

- ディスクデバイスリソースが利用可能になるまで、HAStorage または HAStoragePlus のリソースを含む同じリソースグループ内にあるほかのリソースの START メソッドを待機させることによって、ディスクデバイスとリソースグループの起動の順番を調整できます。
- AffinityOn を True に設定することで、リソースグループとディスクデバイスグループを同一ノード上におき、ディスクに負荷がかかることの多いデータサービスのパフォーマンスを向上させます。

HAStoragePlus は、マウントされていない状態であることが確認されたグローバルファイルシステムをマウントすることもできます。詳細については、17 ページの「クラスタファイルシステムの構成の計画」を参照してください。

注 – HAStorage または HAStoragePlus リソースがオンラインの間にデバイスグループが別のノードに切り替えられた場合、AffinityOn の設定は無視され、リソースグループはデバイスグループと共に別のノードに移行することはありません。一方、リソースグループが別のノードに切り替わった場合、AffinityOn が True に設定されていると、デバイスグループはリソースグループと一緒に新しいノードに移動します。

ディスクデバイスグループとリソースグループ間の関係については、80 ページの「リソースグループとディスクデバイスグループ間での起動の同期」を参照してください。追加情報は、SUNW.HAStorage(5) および SUNW.HAStoragePlus(5) のマニュアルページにあります。

VxFS などのファイルシステムをローカルモードでマウントする方法については、86 ページの「高可用性ローカルファイルシステムの有効化」を参照してください。詳細は、SUNW.HAStoragePlus(5) のマニュアルページを参照してください。

データサービスが HAStorage または HAStoragePlus を必要とするかどうかを確認する方法

- データサービスリソースグループがノードリストを持っており、その一部のノードが記憶装置に直接接続されていない場合は、リソースグループ内で HAStorage または HAStoragePlus リソースを構成し、ほかのデータサービスリソースの依存性を HAStorage リソースまたは HAStoragePlus リソースに設定する必要があ

ります。この条件によって、ストレージとデータサービス間の起動の順番が調整されます。

- ディスクに負荷がかかるデータサービスの場合 (Sun Cluster HA for Oracle や Sun Cluster HA for NFS など)、次の作業を行う必要があります。
  - `HAStorage` リソースまたは `HAStoragePlus` リソースをデータリソースグループに追加します。
  - `HAStorage` リソースまたは `HAStoragePlus` リソースをオンラインに切り替えます。
  - データサービスリソースの依存性を `HAStorage` リソースまたは HAStoragePlus リソースに設定します。
  - `AffinityOn` を `True` に設定します。

    これらの作業を行うと、リソースグループとディスクデバイスグループは同じノード上に配置されます。
  - フェイルバック設定は、リソースグループとデバイスグループで同一にする必要があります。
- 起動時に全ファイルを読み込むなどによりデータサービスがディスクにさほど負荷をかけない場合 (Sun Cluster HA for DNS など) は、`HAStorage` または `HAStoragePlus` リソースタイプの構成を省略することもできます。

`HAStorage` または `HAStoragePlus` の選択

データサービスリソースグループ内で `HAStorage` リソースと `HAStoragePlus` リソースのどちらを作成すべきかを決定するには、以下の基準を考慮してください。

- Sun Cluster 3.0 5/02 または Sun Cluster 3.1 を使用する場合は、`HAStoragePlus` を使用してください (フェイルオーバーが構成された Sun Cluster にローカルアクセスを構成するようにファイルシステムを統合するには、Sun Cluster 3.0 5/02 または Sun Cluster 3.1 にアップグレードし、`HAStoragePlus` リソースタイプを使用してください。詳細については、17 ページの「クラスタファイルシステムの構成の計画」を参照してください。)
- Sun Cluster 3.0 12/01 以前を使用している場合は、`HAStorage` を使用してください。

考慮事項

この節の情報は、データサービスのインストールまたは構成について計画する場合に利用してください。これらの情報に目を通すことで、ユーザーの決定がデータサービスのインストールと構成に及ぼす影響について理解できるでしょう。特定のデータサービスについては、そのデータサービスのマニュアルを参照してください。

- データサービスが入出力中心で、クラスタとして大規模のディスクで構成している場合にディスクに障害が発生すると、入出力サブシステムが再試行するので、アプリケーションは遅延を感じることがあります。入出力サブシステムが再試行し、障害から回復するまで、数分かかることもあります。この遅延によって、最終的にディスクが自分自身で回復したとしても、Sun Cluster がアプリケーションを別のノードにフェイルオーバーすることがあります。このような場合のフェイルオーバーを回避するには、データサービスのデフォルトの検証タイムアウト値を増やしてみてください。データサービスのタイムアウトについての詳細や、タイムアウト値を増やす方法については、ご購入先にお問い合わせください。
- よりよいパフォーマンスを保つために、ストレージに直結されたクラスタノードにデータサービスをインストールし、構成してください。
- クラスタノード上で動作するクライアントアプリケーションは、HA データサービスの論理 IP アドレスにマッピングしてはなりません。フェイルオーバー後、このような論理 IP アドレスは存在しなくなり、クライアントが切断されたままになる可能性があります。

ノードリストプロパティ

データサービスを構成するときに、次の 3 つのノードリストを指定できます。

1. `installed_nodes` – データサービスのリソースタイプのプロパティ。このプロパティには、リソースタイプがインストールされ、実行が有効になるクラスタノード名の一覧が含まれます。
2. `nodelist` — リソースグループのプロパティ。優先順位に基づいて、グループをオンラインにできるクラスタノード名の一覧が含まれます。これらのノードは、リソースグループの潜在的主ノードまたはマスターです。フェイルオーバーサービスについては、リソースグループノードリストを 1 つだけ設定します。スケーラブルサービスの場合は、2 つのリソースグループを設定するため、ノードリストも 2 つ必要になります。一方のリソースグループとノードリストには、共有アドレスをホストするノードが含まれます。このノードリストは、スケーラブルリソースが依存するフェイルオーバーリソースグループを構成します。もう一方のリソースグループとそのノードリストは、アプリケーションリソースをホストするノードを識別します。アプリケーションリソースは、共有アドレスに依存します。共有アドレスを含むリソースグループ用のノードリストは、アプリケーションリソース用のノードリストのスーパーセットになる必要があるためです。
3. `auxnodelist` — 共有アドレスリソースのプロパティ。このプロパティは、クラスタノードを識別する物理ノード ID の一覧が含まれます。このクラスタノードは共有アドレスをホストできますが、フェイルオーバー時に主ノードになることはありません。これらのノードは、リソースグループのノードリストで識別されるノードとは、相互に排他的な関係になります。このノードリストは、スケーラブルサービスにのみ適用されます。詳細は、`scrgadm`(1M) のマニュアルページを参照してください。

インストールと構成プロセスの概要

データサービスをインストールして構成するには、次の手順を使用します。

- パッケージが提供されているインストールメディアから、データサービスパッケージをインストールします。
  - Sun Java™ Enterprise System CD
  - Sun Java Enterprise System Accessory CD Volume 3
- クラスタ環境で実行するアプリケーションをインストールして構成する。
- データサービスが使用するリソースおよびリソースグループを構成する。データサービスを構成するときは、Resource Group Manager (RGM) によって管理される、リソースタイプ、リソース、リソースグループを指定します。これらの手順は、各データサービスに関するマニュアルで説明されています。

データサービスのインストールと構成を開始する前に『*Sun Cluster* ソフトウェアのインストール *(Solaris OS* 版*)*』を参照してください。このマニュアルには、データサービスソフトウェアパッケージのインストール方法と、ネットワークリソースが使用する Internet Protocol Network Multipathing (IP Networking Multipathing) グループを構成する方法を確認してください。

注 – 以下のデータサービスのインストールと構成には、SunPlex™ Manager を使用できます。Sun Cluster HA for Oracle、Sun Cluster HA for Sun Java System Web Server、Sun Cluster HA for Sun Java System Directory Server、 Sun Cluster HA for Apache、Sun Cluster HA for DNS、および Sun Cluster HA for NFS。Sun Cluster HA for Oracle および Sun Cluster HA for Apache は、SPARC ベースのクラスタでのみサポートされているので注意してください。詳細については、SunPlex Manager のオンラインヘルプを参照してください。

インストールと構成の作業の流れ

表 1–1 に、Sun Cluster フェイルオーバーデータサービスのインストールおよび構成作業と、その手順が説明されている参照先を示します。

表 **1–1** Task Map: Sun Cluster データサービスのインストールと構成

タスク	参照箇所
Solaris と Sun Cluster ソフトウェアのインストール	『*Sun Cluster* ソフトウェアのインストール *(Solaris OS* 版*)*』

表 1–1 Task Map: Sun Cluster データサービスのインストールと構成　(続き)

タスク	参照箇所
IP ネットワークマルチパス グループの設定	『*Sun Cluster* ソフトウェアのインストール *(Solaris OS* 版)』
多重ホストディスクの設定	『*Sun Cluster* ソフトウェアのインストール *(Solaris OS* 版)』
リソースとリソースグループの計画	付録 C
アプリケーションバイナリの格納先の決定 (nsswitch.conf の構成)	16 ページの「アプリケーションバイナリの格納先の決定」 17 ページの「nsswitch.conf ファイルの内容の確認」
アプリケーションソフトウェアのインストールと構成	該当する Sun Cluster データサービスブック
データサービスソフトウェアパッケージのインストール	『*Sun Cluster* ソフトウェアのインストール *(Solaris OS* 版)』または該当する Sun Cluster データサービスブック
データサービスの登録と構成	該当する Sun Cluster データサービスブック

SPARC: 例

この節では、例として高可用性フェイルオーバーデータサービスとして設定されている Oracle アプリケーション用に、リソースタイプ、リソース、リソースグループを設定する方法を紹介します。

この例とスケーラブルデータサービスの例では、ネットワークリソースを含むフェイルオーバーリソースグループが異なります。さらに、スケーラブルデータサービスには、アプリケーションリソースごとに別のリソースグループ (スケーラブルリソースグループ) が必要です。

Oracle アプリケーションは、サーバーとリスナーの 2 つのコンポーネントを持ちます。Sun は Sun Cluster HA for Oracle データサービスを提供しているため、これらのコンポーネントはすでに Sun Cluster リソースタイプに対応付けられています。これら両方のリソースタイプが、リソースとリソースグループに関連付けられます。

この例は、フェイルオーバーデータサービスの例なので、論理ホスト名ネットワークリソースを使用し、主ノードから二次ノードにフェイルオーバーする IP アドレスを使用します。フェイルオーバーリソースグループに論理ホスト名リソースを入れ、Oracle サーバーリソースとリスナーリソースを同じリソースグループに入れます。この順に入れることで、フェイルオーバーを行うすべてのリソースが 1 つのグループになります。

Sun Cluster HA for Oracle をクラスタ上で実行するには、次のオブジェクトを定義する必要があります。

- LogicalHostname リソースタイプ – このリソースタイプは組み込まれているため、明示的に登録する必要はありません。
- Oracle リソースタイプ – Sun Cluster HA for Oracle は、2 つの Oracle リソースタイプ (データベースサーバーとリスナー) を定義します。
- 論理ホスト名リソース – これらのリソースは、ノードで障害が発生した場合にフェイルオーバーする IP アドレスをホストします。
- Oracle リソース – Sun Cluster HA for Oracle— 用に 2 つのリソースインスタンス (サーバーとリスナー) を指定する必要があります。
- フェイルオーバーリソースグループ – 1 つのグループでフェイルオーバーを行う、Oracle サーバーとリスナー、および論理ホスト名リソースで構成されています。

データサービスリソースを管理するためのツール

この節では、インストールや構成の作業に使用するツールについて説明します。

SunPlex Manager グラフィカルユーザーインタフェース (GUI)

SunPlex Manager は、次の作業を実行できる Web ベースのツールです。

- クラスタのインストール
- クラスタの管理
- リソースやリソースグループの作成と構成
- Sun Cluster ソフトウェアを使ったデータサービスの構成

SunPlex Manager を使用し、クラスタソフトウェアをインストールする方法については、『 *Sun Cluster* ソフトウェアのインストール *(Solaris OS* 版*)*』を参照してください。管理作業については、SunPlex Manager のオンラインヘルプを参照してください。

SPARC: Sun Management Center GUI 用の Sun Cluster モジュール

Sun Cluster モジュールを使用すると、クラスタの監視やリソースおよびリソースグループに対する処理の一部を Sun Management Center GUI から行えます。Sun Cluster モジュールのインストール要件と手順については、『*Sun Cluster* ソフトウェアのインストール *(Solaris OS* 版*)*』を参照してください。Sun Management Center の詳細は、`http://docs.sun.com` にある Sun Management Center ソフトウェアのマニュアルを参照してください。

scsetup ユーティリティー

scsetup(1M) ユーティリティーは、メニュー主導型のインタフェースで、Sun Cluster の一般的な管理に使用できます。このユーティリティーは、さらに、データサービスのリソースやリソースグループの構成にも使用できます。この場合には、scsetup のメインメニューからオプション 2 を選択して、「Resource Group Manager」というサブメニューを起動してください。

scrgadm コマンド

scrgadm コマンドにより、データサービスリソースの登録や構成を行うことができます。データサービスの登録と構成の方法については、データサービスのブックを参照してください。たとえば、Sun Cluster HA for Oracle を使用する場合は、『*Sun Cluster Data Service for Oracle* ガイド *(Solaris OS* 版*)*』の手順を参照してください。第 2 章 にも、scrgadm コマンドを使ってデータサービスリソースを管理する方法が記載されています。最後に、追加情報については、scrgadm(1M) のマニュアルページを参照してください。

データサービスリソースの管理作業

次の表に、データサービスリソースの管理作業に使用できるツール (コマンド行以外の) を示します。これらの作業の詳細や、関連する手順をコマンド行から行う方法については、第 2 章を参照してください。

表 1–2 データサービスリソースの管理作業に使用できるツール

タスク	SunPlex Manager	SPARC:Sun Management Center	scsetup ユーティリティー
リソースタイプを登録する	Yes	いいえ	Yes
リソースグループを作成	Yes	いいえ	Yes
リソースグループへソースを追加する	Yes	いいえ	Yes
リソースグループをオンラインにする	Yes	Yes	いいえ
リソースグループを削除	Yes	Yes	いいえ
リソースを削除する	Yes	Yes	いいえ
リソースグループの現在の主ノードを切り替える	Yes	いいえ	いいえ
リソースを使用不可にする	Yes	Yes	いいえ

表 **1–2** データサービスリソースの管理作業に使用できるツール　(続き)

タスク	**SunPlex Manager**	**SPARC:Sun Management Center**	`scsetup` ユーティリティー
無効なリソースのリソースグループを非管理状態にする	Yes	いいえ	いいえ
リソースタイプ、リソースグループ、リソース構成情報を表示する	Yes	Yes	いいえ
リソースプロパティを変更する	Yes	いいえ	いいえ
リソースの `STOP_FAILED` エラーフラグを消去する	Yes	いいえ	いいえ
ノードをリソースグループに追加する	Yes	いいえ	いいえ

第 2 章

データサービスリソースの管理

この章では、scrgadm(1M) コマンドを使って、リソースや、リソースグループ、リソースタイプを管理する手順を説明します。手順を実行するその他のツールについては、24 ページの「データサービスリソースを管理するためのツール」を参照してください。

この章では、次の手順について説明します。

- 81 ページの「新しいリソース用に HAStorage リソースタイプを設定する」
- 83 ページの「既存のリソース用に HAStorage リソースタイプを設定する」
- 87 ページの「HAStoragePlus リソースタイプを設定する」
- 90 ページの「オンラインの HAStoragePlus リソースにファイルシステムを追加する」
- 92 ページの「オンラインの HAStoragePlus リソースからファイルシステムを削除する」
- 95 ページの「HAStoragePlus リソースの変更後に障害から回復する」
- 107 ページの「RGOffload リソースを設定する」
- 111 ページの「リソースグループ、リソース型、およびリソースが構成されていないクラスタに構成データを複製する」
- 112 ページの「リソースグループ、リソース型、およびリソースが構成されているクラスタの構成データをアップグレードする」

リソースタイプ、リソースグループ、およびリソースに関する概要情報については、第 1 章 および『*Sun Cluster* の概念 *(Solaris OS* 版*)*』マニュアルを参照してください。

データサービスリソースの管理

表 2–1 に、データサービスリソースの管理作業を説明している節を示します。

表 **2–1** Task Map: データサービス管理

タスク	参照箇所
リソースタイプを登録する	31 ページの「リソースタイプを登録する」
リソースタイプをアップグレードする	34 ページの「既存のリソースを新バージョンのリソースタイプに移行する」 33 ページの「アップグレードされたリソースタイプをインストールして登録する」
フェイルオーバーリソースグループまたはスケーラブルリソースグループの作成	39 ページの「フェイルオーバーリソースグループを作成する」 40 ページの「スケーラブルリソースグループを作成する」

表 2–1 Task Map: データサービス管理　(続き)

タスク	参照箇所
論理ホスト名または共有アドレス、データサービスリソースをリソースグループに追加する	43 ページの「論理ホスト名リソースをリソースグループに追加する」 44 ページの「共有アドレスリソースをリソースグループに追加する」 46 ページの「フェイルオーバーアプリケーションリソースをリソースグループに追加する」 48 ページの「スケーラブルアプリケーションリソースをリソースグループに追加する」
リソースとリソースモニターを有効にし、リソースグループを管理し、リソースグループおよび関連するリソースをオンラインにする	51 ページの「リソースグループをオンラインにする」
リソース自体とは関係なく、リソースモニターだけを無効または有効にする	53 ページの「リソース障害モニターを無効にする」 53 ページの「リソース障害モニターを有効にする」
クラスタからリソースタイプを削除する	54 ページの「リソースタイプを削除する」
クラスタからリソースグループを削除する	55 ページの「リソースグループを削除する」
リソースグループからリソースを削除する	57 ページの「リソースを削除する」
リソースグループの稼動系を切り替える	58 ページの「リソースグループの主ノードを切り替える」
リソースを無効にし、そのリソースグループをUNMANAGEDに移行する	60 ページの「リソースを無効にしてリソースグループを非管理状態に移行する」
リソースタイプ、リソースグループ、リソース構成情報を表示する	62 ページの「リソースタイプ、リソースグループ、リソース構成情報の表示」
リソースタイプ、リソースグループ、リソースプロパティの変更	63 ページの「リソースタイププロパティを変更する」 65 ページの「リソースグループプロパティを変更する」 65 ページの「リソースプロパティを変更する」
失敗した Resource Group Manager (RGM) プロセスのエラーフラグの消去	68 ページの「リソースの STOP_FAILED エラーフラグを消去する」

表 **2–1** Task Map: データサービス管理　(続き)

タスク	参照箇所
組み込みリソースタイプ LogicalHostname および SharedAddress の再登録	71 ページの「事前登録されているリソースタイプを誤って削除した後に再登録する」
組み込みリソースタイプ LogicalHostname および SharedAddress のアップグレード	33 ページの「リソースタイプの更新」 69 ページの「事前登録されているリソースタイプのアップグレード」
ネットワークリソースのネットワークインタフェース ID リストの更新と、リソースグループのノードリストの更新	73 ページの「リソースグループにノードを追加する」
リソースグループからノードを削除する	75 ページの「リソースグループからノードを削除する」
リソースグループとディスクデバイスグループ間で起動の同期をとるために、リソースグループの HAStorage または HAStoragePlus を設定する	81 ページの「新しいリソース用に HAStorage リソースタイプを設定する」
ディスク入出力負荷が高いフェイルオーバーデータサービスに対応するように、HAStoragePlus を設定してローカルファイルシステムの可用性を高める	87 ページの「HAStoragePlus リソースタイプを設定する」
高可用性ファイルシステムのリソースをオンラインのままで変更する	89 ページの「高可用性ファイルシステムのリソースをオンラインのままで変更する」
HAStoragePlus リソースタイプをアップグレードする	33 ページの「リソースタイプの更新」 96 ページの「HAStoragePlus リソースタイプのアップグレード」
リソースグループをオンラインのままでクラスタノード間で分散する	98 ページの「オンラインのリソースグループをクラスタノード間で分散する」
重要なデータサービスのためにノードを自動的に開放するようにリソースタイプを設定する	107 ページの「RGOffload リソースを設定する」
リソースグループ、リソースタイプ、およびリソースの構成データを複製およびアップグレードする	111 ページの「リソースグループ、リソースタイプ、およびリソースの構成データを複製およびアップグレードする」
Sun Cluster データベース用に障害モニターを調整する	113 ページの「Sun Cluster データベース用に障害モニターを調整する」

注 – この章では、scrgadm(1M) コマンドを使用し、これらの作業を完了する手順について解説します。これ以外のツールを使ってリソースを管理することもできます。これらの方法については、24 ページの「データサービスリソースを管理するためのツール」を参照してください。

Sun Cluster データサービスの構成と管理

Sun Cluster の構成は、複数の手順から成る単一の作業です。これらの手順により次の作業を実行できます。

- リソースタイプの登録
- リソースタイプのアップグレード
- リソースグループの作成
- リソースグループへのリソースの追加
- リソースをオンラインにする

データサービスの構成を変更するには、初期構成が終わった後で次の各手順を使用します。たとえば、リソースタイプ、リソースグループ、およびリソースプロパティを変更する場合は、63 ページの「リソースタイプ、リソースグループ、リソースプロパティの変更」へ進んでください。

リソースタイプの登録

リソースタイプは、指定されたタイプのすべてのリソースに適用される共通のプロパティとコールバックメソッドの仕様を提供します。リソースタイプは、そのタイプのリソースを作成する前に登録する必要があります。リソースタイプについての詳細は、第 1 章を参照してください。

▼ リソースタイプを登録する

この手順を実行するには、登録するリソースタイプに、データサービス名の略語で名前をつける必要があります。Sun Cluster に標準添付されているデータサービスのリソースタイプ名の詳細は、Sun Cluster のリリースノートを参照してください。

追加情報については、scrgadm(1M) のマニュアルページを参照してください。

注 – この手順は、任意のクラスタノードから実行します。

1. クラスタメンバー上でスーパーユーザーになります。
2. リソースタイプを登録します。

   # **scrgadm -a -t** *resource-type*

`-a`	指定したリソースタイプを追加します。
`-t` *resource-type*	追加するリソースタイプの名前を指定します。指定する事前定義済みの名前を判別するには、Sun Cluster のリリースノートを参照してください。

3. 登録されたリソースタイプを確認します。

   # **scrgadm -pv -t** *resource-type*

例 – リソースタイプの登録

次に、Sun Cluster HA for Sun Java System Web Server (内部名 `iws`) を登録する例を示します。

```
# scrgadm -a -t SUNW.iws
# scrgadm -pv -t SUNW.iws
リソースタイプ 名前:                                    SUNW.iws
  (SUNW.iws) リソースタイプ 説明:                       None registered
  (SUNW.iws) リソースタイプ ベースディレクトリ:         /opt/SUNWschtt/bin
  (SUNW.iws) リソースタイプ 単一のインスタンス:         False
  (SUNW.iws) リソースタイプ 初期ノード:                 All potential masters
  (SUNW.iws) リソースタイプ フェイルオーバー:           False
  (SUNW.iws) リソースタイプ バージョン:                 1.0
  (SUNW.iws) リソースタイプ API バージョン:             2
  (SUNW.iws) リソースタイプ ノードにインストールされている: All
  (SUNW.iws) リソースタイプ パッケージ:                 SUNWschtt
```

次に進む手順

リソースタイプを登録したあと、リソースグループを作成し、リソースをそのリソースグループに追加できます。詳細は、39 ページの「リソースグループの作成」を参照してください。

リソースタイプの更新

新バージョンのリソースタイプがリリースされる際には、そのアップグレードされたリソースタイプをインストールして登録できます。また、既存のリソースを新しいリソースタイプバージョンにアップグレードすることも可能です。この節では、次の 2 つの作業、アップグレードされたリソースタイプをインストールして登録する方法と、既存のリソースを新しいリソースタイプバージョンにアップグレードする方法について説明します。

- 33 ページの「アップグレードされたリソースタイプをインストールして登録する」
- 34 ページの「既存のリソースを新バージョンのリソースタイプに移行する」

▼ アップグレードされたリソースタイプをインストールして登録する

この作業は、scsetup の「リソースグループ」オプションを使用しても行えます。scsetup の詳細は、scsetup(1M) のマニュアルページを参照してください。

1. すべてのクラスタノードに、リソースタイプアップグレードパッケージをインストールします。

   注 – リソースタイプパッケージがどのノードにもインストールされていない場合は、 作業が別途必要です (手順 3)。

   リソースタイプのアップグレードパッケージをインストールする際にノードを非クラスタモードで起動する必要があるかどうかは、アップグレードドキュメントに示されています。ダウンタイムを避けるには、パッケージをインストールするノードを非クラスタモード、残りのノードをクラスタモードに設定した状態で、一度に 1 台のノードに限定してローリングアップグレード方式で新しいパッケージを追加します。

2. この新しいリソースタイプバージョンを登録します。

   ```
   scrgadm -a -t resource_type -f path_to_new_RTR_file
   ```

   新しいリソースタイプの名前は次の形式をとります。

   ```
   vendor_id.rtname:version
   ```

   登録した新しいリソースタイプを表示するには、scrgadm -p または scrgadm -pv (詳細表示) を使用してください。

3. 新しいリソースタイプをインストールしないノードがある場合は、実際にリソースタイプをインストールしたノードを **Installed_nodes** プロパティに設定します。

   # **scrgadm -c -t** *resource_type* **-h** *installed_node_list*

新バージョンのリソースタイプは、次の点で旧バージョンと異なっている可能性があります。

- リソースタイププロパティの設定
- 標準プロパティ、拡張プロパティを含む宣言済みリソースプロパティ
- リソースプロパティの属性 (default、min、max、arraymin、arraymax、または tunability)
- 宣言済みメソッド
- メソッドやモニターの実装

▼ 既存のリソースを新バージョンのリソースタイプに移行する

この作業は、scsetup の「リソースグループ」オプションを使用しても行えます。scsetup の詳細は、scsetup(1M) のマニュアルページを参照してください。

新しいバージョンタイプに移行する方法は、既存のリソースタイプバージョンと、新バージョンにおける変更によって決まります。移行が可能かどうかは、リソースタイプのアップグレードドキュメントに記載されています。移行がサポートされていない場合は、リソースを削除してアップグレードされた新しいリソースに交換するか、あるいはそのリソースを古いリソースタイプバージョンのままにするかを検討してください。

既存のリソースを移行する場合は、以下の値が変化する可能性があります。

デフォルトのプロパティ値
: アップグレードされたリソースタイプバージョンがデフォルトプロパティに新しいデフォルト値を宣言している場合は、既存のリソースはこの新しいデフォルト値を継承します。

  既存のプロパティ設定が適切かどうかは、新しいリソースタイプバージョンの VALIDATE メソッドによってチェックされます。この設定が不適切な場合は、プロパティを編集して適切な値に変更してください。プロパティの編集方法は、手順 3 を参照してください。

リソースタイプ名
: RTR ファイルには、リソースタイプの完全修飾名の形成に使用される以下のプロパティが含まれます。

  - Vendor_id
  - Resource_type

- RT_Version

アップグレードされたリソースタイプバージョンは、その登録時に vendor_id.rtname:version として保存されます。新バージョンに移行されたリソースには、上記のプロパティから構成される新しい Type プロパティが存在します。

Type_version リソースプロパティ
リソースのタイプの RT_Version プロパティは、標準のリソースプロパティ Type_version に格納されます。Type_Version プロパティは、RTR ファイルには現れません。次のコマンドを使用して Type_Version プロパティを編集してください。

scrgadm -c -j *resource* **-y Type_version=***new_version*

1. 既存のリソースを新しいリソースタイプバージョンに移行する前に、新しいリソースタイプに付属しているアップグレードマニュアルに目を通し、移行が可能かどうかを確認してください。

   このマニュアルには、移行を実施すべきタイミングが記されています。

   - 任意の時点 (Anytime)
   - リソースが監視されていないとき
   - リソースがオフラインのとき
   - リソースが無効なとき
   - リソースグループが管理されていないとき

   注 – いつでも移行できるリソースを移行した後、リソースの検証においてと、リソースタイプのバージョンが正しく表示されないことがあります。このような状況が発生した場合、リソースの障害モニターを一度無効にし、有効にし直すると、リソースの検証において、リソースタイプのバージョンが正しく表示されます。

   移行がサポートされていない場合は、リソースを削除してアップグレードされた新しいリソースバージョンに置き換えるか、そのリソースを古いリソースタイプバージョンのままにしておく必要があります。

2. 移行するリソースタイプのリソースごとに、アップグレードマニュアルに記載されている方法でそのリソースグループのリソースの状態を適切な状態に変更してください。次に例を示します。

   リソースの監視を解除する必要がある場合:

   ```
   scswitch -M -n -j resource
   ```

   リソースをオフラインにする必要がある場合:

   ```
   scswitch -n -j resource
   ```

   リソースを無効にする必要がある場合:

   ```
   scswitch -n -j resource
   ```

リソースグループを非管理状態にする必要がある場合:

```
scsswitch -n -j resource-group
scswitch -F -g resource_group
scswitch -u -g resource_group
```

3. 移行するリソースタイプのリソースごとに、リソースを編集し、その **Type_version** プロパティを新バージョンに変更します。

```
scrgadm -c -j resource -y Type_version=new_version \
-x extension_property=new_value -y extension_property=new_value
```

必要に応じ、-x または -y オプション を追加して同じコマンドを実行し 、同じリソースのほかのプロパティを編集して適切な値に変更します。

4. 手順 2 で入力したコマンドを逆に指定することにより、リソースまたはリソースグループの前の状態に戻します。次に例を示します。

リソースを監視状態に戻す場合:

```
scswitch -M -e -j resource
```

リソースを有効な状態に戻す場合:

```
scswitch -e -j resource
```

リソースグループをオンラインの管理状態に戻す場合:

```
scswitch -o -g resource_group
scswitch -Z -g resource_group
```

例 1 – 既存のリソースを新しいリソースタイプバージョンに移行する

この例は、既存のリソースを新しいリソースタイプバージョンに移行する方法を示しています。新しいリソースタイプパッケージのメソッドは、新しいパスに配置されています。インストール時にメソッドは上書きされないため、アップグレードされたリソースタイプのインストールが完了するまでリソースを無効にする必要はありません。

この実例では、次のことを前提としています。

- 新しいリソースタイプバージョンは 2.0 である
- 移行を実行すべきタイミングは「リソースがオフラインのとき」である
- リソース名は「myresource」である
- リソースタイプ名は「myrt」である
- 新しい RTR ファイルは `/opt/XYZmyrt/etc/XYZ.myrt` に配備されている
- 移行の対象となるリソースに依存していない
- 所属しているリソースグループをオンラインの状態にしたまま、移行の対象となるリソースをオフラインに切り替えることができる

（ベンダーのディレクトリに従ってすべてのノード上で新しいパッケージをインストールする）

```
# scrgadm -a -t myrt -f /opt/XYZmyrt/etc/XYZ.myrt
```

```
# scswitch -n -j myresource
# scrgadm -c -j myresource -y Type_version=2.0
# scswitch -e -j myresource
```

例 2 – 既存のリソースを新しいリソースタイプバージョンに移行する

この例は、既存のリソースを新しいリソースタイプバージョンに移行する方法を示しています。新しいリソースタイプパッケージには、モニターと RTR ファイルしか含まれていません。モニターはインストール時に上書きされるため、アップグレードされたリソースタイプをインストールする前にリソースを無効にする必要があります。

この実例では、次のことを前提としています。

- 新しいリソースタイプバージョンは 2.0 である
- 移行を実行すべきタイミングは「リソースのマウントが解除しているとき」である
- リソース名は「myresource」である
- リソースタイプ名は「myrt」である
- 新しい RTR ファイルは `/opt/XYZmyrt/etc/XYZ.myrt` に配備されている

```
# scswitch -M -n -j myresource
(Install the new package according to vendor's directions.)
# scrgadm -a -t myrt -f /opt/XYZmyrt/etc/XYZ.myrt
# scrgadm -c -j myresource -y Type_version=2.0
# scswitch -M -e -j myresourcee
```

リソースタイプのダウングレード

リソースをダウングレードして古いバージョンのリソースタイプにすることができます。古いリソースタイプバージョンにダウングレードする場合は、新しいリソースタイプバージョンにアップグレードする場合よりも条件が厳しくなります。まず、リソースグループの管理を解除する必要があります。アップグレードが可能なリソースタイプバージョンにしかダウングレードできないということにも注意してください。アップグレード可能なバージョンは `scrgadm!)-p` コマンドを使用して確認できます。アップグレード可能なバージョンの場合、接尾辞 *:version* が表示されます。

▼ 古いバージョンのリソースタイプにダウングレードする方法

リソースをダウングレードして古いバージョンのリソースタイプにすることができます。古いリソースタイプバージョンにダウングレードする場合は、新しいリソースタイプバージョンにアップグレードする場合よりも条件が厳しくなります。まず、リ

ソースグループの管理を解除する必要があります。アップグレードが可能なリソースタイプバージョンにしかダウングレードできないということにも注意してください。アップグレード可能なバージョンは scrgadm!)-p コマンドを使用して確認できます。アップグレード可能なバージョンの場合、接尾辞 :*version* が表示されます。

1. ダウングレードしたいリソースを含んでいるリソースグループをオフラインに切り替えます。

   scswitch -F -g *resource_group*

2. ダウングレードするリソースと、そのリソースグループ内のすべてのリソースを無効にします。

   scswitch -n -j *resource_to_downgrade*
   scswitch -n -j *resource1*
   scswitch -n -j *resource2*
   scswitch -n -j *resource3*
   ...

   注 – リソースの無効化は、依存性の高いもの (アプリケーションリソース) から開始し、もっとも依存性の低いもの (ネットワークアドレスリソース) で終了するように行なってください。

   ---

3. リソースグループを非管理状態に切り替えます。

   scswitch -u -g *resource_group*

4. ダウングレード後のリソースタイプバージョンとする古いリソースバージョンがクラスタ内にまだ登録されているかどうか確認します。
   - 登録されている場合は、次の手順に進みます。
   - 登録されていない場合は、希望する旧バージョンを登録し直します。

     scrgadm -a -t *resource_type_name*

5. 希望する旧バージョンを **Type_version** に指定し、リソースをダウングレードします。

   scrgadm -c -j *resource_to_downgrade* **-y Type_version=***old_version*

   必要に応じて、同じコマンドを使って、同じリソースのその他のプロパティに適切な値を設定します。

6. ダウングレードしたリソースを含んでいるリソースグループを管理状態にし、すべてのリソースを有効にしたあと、このグループをオンラインに切り替えます。

   scswitch -Z -g *resource_group*

リソースグループの作成

リソースグループには、一連のリソースが含まれており、これらすべてのリソースは指定のノードまたはノード群で共にオンラインまたはオフラインになります。リソースを配置する前に、空のリソースグループを作成します。

リソースグループには、フェイルオーバーとスケーラブルの 2 つの種類があります。フェイルオーバーリソースグループの場合、同時にオンラインにできるのは 1 つのノードでのみです。一方、スケーラブルリソースグループの場合は、同時に複数のノードでオンラインにできます。

以下の手順では、`scrgadm`(1M) コマンドを使用し、データサービスを登録、構成する方法について解説します。

リソースグループに関する概念情報については、「第 1 章」および『*Sun Cluster* の概念 *(Solaris OS* 版*)*』マニュアルを参照してください。

▼ フェイルオーバーリソースグループを作成する

フェイルオーバーリソースグループは、ネットワークアドレス (組み込みリソースタイプの `LogicalHostname` や `SharedAddress` など) と、フェイルオーバーリソース (フェイルオーバーデータサービスのためのデータサービスアプリケーションリソースなど) を含みます。ネットワークリソースは、データサービスがフェイルオーバーまたはスイッチオーバーする場合に、依存するデータサービスリソースと共に、クラスタノード間を移動します。

追加情報については、`scrgadm`(1M) のマニュアルページを参照してください。

注 **–** この手順は、任意のクラスタノードから実行します。

---

1. クラスタメンバー上でスーパーユーザーになります。
2. フェイルオーバーリソースグループを作成します。

```
# scrgadm -a -g resource-group [-h nodelist]
```

`-a`	指定したリソースグループを追加します。
`-g` *resource-group*	追加するフェイルオーバーリソースグループの名前を指定します。任意の名前の先頭文字は ASCII にする必要があります。
`-h` *nodelist*	このリソースグループをマスターできるノードの順位リストを指定します (省略可能)。このリストを指定しない場合は、デフォルトでクラスタ内のすべてのノードになります。

3. リソースグループが作成されていることを確認します。

```
# scrgadm -pv -g resource-group
```

例 – フェイルオーバーリソースグループの作成

次に、2 つのノード (phys-schost-1、phys-schost-2) でマスターできるフェイルオーバーリソースグループ (resource-group-1) を追加する例を示します。

```
# scrgadm -a -g resource-group-1 -h phys-schost1,phys-schost-2
# scrgadm -pv -g resource-group-1
リソースグループ 名前:                                    resource-group-1
  (resource-group-1) リソースグループ RG_description:      <NULL>
  (resource-group-1) リソースグループ management state:    Unmanaged
  (resource-group-1) リソースグループ Failback:            False
  (resource-group-1) リソースグループ Nodelist:            phys-schost-1
                                                          phys-schost-2
  (resource-group-1) リソースグループ Maximum_primaries:   1
  (resource-group-1) リソースグループ Desired_primaries:   1
  (resource-group-1) リソースグループ RG_dependencies:     <NULL>
  (resource-group-1) リソースグループ mode:                Failover
  (resource-group-1) リソースグループ network dependencies: True
  (resource-group-1) リソースグループ Global_resources_used: All
  (resource-group-1) リソースグループ Pathprefix:
```

次に進む手順

フェイルオーバーリソースグループを作成した後で、そのリソースグループにアプリケーションリソースを追加できます。手順については、42 ページの「リソースグループへのリソースの追加」を参照してください。

▼ スケーラブルリソースグループを作成する

スケーラブルリソースグループは、スケーラブルサービスと共に使用されます。共有アドレス機能は、スケーラブルサービスの多数のインスタンスを 1 つのサービスとして扱える Sun Cluster のネットワーキング機能です。まず、スケーラブルリソースが依存する共有アドレスを含むフェイルオーバーリソースグループを作成しなければなりません。次にスケーラブルリソースグループを作成し、そのグループにスケーラブルリソースを追加します。

追加情報については、scrgadm(1M) のマニュアルページを参照してください。

注 – この手順は、任意のクラスタノードから実行します。

---

1. クラスタメンバー上でスーパーユーザーになります。

2. スケーラブルリソースが使用する共有アドレスを保持するフェイルオーバーリソースグループを作成します。

3. スケーラブルリソースグループを作成します。

```
# scrgadm -a -g resource-group \
-y Maximum_primaries=m \
-y Desired_primaries=n \
-y RG_dependencies=depend-resource-group \
[-h nodelist]
```

-a
スケーラブルリソースグループを追加します。

-g *resource-group*
追加するスケーラブルリソースグループの名前を指定します。

-y Maximum_primaries =*m*
このリソースグループのアクティブな主ノードの最大数を指定します。

-y Desired_primaries =*n*
リソースグループが起動するアクティブな主ノードの数を指定します。

-y RG_dependencies [ ] =*depend-resource-group*
作成されるリソースグループが依存する共有アドレスリソースを含むリソースグループを指定します。

-h *nodelist*
リソースグループを利用できるノードのリストを指定します (省略可能)。このリストを指定しない場合は、デフォルトですべてのノードになります。

4. スケーラブルリソースグループが作成されていることを確認します。

```
# scrgadm -pv -g resource-group
```

例 – スケーラブルリソースグループの作成

次に、2 つのノード (phys-schost-1、phys-schost-2) でホストされるスケーラブルリソースグループ (resource-group-1) を追加する例を示します。スケーラブルリソースグループは、共有アドレスを含むフェイルオーバーリソースグループ (resource-group-2) に依存します。

```
# scrgadm -a -g resource-group-1 \
-y Maximum_primaries=2 \
-y Desired_primaries=2 \
-y RG_dependencies=resource-group-2 \
-h phys-schost-1,phys-schost-2
# scrgadm -pv -g resource-group-1
リソースグループ 名前:                                          resource-group-1
  (resource-group-1) リソースグループ RG_description:          <NULL>
  (resource-group-1) リソースグループ management state:        Unmanaged
```

```
(resource-group-1) リソースグループ Failback:                 False
(resource-group-1) リソースグループ Nodelist:                 phys-schost-1
                                                    phys-schost-2
(resource-group-1) リソースグループ Maximum_primaries:        2
(resource-group-1) リソースグループ Desired_primaries:        2
(resource-group-1) リソースグループ RG_dependencies:          resource-group-2
(resource-group-1) リソースグループ mode:                     Scalable
(resource-group-1) リソースグループ network dependencies:     True
(resource-group-1) リソースグループ Global_resources_used:    All
(resource-group-1) リソースグループ Pathprefix:
```

次に進む手順

スケーラブルリソースグループを作成したあと、そのリソースグループにスケーラブルアプリケーションリソースを追加できます。詳細は、48 ページの「スケーラブルアプリケーションリソースをリソースグループに追加する」を参照してください。

リソースグループへのリソースの追加

リソースは、リソースタイプをインスタンス化したものです。リソースは、RGM によって管理される前に、リソースグループに追加する必要があります。この節では、3 種類のリソースタイプについて説明します。

- 論理ホスト名リソース。
- 共有アドレスリソース。
- データサービス (アプリケーション) リソース。

論理ホスト名リソースと共有アドレスリソースは、常にフェイルオーバーリソースグループに追加してください。フェイルオーバーデータサービス用のデータサービスリソースは、フェイルオーバーリソースグループに追加してください。フェイルオーバーリソースグループは、そのデータサービス用の論理ホスト名リソースとアプリケーションリソースの両方を含みます。スケーラブルリソースグループは、スケーラブルサービス用のアプリケーションリソースだけを含んでいます。スケーラブルサービスが依存する共有アドレスリソースは、別のフェイルオーバーリソースグループに存在する必要があります。データサービスをクラスタノード全体に渡って提供するには、スケーラブルアプリケーションリソースと共有アドレスリソース間の依存性を指定する必要があります。

リソースに関する詳細は、『*Sun Cluster* の概念 *(Solaris OS* 版*)*』マニュアルおよび第 1 章 を参照してください。

▼ 論理ホスト名リソースをリソースグループに追加する

この手順を実行するには、次の情報が必要になります。

- リソースを追加するフェイルオーバーリソースグループの名前
- リソースグループに追加するホスト名

注 – 論理ホスト名リソースをリソースグループに追加すると、リソースの拡張プロパティはデフォルト値に設定されます。デフォルト以外の値を指定するには、リソースをリソースグループに追加した後、そのリソースを変更する必要があります。詳細は、67 ページの「論理ホスト名リソースまたは共有アドレスリソースを変更する」を参照してください。

追加情報については、`scrgadm`(1M) のマニュアルページを参照してください。

注 – この手順は、任意のクラスタノードから実行します。

---

1. クラスタメンバー上でスーパーユーザーになります。
2. 論理ホスト名リソースをリソースグループに追加します。

```
# scrgadm -a -L [-j resource] -g resource-group -l hostnamelist, ... [-n netiflist]
```

`-a`	論理ホスト名リソースを追加します。
`-L`	論理ホスト名リソースの形式を指定します。
`-j` *resource*	リソース名を指定します (省略可能)。このオプションを指定しない場合は、デフォルトで `-l` オプションで最初に指定したホスト名になります。
`-g` *resource-group*	リソースを配置するリソースグループの名前を指定します。
`-l` *hostnamelist, …*	クライアントがリソースグループでサービスと通信する UNIX ホスト名 (論理ホスト名) をコマンドで区切って指定します。
`-n` *netiflist*	各ノード上の IP ネットワークマルチパス グループをコンマで区切って指定します (省略可能)。 *netiflist* の各要素は、`netif@node` の形式で指定する必要があります。`netif` は IP ネットワークマルチパス グループ名 (`sc_ipmp0` など) として指定できます。ノードは、`sc_ipmp0@1`、`sc_ipmp@phys-schost-1` などのノード名またはノード IDで特定できます。

注 – 現バージョンの Sun Cluster では、netif にアダプタ名を使用できません。

3. 論理ホスト名リソースが追加されていることを確認します。

   # **scrgadm -pv -j** *resource*

   リソースを追加すると、Sun Cluster ソフトウェアは、そのリソースの妥当性を検査します。妥当性が確認されると、そのリソースを有効にできるとともに、そのリソースグループを RGM の管理下に置くことが可能です。妥当性の検査に失敗すると、scrgadm コマンドはエラーメッセージを生成して終了します。妥当性の検査に失敗した場合は、エラーメッセージについて各ノード上の syslog を調べてください。メッセージは、妥当性の検査を実施したノードで表示されます。必ずしも scrgadm コマンドを実行したノードで表示されるわけではありません。

例 – 論理ホスト名リソースのリソースグループへの追加

次に、論理ホスト名リソース (resource-1) をリソースグループ (resource-group-1) に追加する例を示します。

```
# scrgadm -a -L -j resource-1 -g resource-group-1 -l schost-1
# scrgadm -pv -j resource-1
Res Group name: resource-group-1
(resource-group-1) リソース 名前:                                resource-1
  (resource-group-1:resource-1) リソース R_description:
  (resource-group-1:resource-1) リソース リソースタイプ:        SUNW.LogicalHostname
  (resource-group-1:resource-1) リソース リソースグループ名:  resource-group-1
  (resource-group-1:resource-1) リソース 有効:                False
  (resource-group-1:resource-1) リソース 有効なモニター:      True
```

次に進む手順

論理ホスト名リソースを追加したあと、51 ページの「リソースグループをオンラインにする」の手順に従って、このリソースをオンラインにします。

▼ 共有アドレスリソースをリソースグループに追加する

この手順を実行するには、次の情報が必要になります。

- リソースを追加するリソースグループの名前。このグループは、前の手順で作成したフェイルオーバーリソースグループでなければなりません。

- リソースグループに追加するホスト名。

注 – 共有アドレスリソースをリソースグループに追加すると、リソースの拡張プロパティはデフォルト値に設定されます。デフォルト以外の値を指定するには、リソースをリソースグループに追加した後、そのリソースを変更する必要があります。詳細は、67 ページの「論理ホスト名リソースまたは共有アドレスリソースを変更する」を参照してください。

追加情報については、`scrgadm`(1M) のマニュアルページを参照してください。

注 – この手順は、任意のクラスタノードから実行します。

---

1. クラスタメンバー上でスーパーユーザーになります。
2. 共有アドレスリソースをリソースグループに追加します。

```
# scrgadm -a -S [-j resource] -g resource-group -l hostnamelist, ... \
[-X auxnodelist] [-n netiflist]
```

`-a`	共有アドレスリソースを追加します。
`-S`	共有アドレスリソースの形式を指定します。
`-j` *resource*	リソース名を指定します (省略可能)。このオプションを指定しない場合は、デフォルトで `-l` オプションで最初に指定したホスト名になります。
`-g` *resource-group*	リソースグループの名前を指定します。
`-l` *hostnamelist, …*	共有アドレスホスト名をコンマで区切って指定します。
`-X` *auxnodelist*	共有アドレスをホストできるクラスタノード (ただし、フェイルオーバー時に稼動系として使用されない) を識別する物理ノード名または ID をコンマで区切って指定します。これらのノードは、リソースグループのノードリストで潜在的マスターとして識別されるノードと相互に排他的です。
`-n` *netiflist*	各ノード上の IP ネットワークマルチパス グループをコンマで区切って指定します (省略可能)。 *netiflist* の各要素は、`netif@node` の形式で指定する必要があります。`netif` は IP ネットワークマルチパス グループ名 (`sc_ipmp0` など) として指定できます。ノードは、`sc_ipmp0@1`、`sc_ipmp@phys-schost-1` などのノード名またはノード IDで特定できます。

注 – 現バージョンの Sun Cluster では、netif にアダプタ名を使用できません。

3. 共有アドレスリソースが追加され、妥当性が検査されていることを確認します。

   # **scrgadm -pv -j** *resource*

   リソースを追加すると、Sun Cluster ソフトウェアは、そのリソースの妥当性を検査します。妥当性が確認されると、そのリソースを有効にできるとともに、そのリソースグループを RGM の管理下に置くことが可能です。妥当性の検査に失敗すると、scrgadm コマンドはエラーメッセージを生成して終了します。妥当性の検査に失敗した場合は、エラーメッセージについて各ノード上の syslog を調べてください。メッセージは、妥当性の検査を実施したノードで表示されます。必ずしも scrgadm コマンドを実行したノードで表示されるわけではありません。

例 – 共有アドレスリソースのリソースグループへの追加

次に、共有アドレスリソース (resource-1) をリソースグループ (resource-group-1) に追加する例を示します。

```
# scrgadm -a -S -j resource-1 -g resource-group-1 -l schost-1
# scrgadm -pv -j resource-1
(resource-group-1) リソース 名前:                                    resource-1
    (resource-group-1:resource-1) リソース R_description:
    (resource-group-1:resource-1) リソース リソースタイプ:          SUNW.SharedAddress
    (resource-group-1:resource-1) リソース リソースグループ名:  resource-group-1
    (resource-group-1:resource-1) リソース 有効:                 False
    (resource-group-1:resource-1) リソース 有効なモニター:       True
```

次に進む手順

共有リソースを追加したあと、51 ページの「リソースグループをオンラインにする」の手順に従ってリソースを有効にします。

▼ フェイルオーバーアプリケーションリソースをリソースグループに追加する

フェイルオーバーアプリケーションリソースは、以前にフェイルオーバーリソースグループに作成した論理ホスト名を使用するアプリケーションリソースです。

この手順を実行するには、次の情報が必要になります。

- リソースを追加するフェイルオーバーリソースグループの名前

- リソースが属するリソースタイプの名前
- アプリケーションリソースが使用する論理ホスト名リソース。これは、以前に同じリソースグループに含めた論理ホスト名になります。

追加情報については、scrgadm(1M) のマニュアルページを参照してください。

注 – この手順は、任意のクラスタノードから実行します。

---

1. クラスタメンバー上でスーパーユーザーになります。
2. フェイルオーバーアプリケーションリソースをリソースグループに追加します。

```
# scrgadm -a -j resource -g resource-group -t resource-type \
[-x Extension_property=value, ...] [-y Standard_property=value, ...]
```

-a	リソースを追加します。
-j *resource*	追加するリソースの名前を指定します。
-g *resource-group*	以前に作成したフェイルオーバーリソースグループの名前を指定します。
-t *resource-type*	リソースが属するリソースタイプの名前を指定します。
-x *Extension_property =value, …*	特定のデータサービスに依存する拡張プロパティをコンマで区切って指定します。データサービスがこのプロパティの指定が必要かどうかについては、各データサービスのマニュアルを参照してください。
-y *Standard_property =value, …*	特定のデータサービスに依存する標準プロパティをコンマで区切って指定します。データサービスがこのプロパティの指定が必要かどうかについては、各データサービスのマニュアルと付録 A を参照してください。

注 – 別のプロパティを設定することもできます。詳細は、付録 A とフェイルオーバーデータサービスのインストールと構成に関するマニュアルを参照してください。

---

3. フェイルオーバーアプリケーションリソースが追加され、妥当性が検査されていることを確認します。

```
# scrgadm -pv -j resource
```

リソースを追加すると、Sun Cluster ソフトウェアは、そのリソースの妥当性を検査します。妥当性が確認されると、そのリソースを有効にできるとともに、そのリソースグループを RGM の管理下に置くことが可能です。妥当性の検査に失敗すると、scrgadm コマンドはエラーメッセージを生成して終了します。妥当性の検査に失敗した場合は、エラーメッセージについて各ノード上の syslog を調べてください。メッセージは、妥当性の検査を実施したノードで表示されます。必ずしも scrgadm コマンドを実行したノードで表示されるわけではありません。

例 – フェイルオーバーアプリケーションリソースのリソースグループへの追加

次に、リソース (resource-1) をリソースグループ (resource-group-1) に追加する例を示します。リソースは、以前に定義したフェイルオーバーリソースグループと同じリソースグループに存在している論理ホスト名リソース (schost-1、schost-2) に依存しています。

```
# scrgadm -a -j resource-1 -g resource-group-1 -t resource-type-1 \
-y Network_resources_used=schost-1,schost2 \
# scrgadm -pv -j resource-1
(resource-group-1) リソース 名前:                                  resource-1
    (resource-group-1:resource-1) リソース  R_description:
    (resource-group-1:resource-1) リソース  リソースタイプ:        resource-type-1
    (resource-group-1:resource-1) リソース  リソースグループ名:  resource-group-1
    (resource-group-1:resource-1) リソース  有効:                 False
    (resource-group-1:resource-1) リソース  有効なモニター:       True
```

次に進む手順

フェイルオーバーアプリケーションリソースを追加したあと、51 ページの「リソースグループをオンラインにする」の手順に従ってリソースを有効にします。

▼ スケーラブルアプリケーションリソースをリソースグループに追加する

スケーラブルアプリケーションリソースは、フェイルオーバーリソースグループに共有アドレスを使用するアプリケーションリソースです。

この手順を実行するには、次の情報が必要になります。

- リソースを追加するスケーラブルリソースグループの名前
- リソースが属するリソースタイプの名前
- スケーラブルサービスリソースが使用する共有アドレスリソース。これは、以前にフェイルオーバーリソースグループに含めた共有アドレスになります。

追加情報については、scrgadm(1M) のマニュアルページを参照してください。

注 – この手順は、任意のクラスタノードから実行します。

---

1. クラスタメンバー上でスーパーユーザーになります。
2. スケーラブルアプリケーションリソースをリソースグループに追加します。

```
# scrgadm -a -j resource -g resource-group -t resource-type \
-y Network_resources_used=network-resource[,network-resource...] \
-y Scalable=True
[-x Extension_property=value, ...] [-y Standard_property=value, ...]
```

-a
リソースを追加します。

-j *resource*
追加するリソースの名前を指定します。

-g *resource-group*
以前に作成したスケーラブルサービスリソースグループの名前を指定します。

-t *resource-type*
このリソースが属するリソースタイプの名前を指定します。

-y Network_resources_used = *network-resource*[,*network-resource* ...]
このリソースが依存するネットワークリソース (共有アドレス) のリストを指定します。

-y Scalable[ ] =True
このリソースがスケーラブルであることを指定します。

-x *Extension_property =value, …*
特定のデータサービスに依存する拡張プロパティをコンマで区切って指定します。データサービスがこのプロパティの指定が必要かどうかについては、各データサービスのマニュアルを参照してください。

-y *Standard_property =value, …*
特定のデータサービスに依存する標準プロパティをコンマで区切って指定します。データサービスがこのプロパティの指定が必要かどうかについては、各データサービスのマニュアルと付録 A を参照してください。

-y *Standard_property =value, …*
特定のデータサービスに依存する標準プロパティをコンマで区切って指定します。データサービスがこのプロパティの指定が必要かどうかについては、各データサービスのマニュアルと付録 A を参照してください。

注 – 別のプロパティを設定することもできます。構成可能なほかのプロパティについては、付録 A とスケーラブルデータサービスのインストールと構成に関するマニュアルを参照してください。スケーラブルサービスの場合は、通常、Port_list、Load_balancing_weights、Load_balancing_policy プロパティを設定します (付録 A を参照)。

3. スケーラブルアプリケーションリソースが追加され、妥当性が検査されていることを確認します。

   # **scrgadm -pv -j** *resource*

   リソースを追加すると、Sun Cluster ソフトウェアは、そのリソースの妥当性を検査します。妥当性が確認されると、そのリソースを有効にできるとともに、そのリソースグループを RGM の管理下に置くことが可能です。妥当性の検査に失敗すると、scrgadm コマンドはエラーメッセージを生成して終了します。妥当性の検査に失敗した場合は、エラーメッセージについて各ノード上の syslog を調べてください。メッセージは、妥当性の検査を実施したノードで表示されます。必ずしも scrgadm コマンドを実行したノードで表示されるわけではありません。

例 – スケーラブルアプリケーションリソースのリソースグループへの追加

次に、リソース (resource-1) をリソースグループ (resource-group-1) に追加する例を示します。resource-group-1 は、使用されているネットワークアドレス (以下の例の schost-1 と schost-2) を含むフェイルオーバーリソースグループに依存することに注意してください。リソースは、共有アドレスリソース (schost-1 と schost-2) に依存し、以前に定義した 1 つまたは複数のフェイルオーバーリソースグループに存在する必要があります。

```
# scrgadm -a -j resource-1 -g resource-group-1 -t resource-type-1 \
-y Network_resources_used=schost-1,schost-2 \
-y Scalable=True
# scrgadm -pv -j resource-1
(resource-group-1) リソース 名前:                                    resource-1
    (resource-group-1:resource-1) リソース R_description:
    (resource-group-1:resource-1) リソース リソースタイプ:         resource-type-1
    (resource-group-1:resource-1) リソース リソースグループ名:  resource-group-1
    (resource-group-1:resource-1) リソース 有効:                 False
    (resource-group-1:resource-1) リソース 有効なモニター:       True
```

次に進む手順

スケーラブルアプリケーションリソースを追加したあと、51 ページの「リソースグループをオンラインにする」の手順に従って、リソースを有効にします。

リソースグループをオンラインにする

リソースが HA サービスの提供を開始できるようにするには、リソースグループのリソースおよびリソースモニターを有効にし、リソースグループを管理状態にし、リソースグループをオンラインにする必要があります。これらの作業は各々実行できますが、次に示すように 1 つの手順で実行することもできます。詳細は、`scswitch` (1M) のマニュアルページを参照してください。

注 – この手順は、任意のクラスタノードから実行します。

▼ リソースグループをオンラインにする

1. クラスタメンバー上でスーパーユーザーになります。
2. リソースを有効にして、リソースグループをオンラインにします。

   `# scswitch -Z -g` *rg-list*

   リソースモニターを無効にしている場合は、これらも有効になります。

   注 – 意図的にリソース (または、障害) モニターを無効にしており、これ以降も無効にしておく場合は、`-Z` オプションではなく、`-z` オプションを指定します。

   `-Z` まず、リソースグループのリソースと障害モニターを有効にすることによって、リソースグループをオンラインにします。

   `-g` *rg-list* オンラインにするリソースグループの名前をコンマで区切って指定します。これらのリソースグループは存在する必要があります。このリストには、1 つまたは複数のリソースグループ名を指定できます。

   `-g` *rg-list* オプションは省略できます。このオプションを省略した場合、すべてのリソースグループがオンラインになります。

   注 – オンラインにしようとしている任意のリソースグループがほかのリソースグループに対して強いアフィニティを宣言している場合、この操作は失敗します。詳細は、98 ページの「オンラインのリソースグループをクラスタノード間で分散する」を参照してください。

   ---

3. リソースがオンラインになっていることを確認します。

任意のクラスタノードで次のコマンドを実行し、Resource Group State のフィールドを調べ、ノードリストで指定されたノードで各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。

```
# scstat -g
```

例 – リソースグループをオンラインにする

次に、リソースグループ (resource-group-1) をオンラインにし、その状態を確認する例を示します。

```
# scswitch -Z -g resource-group-1
# scstat -g
```

次に進む手順

リソースグループがオンラインになれば、リソースグループが構成されて使用する準備が整ったことになります。リソースやノードで障害が発生した場合は、RGM は別のノードでそのリソースグループをオンラインに切り替えることでリソースグループの可用性を維持します。

リソースモニターの無効化と有効化

次の各手順では、リソース自体とは関係なくリソースフォルトモニターだけを無効または有効にします。したがって、フォルトモニターが無効にされても、そのリソース自体は正常に動作を続けます。ただし、フォルトモニターが無効になっていると、データサービスに障害が発生しても、障害回復は自動的には開始されません。

追加情報については、scswitch(1M) のマニュアルページを参照してください。

注 – この手順は任意のノードから実行できます。

▼ リソース障害モニターを無効にする

1. クラスタメンバー上でスーパーユーザーになります。
2. リソース障害モニターを無効にします。

   ```
   # scswitch -n -M -j resource
   ```

   -n　　リソースまたはリソースモニターを無効にします。

   -M　　指定されたリソースのフォルトモニターを無効にします。

   -j *resource*　　リソースの名前

3. リソースフォルトモニターが無効になっていることを確認します。
   各クラスタノードで次のコマンドを実行し、監視されるフィールド (RS Monitored) を確認します。

   ```
   # scrgadm -pv
   ```

例–リソース障害モニターを無効にする

この例では、リソースフォルトモニターを無効にします。

```
# scswitch -n -M -j resource-1
# scrgadm -pv
...
RS Monitored: no...
```

▼ リソース障害モニターを有効にする

1. クラスタメンバー上でスーパーユーザーになります。
2. リソースフォルトモニターを有効にします。

   ```
   # scswitch -e -M -j resource
   ```

   -e　　リソースまたはリソースモニターを有効にします。

   -M　　指定されたリソースの障害モニターを有効にします。

   -j *resource*　　リソースの名前を指定します。

3. リソース障害モニターが有効になっていることを確認します。
   各クラスタノードで次のコマンドを実行し、監視されるフィールド (RS Monitored) を確認します。

   ```
   # scrgadm -pv
   ```

例–リソース障害モニターを有効にする

この例では、リソースフォルトモニターを有効にします。

```
# scswitch -e -M -j resource-1
# scrgadm -pv
...
RS Monitored: yes...
```

リソースタイプの削除

使用されていないリソースタイプを削除する必要はありませんが、次の手順を使用して削除できます。

追加情報については、scrgadm(1M) および scswitch(1M) のマニュアルページを参照してください。

注 – この手順は、任意のクラスタノードから実行します。

▼ リソースタイプを削除する

リソースタイプを削除する前に、クラスタ内のすべてのリソースグループにある、そのタイプのリソースをすべて無効にし、削除する必要があります。scrgadm -pv コマンドを使用し、クラスタ内のリソースとリソースグループを確認します。

1. クラスタメンバー上でスーパーユーザーになります。
2. 削除するリソースタイプの各リソースを無効にします。

   # **scswitch -n -j** *resource*

   -n　　リソースを無効にします。

   -j *resource*　　無効にするリソースの名前を指定します。
3. 削除するリソースタイプの各リソースを削除します。

   # **scrgadm** -**r** -**j** *resource*

   -r　　指定したリソースを削除します。

`-j` 削除するリソースの名前を指定します。

4. リソースタイプを削除します。

```
# scrgadm -r -t resource-type
```

`-r` 指定したリソースタイプを削除します。

`-t resource-type` 削除するリソースタイプの名前を指定します。

5. リソースタイプが削除されていることを確認します。

```
# scrgadm -p
```

例 – リソースタイプの削除

次に、リソースタイプのすべてのリソース (`resource-type-1`) を無効にして削除したあとで、そのリソースタイプ自体を削除する例を示します。この例では、`resource-1` は、リソースタイプ `resource-type-1` のリソースです。

```
# scswitch -n -j resource-1
# scrgadm -r -j resource-1
# scrgadm -r -t resource-type-1
```

リソースグループの削除

リソースグループを削除するには、最初に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必要があります。

追加情報については、`scrgadm`(1M) および `scswitch`(1M) のマニュアルページを参照してください。

注 – この手順は、任意のクラスタノードから実行します。

▼ リソースグループを削除する

1. クラスタメンバー上でスーパーユーザーになります。

2. 次のコマンドを実行し、リソースグループをオフラインに切り替えます。

```
# scswitch -F -g resource-group
```

`-F` リソースグループをオフラインに切り替えます。

`-g` *resource-group* オフラインにするリソースグループの名前を指定します。

3. リソースグループに含まれているすべてのリソースを無効にします。

   `scrgadm -pv` コマンドを使用し、リソースグループ内のリソースを表示できます。リソースグループ内の削除するすべてのリソースを無効にします。

   ```
   # scswitch -n -j resource
   ```

   `-n` リソースを無効にします。

   `-j` *resource* 無効にするリソースの名前を指定します。

   依存性のあるデータサービスリソースがリソースグループに存在する場合、そのリソースを無効にするには、依存するすべてのリソースを無効にする必要があります。

4. リソースグループからすべてのリソースを削除します。

   `scrgadm` コマンドを使用して次の操作を行います。

   - リソースの削除
   - リソースグループの削除

   ```
   # scrgadm -r -j resource
   # scrgadm -r -g resource-group
   ```

   `-r` 指定したリソースやリソースグループを削除します。

   `-j` *resource* 削除するリソースの名前を指定します。

   `-g` *resource-group* 削除するリソースグループの名前を指定します。

5. リソースグループが削除されていることを確認します。

   ```
   # scrgadm -p
   ```

例 — リソースグループの削除

次に、リソースグループ (`resource-group-1`) のリソース (`resource-1`) を削除したあとで、そのリソースグループ自体を削除する例を示します。

```
# scswitch -F -g resource-group-1
# scrgadm -r -j resource-1
# scrgadm -r -g resource-group-1
```

リソースの削除

リソースグループからリソースを削除する前に、そのリソースを無効にします。

追加情報については、scrgadm(1M) および scswitch(1M) のマニュアルページを参照してください。

注 – この手順は、任意のクラスタノードから実行します。

▼ リソースを削除する

1. クラスタメンバー上でスーパーユーザーになります。
2. 削除するリソースを無効にします。

   # **scswitch -n -j** *resource*

   -n　　リソースを無効にします。

   -j *resource*　　無効にするリソースの名前を指定します。
3. リソースを削除します。

   # **scrgadm -r -j** *resource*

   -r　　指定したリソースを削除します。

   -j *resource*　　削除するリソースの名前を指定します。
4. リソースが削除されていることを確認します。

   # **scrgadm -p**

例 – リソースの削除

次に、リソース resource-1 を無効にして削除する例を示します。

```
# scswitch -n -j resource-1
# scrgadm -r -j resource-1
```

リソースグループの主ノードの切り替え

以下の手順を使用し、リソースグループの現在の主ノードを別のノードに切り替え (スイッチオーバー)、新しい主ノードにすることができます。

追加情報については、`scrgadm`(1M) および `scswitch`(1M) のマニュアルページを参照してください。

注 – この手順は、任意のクラスタノードから実行します。

▼ リソースグループの主ノードを切り替える

この手順を実行するには、次の情報が必要になります。

- スイッチオーバーするリソースグループの名前
- リソースグループをオンラインにする、またはオンラインを維持するノードの名前。スイッチオーバーを行うリソースグループの、待機系として設定されているクラスタノードを指定する必要があります。リソースグループの潜在的主ノードの一覧を表示するには、`scrgadm -pv` コマンドを使用します。

1. クラスタメンバー上でスーパーユーザーになります。
2. 稼動系を待機系に切り替えます。

```
# scswitch -z -g resource-group -h nodelist
```

`-z`	指定したリソースグループをオンラインに切り替えます。
`-g` *resource-group*	切り替えるリソースグループの名前を指定します。
`-h` *nodelist*	リソースグループをオンラインにするか、オンラインのままにしておくノードの名前をコンマで区切って指定します。このリストには、1 つまたは複数のノード名を指定できます。このリソースグループは、このノード以外のすべてのノードでオフラインに切り替えられます。

注 – 切り替えようとしている任意のリソースグループが他のリソースグループに対して強いアフィニティを宣言している場合、その操作は失敗するか、委託されます。詳細は、98 ページの「オンラインのリソースグループをクラスタノード間で分散する」を参照してください。

---

3. リソースグループが新しい稼動系に切り替えられていることを確認します。

   次のコマンドを実行し、スイッチオーバーされたリソースグループの状態に関する出力を調べます。

   ```
   # scstat -g
   ```

例 – リソースグループを新しい主ノードに切り替える

次に、リソースグループ (resource-group-1) を現在の主ノード (phys-schost-1) から、潜在的主ノード (phys-schost-2) へ切り替える例を示します。まず、リソースグループが phys-schost-1 でオンラインになっていることを確認します。続いて、切り替えを行います。最後に、そのグループが phys-schost-2 でオンラインに切り替えられたことを確認します。

```
phys-schost-1# scstat -g
...
Resource Group Name:          resource-group-1
  Status
    Node Name:                phys-schost-1
    Status:                   Online

    Node Name:                phys-schost-2
    Status:                   Offline
...
phys-schost-1# scswitch -z -g resource-group-1 -h phys-schost-2
phys-schost-1# scstat -g
...
Resource Group Name:          resource-group-1
  Status
    Node Name:                phys-schost-2
    Status:                   Online

    Node Name:                phys-schost-1
    Status:                   Offline
...
```

リソースの無効化とリソースグループの UNMANAGED 状態への移行

リソースグループは、そのリソースグループに対して管理手順を実施する前に、UNMANAGED 状態に移行する必要があります。リソースグループを UNMANAGED 状態に移行する前に、リソースグループに含まれるすべてのリソースを無効にし、リソースグループをオフラインにする必要があります。

追加情報については、scrgadm(1M) および scswitch(1M) のマニュアルページを参照してください。

注 – この手順は、任意のクラスタノードから実行します。

▼ リソースを無効にしてリソースグループを非管理状態に移行する

この手順を実行するには、次の情報が必要になります。

- 無効にするリソースの名前
- UNMANAGED 状態に移行するリソースグループの名前

この手順に必要なリソースとリソースグループの名前を判断するには、scrgadm -pv コマンドを使用します。

注 – 共有アドレスリソースを無効にした後でも、そのリソースはいくつかのホストからの ping(1M) コマンドに応答することがあります。無効にした共有アドレスリソースが ping コマンドに応答しないようにするには、そのリソースのリソースグループを UNMANAGED 状態にする必要があります。

---

1. クラスタメンバー上でスーパーユーザーになります。
2. リソースを無効にします。
   この手順を、リソースグループ内のすべてのリソースに対して実行します。

   # **scswitch -n -j** *resource*

   -n　　リソースを無効にします。

   -j *resource*　　無効にするリソースの名前を指定します。
3. 次のコマンドを実行し、リソースグループをオフラインに切り替えます。

   # **scswitch** -**F** -**g** *resource-group*

   -F　　リソースグループをオフラインに切り替えます。

   -g *resource-group*　　オフラインにするリソースグループの名前を指定します。
4. リソースグループを非管理状態にします。

   # **scswitch -u -g** *resource-group*

   -u　　指定したリソースグループを非管理状態にします。

-g *resource-group*　UNMANAGED 状態にするリソースグループの名前を指定します。

5. リソースが無効になり、リソースグループが **UNMANAGED** 状態になっていることを確認します。

```
# scrgadm -pv -g resource-group
```

例 – リソースの無効化とリソースグループの非管理状態への移行

次に、リソース (resource-1) を無効にし、リソースグループ (resource-group-1) を非管理状態に移行する例を示します。

```
# scswitch -n -j resource-1
# scswitch -F -g resource-group-1
# scswitch -u -g resource-group-1
# scrgadm -pv -g resource-group-1
リソースグループ 名前:                                          resource-group-1
  (resource-group-1) リソースグループ RG_description:            <NULL>
  (resource-group-1) リソースグループ management state:          Unmanaged
  (resource-group-1) リソースグループ Failback:                  False
  (resource-group-1) リソースグループ Nodelist:                  phys-schost-1
                                                               phys-schost-2
  (resource-group-1) リソースグループ Maximum_primaries:         2
  (resource-group-1) リソースグループ Desired_primaries:         2
  (resource-group-1) リソースグループ RG_dependencies:           <NULL>
  (resource-group-1) リソースグループ mode:                      Failover
  (resource-group-1) リソースグループ network dependencies:      True
  (resource-group-1) リソースグループ Global_resources_used:     All
  (resource-group-1) リソースグループ Pathprefix:

  (resource-group-1) リソース 名前:                               resource-1
    (resource-group-1:resource-1) リソース R_description:
    (resource-group-1:resource-1) リソース リソースタイプ:        SUNW.apache
    (resource-group-1:resource-1) リソース リソースグループ名:    resource-group-1
    (resource-group-1:resource-1) リソース 有効:                True
    (resource-group-1:resource-1) リソース 有効なモニター:      False
    (resource-group-1:resource-1) リソース detached:              False
```

リソースタイプ、リソースグループ、リソース構成情報の表示

リソース、リソースグループ、リソースタイプで管理手順を実施する前に、この手順を使用し、これらのオブジェクトの現在の構成設定を表示します。

追加情報については、scrgadm(1M) および scswitch(1M) のマニュアルページを参照してください。

注 – この手順は、任意のクラスタノードから実行します。

リソースタイプ、リソースグループ、リソース構成情報の表示

scrgadm コマンドは、構成状態に関する次の 3 つのレベルの情報を表示します。

- --p オプションを指定した場合は、リソースタイプ、リソースグループ、リソースのプロパティ値に関する最小限の情報が表示されます。
- -pv オプションを指定した場合は、ほかのリソースタイプ、リソースグループ、リソースプロパティに関する詳細が表示されます。
- -pvv オプションを指定した場合は、リソースタイプメソッド、拡張プロパティ、すべてのリソースとリソースグループのプロパティを含む、詳細情報が表示されます。

また、表示したいオブジェクトの名前の後に -t (リソースタイプ)、-g (リソースグループ)、および -j (リソース) オプションを指定することによって、特定のリソースタイプ、リソースグループ、またはリソースのステータス情報を確認できます。たとえば、次のコマンドは、リソース apache-1 のみについて、特定の情報を表示することを指定します。

```
# scrgadm -p[v[v]] -j apache-1
```

詳細については、scrgadm(1M) のマニュアルページを参照してください。

リソースタイプ、リソースグループ、リソースプロパティの変更

Sun Cluster は、リソースタイプ、リソースグループ、およびリソースを構成するための標準プロパティを定義します。これらの標準プロパティについては、次の節を参照してください。

- 119 ページの「リソースタイププロパティ」
- 126 ページの「リソースのプロパティ」
- 137 ページの「リソースグループのプロパティ」

また、リソースには、リソースを表現するデータサービスの拡張プロパティも事前定義されています。データサービスの拡張プロパティについては、データサービスのマニュアルを参照してください。

プロパティを変更できるかどうかを判断するには、そのプロパティの説明において、プロパティの調整エントリを参照してください。

次の手順に、リソースタイプ、リソースグループ、およびリソースを構成するためのプロパティを変更する方法について説明します。

▼ リソースタイププロパティを変更する

この手順を実行するには、次の情報が必要になります。

- 変更するリソースタイプの名前
- 変更するリソースタイププロパティの名前。リソースタイプの場合、特定のプロパティだけを変更できます。プロパティを変更できるかどうかを判断するには、119 ページの「リソースタイププロパティ」において、プロパティの調整エントリを参照してください。

注 – `Installed_nodes` プロパティは明示的には変更できません。このプロパティを変更するには、`scrgadm` コマンドの `-h` *installed-node-list* オプションを指定します。

注 – この手順は、任意のクラスタノードから実行します。

---

1. クラスタメンバー上でスーパーユーザーになります。

2. **scrgadm** コマンドを使用し、この手順に必要なリソースタイプの名前を判断します。

   ```
   # scrgadm -pv
   ```

3. リソースタイププロパティを変更します。

   リソースタイプの場合、特定のプロパティだけを変更できます。プロパティを変更できるかどうかを判断するには、119 ページの「リソースタイププロパティ」において、プロパティの調整エントリを参照してください。

```
# scrgadm -c -t resource-type [-h installed-node-list] [-y property=new-value]
```

`-c`	指定したリソースタイププロパティを変更します。
`-t` *resource-type*	リソースタイプの名前を指定します。
`-h` *installed-node-list*	このリソースタイプがインストールされるノードの名前を指定します。
`-y` *property* `=`*new-value*	変更する標準プロパティの名前と、そのプロパティの新しい値を指定します。 `Installed_nodes` プロパティは明示的には変更できません。このプロパティを変更するには、`scrgadm` コマンドの `-h` *installed-node-list* オプションを指定します。

4. リソースタイププロパティが変更されていることを確認します。

   ```
   # scrgadm -pv -t resource-type
   ```

例 – リソースタイププロパティの変更

次に、`SUNW.apache` プロパティを変更し、このリソースタイプが 2 つのノード (`phys-schost-1` および `phys-schost-2`) にインストールされるように定義する例を示します。

```
# scrgadm -c -t SUNW.apache -h phys-schost-1,phys-schost-2
# scrgadm -pv -t SUNW.apache
リソースタイプ 名前:                                    SUNW.apache
  (SUNW.apache) リソースタイプ 説明:                    Apache Resource Type
  (SUNW.apache) リソースタイプ ベースディレクトリ:      /opt/SUNWscapc/bin
  (SUNW.apache) リソースタイプ 単一のインスタンス:      False
  (SUNW.apache) リソースタイプ 初期ノード:              All potential masters
  (SUNW.apache) リソースタイプ フェイルオーバー:        False
  (SUNW.apache) リソースタイプ バージョン:              1.0
  (SUNW.apache) リソースタイプ API バージョン:          2
  (SUNW.apache) リソースタイプ ノードにインストールされている: phys-schost1 phys-schost-2
  (SUNW.apache) リソースタイプ パッケージ:              SUNWscapc
```

▼ リソースグループプロパティを変更する

この手順を実行するには、次の情報が必要になります。

- 変更するリソースグループの名前
- 変更するリソースグループプロパティの名前とその新しいプロパティ値

この手順では、リソースグループプロパティの変更方法について説明しています。リソースグループプロパティの一覧については、付録 A を参照してください。

注 – この手順は、任意のクラスタノードから実行します。

---

1. クラスタメンバー上でスーパーユーザーになります。
2. リソースグループプロパティを変更します。

   # **scrgadm** -**c** -**g** *resource-group* -**y** *property*=*new_value*

-c	指定したプロパティを変更します。
-g *resource-group*	リソースグループの名前を指定します。
-y *property*	変更するプロパティの名前を指定します。

3. リソースグループプロパティが変更されていることを確認します。

   # **scrgadm -pv -g** *resource-group*

例 – リソースグループプロパティの変更

次に、リソースグループ (resource-group-1) の Failback プロパティを変更する例を示します。

```
# scrgadm -c -g resource-group-1 -y Failback=True
# scrgadm -pv -g resource-group-1
```

▼ リソースプロパティを変更する

この手順を実行するには、次の情報が必要になります。

- 変更するプロパティを持つリソースの名前
- 変更するプロパティの名前

この手順は、リソースプロパティの変更方法について説明しています。リソースグループプロパティの一覧については、付録 A を参照してください。

注 – この手順は、任意のクラスタノードから実行します。

1. クラスタメンバー上でスーパーユーザーになります。
2. **scrgadm -pvv** コマンドを実行し、現在のリソースプロパティ設定を表示します。

   # **scrgadm -pvv -j** *resource*

3. リソースプロパティを変更します。

scrgadm -c -j *resource* **-y** *property*=*new_value* | **-x** *extension_property*=*new_value*

-c	指定したプロパティを変更します。
-j *resource*	リソースの名前を指定します。
-y *property* =*new_value*	変更する標準プロパティの名前を指定します。
-x *extension_property* =*new_value*	変更する拡張プロパティの名前を指定します。データサービスの拡張プロパティについては、データサービスのマニュアルを参照してください。

4. リソースプロパティが変更されていることを確認します。

   # **scrgadm pvv -j** *resource*

例 – 標準リソースプロパティの変更

次に、リソース (resource-1) のシステム定義プロパティ (Start_timeout) の変更例を示します。

```
# scrgadm -c -j resource-1 -y start_timeout=30
# scrgadm -pvv -j resource-1
```

例 – 拡張リソースプロパティの変更

次に、リソース (resource-1) の拡張プロパティ (Log_level) の変更例を示します。

```
# scrgadm -c -j resource-1 -x Log_level=3
# scrgadm -pvv -j resource-1
```

▼ 論理ホスト名リソースまたは共有アドレスリソースを変更する

デフォルトでは、論理ホスト名リソースと共有アドレスリソースは名前解決にネームサービスを使用します。 同じクラスタ上で動作するネームサービスを使用するようにクラスタを構成することも可能です。論理ホスト名リソースまたは共有アドレスリソースがフェイルオーバーされると、そのクラスタ上で動作しているネームサービスもフェイルオーバーされます。論理ホスト名リソースまたは共有アドレスリソースが使用するネームサービスがフェイルオーバーしている場合、このリソースはフェイルオーバーできません。

注 – 同じクラスタ上で動作しているネームサービスを使用するようにクラスタを構成すると、そのクラスタ上のほかのサービスの可用性を損なう可能性があります。

このようなフェイルオーバーの失敗を防ぐには、ネームサービスをバイパスするように論理ホスト名リソースまたは共有アドレスリソースを変更します。ネームサービスをバイパスするようにリソースを変更するには、リソースの `CheckNameService` 拡張プロパティを `false` に設定します。`CheckNameService` プロパティはいつでも変更できます。

注 – リソースタイプのバージョンが 2 より前の場合、リソースを変更する前に、まず、リソースタイプをアップグレードする必要があります。詳細は、69 ページの「事前登録されているリソースタイプのアップグレード」を参照してください。

---

1. クラスタメンバー上でスーパーユーザーになります。
2. リソースプロパティを変更します。

```
# scrgadm -c -j resource -x CheckNameService=false
```

`-j` *resource*	変更する論理ホスト名リソースまたは共有アドレスリソースの名前を指定します。
`-y CheckNameService=false`	リソースの `CheckNameService` 拡張プロパティを `false` に設定します。

リソースの STOP_FAILED エラーフラグの消去

Failover_mode リソースプロパティが NONE または SOFT に設定されているときに、リソースの STOP に失敗した場合は、個々のリソースは STOP_FAILED 状態になり、リソースグループは ERROR_STOP_FAILED 状態になります。この状態のリソースグループは、ノード上でオンラインにできません。また、リソースの作成や削除、リソースグループやリソースプロパティの変更などの編集操作を行うこともできません。

▼ リソースの STOP_FAILED エラーフラグを消去する

この手順を実行するには、次の情報が必要になります。

- リソースが STOP_FAILED であるノードの名前
- STOP_FAILED 状態になっているリソースとリソースグループの名前

追加情報については、scswitch(1M) のマニュアルページを参照してください。

注 – この手順は、任意のクラスタノードから実行します。

---

1. クラスタメンバー上でスーパーユーザーになります。
2. **STOP_FAILED** 状態のリソースと、どのノードでこの状態なのかを確認します。

   ```
   # scstat -g
   ```

3. **STOP_FAILED** 状態になっているノード上で、リソースとそのモニターを手作業で停止します。

   この手順では、プロセスを強制終了するか、リソースタイプ固有のコマンドまたは別のコマンドを実行する必要があります。

4. リソースを手作業で停止したすべてのノード上で、これらのリソースの状態を手作業で **OFFLINE** に設定します。

   ```
   # scswitch -c -h nodelist -j resource -f STOP_FAILED
   ```

   -c フラグを消去します。

-h *nodelist*　リソースが STOP_FAILED 状態であるノードの名前をコンマで区切って指定します。このリストには、1 つまたは複数のノード名を指定できます。

-j *resource*　オフラインにするリソースの名前を指定します。

-f STOP_FAILED　フラグ名を指定します。

5. 手順 4 で **STOP_FAILED** フラグを消去したノード上で、リソースグループの状態を調べます。

   リソースグループの状態は、OFFLINE または ONLINE になっています。

   ```
   # scstat -g
   ```

   コマンド scstat -g は、リソースグループの状態が ERROR_STOP_FAILED のままかを示します。リソースグループがまだ ERROR_STOP_FAILED 状態の場合は、scswitch コマンドを実行し、該当するノード上でリソースグループをオフラインに切り替えます。

   ```
   # scswitch -F -g resource-group
   ```

   -F　グループをマスターできるすべてのノード上でリソースグループをオフラインにします。

   -g *resource-group*　オフラインに切り替えるリソースグループの名前を指定します。

   この状況は STOP メソッドに失敗し、停止に失敗したリソースがリソースグループ内のほかのノードの依存性を持っているときに、リソースグループをオフラインに切り替えた場合に発生します。これ以外の状況では、手順 4 のコマンドをすべての STOP_FAILED リソースで実行することによって、リソースグループは自動的に ONLINE または OFFLINE 状態に戻ります。

   これで、リソースグループを ONLINE 状態に切り替えることができます。

事前登録されているリソースタイプのアップグレード

Sun Cluster 3.1 9/04 では、次の事前登録されているリソースタイプが拡張されています。

- SUNW.LogicalHostname は、論理ホスト名を表現します。
- SUNW.SharedAddress は、共有アドレスを表現します。

これらのリソースタイプが拡張された目的は、名前解決用のネームサービスをバイパスするように論理ホスト名リソースと共有アドレスリソースを変更できるようにするためです。

以下の条件が当てはまる場合は、これらのリソースタイプをアップグレードします。

- 以前のバージョンの Sun Cluster からアップグレードしている場合。
- リソースタイプの新機能を使用する必要がある場合。

リソースタイプをアップグレードする方法については、33 ページの「リソースタイプの更新」を参照してください。以下の各項では、事前登録されているリソースタイプのアップグレードに必要な情報について説明します。

新しいリソースタイプバージョンの登録に関する情報

次の表に、各事前登録されているリソースタイプと Sun Cluster のリリース間の関係を示します。Sun Cluster のリリースは、リソースタイプが導入されたバージョンを表します。

リソースタイプ	リソースタイプバージョン	**Sun Cluster** のリリース
`SUNW.LogicalHostname`	1.0	3.0
	2	3.1 9/04
`SUNW.SharedAddress`	1.0	3.0
	2	3.1 9/04

登録されているリソースタイプのバージョンを調べるには、次のどちらかのコマンドを使用します。

- `scrgadm -p`
- `scrgadm -pv`

例 **2–1** `SUNW.LogicalHostname` リソースタイプの新しいバージョンの登録

この例では、アップグレード中、 `SUNW.LogicalHostname` リソースタイプのバージョン 2 を登録するためのコマンドを示します。

```
# scrgadm -a -t SUNW.LogicalHostname:2
```

リソースタイプの既存インスタンスの移行に関する情報

次に、事前登録されているリソースタイプのインスタンスを移行する必要がある情報を示します。

- 移行はいつでも実行できます。

- 事前登録されているリソースタイプの新機能を使用する必要がある場合、Type_version プロパティの値が 2 である必要があります。
- ネームサービスをバイパスするようにリソースを変更するには、リソースの CheckNameService 拡張プロパティを false に設定します。

例 **2–2** 論理ホスト名リソースの移行

この例では、論理ホスト名リソース lhostrs を移行するためのコマンドを示します。移行の結果として、このリソースは名前解決用のネームサービスをバイパスするように変更されます。

```
# scrgadm -c -j lhostrs -y Type_version=2 -x CheckNameService=false
```

事前登録されているリソースタイプを誤って削除した後の再登録

事前登録されているリソースタイプには、SUNW.LogicalHostname と SUNW.SharedAddress があります。すべての論理ホスト名と共有アドレスリソースがこれらのリソースタイプを使用します。これら 2 つのリソースタイプは、誤って削除した場合を除き、登録する必要はありません。誤ってリソースタイプを削除した場合は、次の手順を使用して再登録してください。

注 – 事前登録されているリソースタイプをアップグレードしている場合は、69 ページの「事前登録されているリソースタイプのアップグレード」の指示に従って、新しいリソースタイプのバージョンを登録してください。

追加情報については、scrgadm(1M) のマニュアルページを参照してください。

注 – この手順は、任意のクラスタノードから実行します。

▼ 事前登録されているリソースタイプを誤って削除した後に再登録する

- リソースタイプを再登録します。

```
# scrgadm -a -t SUNW.resource-type
```

-a　　リソースタイプを追加します。

`-t SUNW.`*resource-type*　追加する (再登録する) リソースタイプを指定します。リソースタイプは、SUNW.LogicalHostname または SUNW.SharedAddress のいずれかになります。

例 - 事前登録されているリソースタイプを誤って削除した後に再登録する

次に、LogicalHostname リソースタイプを再登録する例を示します。

```
# scrgadm -a -t SUNW.LogicalHostname
```

リソースグループへのノードの追加と削除

この節の手順では、次の作業を行います。

- リソースグループの追加のマスターとなるクラスタノードを構成する
- リソースグループからノードを削除する

ノードの追加や削除をフェイルオーバーリソースグループに対して行うのか、スケーラブルリソースグループに対して行うのかによって、手順は異なります。

フェイルオーバーリソースグループは、フェイルオーバーとスケーラブルの両方のサービスによって使用されるネットワークリソースを含みます。クラスタに接続される各 IP サブネットワークは、指定された独自のネットワークリソースを持ち、フェイルオーバーリソースグループに含まれます。このネットワークリソースは、論理ホスト名または共有アドレスリソースのいずれかになります。各ネットワークリソースは、それが使用する IP ネットワークマルチパスグループのリストを含んでいます。フェイルオーバーリソースグループの場合は、リソースグループ (netiflist リソースプロパティ) に含まれる各ネットワークリソースに対し、IP ネットワークマルチパスグループの完全なリストを更新する必要があります。

スケーラブルリソースグループの場合は、スケーラブルグループをホストの新しいセット上でマスターされるように変更するほかに、スケーラブルリソースによって使用されるネットワークリソースを含むフェイルオーバーグループのための手順も実行する必要があります。

追加情報については、scrgadm(1M) のマニュアルページを参照してください。

注 – 任意のクラスタノードから、以下に説明する手順のいずれかを実行します。

リソースグループにノードを追加する

ノードをリソースグループに追加する手順は、リソースグループがスケーラブルリソースグループか、またはフェイルオーバーリソースグループかによって異なります。詳細の手順については、以下の節を参照してください。

- 73 ページの「スケーラブルリソースグループにノードを追加する」
- 74 ページの「フェイルオーバーリソースグループにノードを追加する」

この手順を実行するには、次の情報が必要になります。

- すべてのクラスタノードの名前とノード ID
- ノードが追加されるリソースグループの名前
- すべてのノード上のリソースグループによって使用されるネットワークリソースをホストする IP ネットワークマルチパスグループの名前

さらに、新しいノードがすでにクラスタメンバーになっていることも確認してください。

▼ スケーラブルリソースグループにノードを追加する

1. リソースグループ内のスケーラブルリソースが使用する各ネットワークリソースごとに、そのネットワークリソースが配置されているリソースグループが新しいノードで実行されるようにします。

   詳細は、以下の作業の手順 1 から手順 4 を参照してください。

2. スケーラブルリソースグループをマスターできるノードのリスト **(`nodelist` リソースグループプロパティ)** に新しいノードを追加します。

   この手順は、`nodelist` の値を上書きするため、リソースグループをマスターできるすべてのノードをここに含める必要があります。

   ```
   # scrgadm -c -g resource-group -h nodelist
   ```

`-c`	リソースグループを変更します。
`-g` *resource-group*	ノードが追加されるリソースグループの名前を指定します。
`-h` *nodelist*	リソースグループをマスターできるノードの名前をコンマで区切って指定します。

3. **(省略可能)** スケーラブルリソースの **`Load_balancing_weights`** プロパティを更新し、リソースグループに追加するノードにウエイトを割り当てます。

   ウエイトを割り当てない場合は、デフォルトで 1 になります。詳細は、`scrgadm` (1M) のマニュアルページを参照してください。

▼ フェイルオーバーリソースグループにノードを追加する

1. 現在のノードリスト、およびリソースグループ内の各リソース用に構成された **IP** ネットワークマルチパスグループの現在のリストを表示します。

```
# scrgadm -pvv -g resource-group | grep -i nodelist
# scrgadm -pvv -g resource-group | grep -i netiflist
```

注 – `nodelist` と `netiflist` のコマンド行出力では、ノード名でノードが識別されます。ノード ID を識別するには、コマンド `scconf -pv | grep -i` *node_id* を実行してください。

---

2. ノードの追加によって影響を受けるネットワークリソースの **`netiflist`** を更新します。

   この手順は、*netiflist* の値を上書きするため、すべての IP ネットワークマルチパスグループをここに含める必要があります。

```
# scrgadm -c -j network-resource -x netiflist=netiflist
```

`-c`	ネットワークリソースを変更します。
`-j` *network-resource*	*netiflist* エントリ上でホストされているネットワークリソースの名前 (論理ホスト名または共有アドレス) を指定します。
`-x netiflist =`*netiflist*	各ノード上の IP ネットワークマルチパスグループをコンマで区切って指定します。*netiflist* の各要素は、`netif@node` の形式にする必要があります。`netif` は IP ネットワークマルチパス グループ名 (`sc_ipmp0` など) として指定できます。ノードには、ノード名またはノード ID (`sc_ipmp0@1`、`sc_ipmp@phys-schost-1` など) を指定できます。

3. このリソースグループをマスターできるすべてのノードを含めるように、ノードリストを更新します。

   この手順は、`nodelist` の値を上書きするため、リソースグループをマスターできるすべてのノードをここに含める必要があります。

```
# scrgadm -c -g resource-group -h nodelist
```

`-c`	リソースグループを変更します。
`-g` *resource-group*	ノードが追加されるリソースグループの名前を指定します。
`-h` *nodelist*	リソースグループをマスターできるノードの名前をコンマで区切って指定します。

4. 更新された情報を確認します。

```
# scrgadm -pvv -g resource-group | grep -i nodelist
# scrgadm -pvv -g resource-group | grep -i netiflist
```

例 – リソースグループへのノードの追加

次に、リソースグループ (resource-group-1) にノード (phys-schost-2) を追加する例を示します。このリソースグループは、論理ホスト名リソース (schost-2) を含んでいます。

```
# scrgadm -pvv -g resource-group-1 | grep -i nodelist
(resource-group-1) リソースグループ Nodelist:    phys-schost-1 phys-schost-3
# scrgadm -pvv -g resource-group-1 | grep -i netiflist
(resource-group-1:schost-2) リソース property name: NetIfList
(resource-group-1:schost-2:NetIfList) リソース property class: extension
(resource-group-1:schost-2:NetIfList) List of IP Networking Multipathing
interfaces on each node
(resource-group-1:schost-2:NetIfList) リソース property type: stringarray
(resource-group-1:schost-2:NetIfList) リソース property value: sc_ipmp0@1 sc_ipmp0@3
```

(ノード 1 と 3 のみが IP ネットワークマルチパスグループに割り当てられています。ノード 2 用の IP ネットワークマルチパスグループを追加する必要があります。)

```
# scrgadm -c -j schost-2 -x netiflist=sc_ipmp0@1,sc_ipmp0@2,sc_ipmp0@3
# scrgadm -c -g resource-group-1 -h phys-schost-1,phys-schost-2,phys-schost-3
# scrgadm -pvv -g resource-group-1 | grep -i nodelist
(resource-group-1) リソースグループ Nodelist:     phys-schost-1 phys-schost-2
                                                  phys-schost-3
# scrgadm -pvv -g resource-group-1 | grep -i netiflist
(resource-group-1:schost-2:NetIfList) リソース property value: sc_ipmp0@1 sc_ipmp0@2
                                                               sc_ipmp0@3
```

リソースグループからノードを削除する

ノードをリソースグループから削除する手順は、リソースグループがスケーラブルリソースグループであるか、またはフェイルオーバーリソースグループであるかによって異なります。詳細の手順については、以下の節を参照してください。

- 76 ページの「スケーラブルリソースグループからノードを削除する」
- 77 ページの「フェイルオーバーリソースグループからノードを削除する」
- 79 ページの「共有アドレスリソースを含むフェイルオーバーリソースグループからノードを削除する」

具体例は、80 ページの「例 – リソースグループからのノードの削除」 を参照してください。

この手順を実行するには、次の情報が必要になります。

- すべてのクラスタノードの名前とノード ID

```
# scconf -pv | grep "Node ID"
```

- ノードが削除されるリソースグループまたはグループの名前

```
# scrgadm -pv | grep "Res Group Nodelist"
```

- すべてのノード上のリソースグループによって使用されるネットワークリソースをホストする IP ネットワークマルチパスグループの名前

```
# scrgadm -pvv | grep "NetIfList.*value"
```

さらに、削除するノード上でリソースグループがマスターされていないことを確認してください。削除するノード上でマスターされている場合は、`scswitch` コマンドを実行し、そのノードでリソースグループをオフラインに切り替えてください。次の `scswitch` コマンドは、指定されたノードからリソースグループをオフラインにします。この場合、*new-masters* にこのノードが含まれていてはなりません。

```
# scswitch -z -g resource-group -h new-masters
```

`-g` *resource-group*	オフラインに切り替えるリソースグループ (削除するノードでマスターされている) の名前を指定します。
`-h` *new-masters*	このリソースグループを現在マスターできるノードを指定します。

追加情報については、`scswitch`(1M) のマニュアルページを参照してください。

注意 – すべてのリソースグループからノードを削除する場合で、スケーラブルサービス構成を使用するときは、最初にスケーラブルリソースグループからそのノードを削除してください。続いて、フェイルオーバーグループからそのノードを削除してください。

▼ スケーラブルリソースグループからノードを削除する

スケーラブルサービスは、次に示すように 2 つのリソースグループとして構成されます。

- 1 つは、スケーラブルサービスリソースを含むスケーラブルグループです。
- もう 1 つは、スケーラブルサービスリソースが使用する共有アドレスリソースを含むフェイルオーバーグループです。

スケーラブルリソースグループの `RG_dependencies` プロパティは、フェイルオーバーリソースグループへの依存性を使用してスケーラブルグループを構成するように設定されます。このプロパティの詳細は、付録 A を参照してください。

スケーラブルサービスの構成に関する詳細は、『 *Sun Cluster* の概念 *(Solaris OS* 版*)* 』のマニュアルを参照してください。

スケーラブルリソースグループからノードを削除すると、そのスケーラブルサービスはそのノード上でオンラインにすることができなくなります。スケーラブルリソースグループからノードを削除するには、以下の作業を行なってください。

1. スケーラブルリソースグループをマスターできるノードのリスト **(nodelist** リソースグループプロパティ**)** からノードを削除します。

   # **scrgadm -c -g** *scalable-resource-group* **-h** *nodelist*

`-c`	リソースグループを変更します。
`-g` *scalable-resource-group*	ノードが削除されるリソースグループの名前を指定します。
`-h` *nodelist*	当該リソースグループをマスターできるノードの名前をコンマで区切って指定します。

2. **(**省略可能**)** 共有アドレスリソースが入ったフェイルオーバーリソースグループからノードを削除します。

   詳細は、79 ページの「共有アドレスリソースを含むフェイルオーバーリソースグループからノードを削除する」を参照してください。

3. **(**省略可能**)** スケーラブルリソースの **Load_balancing_weights** プロパティを更新し、リソースグループから削除するノードのウエイトを削除します。

   詳細は、`scrgadm`(1M) のマニュアルページを参照してください。

▼ フェイルオーバーリソースグループからノードを削除する

フェイルオーバーリソースグループからノードを削除するには、以下の作業を行なってください。

注意 – すべてのリソースグループからノードを削除する場合で、スケーラブルサービス構成を使用するときは、最初にスケーラブルリソースグループからそのノードを削除してください。続いて、この方法を使用してフェイルオーバーグループからノードを削除してください。

注 – フェイルオーバーリソースグループにスケーラブルサービスが使用する共有アドレスリソースが含まれる場合は、79 ページの「共有アドレスリソースを含むフェイルオーバーリソースグループからノードを削除する」を参照してください。

1. このリソースグループをマスターできるすべてのノードを含めるように、ノードリストを更新します。

この手順はノードを削除してノードリストの値を上書きするため、リソースグループをマスターできるすべてのノードをここに含める必要があります。

```
# scrgadm -c -g failover-resource-group -h nodelist
```

`-c`	リソースグループを変更します。
`-g` *failover-resource-group*	ノードが削除されるリソースグループの名前を指定します。
`-h` *nodelist*	当該リソースグループをマスターできるノードの名前をコンマで区切って指定します。

2. リソースグループ内の各リソース用に構成した **IP** ネットワークマルチパスグループの現在のリストを表示します。

```
# scrgadm -pvv -g failover-resource-group | grep -i netiflist
```

3. ノードの削除によって影響を受けるネットワークリソースの **`netiflist`** を更新します。

この手順は `netiflist` の値を上書きするため、すべての IP ネットワークマルチパスグループをここに含める必要があります。

```
# scrgadm -c -j network-resource -x netiflist=netiflist
```

注 – 上記コマンド行の出力は、ノード 名によってノードを識別します。ノード ID を識別するには、コマンド `scconf -pv | grep "ノード ID"` を実行してください。

`-c`	ネットワークリソースを変更します。
`-j` *network-resource*	`netiflist` エントリ上でホストされているネットワークリソースの名前を指定します。
`-x netiflist=`*netiflist*	各ノード上の IP ネットワークマルチパスグループをコンマで区切って指定します。*netiflist* の各要素は、`netif@node` の形式にする必要があります。`netif` は IP ネットワークマルチパス グループ名 (`sc_ipmp0` など) として指定できます。ノードは、ノード名またはノード ID (`sc_ipmp0@1`、`sc_ipmp@phys-schost-1` など) で識別できます。

注 – 現バージョンの Sun Cluster では、`netif` にアダプタ名を使用できません。

---

4. 更新された情報を確認します。

```
# scrgadm -pvv -g failover-resource-group | grep -i nodelist
# scrgadm -pvv -g failover-resource-group | grep -i netiflist
```

▼ 共有アドレスリソースを含むフェイルオーバーリソースグループからノードを削除する

スケーラブルサービスが使用する共有アドレスリソースを含むフェイルオーバーリソースグループでは、ノードは次の場所に現れます。

- フェイルオーバーリソースグループのノードリスト
- 共有アドレスリソースの `auxnodelist`

フェイルオーバーリソースグループのノードリストからノードを削除するには、77 ページの「フェイルオーバーリソースグループからノードを削除する」に示されている作業を行なってください。

共有アドレスリソースの `auxnodelist` を変更するには、共有アドレスリソースを削除して作成し直す必要があります。

フェイルオーバーグループのノードリストからノードを削除すると、そのノード上の共有アドレスリソースを継続して使用し、スケーラブルサービスを提供できます。このためには、共有アドレスリソースの `auxnodelist` にそのノードを追加する必要があります。`auxnodelist` にノードを追加するには、以下の作業を行なってください。

注 – 以下の作業は、共有アドレスリソースの `auxnodelist` からノードを削除するためにも使用できます。`auxnodelist` からノードを削除するには、共有アドレスリソースを削除して作成し直す必要があります。

---

1. スケーラブルサービスリソースをオフラインに切り替えます。
2. フェイルオーバーリソースグループから共有アドレスリソースを削除します。
3. 共有アドレスリソースを作成します。

   フェイルオーバーリソースグループから削除したノードのノード ID またはノード名を `auxnodelist` に追加します。

   ```
   # scrgadm -a -S -g failover-resource-group \
    -l shared-address -X new-auxnodelist
   ```

failover-resource-group	共有アドレスリソースを含めるために使用されたフェイルオーバーリソースグループの名前。
shared-address	共有アドレスの名前。
new-auxnodelist	妥当なノードの追加または削除によって変更された新しい `auxnodelist`。

例 – リソースグループからのノードの削除

次に、リソースグループ (resource-group-1) からノード (phys-schost-3) を削除する例を示します。このリソースグループは、論理ホスト名リソース (schost-1) を含んでいます。

```
# scrgadm -pvv -g resource-group-1 | grep -i nodelist
(resource-group-1) リソースグループ Nodelist:       phys-schost-1 phys-schost-2
                                                phys-schost-3
# scrgadm -c -g resource-group-1 -h phys-schost-1,phys-schost-2
# scrgadm -pvv -g resource-group-1 | grep -i netiflist
(resource-group-1:schost-1) リソース property name: NetIfList
(resource-group-1:schost-1:NetIfList) リソース property class: extension
(resource-group-1:schost-1:NetIfList) List of IP Networking Multipathing
interfaces on each node
(resource-group-1:schost-1:NetIfList) リソース property type: stringarray
(resource-group-1:schost-1:NetIfList) リソース property value: sc_ipmp0@1 sc_ipmp0@2
                                                          sc_ipmp0@3
```

(*sc_ipmp0@3* は削除される *IP* ネットワークマルチパスグループです。)

```
# scrgadm -c  -j schost-1 -x  netiflist=sc_ipmp0@1,sc_ipmp0@2
# scrgadm -pvv -g resource-group-1 | grep -i nodelist
(resource-group-1) リソースグループ Nodelist:       phys-schost-1 phys-schost-2
# scrgadm -pvv -g resource-group-1 | grep -i netiflist
(resource-group-1:schost-1:NetIfList) リソース property value: sc_ipmp0@1 sc_ipmp0@2
```

リソースグループとディスクデバイスグループ間での起動の同期

クラスタが起動した後、あるいは、サービスが別のノードにフェイルオーバーした後、グローバルデバイスとクラスタファイルシステムが利用できるようになるまでには、しばらく時間がかかることがあります。ただし、データサービスは、データサービスが依存する広域デバイスとクラスタファイルシステムがオンラインになる前に、START メソッドを実行できます。この例では、START メソッドがタイムアウトするため、データサービスが使用するリソースグループの状態をリセットし、データサービスを手動で再起動する必要があります。リソースタイプ HAStorage と HAStoragePlus は、広域デバイスとクラスタファイルシステムを監視し、同じリソースグループ内のほかのリソースが利用可能になるまでそれらの START メソッドを待機させます(どのリソースタイプを作成するかを決定するには、20 ページの「HAStorage または HAStoragePlus の選択」 を参照してください)。このような追加の管理作業を軽減するには、グローバルデバイスやクラスタファイルシステムに依存するデータサービスリソースを持つすべてのリソースグループに、HAStorage または HAStoragePlus を設定してください。

HAStorage リソースタイプの作成については、81 ページの「新しいリソース用に HAStorage リソースタイプを設定する」を参照してください。

HAStoragePlus リソースタイプの作成については、87 ページの「HAStoragePlus リソースタイプを設定する」を参照してください。

▼ 新しいリソース用に HAStorage リソースタイプを設定する

HAStorage は、今後の Sun Cluster でサポートされなくなる可能性があります。同等の機能が HAStoragePlus でサポートされています。HAStorage から HAStoragePlus へアップグレードするには、83 ページの「HAStorage から HAStoragePlus へのアップグレード」を参照してください。

次の例では、リソースグループ resource-group-1 は、次の 3 つのデータサービスを含んでいます。

- Sun Java System Web Server (/global/resource-group-1 に依存する)
- Oracle (/dev/global/dsk/d5s2 に依存する)
- NFS (dsk/d6 に依存する)

新しいリソースに対し、HAStorage リソースの hastorage-1 を resource-group-1 に作成するには、80 ページの「リソースグループとディスクデバイスグループ間での起動の同期」 を読み、その後次の手順を実行します。

HAStoragePlus リソースタイプを作成するには、86 ページの「高可用性ローカルファイルシステムの有効化」を参照してください。

1. クラスタメンバー上でスーパーユーザーになります。
2. リソースグループ **resource-group-1** を作成します。

   ```
   # scrgadm -a -g resource-group-1
   ```

3. リソースタイプが登録されているかどうかを調べます。
   次のコマンドは、登録されているリソースタイプのリストを出力します。

   ```
   # scrgadm -p | egrep Type
   ```

4. 必要であれば、リソースタイプを登録します。

   ```
   # scrgadm -a -t SUNW.HAStorage
   ```

5. **HAStorage** リソースである **hastorage-1** を作成し、サービスパスを定義します。

```
# scrgadm -a -j hastorage-1 -g resource-group-1 -t SUNW.HAStorage \
-x ServicePaths=/global/resource-group-1,/dev/global/dsk/d5s2,dsk/d6
```

   ServicePaths には、次の値を含むことができます。

- 広域デバイスグループ名 (例:nfs-dg)
- 広域デバイスのパス (例:/dev/global/dsk/d5s2 または dev/d6)
- クラスタファイルシステムのマウントポイント (例:/global/nfs)

注 – ServicePaths にクラスタファイルシステムパスが含まれる場合、広域デバイスグループはそれらに対応するリソースグループと共に使用されない場合があります。

---

6. **hastorage-1** リソースを有効にします。

   ```
   # scswitch -e -j hastorage-1
   ```

7. リソース **Sun Java System Web Server**、**Oracle**、**NFS** を **resource-group-1** に追加し、これらの依存性を **hastorage-1** に設定します。

   たとえば、Sun Java System Web Server の場合、次のコマンドを実行します。

```
# scrgadm -a -j resource \-g resource-group-1 -t SUNW.iws \
-x Confdir_list=/global/iws/schost-1 -y Scalable=False \
-y Network_resources_used=schost-1 -y Port_list=80/tcp \
-y Resource_dependencies=hastorage-1
```

8. リソースの依存性を正しく構成したかを確認します。

   ```
   # scrgadm -pvv -j resource | egrep strong
   ```

9. **resource-group-1** を **MANAGED** 状態に設定し、オンラインにします。

   ```
   # scswitch -Z -g resource-group-1
   ```

HAStorage リソースタイプは、別の拡張プロパティ (AffinityOn) を含みます。この拡張プロパティは、HAStorage が ServicePaths で定義されている広域デバイスおよびクラスタファイルシステムの類似性スイッチオーバーを実行する必要があるかどうかを指定するブール値です。詳細は、SUNW.HAStorage(5) のマニュアルページを参照してください。

注 – リソースグループがスケーラブルの場合、HAStorage と HAStoragePlus は AffinityOn が TRUE に設定されることを許可しません。スケーラブルリソースグループについては、HAStorage と HAStoragePlus は AffinityOn 値をチェックし、この値を内部的に FALSE に設定し直します。

▼ 既存のリソース用に HAStorage リソースタイプを設定する

HAStorage は、今後の Sun Cluster でサポートされなくなる可能性があります。同等の機能が HAStoragePlus でサポートされています。HAStorage から HAStoragePlus へアップグレードするには、83 ページの「HAStorage から HAStoragePlus へのアップグレード」を参照してください。

既存のリソースのために HAStorage リソースを作成するには、80 ページの「リソースグループとディスクデバイスグループ間での起動の同期」を読み、その後以下の作業を行なってください。

1. リソースタイプが登録されているかどうかを調べます。
   次のコマンドは、登録されているリソースタイプのリストを出力します。

   ```
   # scrgadm -p | egrep Type
   ```

2. 必要であれば、リソースタイプを登録します。

   ```
   # scrgadm -a -t SUNW.HAStorage
   ```

3. **HAStorage** リソースである **hastorage-1** を作成します。

   ```
   # scrgadm -a -g resource-group -j hastorage-1 -t SUNW.HAStorage \
   -x ServicePaths= ... -x AffinityOn=True
   ```

4. **hastorage-1** リソースを有効にします。

   ```
   # scswitch -e -j hastorage-1
   ```

5. 必要に応じて既存の各リソースについて依存性を設定します。

   ```
   # scrgadm -c -j resource -y Resource_Dependencies=hastorage-1
   ```

6. リソースの依存性を正しく構成したかを確認します。

   ```
   # scrgadm -pvv -j resource | egrep strong
   ```

HAStorage から HAStoragePlus へのアップグレード

HAStorage は、今後の Sun Cluster でサポートされなくなる可能性があります。同等の機能が HAStoragePlus でサポートされています。HAStorage から HAStorage へアップグレードする方法については、以下の節を参照してください。

デバイスグループまたは CFS を使用している場合に HAStorage から HAStoragePlus へアップグレードする

HAStorage は、今後の Sun Cluster でサポートされなくなる可能性があります。同等の機能が HAStoragePlus でサポートされています。デバイスグループまたは CFS を使用している場合に HAStorage から HAStoragePlus にアップグレードするには、以下の作業を行なってください。

この例では、HAStorage で単純な HA-NFS リソースが有効になっています。ServicePaths はディスクグループ `nfsdg` で、AffinityOn プロパティは TRUE です。さらに、この HA-NFS リソースは `Resource_Dependencies` を HAStorage リソースに設定しています。

1. **HAStorage** に対するアプリケーションリソースの依存性を除去します。

   ```
   # scrgadm -c -j nfsserver-rs -y Resource_Dependencies=""
   ```

2. **HAStorage** リソースを無効にします。

   ```
   # scswitch -n -j nfs1storage-rs
   ```

3. アプリケーションリソースグループから **HAStorage** リソースを削除します。

   ```
   # scrgadm -r -j nfs1storage-rs
   ```

4. **HAStorage** リソースタイプの登録を解除します。

   ```
   # scrgadm -r -t SUNW.HAStorage
   ```

5. **HAStoragePlus** リソースタイプを登録します。

   ```
   # scrgadm -a -t SUNW.HAStoragePlus
   ```

6. **HAStoragePlus** リソースを作成します。

   ファイルシステムのマウントポイントを指定するには、次のテキストを入力してください。

   ```
   # scrgadm -a -j nfs1-hastp-rs -g nfs1-rg -t \
   SUNW.HAStoragePlus -x FilesystemMountPoints=/global/nfsdata -x \
   AffinityOn=True
   ```

   グローバルデバイスパスを指定するには、次のテキストを入力してください。

   ```
   # scrgadm -a -j nfs1-hastp-rs -g nfs1-rg -t \
   SUNW.HAStoragePlus -x GlobalDevicePaths=nfsdg -x AffinityOn=True
   ```

注 – HAStorage の ServicePaths プロパティではなく、HAStoragePlus の GlobalDevicePaths または FilesystemMountPoints プロパティを使用する必要があります。FilesystemMountPoints 拡張プロパティは、/etc/vfstab で指定されたシーケンスと一致する必要があります。

7. **HAStoragePlus** リソースを有効にします。

```
# scswitch -e -j nfs1-hastp-rs
```

8. アプリケーションサーバーと**HAStoragePlus** との間の依存性を設定します。

```
# scrgadm -c -j nfsserver-rs -y \
Resource_Depencencies=nfs1=hastp-rs
```

CFS による HAStorage からフェイルオーバーファイルシステムによる HAStoragePlus へアップグレードする

HAStorage は、今後の Sun Cluster でサポートされなくなる可能性があります。同等の機能が HAStoragePlus でサポートされています。CFS による HAStorage から Failover Filesystem (FFS) による HAStoragePlus にアップグレードするには、以下の作業を行なってください。

この例では、HAStorage で単純な HA-NFS リソースが有効になっています。ServicePaths はディスクグループ nfsdg で、AffinityOn プロパティは TRUE です。さらに、この HA-NFS リソースは Resource_Dependencies を HAStorage リソースに設定しています。

1. **HAStorage** リソースに対するアプリケーションリソースの依存性を除去します。

```
# scrgadm -c -j nfsserver-rs -y Resource_Dependencies=""
```

2. **HAStorage** リソースを無効にします。

```
# scswitch -n -j nfs1storage-rs
```

3. アプリケーションリソースグループから **HAStorage** リソースを削除します。

```
# scrgadm -r -j nfs1storage-rs
```

4. **HAStorage** リソースタイプの登録を解除します。

```
# scrgadm -r -t SUNW.HAStorage
```

5. **/etc/vfstab** を変更して広域フラグを削除し、「**mount at boot**」を「**no**」に変更します。

6. **HAStoragePlus** リソースを作成します。

   ファイルシステムのマウントポイントを指定するには、次のテキストを入力してください。

   ```
   # scrgadm -a -j nfs1-hastp-rs -g nfs1-rg -t \
   SUNW.HAStoragePlus -x FilesystemMountPoints=/global/nfsdata -x \
   AffinityOn=True
   ```

   グローバルデバイスパスを指定するには、次のテキストを入力してください。

   ```
   # scrgadm -a -j nfs1-hastp-rs -g nfs1-rg -t \
   SUNW.HAStoragePlus -x GlobalDevicePaths=nfsdg -x AffinityOn=True
   ```

   注 – HAStorage の ServicePaths プロパティではなく、HAStoragePlus の GlobalDevicePaths または FilesystemMountPoints プロパティを使用する必要があります。FilesystemMountPoints 拡張プロパティは、`/etc/vfstab` で指定されたシーケンスと一致する必要があります。

   ---

7. **HAStoragePlus** リソースを有効にします。

   ```
   # scswitch -e -j nfs1-hastp-rs
   ```

8. アプリケーションサーバーと**HAStoragePlus** との間の依存性を設定します。

   ```
   # scrgadm -c -j nfsserver-rs -y \
   Resource_Depencencies=nfs1=hastp-rs
   ```

高可用性ローカルファイルシステムの有効化

`HAStoragePlus` リソースタイプを使用すると、ローカルファイルシステムを Sun Cluster 環境内で高可用性にすることができます。このためには、ローカルファイルシステムのパーティションがグローバルディスクグループに存在し、アフィニティスイッチオーバーが有効であり、Sun Cluster 環境がフェイルオーバー用に構成されている必要があります。これによって、多重ホストディスクに直接接続された任意のホストから、多重ホストディスク上の任意のファイルシステムにアクセスできるようになります。(HAStoragePlus では、ルートファイルシステムを高可用性にすることはできません)。フェイルバック設定は、リソースグループとデバイスグループで同一にする必要があります。

入出力の多いデータサービスの中には、HA ローカルファイルシステムの使用が強く望まれるものがあります。このため、このようなデータサービスの登録作業と構成作業には、HAStoragePlus リソースタイプを構成する方法が追加されています。これらのデータサービスの HAStoragePlus リソースタイプを設定する手順については、以下の節を参照してください。

- 『*Sun Cluster Data Service for Oracle* ガイド *(Solaris OS* 版*)*』の「Sun Cluster HA for Oracle の登録と構成」
- 『*Sun Cluster Data Service for Sybase ASE* ガイド *(Solaris OS* 版*)*』の「Sun Cluster HA for Sybase ASE の登録と構成」

ほかのデータサービスの HAStoragePlus リソースタイプを設定する方法については、87 ページの「HAStoragePlus リソースタイプを設定する」を参照してください。

注 – この節では、HAStoragePlus リソースタイプを UNIX ファイルシステムで使用する方法について説明します。HAStoragePlus リソースタイプを Sun StorEdge™ QFS ファイルシステムで使用する方法については、Sun StorEdge QFSSunStorEdgeQFS のマニュアルを参照してください。

▼ HAStoragePlus リソースタイプを設定する

HAStoragePlus リソースタイプは Sun Cluster 3.0 5/02 で導入されています。この新しいリソースタイプは、HAStorage と同じ機能を果たし、リソースグループとディスクデバイスグループ間で起動を同期します。HAStoragePlus リソースタイプには、ローカルファイルシステムを高可用性にするための機能が追加されています。(ローカルファイルシステムの可用性を高めるための背景情報については、86 ページの「高可用性ローカルファイルシステムの有効化」を参照してください)。これら 2 つの機能を両方とも使用するには、HAStoragePlus リソースタイプを設定します。

HAStoragePlus を設定するには、ローカルファイルシステムのパーティションがグローバルディスクグループに存在し、アフィニティスイッチオーバーが有効であり、かつ Sun Cluster 環境がフェイルオーバー用に構成されている必要があります。

次の例では、簡単な NFS サービスを使用して、ローカルにマウントされたディレクトリ /global/local-fs/nfs/export/ からホームディレクトリのデータを共有します。この例では、次の条件を前提にしています。

- マウントポイント /global/local-fs/nfs は、UFS ローカルファイルシステムを Sun Cluster 広域デバイスのパーティションにマウントするために使用されます。
- /global/local-fs/nfs ファイルシステムの /etc/vfstab エントリには、このファイルシステムがローカルファイルシステムで、マウントブートフラグが「no」であるよう指定されている必要があります。

- PathPrefix ディレクトリ (HA-NFS が管理情報と状態情報を保守するために使用するディレクトリ) は、マウントするファイルシステムのルートディレクトリ (たとえば、/global/local-fs/nfs) 上に存在します。

1. クラスタメンバー上でスーパーユーザーになります。
2. リソースタイプが登録されているかどうかを調べます。
   次のコマンドは、登録されているリソースタイプのリストを出力します。

   ```
   # scrgadm -p | egrep Type
   ```

3. 必要であれば、リソースタイプを登録します。

   ```
   # scrgadm -a -t SUNW.nfs
   ```

4. フェイルオーバーリソースグループである **nfs-r** を作成します。

```
# scrgadm -a -g nfs-rg -y PathPrefix=/global/local-fs/nfs
```

5. タイプ **SUNW.LogicalHostname** の論理ホストリソースを作成します。

   ```
   # scrgadm -a -j nfs-lh-rs -g nfs-rg -L -l log-nfs
   ```

6. クラスタに **HAStoragePlus** リソースタイプを登録します。

```
# scrgadm -a -t SUNW.HAStoragePlus
```

7. タイプ **HAStoragePlus** のリソース **nfs-hastp-rs** を作成します。

```
# scrgadm -a -j nfs-hastp-rs -g nfs-rg -t SUNW.HAStoragePlus \
-x FilesystemMountPoints=/global/local-fs/nfs \
-x AffinityOn=TRUE
```

注 – FilesystemMountPoints 拡張プロパティは、1 つ以上のファイルシステムマウントポイントをリストの形式で指定するために使用できます。このリストには、ローカルファイルシステムマウントポイントとグローバルファイルシステムマウントポイントの両方を含めることができます。ブートフラグでのマウントは、広域ファイルシステムの HAStoragePlus によって無視されます。

---

8. リソースグループ **nfs-rg** をクラスタノード上でオンラインにします。
   このノードは、/global/local-fs/nfs ファイルシステムの実際のグローバルデバイスのパーティション用の稼働系になります。次に、ファイルシステム /global/local-fs/nfs は当該ノード上にローカルにマウントされます。

   ```
   # scswitch -Z -g nfs-rg
   ```

9. **SUNW.nfs** リソースタイプをクラスタに登録します。タイプ **SUNW.nfs** のリソース **nfs-rs** を作成して、リソース **nfs-hastp-rs** へのリソース依存関係を指定します。
   dfstab.nfs-rs が /global/local-fs/nfs/SUNW.nfs に作成されます。

```
# scrgadm -a -t SUNW.nfs
# scrgadm -a -g nfs-rg -j nfs-rs -t SUNW.nfs \
-y Resource_dependencies=nfs-hastp-rs
```

注 – nfs リソースに依存関係を設定するには、nfs-hastp-rs リソースがオンラインである必要があります。

10. リソース **nfs-rs** をオンラインにします。

```
# scswitch -Z -g nfs-rg
```

注意 – 切り替えは、リソースグループレベルに限定して行なってください。デバイスグループレベルで切り替えを行うと、リソースグループが混乱し、フェイルオーバーが発生します。

これで、サービスを新しいノードに移行するときには常に、/global/local-fs/nfs 用のプライマリ入出力パスはオンラインになり、NFS サーバーに配置されます。ファイルシステム /global/local-fs/nfs は NFS サーバーが起動する前にローカルにマウントされます。

高可用性ファイルシステムのリソースをオンラインのままで変更する

ファイルシステムを表現しているリソースを変更している間でも、高可用性ファイルシステムは利用できる必要があります。たとえば、ストレージが動的に提供されている場合、ファイルシステムは利用できる必要があります。このような状況では、高可用性ファイルシステムを表現しているリソースをオンラインのままで変更します。

Sun Cluster 環境では、高可用性ファイルシステムは HAStoragePlus リソースで表現されます。Sun Cluster では、HAStoragePlus をオンラインのままで変更するには、次のようにします。

- ファイルシステムを HAStoragePlus リソースに追加する
- ファイルシステムを HAStoragePlus リソースから削除する

注 – Sun Cluster では、ファイルシステムの名前はオンラインのままでは変更できません。

▼ オンラインの HAStoragePlus リソースにファイルシステムを追加する

HAStoragePlus リソースにファイルシステムを追加するとき、HAStoragePlus リソースはローカルファイルシステムをグローバルファイルシステムとは別に処理します。

- HAStoragePlus リソースは常に、ローカルファイルシステムを自動的にマウントします。
- HAStoragePlus リソースがグローバルファイルシステムを自動的にマウントするのは、HAStoragePlus リソースの AffinityOn 拡張プロパティが True の場合だけです。

AffinityOn 拡張プロパティについては、80 ページの「リソースグループとディスクデバイスグループ間での起動の同期」を参照してください。

1. クラスタの1つのノード上で、スーパーユーザーになります。
2. クラスタの各ノードの **/etc/vfstab** ファイルにおいて、追加しようとしている各ファイルシステムのマウントポイント用のエントリを追加します。

   エントリごとに、mount at boot フィールドと mount options フィールドを次のように設定します。

   - mount at boot フィールドを no に設定します。
   - ファイルシステムがグローバルファイルシステムの場合、global オプションを含むように mount options フィールドを設定します。
3. **HAStoragePlus** リソースがすでに管理しているファイルシステムのマウントポイントのリストを取得します。

   ```
   # scha_resource_get -O extension -R hasp-resource -G hasp-rg \
   FileSystemMountPoints
   ```

-R *hasp-resource*	ファイルシステムを追加する先の HAStoragePlus リソースを指定します。
-G *hasp-rg*	HAStoragePlus リソースを含むリソースグループを指定します。

4. **HAStoragePlus** リソースの **FileSystemMountPoints** 拡張プロパティを変更して、次のマウントポイントを含むようにします。

   - HAStoragePlus リソースがすでに管理しているファイルシステムのマウントポイント
   - HAStoragePlus リソースに追加しようとしているファイルシステムのマウントポイント

   ```
   # scrgadm -c -j hasp-resource -x FileSystemMountPoints="mount-point-list"
   ```

-j *hasp-resource*
ファイルシステムを追加する先の HAStoragePlus リソースを指定します。

-x FileSystemMountPoints="*mount-point-list* "
HAStoragePlus リソースがすでに管理しているファイルシステムのマウントポイントと、追加しようとしているファイルシステムのマウントポイントをコンマで区切って指定します。

5. **HAStoragePlus** リソースのマウントポイントのリストと手順 **4**で指定したリストが一致していることを確認します。

```
# scha_resource_get -O extension -R hasp-resource -G hasp-rg \
 FileSystemMountPoints
```

-R *hasp-resource* ファイルシステムを追加する先の HAStoragePlus リソースを指定します。

-G *hasp-rg* HAStoragePlus リソースを含むリソースグループを指定します。

6. **HAStoragePlus** リソースがオンラインであり、障害が発生していないことを確認します。

HAStoragePlus リソースがオンラインであるが、障害が発生している場合、リソースの確認は成功しますが、HAStoragePlus によるファイルシステムのマウントは失敗します。

```
# scstat -g
```

例 **2–3** オンラインの HAStoragePlus リソースへのファイルシステムの追加

次に、オンラインの HAStoragePlus リソースにファイルシステムを追加する例を示します。

- HAStoragePlus リソースは rshasp という名前であり、リソースグループ rghasp に含まれます。
- rshasp という名前の HAStoragePlus リソースはすでに、マウントポイントが /global/global-fs/fs1 であるファイルシステムを管理しています。
- 追加しようとしているファイルシステムのマウントポイントは /global/global-fs/fs2 です。

この例では、各クラスタノード上の /etc/vfstabファイルにはすでに、追加しようとしているファイルシステムのエントリが含まれていると仮定します。

```
# scha_resource_get -O extension -R rshasp -G rghasp FileSystemMountPoints
STRINGARRAY
/global/global-fs/fs1
# scrgadm -c -j rshasp \
-x FileSystemMountPoints="/global/global-fs/fs1,/global/global-fs/fs2"
# scha_resource_get -O extension -R rshasp -G rghasp FileSystemMountPoints
STRINGARRAY
/global/global-fs/fs1
/global/global-fs/fs2
```

例 2–3 オンラインの HAStoragePlus リソースへのファイルシステムの追加　(続き)

```
# scstat -g

 -- Resource Groups and Resources --

             Group Name      Resources
             ----------      ---------
  Resources: rghasp          rshasp


 -- Resource Groups --

             Group Name      Node Name    State
             ----------      ---------    -----
      Group: rghasp          node46       Offline
      Group: rghasp          node47       Online


 -- Resources --

             Resource Name   Node Name    State     Status Message
             -------------   ---------    -----     --------------
   Resource: rshasp          node46       Offline   Offline
   Resource: rshasp          node47       Online    Online
```

▼ オンラインの HAStoragePlus リソースからファイルシステムを削除する

HAStoragePlus リソースからファイルシステムを削除するとき、HAStoragePlus リソースはローカルファイルシステムをグローバルファイルシステムとは別に処理します。

- HAStoragePlus リソースは常に、ローカルファイルシステムを自動的にアンマウントします。
- HAStoragePlus リソースがグローバルファイルシステムを自動的にアンマウントするのは、HAStoragePlus リソースの AffinityOn 拡張プロパティが True の場合だけです。

AffinityOn 拡張プロパティについては、80 ページの「リソースグループとディスクデバイスグループ間での起動の同期」を参照してください。

注意 – オンラインの HAStoragePlus リソースからファイルシステムを削除する前には、そのファイルシステムを使用しているアプリケーションが存在しないことを確認してください。オンラインの HAStoragePlus リソースからファイルシステムを削除すると、そのファイルシステムは強制的にアンマウントされます。アプリケーションが使用しているファイルシステムが強制的にアンマウントされると、そのアプリケーションは異常終了またはハングする可能性があります。

1. クラスタの1つのノード上で、スーパーユーザーになります。
2. **HAStoragePlus** リソースがすでに管理しているファイルシステムのマウントポイントのリストを取得します。

   ```
   # scha_resource_get -O extension -R hasp-resource -G hasp-rg \
   FileSystemMountPoints
   ```

   -R *hasp-resource* ファイルシステムを削除する元の HAStoragePlus リソースを指定します。

   -G *hasp-rg* HAStoragePlus リソースを含むリソースグループを指定します。

3. **HAStoragePlus** リソースの **FileSystemMountPoints** 拡張プロパティを変更して、**HAStoragePlus** リソースに残すファイルシステムのマウントポイントだけを含むようにします。

   ```
   # scrgadm -c -j hasp-resource -x FileSystemMountPoints="mount-point-list"
   ```

   -j *hasp-resource*
   ファイルシステムを削除する元の HAStoragePlus リソースを指定します。

   -x FileSystemMountPoints="*mount-point-list* "
   HAStoragePlus リソースに残そうとしているファイルシステムのマウントポイントをコンマで区切って指定します。このリストには、削除しようとしているファイルシステムのマウントポイントが含まれていてはなりません。

4. **HAStoragePlus** リソースのマウントポイントのリストと手順 **3**で指定したリストが一致していることを確認します。

   ```
   # scha_resource_get -O extension -R hasp-resource -G hasp-rg \
   FileSystemMountPoints
   ```

   -R *hasp-resource* ファイルシステムを削除する元の HAStoragePlus リソースを指定します。

   -G *hasp-rg* HAStoragePlus リソースを含むリソースグループを指定します。

5. **HAStoragePlus** リソースがオンラインであり、障害が発生していないことを確認します。

HAStoragePlus リソースがオンラインであるが、障害が発生している場合、リソースの確認は成功しますが、HAStoragePlus によるファイルシステムのアンマウントは失敗します。

```
# scstat -g
```

6. **(省略可能)** クラスタの各ノードの **/etc/vfstab** ファイルから、削除しようとしている各ファイルシステムのマウントポイント用のエントリを削除します。

例 **2–4** オンラインの HAStoragePlus リソースからのファイルシステムの削除

次に、オンラインの HAStoragePlus リソースからファイルシステムを削除する例を示します。

- HAStoragePlus リソースは rshasp という名前であり、リソースグループ rghasp に含まれます。
- rshasp という名前の HAStoragePlus リソースはすでに、次のようなマウントポイントのファイルシステムを管理しています。
  - /global/global-fs/fs1
  - /global/global-fs/fs2
- 削除しようとしているファイルシステムのマウントポイントは /global/global-fs/fs2 です。

```
# scha_resource_get -O extension -R rshasp -G rghasp FileSystemMountPoints
STRINGARRAY
/global/global-fs/fs1
/global/global-fs/fs2
# scrgadm -c -j rshasp -x FileSystemMountPoints="/global/global-fs/fs1"
# scha_resource_get -O extension -R rshasp -G rghasp FileSystemMountPoints
STRINGARRAY
/global/global-fs/fs1
 # scstat -g

 -- Resource Groups and Resources --

             Group Name      Resources
             ----------      ---------
  Resources: rghasp          rshasp


 -- Resource Groups --

             Group Name      Node Name    State
             ----------      ---------    -----
     Group: rghasp          node46       Offline
     Group: rghasp          node47       Online


 -- Resources --

             Resource Name   Node Name    State     Status Message
```

例 2–4 オンラインの HAStoragePlus リソースからのファイルシステムの削除　(続き)

```
          -------------   ---------    -----     --------------
Resource: rshasp          node46       Offline   Offline
Resource: rshasp          node47       Online    Online
```

▼ HAStoragePlus リソースの変更後に障害から回復する

FileSystemMountPoints 拡張プロパティの変更中に障害が発生した場合、HAStoragePlus リソースの状態はオンラインであり、かつ、障害が発生しています。障害を修正した後、HAStoragePlus の状態はオンラインです。

1. 変更が失敗した原因となる障害を判断します。

   ```
   # scstat -g
   ```

   障害が発生した HAStoragePlus リソースの状態メッセージは、その障害を示します。可能性のある障害は、次のとおりです。

   - ファイルシステムが存在するはずのデバイスが存在しません。
   - fsck コマンドによるファイルシステムの修復が失敗しました。
   - 追加しようとしたファイルシステムのマウントポイントが存在しません。
   - 追加しようとしたファイルシステムがマウントできません。
   - 削除しようとしたファイルシステムがアンマウントできません。

2. 変更が失敗した原因となる障害を修正します。

3. **HAStoragePlus** リソースの **FileSystemMountPoints** 拡張プロパティを変更する手順を繰り返します。

   ```
   # scrgadm -c -j hasp-resource -x FileSystemMountPoints="mount-point-list"
   ```

   -j *hasp-resource*
   変更しようとしている HAStoragePlus リソースを指定します。

   -x FileSystemMountPoints="*mount-point-list* "
   高可用性ファイルシステムの変更が失敗したときに指定したマウントポイントをコンマで区切って指定します。

4. **HAStoragePlus** リソースがオンラインであり、障害が発生していないことを確認します。

   ```
   # scstat -g
   ```

例 2–5 障害が発生した HAStoragePlus リソースの状態

次に、障害が発生した HAStoragePlus リソースの状態の例を示します。fsck コマンドによるファイルシステムの修復が失敗したため、このリソースには障害が発生しています。

例 2–5 障害が発生した HAStoragePlus リソースの状態　(続き)

```
# scstat -g
 -- Resource Groups and Resources --

             Group Name      Resources
             ----------      ---------
  Resources: rghasp          rshasp


 -- Resource Groups --

             Group Name      Node Name    State
             ----------      ---------    -----
      Group: rghasp          node46       Offline
      Group: rghasp          node47       Online


 -- Resources --

           Resource Name   Node Name    State   Status Message
           -------------   ---------    -----   --------------
 Resource: rshasp          node46       Offline Offline
 Resource: rshasp          node47       Online  Online Faulted - Failed
to fsck: /mnt.
```

HAStoragePlus リソースタイプのアップグレード

Sun Cluster 3.1 9/04 では、HAStoragePlus リソースタイプは高可用性ファイルシステムをオンラインのままで変更できるように拡張されました。HAStoragePlus リソースタイプのアップグレードは、次のすべての条件が満たされる場合に行ってください。

- 以前のバージョンの Sun Cluster からアップグレードしている場合。
- HAStoragePlus リソースタイプの新機能を使用する必要がある場合。

リソースタイプをアップグレードする方法については、33 ページの「リソースタイプの更新」を参照してください。以下の各項では、HAStoragePlus リソースタイプのアップグレードに際して必要になる情報について説明します。

新しいリソースタイプバージョンの登録に関する情報

次の表に、リソースタイプのバージョンと Sun Cluster データサービスのリリースの関係を示します。Sun Cluster のリリースは、リソースタイプが導入されたバージョンを表します。

リソースタイプバージョン	**Sun Cluster** のリリース
1.0	3.0 5/02
2	3.1 9/04

登録されているリソースタイプのバージョンを調べるには、次のどちらかのコマンドを使用します。

- `scrgadm -p`
- `scrgadm -pv`

このリソースタイプのリソースタイプ登録 (RTR) ファイルは `/usr/cluster/lib/rgm/rtreg/SUNW.HAStoragePlus` です。

リソースタイプの既存インスタンスの移行に関する情報

`HAStoragePlus` リソースタイプのインスタンスを移行する際には、次の点に注意してください。

- 移行はいつでも実行できます。
- `HAStoragePlus` リソースタイプの新機能を使用する場合は、`Type_version` プロパティに設定する必要がある値は `2` です。

オンラインのリソースグループをクラスタノード間で分散する

可用性を最大化するため、あるいは、性能を最適化するため、いくつかのサービスの組み合わせは、特定のオンラインのリソースグループをクラスタノード間で分散する必要があります。オンラインのリソースグループを分散するということは、リソースグループ間でアフィニティを作成するということであり、次のような理由で行われます。

- 初めてリソースグループをオンラインにするときに必要な分散を強制的に実行するため
- リソースグループのフェイルオーバーまたはスイッチオーバーの後に必要な分散を保持しておくため

この節では、次のような例を使用しながら、リソースグループのアフィニティを使用して、オンラインのリソースグループをクラスタノード間で分散する方法について説明します。

- あるリソースグループと別のリソースグループを強制的に同じ場所に配置する
- あるリソースグループと別のリソースグループをできる限り同じ場所に配置する
- リソースグループの集合の負荷をクラスタノード間で均等に分配する
- 重要なサービスに優先権を指定する
- リソースグループのフェイルオーバーまたはスイッチオーバーを委託する
- リソースグループ間のアフィニティを組み合わせて、複雑な動作を指定する

リソースグループのアフィニティ

リソースグループ間のアフィニティは、複数のリソースグループが同時にオンラインになる可能性があるノードを制限します。各アフィニティにおいて、ソースのリソースグループには 1 つまたは複数のターゲットのリソースグループに対するアフィニティを宣言します。リソースグループ間にアフィニティを作成するには、ソースの `RG_affinities` リソースグループプロパティを次のように設定します。

`-y RG_affinities=`*operator target-rg-list*

注 – *operator* と *target-rg-list* の間にスペースを入れてはなりません。

---

operator 作成しようとしているアフィニティのタイプを指定します。詳細は、表 2–2を参照してください。

target-rg-list 作成しようとしているアフィニティのターゲットであるリソースグループをコンマで区切って指定します。このリストには、1 つまたは複数のリソースグループを指定できます。

表 2–2 リソースグループ間のアフィニティのタイプ

オペレータ	アフィニティのタイプ	効果
+	弱い肯定的なアフィニティ	ソースは、できる限り、ターゲットがオンラインである (あるいは、起動している) 1 つまたは複数のノード上でオンラインになります。つまり、ソースとターゲットは異なるノード上でオンラインになることもあります。
++	強い肯定的なアフィニティ	ソースは、ターゲットがオンラインである (あるいは、起動している) 1 つまたは複数のノード上でのみオンラインになります。つまり、ソースとターゲットは異なるノード上でオンラインになることはありません。
-	弱い否定的なアフィニティ	ソースは、可能であれば、ターゲットがオンラインでない (あるいは、起動していない) 1 つまたは複数のノード上でオンラインになります。つまり、ソースとターゲットは同じノード上でオンラインになることもあります。
--	強い否定的なアフィニティ	ソースは、ターゲットがオンラインでない 1 つまたは複数のノード上でのみオンラインになります。つまり、ソースとターゲットは同じノード上でオンラインになることはありません。
+++	フェイルオーバー委託付きの強い肯定的なアフィニティ	強い肯定的なアフィニティと似ていますが、ソースによるフェイルオーバーはターゲットに委託されます。詳細は、104 ページの「リソースグループのフェイルオーバーまたはスイッチオーバーを委託する」を参照してください。

弱いアフィニティは、`Nodelist` 優先順位より優先されます。

その他のリソースグループの現在の状態によっては、任意のノード上で、強いアフィニティが成立しないことがあります。このような状況では、アフィニティのソースであるリソースグループはオフラインのままです。その他のリソースグループの状態が変更され、強いアフィニティが成立できるようになると、アフィニティのソースであるリソースグループはオンラインに戻ります。

注 – 1 つのソースリソースグループに複数のターゲットリソースグループに対する強いアフィニティを宣言するときには注意してください。宣言されたすべての強いアフィニティが成立しない場合、ソースリソースグループはオフラインのままになるためです。

あるリソースグループと別のリソースグループを強制的に同じ場所に配置する

あるリソースグループのサービスが別のリソースグループのサービスに強く依存する場合、これらのリソースグループは両方とも同じノード上で動作する必要があります。たとえば、あるアプリケーションがお互いに依存する複数のサービスのデーモンから構成される場合、すべてのデーモンは同じノード上で動作する必要があります。

このような状況では、依存するサービスのリソースグループを、強制的に、依存されるサービスのリソースグループと同じ場所に配置するように指定します。あるリソースグループを強制的に別のリソースグループと同じ場所に配置するには、あるリソースグループに別のリソースグループに対する強い肯定的なアフィニティを宣言します。

```
# scrgadm -c|-a -g source-rg -y RG_affinities=++target-rg
```

`-g` *source-rg*
: 強い肯定的なアフィニティのソースであるリソースグループを指定します。このリソースグループは、別のリソースグループに対する強い肯定的なアフィニティを宣言するリソースグループです。

`-y RG_affinities=++`*target-rg*
: 強い肯定的なアフィニティのターゲットであるリソースグループを指定します。このリソースグループは、強い肯定的なアフィニティが宣言されるリソースグループです。

強い肯定的なアフィニティを宣言しているソースのリソースグループは、ターゲットのリソースグループに従います。しかし、強い肯定的なアフィニティを宣言しているソースのリソースグループは、ターゲットのリソースグループが動作していないノードにはフェイルオーバーできません。

注 – フェイルオーバーされないのは、リソースモニターが起動したフェイルオーバーだけです。ソースとターゲットの両方のリソースグループが動作しているノードに障害が発生した場合、これらのリソースグループは、正常に動作している同じノード上で再起動されます。

たとえば、リソースグループ `rg1` にリソースグループ `rg2` に対する強い肯定的なアフィニティが宣言されていると仮定します。 `rg2` が別のノードにフェイルオーバーすると`rg1` もそのノードにフェイルオーバーします。この場合、`rg1` 内のすべてのリソースが操作可能であるとしても、`rg1` はそのノードにフェイルオーバーします。しかし、`rg1` 内のリソースによって、`rg2` が動作していないノードに `rg1` がフェイルオーバーしようとした場合、このフェイルオーバーはブロックされます。

強い肯定的なアフィニティを宣言しているリソースグループをフェイルオーバーする必要がある場合、そのフェイルオーバーは委託する必要があります。詳細は、104 ページの「リソースグループのフェイルオーバーまたはスイッチオーバーを委託する」を参照してください。

例 **2–6** あるリソースグループと別のリソースグループを強制的に同じ場所に配置する

この例では、リソースグループ rg1 を変更して、リソースグループ rg2 に対する強い肯定的なアフィニティを宣言するためのコマンドを示します。このアフィニティを宣言すると、rg1 は rg2 が動作しているノード上だけでオンラインになります。この例では、両方のリソースグループが存在していると仮定します。

```
# scrgadm -c -g rg1 -y RG_affinities=++rg2
```

あるリソースグループと別のリソースグループをできる限り同じ場所に配置する

あるリソースグループのサービスが別のリソースグループのサービスを使用していることがあります。結果として、これらのサービスは、同じノード上で動作する場合にもっとも効率よく動作します。たとえば、データベースを使用するアプリケーションは、そのアプリケーションとデータベースが同じノード上で動作する場合に、もっとも効率よく動作します。しかし、これらのサービスは異なるノード上で動作してもかまいません。なぜなら、リソースグループのフェイルオーバーの増加よりも効率の低下のほうが被害が小さいためです。

このような状況では、両方のリソースグループを、できる限り、同じ場所に配置するように指定します。あるリソースグループと別のリソースグループをできる限り同じ場所に配置するには、あるリソースグループに別のリソースグループに対する弱い肯定的なアフィニティを宣言します。

```
# scrgadm -c|-a -g source-rg -y RG_affinities=+target-rg
```

-g *source-rg*
: 弱い肯定的なアフィニティのソースであるリソースグループを指定します。このリソースグループは、別のリソースグループに対する弱い肯定的なアフィニティを宣言するリソースグループです。

-y RG_affinities=+*target-rg*
: 弱い肯定的なアフィニティのターゲットであるリソースグループを指定します。このリソースグループは、弱い肯定的なアフィニティを宣言するリソースグループです。

あるリソースグループに別のリソースグループに対する弱い肯定的なアフィニティを宣言することによって、両方のリソースグループが同じノードで動作する確率が上がります。弱い肯定的なアフィニティのソースは、まず、そのアフィニティのターゲットがすでに動作しているノード上でオンラインになろうとします。しかし、弱い肯定的なアフィニティのソースは、そのアフィニティのターゲットがリソースモニターによってフェイルオーバーされても、フェイルオーバーしません。同様に、弱い肯定的なアフィニティのソースは、そのアフィニティのターゲットがスイッチオーバーされても、フェイルオーバーしません。どちらの状況でも、ソースがすでに動作しているノード上では、ソースはオンラインのままです。

注 – ソースとターゲットの両方のリソースグループが動作しているノードに障害が発生した場合、これらのリソースグループは、正常に動作している同じノード上で再起動されます。

例 **2–7** あるリソースグループと別のリソースグループをできる限り同じ場所に配置する

この例では、リソースグループ `rg1` を変更して、リソースグループ `rg2` に対する弱い肯定的なアフィニティを宣言するためのコマンドを示します。このアフィニティを宣言すると、`rg1` と `rg2` はまず、同じノード上でオンラインになろうとします。しかし、`rg2` 内のリソースによって `rg2` がフェイルオーバーしても、`rg1` は最初にオンラインになったノード上でオンラインのままです。この例では、両方のリソースグループが存在していると仮定します。

```
# scrgadm -c -g rg1 -y RG_affinities=+rg2
```

リソースグループの集合の負荷をクラスタノード間で均等に分配する

リソースグループの集合の各リソースグループには、クラスタの同じ負荷をかけることができます。このような状況では、リソースグループをクラスタ間で均等に分散することによって、クラスタの負荷の均衡をとることができます。

リソースグループの集合のリソースグループをクラスタノード間で均等に分散するには、各リソースグループに、リソースグループの集合のほかのリソースグループに対する弱い否定的なアフィニティを宣言します。

```
# scrgadm -c|-a -g source-rg -y RG_affinities=-target-rg-list
```

`-g` *source-rg*
: 弱い否定的なアフィニティのソースであるリソースグループを指定します。このリソースグループは、その他のリソースグループに対する弱い否定的なアフィニティを宣言するリソースグループです。

`-y RG_affinities=-`*target-rg-list*
: 弱い指定的なアフィニティのターゲットであるリソースグループをコンマで区切って指定します。これらのリソースグループは、弱い否定的なアフィニティが宣言されるリソースグループです。

あるリソースグループにその他のリソースグループに対する弱い否定的なアフィニティを宣言することによって、そのリソースグループが常に、もっとも負荷がかかっていないクラスタノード上でオンラインになることが保証されます。 このノード上で動作しているその他のリソースグループは最小数です。したがって、弱い否定的なアフィニティの最小数が違反されます。

例 **2–8** リソースグループの集合の負荷をクラスタノード間で均等に分配する

この例では、リソースグループ rg1、rg2、rg3、および rg4 を変更して、これらのリソースグループを利用可能なクラスタノード間で均等に分散するためのコマンドを示します。この例では、リソースグループ rg1、rg2、rg3、および rg4 が存在していると仮定します。

```
# scrgadm -c -g rg1 RG_affinities=-rg2,-rg3,-rg4
# scrgadm -c -g rg2 RG_affinities=-rg1,-rg3,-rg4
# scrgadm -c -g rg3 RG_affinities=-rg1,-rg2,-rg4
# scrgadm -c -g rg4 RG_affinities=-rg1,-rg2,-rg3
```

重要なサービスに優先権を指定する

クラスタは、重要なサービスと重要でないサービス組み合わせて動作するように構成できます。たとえば、重要な顧客サービスをサポートするデータベースは、重要でない研究タスクと同じクラスタで実行できます。

重要でないサービスが重要なサービスに影響を与えないようにするには、重要なサービスに優先権を指定します。重要なサービスに優先権を指定することによって、重要でないサービスが重要なサービスと同じノード上で動作することを防ぐことができます。

すべてのノードが操作可能であるとき、重要なサービスは重要でないサービスとは異なるノード上で動作します。しかし、重要なサービスに障害が発生すると、このサービスは重要でないサービスが動作しているノードにフェイルオーバーします。このような状況では、重要でないサービスは直ちにオフラインになり、重要なサービスはコンピューティングリソースを完全に利用できるようになります。

重要なサービスに優先権を指定するには、重要でない各サービスのリソースグループに、重要なサービスを含むリソースグループに対する強い否定的なアフィニティを宣言します。

```
# scrgadm -c|-a -g noncritical-rg -y RG_affinities=--critical-rg
```

`-g` *noncritical-rg*
重要でないサービスを含むリソースグループを指定します。このリソースグループは、別のリソースグループに対する強い否定的なアフィニティを宣言するリソースグループです。

`-y RG_affinities=--`*critical-rg*
重要なサービスを含むリソースグループを指定します。このリソースグループは、強い否定的なアフィニティが宣言されるリソースグループです。

強い否定的なアフィニティのソースのリソースグループは、そのアフィニティのターゲットのリソースグループから離れます。

例 **2–9** 重要なサービスに優先権を指定する

この例では、重要でないリソースグループ ncrg1 と ncrg2 を変更して、重要なリソースグループ mcdbrg に重要でないリソースグループよりも高い優先権を与えるためのコマンドを示します。この例では、リソースグループ mcdbrg、ncrg1、および ncrg2 が存在していると仮定します。

```
# scrgadm -c -g ncrg1 RG_affinities=--mcdbrg
# scrgadm -c -g ncrg2 RG_affinities=--mcdbrg
```

リソースグループのフェイルオーバーまたはスイッチオーバーを委託する

強い肯定的なアフィニティのソースリソースグループは、そのアフィニティのターゲットが動作していないノードにはフェイルオーバーまたはスイッチオーバーできません。強い肯定的なアフィニティのソースリソースグループをフェイルオーバーまたはスイッチオーバーする必要がある場合、そのフェイルオーバーはターゲットリソースグループに委託する必要があります。このアフィニティのターゲットがフェイルオーバーするとき、このアフィニティのソースはターゲットと一緒に強制的にフェイルオーバーされます。

注 – ++ オペレータで指定した強い肯定的なアフィニティのソースリソースグループでも、スイッチオーバーする必要がある場合もあります。このような状況では、このアフィニティのターゲットとソースを同時にスイッチオーバーします。

リソースグループのフェイルオーバーまたはスイッチオーバーを別のリソースグループに委託するには、そのリソースグループに、その他のリソースグループに対するフェイルオーバー委託付きの強い肯定的なアフィニティを宣言します。

```
# scrgadm -c|-a -g source-rg -y RG_affinities=+++target-rg
```

-g *source-rg*
フェイルオーバーまたはスイッチオーバーを委託するリソースグループを指定します。このリソースグループは、別のリソースグループに対するフェイルオーバー委託付きの強い肯定的なアフィニティを宣言するリソースグループです。

-y RG_affinities=+++*target-rg*
source-rg がフェイルオーバーまたはスイッチオーバーを委託するリソースグループを指定します。このリソースグループは、フェイルオーバー委託付きの強い肯定的なアフィニティが宣言されるリソースグループです。

あるリソースグループは、最大 1 つのリソースグループに対するフェイルオーバー委託付きの強い肯定的なアフィニティを宣言できます。逆に、あるリソースグループは、その他の任意の数のリソースグループによって宣言されたフェイルオーバー委託付きの強い肯定的なアフィニティのターゲットである可能性があります。

つまり、フェイルオーバー委託付きの強い肯定的なアフィニティは対照的ではありません。ソースがオフラインの場合でも、ターゲットはオンラインになることができます。しかし、ターゲットがオフラインの場合、ソースはオンラインになることができません。

ターゲットが第三のリソースグループに対するフェイルオーバー委託付きの強い肯定的なアフィニティを宣言する場合、フェイルオーバーまたはスイッチオーバーはさらに第三のリソースグループに委託されます。第三のリソースグループがフェイルオーバーまたはスイッチオーバーを実行すると、その他のリソースグループも強制的にフェイルオーバーまたはスイッチオーバーされます。

例 **2–10** リソースグループのフェイルオーバーまたはスイッチオーバーを委託する

この例では、リソースグループ `rg1` を変更して、リソースグループ `rg2` に対するフェイルオーバー委託付きの強い肯定的なアフィニティを宣言するためのコマンドを示します。このアフィニティを宣言すると、 `rg1` はフェイルオーバーまたはスイッチオーバーを `rg2` に委託します。この例では、両方のリソースグループが存在していると仮定します。

```
# scrgadm -c -g rg1 -y RG_affinities=+++rg2
```

リソースグループ間のアフィニティの組み合わせ

複数のアフィニティを組み合わせることによって、より複雑な動作を作成できます。たとえば、関連する複製サーバーにアプリケーションの状態を記録できます。この例におけるノード選択条件は次のとおりです。

- 複製サーバーは、アプリケーションと異なるノード上で動作している必要があります。
- アプリケーションが現在のノードからフェイルオーバーすると、アプリケーションは、複製サーバーが動作しているノードにフェイルオーバーする必要があります。
- アプリケーションが複製サーバーが動作しているノードにフェイルオーバーすると、複製サーバーは異なるノードにフェイルオーバーする必要があります。その他のノードが利用できない場合、複製サーバーはオフラインになる必要があります。

これらの条件を満たすには、アプリケーションと複製サーバーのリソースグループを次のように構成します。

- アプリケーションを含むリソースグループは、複製サーバーを含むリソースグループに対する弱い肯定的なアフィニティを宣言します。
- 複製サーバーを含むリソースグループは、アプリケーションを含むリソースグループに対する強い否定的なアフィニティを宣言します。

例 **2–11** リソースグループ間のアフィニティの組み合わせ

この例では、次のリソースグループ間のアフィニティを組み合わせるためのコマンドを示します。

- リソースグループ `app-rg` は、複製サーバーによって状態を追跡するアプリケーションを示します。

例 **2–11** リソースグループ間のアフィニティの組み合わせ　(続き)

- リソースグループ `rep-rg` は、複製サーバーを示します。

この例では、リソースグループはアフィニティを次のように宣言します。

- リソースグループ `app-rg` は、リソースグループ `rep-rg` に対する弱い肯定的なアフィニティを宣言します。
- リソースグループ `rep-rg` は、リソースグループ `app-rg` に対する強い否定的なアフィニティを宣言します。

この例では、両方のリソースグループが存在していると仮定します。

```
# scrgadm -c -g app-rg RG_affinities=+rep-rg
# scrgadm -c -g rep-rg RG_affinities=--app-rg
```

重要ではないリソースグループをオフロードすることによるノードリソースの解放

注 – 重要でないリソースグループをオフロードするもっとも簡単な方法は、リソースグループ間で強い否定的なアフィニティを使用することです。詳細は、98 ページの「オンラインのリソースグループをクラスタノード間で分散する」を参照してください。

Prioritized Service Management (`RGOffload`) を使用すると、重要なデータサービス用にノードのリソースを自動的に解放できます。`RGOffload` は、重要なフェイルオーバーデータサービスを起動するために、重要でないスケーラブルデータサービスまたはフェイルオーバーデータサービスをオフラインにする必要があるときに使用します。`RGOffload` は、重要でないデータサービスを含むリソースグループをオフロードするときに使用します。

注 – プライオリティが高いデータサービスはフェイルオーバー可能でなければなりません。オフロードするデータサービスは、フェイルオーバーデータサービスでもスケーラブルデータサービスでもかまいません。

▼ RGOffload リソースを設定する

1. クラスタメンバー上でスーパーユーザーになります。
2. **RGOffload** リソースタイプが登録されているかどうかを調べます。
   次のコマンドは、リソースタイプのリストを出力します。

   ```
   # scrgadm -p|egrep SUNW.RGOffload
   ```

3. 必要であれば、リソースタイプを登録します。

   ```
   # scrgadm -a -t SUNW.RGOffload
   ```

4. **RGOffload** リソースでオフロードするリソースグループごとに、**Desired_primaries** をゼロに設定します。

   ```
   # scrgadm -c -g offload-rg -y Desired_primaries=0
   ```

5. **RGOffload** リソースを重要なフェイルオーバーリソースグループに追加して、拡張プロパティを設定します。
   リソースグループを複数のリソースの rg_to_offload リストに追加してはいけません。リソースグループを複数の rg_to_offload リストに追加すると、リソースグループはオフラインになったあとにオンラインになるという動作を繰り返すことになります。
   拡張プロパティについては、109 ページの「RGOffload 拡張プロパティを構成する」を参照してください。

   ```
   # scrgadm -aj rgoffload-resource \
   -t SUNW.RGOffload -g critical-rg \
   -x rg_to_offload=offload-rg-1, offload-rg-2, ... \
   -x continue_to_offload=TRUE \
   -x max_offload_retry=15
   ```

   注 – この場合、rg_to_offload 以外の拡張プロパティはデフォルト値で表示されます。rg_to_offload は、お互いに依存しないリソースグループをコンマで区切ったリストです。このリストには、RGOffload リソースを追加するリソースグループを含めることはできません。
   ---

6. **RGOffload** リソースを有効にします。

   ```
   # scswitch -ej rgoffload-resource
   ```

7. 重要なフェイルオーバーリソースから **RGOffload** への依存関係を設定します。

   ```
   # scrgadm -c -j critical-resource \
   -y Resource_dependencies=rgoffload-resource
   ```

   Resource_dependencies_weak も使用できます。
   Resource_dependencies_weak を RGOffload リソースタイプに使用すると、offload-rg のオフロード中にエラーが発生しても、重要なフェイルオーバーリ

ソースを起動できます。

8. オフロードするリソースグループを、オンラインにします。

   `# scswitch -z -g` *offload-rg, offload-rg-2, ...* `-h [`*nodelist*`]`

   リソースグループは、プライオリティが高いリソースグループがオフラインであるすべてのノード上でオンラインのままになります。障害モニターは、重要なリソースグループがオンラインであるノード上でリソースグループが動作しないようにします。

   オフロードするリソースグループの `Desired_primaries` はゼロに設定されているので (手順 4を参照)、"-Z" オプションを指定しても、このようなリソースグループはオンラインになりません。

9. 重要なフェイルオーバーリソースグループがオンラインでない場合、オンラインにします。

   `# scswitch -Z -g` *critical-rg*

SPARC: 例 – `RGOffload` リソースを構成する

この例では、`RGOffload` リソース (`rgofl`)、RGOffload リソースを含む重要なリソースグループ (oracle_rg)、および重要なリソースグループがオンラインになったときにオフロードされるスケーラブルリソースグループ (`IWS-SC`, `IWS-SC-2`) を構成する方法について説明します。この例では、重要なリソースは `oracle-server-rs` です。

この例では、`oracle_rg`、`IWS-SC`、および `IWS-SC-2` は、クラスタ "triped"、phys-triped-1、phys-triped-2、phys-triped-3 の任意のノード上でマスターできます。

```
[SUNW.RGOffload リソースタイプが登録されているかどうかを判断する]
# scrgadm -p|egrep SUNW.RGOffload

[必要に応じて、リソースタイプを登録する]
# scrgadm -a -t SUNW.RGOffload

[RGOffload によってオフロードされる各リソースグループで、Desired_primaries をゼロに設定する]
# scrgadm -c -g IWS-SC-2 -y Desired_primaries=0
# scrgadm -c -g IWS-SC -y Desired_primaries=0

[プライオリティが高いリソースグループに RGOffload リソースを追加し、拡張プロパティを設定する]
# scrgadm -aj rgofl -t SUNW.RGOffload -g oracle_rg \
-x rg_to_offload=IWS-SC,IWS-SC-2 -x continue_to_offload=TRUE \
-x max_offload_retry=15

[RGOffload リソースを有効にする]
# scswitch -ej rgofl

[プライオリティが高いフェイルオーバーリソースの RGOffload リソースに対する依存性を設定する]
# scrgadm -c -j oracle-server-rs -y Resource_dependencies=rgofl

[オフロードされるリソースグループをすべてのノードでオンラインにする]
```

```
# scswitch -z -g IWS-SC,IWS-SC-2 -h phys-triped-1,phys-triped-2,phys-triped-3
```

[プライオリティが高いフェイルオーバーリソースグループがオンラインでない場合は、それをオンラインにする]

```
# scswitch -Z -g oracle_rg
```

RGOffload 拡張プロパティを構成する

通常、RGOffload リソースを作成するとき、拡張プロパティを構成するには、コマンド行 scrgadm -x *parameter=value* を使用します。Sun Cluster のすべての標準プロパティの詳細は、付録 A を参照してください。

表 2–3 に RGOffload に設定できる拡張プロパティを示します。「調整可能」エントリは、いつプロパティを更新できるかを示します。

表 **2–3** RGOffload 拡張プロパティ

名前/データタイプ	標準
rg_to_offload (文字列)	プライオリティが高いフェイルオーバーリソースグループがノード上で起動するときに、当該ノード上でオフロードする必要があるリソースグループをコンマで区切ったリスト。このリストには、互いに依存するリソースグループが含まれてはいけません。このプロパティにはデフォルト設定値がないので、必ず設定する必要があります。 RGOffload は、rg_to_offload 拡張プロパティに設定されたリソースグループのリストにおける依存関係ループを検査しません。たとえば、リソースグループ RG-B が RG-A に依存する場合、RG-A と RG-B が両方とも rg_to_offload に含まれてはいけません。 初期値: なし 調整: 任意の時点
continue_to_offload (ブール型)	リソースグループのオフロード中にエラーが発生した後に、rg_to_offload リスト内の残りのリソースグループをオフロードし続けるかどうかを示すブール型。 このプロパティは START メソッドだけが使用します。 初期値: True 調整: 任意の時点

表 **2–3** RGOffload 拡張プロパティ　(続き)

名前/データタイプ	標準
max_offload_retry (整数)	クラスタまたはリソースグループの再構成による障害時の起動中に、リソースグループをオフロードしようとする回数。再試行の間には 10 秒の間隔があります。 (オフロードされるリソースグループの数 * max_offload_retry * 10 秒) が RGOffload リソースの Start_timeout よりも小さくなるように max_offload_retry を設定します。この値が Start_timeout の値に近い (あるいは、より大きい) 場合、最大再試行数に到達する前に、RGOffload リソースの START メソッドがタイムアウトする可能性があります。 このプロパティは START メソッドだけが使用します。 初期値: 15 調整: 任意の時点

障害モニター

RGOffload リソースの障害モニター検証は、重要なリソースをマスターするノード上で、rg_to_offload 拡張プロパティに指定されたリソースグループをオフラインにし続けるために使用されます。各検証サイクルでフォルトモニターは、プライオリティが高いリソースをマスターするノード上で、オフロードされるリソースグループ (offload-rg) がオフラインであることを確認します。重要なリソースをマスターするノード上で offload-rg がオンラインである場合、障害モニターは重要なリソースをマスターするノード以外のノード上で offload-rg を起動し、同時に、重要なリソースをマスターするノード上では offload-rg をオフラインにしようとします。

offload-rg の desired_primaries はゼロに設定されているので、この後で利用可能になったノード上では、オフロードするリソースグループは再起動されません。したがって、RGOffload 障害モニターは maximum_primaries に到達するまで、重要なリソースをマスターするノード上では offload-rg をオフラインにしながら、可能な限りのプライマリ上で offload-rg を起動しようとします。

RGOffload は、MAINTENANCE 状態または UNMANAGED 状態でないかぎり、オフロードされたすべてのリソースを起動しようとします。リソースグループを UNMANAGED 状態にするには、scswitch コマンドを使用します。

```
# scswitch -u -g resourcegroup
```

フォルトモニター検証サイクルは、Thorough_probe_interval が実行されたあとにかならず呼び出されます。

リソースグループ、リソースタイプ、およびリソースの構成データを複製およびアップグレードする

2 つのクラスタ上で同じリソース構成データが必要である場合、このデータを 2 番目のクラスタに複製することによって、もう一度同じ設定を行うという面倒な作業を省略できます。scsnapshot を使用して、あるクラスタから別のクラスタにリソース構成情報をコピーします。設定後、問題が生じないように、リソース関係の構成が安定していることを確認します。2 番目のクラスタに情報をコピーする前に、リソース構成に大きな変更を行う必要はありません。

リソースグループ、リソースタイプ、およびリソースの構成データは、クラスタ構成リポジトリ (CCR) から取得でき、シェルスクリプトとして書式化されています。このスクリプトを使用すると、次の作業を実行できます。

- リソースグループ、リソース型、およびリソースが構成されていないクラスタに構成データを複製する
- リソースグループ、リソース型、およびリソースが構成されているクラスタの構成データをアップグレードする

scsnapshot ツールは、CCR に格納されている構成データを取得します。ほかの構成データは無視されます。scsnapshot ツールは、異なるリソースグループ、リソース型、およびリソースの動的な状態については考慮しません。

▼ リソースグループ、リソース型、およびリソースが構成されていないクラスタに構成データを複製する

この手順は、リソースグループ、リソース型、およびリソースが構成されていないクラスタに構成データを複製します。この手順では、あるクラスタから構成データのコピーを取得し、このデータを使用して、別のクラスタ上で構成データを生成します。

1. システム管理者役割を使用して、構成データをコピーしたいクラスタノードにログインします。

   たとえば、node1 にログインすると仮定します。

   システム管理者役割が与える役割によるアクセス制御 (RBAC) 権は、次のとおりです。

   - solaris.cluster.resource.read
   - solaris.cluster.resource.modify

2. クラスタから構成データを取得します。

   ```
   node1 % scsnapshot -s scriptfile
   ```

   scsnapshot ツールは、 *scriptfile* というスクリプトを生成します。scsnapshot ツールの使用方法についての詳細は、scsnapshot(1m) のマニュアルページを参照してください。

3. このスクリプトを編集して、構成データを複製したいクラスタに固有な特徴に合わせます。

   たとえば、スクリプト内にある IP アドレスやホスト名を変更します。

4. このスクリプトを、構成データを複製したい任意のクラスタノードから実行します。

   このスクリプトは、スクリプトが生成されたクラスタとローカルクラスタの特性を比較します。これらの特性が同じでない場合、このスクリプトはエラーを書き込んで終了します。次に、-f オプションを使用してスクリプトを実行し直すかどうかをたずねるメッセージが表示されます。-f オプションを使用した場合、上記のような特性の違いを無視して、スクリプトを強制的に実行します。-f オプションを使用した場合、クラスタ内に不整合がないことを確認します。

   このスクリプトは、Sun Cluster リソースタイプがローカルクラスタ上に存在することを確認します。リソース型がローカルクラスタに存在しない場合、このスクリプトはエラーを書き込んで終了します。もう一度スクリプトを実行する前に、存在しないリソース型をインストールするかどうかをたずねるメッセージが表示されます。

▼ リソースグループ、リソース型、およびリソースが構成されているクラスタの構成データをアップグレードする

この手順は、リソースグループ、リソース型、およびリソースがすでに構成されているクラスタ上の構成データをアップグレードします。この手順は、リソースグループ、リソース型、およびリソースの構成テンプレートを生成するのにも使用できます。

この手順では、cluster1 上の構成データが cluster2 上の構成データに一致するようにアップグレードされます。

1. システム管理者役割を使用して、**cluster1** の任意のノードにログインします。

   たとえば、node1 にログインすると仮定します。

   システム管理者役割が与える RBAC 権は次のとおりです。

   - solaris.cluster.resource.read
   - solaris.cluster.resource.modify

2. **scsnapshot** ツールの **image file** オプションを使用して、クラスタから構成データを取得します。

   ```
   node1% scsnapshot -s scriptfile1 -o imagefile1
   ```

   node1 上で実行するとき、scsnapshot ツールは *scriptfile1* というスクリプトを生成します。このスクリプトは、リソースグループ、リソース型、およびリソースの構成データを *imagefile1* というイメージファイルに格納します。scsnapshot ツールの使用方法についての詳細は、scsnapshot(1M) のマニュアルページを参照してください。

3. **cluster2** のノード上で、手順 **1** から 手順 **2** までの手順を繰り返します。

   ```
   node2 % scsnapshot -s scriptfile2 -o imagefile2
   ```

4. **node1** 上で **cluster2** の構成データを使用して、**cluster1** の構成データをアップグレードするためのスクリプトを生成します。

   ```
   node1 % scsnapshot -s scriptfile3 imagefile1 imagefile2
   ```

   この手順では、手順 2と手順 3で生成したイメージファイルを使用して、 *scriptfile3* という新しいスクリプトを生成します。

5. 手順 **4**で生成したスクリプトを編集して、**cluster1** に固有な特徴に合わせて、**cluster2** に固有なデータを削除します。

6. このスクリプトを **node1** から実行して、構成データをアップグレードします。

   このスクリプトは、スクリプトが生成されたクラスタとローカルクラスタの特性を比較します。これらの特性が同じでない場合、このスクリプトはエラーを書き込んで終了します。次に、-f オプションを使用してスクリプトを実行し直すかどうかをたずねるメッセージが表示されます。-f オプションを使用した場合、上記のような特性の違いを無視して、スクリプトを強制的に実行します。-f オプションを使用した場合、クラスタ内に不整合がないことを確認します。

   このスクリプトは、Sun Cluster リソースタイプがローカルクラスタ上に存在することを確認します。リソース型がローカルクラスタに存在しない場合、このスクリプトはエラーを書き込んで終了します。もう一度スクリプトを実行する前に、存在しないリソース型をインストールするかどうかをたずねるメッセージが表示されます。

Sun Cluster データベース用に障害モニターを調整する

Sun Cluster 製品で提供されるデータサービスには、障害モニターが組み込まれています。障害モニターは、次の機能を実行します。

- データサービスサーバーのプロセスの予期せぬ終了を検出する

- データサービスの健全性の検査

障害モニターは、データサービスが作成されたアプリケーションを表現するリソースに含まれます。このリソースは、データサービスを登録および構成したときに作成します。詳細は、データサービスのマニュアルを参照してください。

障害モニターの動作は、このリソースのシステムプロパティと拡張プロパティによって制御されます。事前に設定された障害モニターの動作は、これらのプロパティのデフォルト値に基づいています。現在の動作は、ほとんどの Sun Cluster システムに適しているはずです。したがって、障害モニターを調整するのは、事前に設定されたこの動作を変更したい場合「だけに」留めるべきです。

障害モニターを調整するには、次の作業が含まれます。

- 障害モニターの検証間隔を設定する。
- 障害モニターの検証タイムアウトを設定する。
- 継続的な障害とみなす基準を定義する。
- リソースのフェイルオーバー動作を指定する。

これらの作業は、データサービスの登録と構成の際に行います。詳細は、データサービスのマニュアルを参照してください。

注 – リソースの障害モニターは、そのリソースを含むリソースグループをオンラインにしたときに起動されます。障害モニターを明示的に起動する必要はありません。

障害モニターの検証間隔の設定

リソースが正しく動作しているかどうかを判断するには、障害モニターで当該リソースを定期的に検証します。障害モニターの検証間隔は、リソースの可用性とシステムの性能に次のような影響を及ぼします。

- 障害モニターの検証間隔は、障害の検出とその障害への対応にどの程度の時間がかかるかに影響を与えます。したがって、障害モニターの検証間隔を短くすると、障害の検出とその障害への対応にかかる時間も短くなります。このような時間の短縮は、リソースの可用性が向上することを意味します。
- 障害モニターの検証では、プロセッササイクルやメモリなどのシステムリソースが使用されます。したがって、障害モニターの検証間隔を短くすると、システムの性能は低下します。

さらに、障害モニターの最適な検証間隔は、リソースの障害への対応にどの程度の時間が必要かによって異なります。この時間は、リソースの複雑さが、リソースの再起動などの操作にかかる時間にどのような影響を及ぼすかに依存します。

障害モニターの検証間隔を設定するには、リソースの `Thorough_probe_interval` システムプロパティを必要な間隔 (秒単位) に設定します。

障害モニターの検証タイムアウトの設定

障害モニターの検証タイムアウトでは、検証に対するリソースからの応答にどのくらいの時間を許すかを指定します。このタイムアウト内にリソースからの応答がないと、障害モニターは、このリソースに障害があるものとみなします。障害モニターの検証に対するリソースの応答にどの程度の時間がかかるかは、障害モニターがこの検証に使用する操作によって異なります。データサービスの障害モニターがリソースを検証するために実行する操作については、データサービスのマニュアルを参照してください。

リソースの応答に要する時間は、障害モニターやアプリケーションとは関係のない次のような要素にも依存します。

- システム構成
- クラスタ構成
- システム負荷
- ネットワークトラフィックの量

障害モニターの検証タイムアウトを設定する場合は、必要なタイムアウト値をリソースの `Probe_timeout` 拡張プロパティに秒単位で指定します。

継続的な障害とみなす基準の定義

一時的な障害による中断を最小限に抑えるために、障害モニターは、このような障害が発生するとこのリソースを再起動します。継続的な障害の場合は、リソースの再起動よりも複雑なアクションをとる必要があります。

- フェイルオーバーリソースの場合は、障害モニターがこのリソースを別のノードにフェイルオーバーします。
- スケーラブルリソースの場合は、障害モニターがこのリソースをオフラインにします。

障害モニターは、指定された再試行間隔の中で、リソースの完全な障害の回数が、指定されたしきい値を超えると障害を継続的であるとみなします。ユーザーは、継続的な障害とみなす基準を定義することによって、 可用性要件とクラスタの性能特性を満たすしきい値や再試行間隔を設定できます。

リソースの完全な障害と部分的な障害

障害モニターは、いくつかの障害を、リソースの「完全な障害」としてみなします。完全な障害は通常、サービスの完全な損失を引き起こします。次に、完全な障害の例を示します。

- データサービスサーバーのプロセスの予期せぬ終了
- 障害モニターがデータサービスサーバーに接続できない

完全な障害が発生すると、障害モニターは再試行間隔内の完全な障害の回数を 1 つ増やします。

障害モニターは、それ以外の障害を、リソースの「部分的な障害」としてみなします。部分的な障害は完全な障害よりも重大ではなく、通常、サービスの低下を引き起こしますが、サービスの完全な損失は引き起こしません。次に、障害モニターがタイムアウトするまでにデータサービスサーバーからの応答が不完全であるという部分的な障害の例を示します。

部分的な障害が発生すると、障害モニターは再試行間隔内の完全な障害の回数を小数点数だけ増やします。部分的な障害は、再試行間隔を過ぎても累積されます。

部分的な障害の次の特性は、データサービスに依存します。

- 障害モニターが部分的な障害とみなす障害のタイプ
- それぞれの部分的な障害が完全な障害の回数に追加する小数点数

データサービスの障害モニターが検出する障害については、データサービスのマニュアルを参照してください。

しきい値や再試行間隔と他のプロパティとの関係

障害のあるリソースが再起動するのに必要な最大時間は、次のプロパティの値を合計したものです。

- `Thorough_probe_interval` システムプロパティ
- `Probe_timeout` 拡張プロパティ

再試行回数がしきい値に達しないうちに再試行間隔がきてしまうのを避けるためには、再試行間隔としきい値の値を次の式に従って計算します。

retry-interval ≥ *threshold* × (*thorough-probe-interval* + *probe-timeout*)

しきい値と再試行間隔を設定するシステムプロパティ

しきい値と再試行間隔を設定するには、リソースの次のようなシステムプロパティを使用します。

- しきい値を設定するには、`Retry_count` システムプロパティを完全な障害の最大値に設定します。
- 再試行間隔を設定する場合には、`Retry_interval` システムプロパティに、必要な間隔を秒数で指定します。

リソースのフェイルオーバー動作を指定する。

リソースのフェイルオーバー動作は、次の障害に対して RGM がどのように応答するかを決定します。

- リソースの起動の失敗

- リソースの停止の失敗
- リソースの障害モニターの停止の失敗

リソースのフェイルオーバー動作を指定するには、リソースの `Failover_mode` システムプロパティを設定します。このプロパティに指定できる値については、126 ページの「リソースのプロパティ」 における `Failover_mode` システムプロパティの説明を参照してください。

付録 A

標準プロパティ

この付録では、標準リソースタイプ、リソースグループ、リソースプロパティについて説明します。また、システム定義プロパティの変更および拡張プロパティの作成に使用するリソースプロパティ属性についても説明します。

この付録は、次の節で構成されています。

リソースタイププロパティ

この節では、Sun Cluster で定義されているリソースタイププロパティについて説明します。プロパティ値は、次のように分類されます (「カテゴリ」の後)。

- 必須 -- プロパティはリソースタイプ登録 (RTR) ファイルに明示的な値を必要とします。そうでない場合、プロパティが属するオブジェクトは作成できません。空白文字または空の文字列を値として指定することはできません。
- 条件付 — このプロパティが存在するためには、RTR ファイル内で宣言する必要があります。宣言されていない場合は、RGM はこのプロパティを作成しないため、管理ユーティリィティで利用できません。空白文字または空の文字列を値として指定できます。プロパティが RTR ファイル内で宣言されており、値が指定されていない場合には、RGM はデフォルト値を使用します。
- 条件付/明示 — このプロパティが存在するためには、明示的に値を指定し、RTR ファイル内で宣言する必要があります。宣言されていない場合は、RGM はこのプロパティを作成しないため、管理ユーティリィティで利用できません。空白文字または空の文字列を値として指定することはできません。

- 任意 — プロパティを RTR ファイル内で宣言できます。宣言しない場合は、RGM はこのプロパティを作成し、デフォルト値を使用します。プロパティが RTR ファイル内で宣言されており、値が指定されていない場合は、RGM は、プロパティが RTR ファイル内で宣言されないときのデフォルト値と同じ値を使用します。

リソースタイププロパティは管理ユーティリティで更新できません。ただし、`Installed_nodes` と `RT_system` は RTR ファイル内で宣言できないため、管理者が設定する必要があります。

最初にプロパティ名が表示され、次に説明が表示されます。

`API_version` (`integer`)
このリソースタイプの実装が使用するリソース管理 API のバージョン。

次に、Sun Cluster の各リリースがサポートする `API_version` の最大値を要約します。

3.1 以前　2

3.1 10/03　3

3.1 4/04　4

3.1 9/04　5

RTR ファイルにおいて、あるリソースタイプの `API_version` に 2 より大きな値を宣言した場合、そのリソースタイプは、宣言した値より小さな最大値しかサポートしないバージョンの Sun Cluster にはインストールされません。たとえば、あるリソースタイプに `API_version=5` を宣言した場合、そのリソースタイプは、3.1 9/04 より前にリリースされたバージョンの Sun Cluster にはインストールされません。

カテゴリ:　オプション

初期値:　2

調整:　いいえ

`Boot` (`string`)
任意のコールバックメソッド。RGM がノード上で呼び出すプログラムのパスを指定します。このプログラムは、このリソース型が管理対象になっているとき、クラスタの結合または再結合を行います。このメソッドは、`Init` メソッドと同様に、このタイプのリソースに対し、初期化アクションを行う必要があります。

カテゴリ:　条件付き / 明示

初期値:　なし

調整:　いいえ

`Failover` (`boolean`)
`TRUE` の場合、複数のノード上で同時にオンラインにできるグループ内にこの型のリソースを構成することはできません。

カテゴリ: オプション

初期値: FALSE

調整: いいえ

Fini (string)
任意のコールバックメソッド。この型のリソースを RGM 管理の対象外にするとき RGM によって呼び出されるプログラムのパスです。

カテゴリ: 条件付き / 明示

初期値: なし

調整: いいえ

Init (string)
任意のコールバックメソッド。この型のリソースを RGM 管理対象にするとき RGM によって呼び出されるプログラムのパスです。

カテゴリ: 条件付き / 明示

初期値: なし

調整: いいえ

Init_nodes (enum)
指定できる値は、RG_primaries (リソースをマスターできるノードのみ) または RT_installed_nodes (このリソース型がインストールされる全てのノード) のいずれかです。RGM が Init、Fini、 Boot、Validate メソッドをコールするノードを示します。

カテゴリ: オプション

初期値: RG_primaries

調整: いいえ

Installed_nodes (string_array)
リソースタイプの実行が許可されるクラスタノード名のリスト。このプロパティは RGM によって自動的に作成されます。クラスタ管理者は値を設定できます。RTR ファイル内には宣言できません。

カテゴリ: クラスタ管理者による構成が可能です。

初期値: すべてのクラスタノード

調整: Yes

Is_logical_hostname (boolean
TRUEは、このリソース型が、フェイルオーバーインターネットプロトコル (IP) アドレスを管理する LogicalHostname リソース型のいずれかのバージョンであることを示します。

カテゴリ: 照会のみ

初期値: デフォルトなし

調整: いいえ

`Is_shared_address` (`boolean`)
`TRUE` は、このリソース型が、 フェイルオーバーインターネットプロトコル (IP) アドレスを管理する `SharedAddress` リソース型のいずれかのバージョンであることを示します。

カテゴリ: 照会のみ

初期値: デフォルトなし

調整: いいえ

`Monitor_check` (`string`)
任意のコールバックメソッド。障害モニターの要求によってこのリソース型のフェイルオーバーを実行する前に、RGM によって呼び出されるプログラムのパスです。

カテゴリ: 条件付き / 明示

初期値: デフォルトなし

調整: いいえ

`Monitor_start` (`string`)
任意のコールバックメソッド。この型のリソースの障害モニターを起動するために RGM によって呼び出されるプログラムのパスです。

カテゴリ: 条件付き / 明示

初期値: デフォルトなし

調整: いいえ

`Monitor_stop` (`string`)
`Monitor_start` が設定されている場合の、必須のコールバックメソッド。この型のリソースの障害モニターを停止するために RGM によって呼び出されるプログラムのパスです。

カテゴリ: 条件付き / 明示

初期値: デフォルトなし

調整: いいえ

`Pkglist` (`string_array`)
リソース型のインストールに含まれている任意のパッケージリストです。

カテゴリ: 条件付き / 明示

初期値: デフォルトなし

調整: いいえ

`Postnet_stop` (`string`)
任意のコールバックメソッド。この型のリソースがネットワークアドレスリソースに依存している場合、このネットワークアドレスリソースの`Stop`メソッドの呼び出し後に RGM によって呼び出されるプログラムのパスです。ネットワークインタフェースが停止するように構成された後、このメソッドは `Stop` アクションを実行する必要があります。

カテゴリ: 条件付き / 明示

初期値: デフォルトなし

調整: いいえ

`Prenet_start` (`string`)
任意のコールバックメソッド。この型のリソースがネットワークアドレスリソースに依存している場合、このネットワークアドレスリソースの `Start` メソッドの呼び出し前に RGM によって呼び出されるプログラムのパスです。ネットワークインタフェースが構成される前、このメソッドは `Start` アクションを実行する必要があります。

カテゴリ: 条件付き / 明示

初期値: デフォルトなし

調整: いいえ

`Resource_type` (`string`)
リソースタイプの名前。現在登録されているリソースタイプ名を表示するには、次のコマンドを使用します。

`scrgadm -p`

Sun Cluster 3.1 以降のリリースでは、リソースタイプ名にバージョンが含まれます (必須)。

`vendor_id.resource_type:version`

リソース型の名前は、 RTR ファイル内に指定された 3 つのプロパティ *Vendor_id*、*Resource_type* 、*RT_version* で構成されます。`scrgadm` コマンドは、ピリオド (.) とコロン (:) の区切り文字を挿入します。リソースタイプ名の最後の部分、`RT_version` には、`RT_version` プロパティと同じ値が入ります。重複を防ぐため、*Vendor_id* には、リソースタイプの作成元の会社のストックシンボルを使用することをお勧めします。Sun Cluster 3.1 以前に作成されたリソースタイプ名では、引き続き次の構文を使用します。

`vendor_id.resource_type`

カテゴリ: 必須

初期値: 空の文字列

調整: いいえ

`RT_basedir` (`string`)
コールバックメソッドの相対パスのを補完するディレクトリパスです。このパスは、リソースタイプパッケージのインストール場所に設定します。スラッシュ (/) で開始する完全なパスを指定する必要があります。すべてのメソッドパス名が絶対パスの場合は、指定しなくてもかまいません。

カテゴリ: 必須 (絶対パスでないメソッドパスがある場合)

初期値: デフォルトなし

調整: いいえ

`RT_description` (`string`)
リソース型の簡単な説明です。

カテゴリ: 条件付き

初期値: 空の文字列

調整: いいえ

`RT_system` (`boolean`)
あるリソースタイプの `RT_system` が `TRUE` に設定されているときに、そのリソースタイプでは、許可されている `scrgadm(1M)` 操作が制限されることを示します。`RT_system` が `TRUE` に設定されているリソースタイプのことを「システムリソースタイプ」と呼びます。`RT_system` の現在の値に関わらず、 `RT_system` プロパティ自身を編集することは制限されません。

カテゴリ: オプション

初期値: `FALSE`

調整: Yes

`RT_version` (`string`)
Sun Cluster 3.1 以降では、このリソースタイプの実装の必須バージョン文字列。`RT_version` は、完全なリソースタイプ名の末尾の部分です。`RT_version` プロパティは Sun Cluster 3.0 では任意でしたが、Sun Cluster 3.1 以降のリリースでは必須です。

カテゴリ: 任意/明示または必須

初期値: デフォルトなし

調整: いいえ

`Single_instance` (`boolean`)
`TRUE` は、この型のリソースがクラスタ内に 1 つだけ存在できることを示します。つまり、この型のリソースが実行されるのは、クラスタ全体で 1 箇所だけです。

カテゴリ: オプション

初期値: `FALSE`

調整: いいえ

`Start` (`string`)
コールバックメソッド。この型のリソースを起動するために RGM によって呼び出されるプログラムのパスです。

カテゴリ: RTR ファイルで `Prenet_start` メソッドが宣言されていないかぎり必須

初期値: デフォルトなし

調整: いいえ

`Stop` (`string`)
コールバックメソッド。この型のリソースを停止するために RGM によって呼び出されるプログラムのパスです。

カテゴリ: RTR ファイルで `Postnet_stop` メソッドが宣言されていないかぎり必須

初期値: デフォルトなし

調整: いいえ

`Update` (`string`)
任意のコールバックメソッド。この型の実行中のリソースのプロパティが変更されたとき RGM によって呼び出されるプログラムのパスです。

カテゴリ: 条件付き / 明示

初期値: デフォルトなし

調整: いいえ

`Validate` (`string`)
任意のコールバックメソッド。この型のリソースのプロパティ値を検査するために呼び出されるプログラムのパスです。

カテゴリ: 条件付き / 明示

初期値: デフォルトなし

調整: いいえ

`Vendor_ID` (`string`)
`Resource_type` を参照してください。

カテゴリ: 条件付き

初期値: デフォルトなし

調整: いいえ

リソースのプロパティ

この節では、Sun Cluster で定義されているリソースプロパティについて説明します。プロパティ値は、次のように分類されます (「カテゴリ」の後)。

- 必須 — 管理者は、管理ユーティリティでリソースを作成するときに、必ず値を指定する必要があります。
- 任意 — 管理者がリソースグループの作成時に値を指定しない場合、システムがデフォルト値を提供します。
- 条件付き – プロパティが **RTR** ファイルで宣言されている場合にのみ、**RGM** がプロパティを作成します。宣言されていない場合プロパティは存在せず、システム管理者はこれを利用できません。RTR ファイルで宣言されている条件付きのプロパティは、デフォルト値が RTR ファイル内で指定されているかどうかによって、必須または任意になります。詳細については、各条件付きプロパティの説明を参照してください。
- 照会のみ– 管理ツールから直接設定できません。

調整は、次のように、リソースプロパティを更新できるかどうか、および、いつ更新できるかを示します。

`NONE` または `FALSE`	Never
`TRUE` または `ANYTIME`	任意の時点 (Anytime)
`AT_CREATION`	リソースをクラスタに追加するとき
`WHEN_DISABLED`	リソースが無効なとき

最初にプロパティ名が表示され、次に説明が表示されます。

`Affinity_timeout` (`integer`)
リソース内のサービスのクライアント IP アドレスからの接続は、この時間 (秒数) 内に同じサーバーノードに送信されます。

このプロパティは、`Load_balancing_policy` が `Lb_sticky` または `Lb_sticky_wild` の場合にかぎり有効です。さらに、`Weak_affinity` が `FALSE` (デフォルト値) に設定されている必要があります。

このプロパティは、スケーラブルサービス専用です。

カテゴリ: オプション

初期値: デフォルトなし

調整: `ANYTIME`

Cheap_probe_interval (integer)
リソースの即時障害検証の呼び出しの間隔 (秒数)。このプロパティは RGM によって作成されます。RTR ファイルに宣言されている場合にかぎり、管理者は使用を許可されます。

RTR ファイル内でデフォルト値が指定されている場合、このプロパティは任意です。Tunable 属性がリソースタイプファイルに指定されていない場合は、プロパティの Tunable 値は WHEN_DISABLED になります。

RTR ファイル内で宣言され、Default 属性が指定されていない場合、このプロパティは必須になります。

カテゴリ: 条件付き

初期値: デフォルトなし

調整: WHEN_DISABLED

拡張プロパティ
そのリソースのタイプの RTR ファイルで宣言される拡張プロパティ。リソースタイプの実装によって、これらのプロパティを定義します。拡張プロパティに設定可能な各属性については、143 ページの「リソースプロパティの属性」を参照してください。

カテゴリ: 条件付き

初期値: デフォルトなし

調整: 特定のプロパティに依存

Failover_mode (enum)
開始メソッド (Prenet_start または Start) が失敗した場合、NONE、SOFT、および HARD はフェイルオーバーの動作だけに影響します。しかし、リソースが正常に起動した場合、 NONE、SOFT 、および HARD は、これ以降にリソースモニターが scha_control (1HA) または scha_control (3HA) で発行したリソースの再起動またはギブオーバーの動作には影響しません。NONE (デフォルト) は、メソッドが失敗した場合に、RGM がリソースの状態を設定し、ユーザーが介入するまで待機することを示します。SOFT は、Start メソッドが失敗した場合に、RGM がリソースグループを異なるノードに移動することを示します。Stop メソッドまたは Monitor_stop メソッドが失敗した場合、RGM はリソースをStop_failed 状態に設定して、リソースグループを Error_stop_failed 状態に設定します。その後、RGM はユーザーが介入するまで待機します。Stop メソッドまたは Monitor_stop メソッドが失敗した場合、NONE と SOFT は同じです。HARD は、Start メソッドが失敗した場合に、RGM がグループを移動することを示します。Stop メソッドまたは Monitor_stop メソッドが失敗した場合、RGM はクラスタノードを中断して、リソースを停止します。Start メソッドまたは Prenet_start メソッドが失敗した場合、HARD、NONE、および SOFT はフェイルオーバーの動作に影響します。

NONE、SOFT、および HARD とは異なり、RESTART_ONLY および LOG_ONLY はすべてのフェイルオーバーの動作に影響します (モニターが scha_control で発行したリソースとリソースグループの再起動、および、リソースモニターが

scha_control で起動したギブオーバーを含む)。RESTART_ONLY は、モニターが scha_control でリソースを再起動できることを示します。しかし、これ以降に scha_control で発行したリソースグループの再起動またはギブオーバーは失敗します。このとき、RGM は Retry_interval 内に Retry_count 回の再起動を許可します。Retry_count を超えた場合、リソースの再起動は許可されません。Failover_mode が LOG_ONLY に設定されている場合、リソースの再起動またはギブオーバーは許可されません。 Failover_mode を LOG_ONLY に設定することは、 Failover_mode を RESTART_ONLY に設定し、 Retry_count をゼロに設定することと同じです。開始メソッドが失敗した場合、RESTART_ONLY および LOG_ONLY は NONE と同じです。つまり、フェイルオーバーは発生せず、リソースは Start_failed 状態になります。

カテゴリ: オプション

初期値: デフォルトなし

調整: ANYTIME

Load_balancing_policy (string)
使用する負荷均衡ポリシーを定義する文字列。このプロパティは、スケーラブルサービス専用です。RTR ファイルに Scalable プロパティが宣言されている場合、RGM は自動的にこのプロパティを作成します。Load_balancing_policy には次の値を設定できます。

Lb_weighted (デフォルト)。Load_balancing_weights プロパティで設定されているウエイトに従って、さまざまなノードに負荷が分散されます。

Lb_sticky。スケーラブルサービスの指定のクライアント (クライアントの IP アドレスで識別される) は、常に同じクラスタノードに送信されます。

Lb_sticky_wild。指定のクライアントの IP アドレスは、ワイルドカードスティッキーサービスの IP アドレスに接続され、IP アドレスが着信したポート番号とは無関係に、常に同じクラスタノードに送信されます。

カテゴリ: 条件付き/任意

初期値: Lb_weighted

調整: AT_CREATION

Load_balancing_weights (string_array)
このプロパティは、スケーラブルサービス専用です。RTR ファイルに Scalable プロパティが宣言されている場合、RGM は自動的にこのプロパティを作成します。形式は、「*weight@node,weight@node*」になります。ここで、*weight* は、指定したノード (*node*) に対する負荷分散の相対的な割り当てを示す整数になります。ノードに分散される負荷の割合は、すべてのウエイトの合計でこのノードのウエイトを割った値になります。たとえば、1@1,3@2 は、ノード 1 が負荷の 1/4 を受け取り、ノード 2 が 3/4 を受け取ることを指定します。空の文字列 (“”) は、負荷を均一に分散することを意味します (デフォルト)。明示的にウエイトを割り当てられていないノードのウエイトは、デフォルトで 1 になります。

Tunable 属性がリソースタイプファイルに指定されていない場合は、プロパティの Tunable 値は ANYTIME (任意の時点) になります。このプロパティを変更すると、新しい接続時にのみ分散が変更されます。

カテゴリ: 条件付き/任意

初期値: 空の文字列 (“”)

調整: ANYTIME

各コールバックメソッドの *method*_timeout (integer)
RGM がメソッドの呼び出しに失敗したと判断するまでの時間 (秒)。

カテゴリ: 条件付き/任意

初期値: RTR ファイルにメソッド自体が宣言されている場合は 3,600 (1 時間)

調整: ANYTIME

Monitored_switch (enum)
クラスタ管理者が管理ユーティリティを使用してモニターを有効または無効にすると、RGM によって Enabled または Disabled に設定されます。Disabled に設定されると、再び有効に設定されるまで、モニターは Start メソッドを呼び出しません。リソースが、モニターのコールバックメソッドを持っていない場合は、このプロパティは存在しません。

カテゴリ: 照会のみ

初期値: デフォルトなし

調整: Never

Network_resources_used (string_array)
リソースが使用する論理ホスト名または共有アドレスネットワークリソースのリスト。スケーラブルサービスの場合、このプロパティは別のリソースグループに存在する共有アドレスリソースを参照する必要があります。フェイルオーバーサービスの場合、このプロパティは同じリソースグループに存在する論理ホスト名または共有アドレスを参照します。RTR ファイルに Scalable プロパティが宣言されている場合、RGM は自動的にこのプロパティを作成します。Scalable が RTR ファイルで宣言されていない場合、Network_resources_used は RTR ファイルで明示的に宣言されていない限り使用できません。

Tunable 属性がリソースタイプファイルに指定されていない場合は、プロパティの Tunable 値は AT_CREATION になります。

注 – このプロパティを CRNP 向けに設定する方法については、SUNW.Event(5) のマニュアルページを参照してください。

カテゴリ: 条件付き/必須

初期値: デフォルトなし

調整: AT_CREATION

各クラスタノード上の Num_resource_restarts (整数)
このプロパティは、RGM によって、このノード上のこのリソースに対して過去 *n* 秒以内に行われた scha_control 、Resource_restart、または Resource_is_restarted の呼び出し回数に設定されるため、直接は設定できません。ここで、*n* はリソースの Retry_interval プロパティの値です。このリソースが scha_control ギブオーバーを実行した場合、ギブオーバーが成功または失敗したかに関わらず、リソースの再起動カウンタは RGM によってゼロにリセットされます。

リソース型が Retry_interval プロパティを宣言していない場合、この型のリソースに Num_resource_restarts プロパティを使用できません。

カテゴリ: 照会のみ

初期値: デフォルトなし

調整: いいえ

各クラスタノード上の Num_rg_restarts (integer)
このプロパティは、RGM によって、このノード上のこのリソースに対して過去 *n* 秒以内に行われた scha_control Restart の呼び出し回数に設定されるため、直接は設定できません。ここで、*n* はリソースの Retry_interval プロパティの値です。リソース型が Retry_interval プロパティを宣言していない場合、この型のリソースに Num_resource_restarts プロパティを使用できません。

カテゴリ: 説明を参照

初期値: デフォルトなし

調整: いいえ

On_off_switch (enum)
クラスタ管理者が管理ユーティリィティを使用してリソースを有効または無効にすると、RGM によって Enabled または Disabled に設定されます。無効にした場合、リソースはオフラインになり、もう一度有効にするまで、コールバックは呼び出されません。

カテゴリ: 照会のみ

初期値: デフォルトなし

調整: Never

Port_list (string_array)
サーバーが待機するポートの番号リストです。各ポート番号の後ろには、スラッシュ (/) とそのポートが使用しているプロトコルが続きます (たとえば、Port_list=80/tcp または Port_list=80/tcp6,40/udp6)。指定できるプロトコル値は次のとおりです。

- tcp (TCP IPv4 の場合)
- tcp6 (TCP IPv6 の場合)
- udp (UDP IPv4 の場合)
- udp6 (UDP IPv6 の場合) Scalable プロパティが RTR ファイルで宣言されている場合、RGM は自動的に Port_list を作成します。それ以外の場合、このプロ

パティは RTR ファイルで明示的に宣言されていないかぎり使用できません。

このプロパティを Apache 用に設定する方法については、『*Sun Cluster* データサービスの計画と管理 *(Solaris OS* 版*)*』を参照してください。

カテゴリ: 条件付き/必須

初期値: デフォルトなし

調整: AT_CREATION

`R_description` (`string`)
リソースの簡単な説明。

カテゴリ: オプション

初期値: 空の文字列

調整: ANYTIME

`Resource_dependencies` (`string_array`)
このリソースが強い依存関係を持っている同じまたは異なるグループ内のリソースのリスト。このリスト内の任意のリソースがオンラインでない場合、このリソースは起動できません。このリソースとリスト内のリソースの 1 つが同時に起動する場合、RGM は、リスト内のリソースが起動するまで待機してから、このリソースを起動します。このリソースの `Resource_dependencies` リスト内のリソースが起動しない場合、このリソースもオフラインのままになります。リスト内のリソースのリソースグループがオフラインのままであるか、リスト内のリソースが `Start_failed` 状態である場合、このリソースのリスト内のリソースは起動しない可能性があります。強い依存関係を持っている異なるリソースグループ内のリソースが起動に失敗したために、このリソースがオフラインのままである場合、このリソースのリソースグループは `Pending_online_blocked` 状態に入ります。

このリソースとリスト内の複数のリソースが同時にオフラインになる場合、このリソースはリスト内のリソースより前に停止します。しかし、このリソースがオンラインのままであったり、停止に失敗した場合でも、リスト内の異なるリソースグループのリソースは停止できます。このリソースが先に無効にならなければ、リスト内のリソースは無効にできません。

デフォルトでは、リソースグループ内では、アプリケーションのリソースはネットワークアドレスリソースに対して、暗黙的に強いリソースの依存関係を持っています。詳細については、137 ページの「リソースグループのプロパティ」 の `Implicit_network_dependencies` を参照してください。

リソースグループ内では、`Prenet_start` メソッドは `Start` メソッドより前に、依存関係順に実行されます。`Postnet_stop` メソッドは `Stop` メソッドより後に、依存関係順に実行されます。異なるリソースグループでは、依存しているリソースは、依存されているリソースが `Prenet_start` および `Start` を実行するまで待機してから、`Prenet_start` を実行します。依存されているリソースは、依存しているリソースが `Stop` および `Postnet_stop` を終了するまで待機してから、`Stop` を実行します。

カテゴリ: オプション

初期値: 空のリスト

調整: ANYTIME

Resource_dependencies_restart (string_array)
このリソースが再起動の依存関係を持っている同じまたは異なるグループ内のリソースのリスト。

このプロパティの動作は Resource_dependencies と似ていますが、再起動の依存関係リスト内にある任意のリソースが再起動した場合、このリソースは再起動されます。リスト内のリソースがオンラインに戻った後、このリソースは再起動されます。

カテゴリ: オプション

初期値: 空のリスト

調整: ANYTIME

Resource_dependencies_weak (string_array)
このリソースが弱い依存関係を持っている同じまたは異なるグループ内のリソースのリスト。弱い依存関係は、メソッドが呼び出される順序を決定します。RGM は、このリスト内のリソースの Start メソッドを呼び出してから、このリソースの Start メソッドを呼び出します。そして、RGM は、このリソースの Stop メソッドを呼び出してから、このリスト内のリソースの Stop メソッドを呼び出します。リスト内のリソースが起動に失敗した場合でも、このリソースは起動できます。

このリソースとその Resource_dependencies_weak リスト内のリソースが同時に起動する場合、RGM は、リスト内のリソースが起動するまで待機してから、このリソースを起動します。リスト内のリソースが起動しない場合 (たとえば、リスト内のリソースのリソースグループがオフラインのままであったり、リスト内のリソースが Start_failed 状態である場合)、このリソースは起動します。このリソースの Resource_dependencies_weak リスト内のリソースが起動すると、このリソースのリソースグループは一時的に Pending_online_blocked 状態に入ることがあります。リスト内のすべてのリソースが起動したとき、あるいは、起動に失敗したとき、このリソースとそのリソースグループは再び Pending_online 状態に入ります。

このリソースとリスト内の複数のリソースが同時にオフラインになる場合、このリソースはリスト内のリソースより前に停止します。このリソースがオンラインのままであったり、停止に失敗した場合でも、リスト内のリソースは停止できます。このリソースを無効にするまで、リスト内のリソースは無効にできません。

リソースグループ内では、Prenet_start メソッドは Start メソッドより前に、依存関係順に実行されます。Postnet_stop メソッドは Stop メソッドより後に、依存関係順に実行されます。異なるリソースグループでは、依存しているリソースは、依存されているリソースが Prenet_start および Start を実行するまで待機してから、Prenet_start を実行します。依存されているリソースは、依存しているリソースが Stop および Postnet_stop を終了するまで待機してから、Stop を実行します。

カテゴリ: オプション

初期値: 空のリスト

調整: ANYTIME

Resource_name (string)
リソースインスタンスの名前です。この名前はクラスタ構成内で一意にする必要があります。リソースが作成されたあとで変更はできません。

カテゴリ: 必須

初期値: デフォルトなし

調整: Never

Resource_project_name (string)
リソースに関連付けられた Solaris プロジェクト名。このプロパティは、CPU の共有、クラスタデータサービスのリソースプールといった Solaris のリソース管理機能に適用できます。RGM は、リソースをオンラインにすると、このプロジェクト名を持つ関連プロセスを起動します。このプロパティが指定されていない場合、プロジェクト名は、リソースが属しているリソースグループの RG_project_name プロパティから取得されます。rg_properties(5) を参照してください。どちらのプロパティも指定されなかった場合、RGM は事前定義済みのプロジェクト名 default を使用します。プロジェクトデータベース内に存在するプロジェクト名を指定する必要があります。また、root ユーザーは、このプロジェクトのメンバーとして構成されている必要があります。このプロパティは、Solaris 9 以降のバージョンでのみサポートされます。

注 – このプロパティの変更は、このリソースが次回起動されるときに有効になります。

カテゴリ: オプション

初期値: Null

調整: ANYTIME

各クラスタノードの Resource_state (enum)
RGM が判断した各クラスタノード上のリソースの状態。使用可能な状態は、Online、Offline、Start_failed、Stop_failed、Monitor_failed、Online_not_monitored、Starting、および Stopping です。

ユーザーはこのプロパティを構成できません。

カテゴリ: 照会のみ

初期値: デフォルトなし

調整: Never

Retry_count (integer)
起動に失敗したリソースをモニターが再起動する回数です。このプロパティは RGM によって作成されます。RTR ファイルに宣言されている場合にかぎり、管理者は使用を許可されます。デフォルト値が RTR ファイルに指定されている場合、Retry_count は任意です。

Tunable 属性がリソースタイプファイルに指定されていない場合は、プロパティの Tunable 値は WHEN_DISABLED になります。

RTR ファイル内で宣言され、Default 属性が指定されていない場合、このプロパティは必須になります。

カテゴリ: 条件付き

初期値: デフォルトなし

調整: WHEN_DISABLED

Retry_interval (integer)
失敗したリソースを再起動するまでの秒数。リソースモニターは、このプロパティと Retry_count を組み合わせて使用します。このプロパティは RGM によって作成されます。RTR ファイルに宣言されている場合にかぎり、管理者は使用を許可されます。デフォルト値が RTR ファイルに指定されている場合、Retry_interval は任意です。

Tunable 属性がリソースタイプファイルに指定されていない場合は、プロパティの Tunable 値は WHEN_DISABLED になります。

RTR ファイル内で宣言され、Default 属性が指定されていない場合、このプロパティは必須になります。

カテゴリ: 条件付き

初期値: デフォルトなし

調整: WHEN_DISABLED

Scalable (boolean)
リソースがスケーラブルであるかどうか、つまり、リソースが Sun Cluster のネットワーキング負荷分散機能を使用するかどうかを表します。

このプロパティが RTR ファイルで宣言されている場合は、そのタイプのリソースに対して、RGM は、次のスケーラブルサービスプロパティを自動的に作成します。つまり、 Affinity_timeout、Load_balancing_policy、Load_balancing_weights、Network_resources_used、Port_list、UDP_affinity、および Weak_affinity です。これらのプロパティは、RTR ファイル内で明示的に宣言されない限り、デフォルト値を持ちます。RTR ファイルに Scalable が宣言されている場合、このプロパティのデフォルトは TRUE です。

RTR ファイルにこのプロパティが宣言されている場合、AT_CREATION 以外の Tunable 属性の割り当ては許可されません。

RTR ファイルにこのプロパティが宣言されていない場合、このリソースはスケーラブルではないため、このプロパティを調整することはできません。RGM は、スケーラビブルサービスプロパティをいっさい設定しません。ただし、Network_resources_used および Port_list プロパティは、スケーラブルサービスでも非スケーラブルサービスでも有用であるため、RTR ファイルに明示的に宣言できます。

このリソースプロパティと Failover リソースタイププロパティの組み合わせと、その説明は次のとおりです。

このリソースプロパティを Failover リソースタイププロパティと組み合わせて使用する方法の詳細については、r_properties(5) を参照してください。

カテゴリ: オプション

初期値: デフォルトなし

調整: AT_CREATION

各クラスタノードの Status (enum)
リソースモニターが scha_resource_setstatus(1HA) または scha_resource_setstatus(3HA) で設定します。指定可能な値は、 OK、degraded、faulted、unknown、および offline です。リソースがオンラインまたはオフラインになったとき、RGM は自動的に Status 値を設定します (Status 値をリソースのモニターまたはメソッドが設定していない場合)。

カテゴリ: 照会のみ

初期値: デフォルトなし

調整: Never

各クラスタノードの Status_msg (string)
リソースモニターによって、Status プロパティと同時に設定されます。リソースがオンラインまたはオフラインになったとき、RGM は自動的にこのプロパティを空の文字列にリセットします (このプロパティをリソースのメソッドが設定していない場合)。

カテゴリ: 照会のみ

初期値: デフォルトなし

調整: Never

Thorough_probe_interval (integer)
高オーバーヘッドのリソース障害検証の呼び出し間隔 (秒)。このプロパティは RGM によって作成されます。RTR ファイルに宣言されている場合にかぎり、管理者は使用を許可されます。デフォルト値が RTR ファイルに指定されている場合、Thorough_probe_interval は任意です。

Tunable 属性がリソースタイプファイルに指定されていない場合は、プロパティの Tunable 値は WHEN_DISABLED になります。

RTR ファイルのプロパティ宣言内に Default 属性が指定されていない場合、このプロパティは必須です。

カテゴリ: 条件付き

初期値: デフォルトなし

調整: WHEN_DISABLED

Type (string)
このリソースがインスタントであるリソースタイプ。

カテゴリ: 必須

初期値: デフォルトなし

調整: Never

Type_version (string)
現在このリソースに関連付けられているリソース型のバージョンを指定します。このプロパティは RTR ファイル内に宣言できません。したがって、RGM によって自動的に作成されます。このプロパティの値は、リソースタイプの RT_version プロパティと等しくなります。リソースの作成時、Type_version プロパティはリソースタイプ名の接尾辞として表示されるだけで、明示的には指定されません。リソースを編集すると、Type_version の値が変更されます。

このプロパティの調整は次のソースから継承されます。

- 現在のリソースタイプのバージョン
- RTR ファイル内の #$upgrade_from ディレクティブ

カテゴリ: 説明を参照

初期値: デフォルトなし

調整: 説明を参照

UDP_affinity (boolean)
true の場合、指定のクライアントからの UDP トラフィックはすべて現在クライアントの TCP トラフィックを処理しているサーバーノードに送信されます。

このプロパティは、Load_balancing_policy が Lb_sticky または Lb_sticky_wild の場合にかぎり有効です。さらに、Weak_affinity が FALSE (デフォルト値) に設定されている必要があります。

このプロパティは、スケーラブルサービス専用です。

カテゴリ: オプション

初期値: デフォルトなし

調整: WHEN_DISABLED

Weak_affinity (boolean)
true の場合、弱い形式のクライアントアフィニティが有効になります。弱い形式のクライアントアフィニティは、1 つのクライアントから複数の接続を同じサーバーノードに送信することを許可します。ただし、次の状態の場合を除きます。

- サーバーのリスナーが起動するのは、たとえば、障害モニターが再起動したとき、リソースがフェイルオーバーまたはスイッチオーバーしたとき、あるいは、ノードが障害の後にクラスタに参加し直したときです。
- 管理アクションによるスケーラブルリソースの `Load_balancing_weights` の変更時

弱いアフィニティはメモリーの消費とプロセッササイクルの点で、デフォルトの形式よりもオーバーヘッドを低く抑えられます。

このプロパティは、`Load_balancing_policy` が `Lb_sticky` または `Lb_sticky_wild` の場合にかぎり有効です。

このプロパティは、スケーラブルサービス専用です。

カテゴリ: オプション

初期値: デフォルトなし

調整: `WHEN_DISABLED`

リソースグループのプロパティ

この節では、Sun Cluster で定義されているリソースグループプロパティについて説明します。プロパティ値は、次のように分類されます (「カテゴリ」の後)。

- 必須 — 管理者は、管理ユーティリィティでリソースグループを作成するときに、必ず値を指定する必要があります。
- 任意 — 管理者がリソースグループの作成時に値を指定しない場合、システムがデフォルト値を提供します。
- 照会のみ– 管理ツールから直接設定できません。

各説明は、そのプロパティが初期設定後に更新できる (はい) か更新できない (いいえ) かを示してい ます。

最初にプロパティ名が表示され、次に説明が表示されます。

`Auto_start_on_new_cluster` (`boolean`)
このプロパティを使用すると、新しいクラスタを形成するとき、Resource Group の自動起動を無効にすることができます。

`TRUE` の場合、クラスタが再起動するとき、Resource Group Manager はリソースグループを自動的に起動して、`Desired_primaries` を有効にしようと試みます。`FALSE` に設定されている場合、クラスタのすべてのノードが同時に再起動したとき、Resource Group Manager はリソースグループを自動的に起動しません。

カテゴリ: オプション

初期値: TRUE

調整: Yes

Desired_primaries (integer)
グループが同時にオンラインになることができるノードの数。

RG_mode プロパティが Failover の場合、このプロパティの値を1 より大きく設定することはできません。RG_mode プロパティが Scalable の場合は、1 より大きな値を設定できます。

カテゴリ: オプション

初期値: 1

調整: Yes

Failback (boolean)
クラスタのメンバーシップが変更されたとき、グループがオンラインになっているノードセットを再計算するかどうかを示すブール値です。再計算により、RGM は優先度の低いノードをオフラインにし、優先度の高いノードをオンラインにすることができます。

カテゴリ: オプション

初期値: FALSE

調整: Yes

Global_resources_used (string_array)
クラスタファイルシステムがこのリソースグループ内のリソースによって使用されるかどうかを指定します。管理者はアスタリスク (*) か空文字列 (“”) を指定できます。すべてのグローバルリソースを指定するときはアスタリスク、グローバルリソースを一切指定しない場合は空文字列を指定します。

カテゴリ: オプション

初期値: すべてのグローバルリソース

調整: Yes

Implicit_network_dependencies (boolean)
TRUE の場合、 RGM は、グループ内のネットワークアドレスリソースで非ネットワークアドレスリソースに対する強い依存を強制します。ネットワークアドレスリソースには、論理ホスト名と共有アドレスリソース型があります。

スケーラブルリソースグループの場合、ネットワークアドレスリソースを含んでいないため、このプロパティは効果がありません。

カテゴリ: オプション

初期値: TRUE

調整: Yes

Maximum_primaries (integer)
グループを同時にオンラインにできるノードの最大数です。

RG_mode プロパティが Failover の場合、このプロパティの値を1 より大きく設定することはできません。RG_mode プロパティが Scalable の場合は、1 より大きな値を設定できます。

カテゴリ: オプション

初期値: 1

調整: Yes

Nodelist (string_array)
優先順位に従ってグループをオンラインにできるクラスタノードのリスト。これらのノードは、リソースグループの潜在的主ノードまたはマスターです。

カテゴリ: オプション

初期値: すべてのクラスタノードのリスト。

調整: Yes

Pathprefix (string)
リソースグループ内のリソースが重要な管理ファイルを書き込むことができるクラスタファイルシステム内のディレクトリ。一部のリソースの必須プロパティです。Pathprefix の値はリソースグループごとに固有の値を指定します。

カテゴリ: オプション

初期値: 空の文字列

調整: Yes

Pingpong_interval (integer)
RGM がリソースグループをオンラインにする場所を決定するときに使用する負でない整数値 (秒)。このプロパティが必要になる条件は次のとおりです。

- 再構成が発生している場合。
- scha_control -O GIVEOVER コマンドまたは scha_control() 関数に SCHA_GIVEOVER 引数を指定して実行します。 再構成が発生したときに、

Pingpong_interval で指定した秒数内に特定のノード上で複数回、リソースグループがオンラインになれなかった場合、 そのノードはリソースグループのホストとしては不適切だと判断され、RGM は別のマスターを探します。リソースグループがオンラインになれない原因は、 Start メソッドまたは Prenet_start メソッドがゼロ以外で終了したか、タイムアウトしたかのどちらかです。

リソースの scha_control コマンドまたは関数の呼び出しによって、Pingpong_interval で指定した秒数内に特定のノード上でリソースグループがオフラインになった場合、別のノードから生じる後続の scha_control() 呼び出しの結果、そのノードはリソースグループのホストとして不適切だと判断されます。

カテゴリ: オプション

初期値: 3,600 (1 時間)

調整: Yes

`Resource_list` (`string_array`)
グループに含まれるリソースのリスト。管理者はこのプロパティを直接設定しません。このプロパティは、管理者がリソースグループにリソースを追加したり、リソースを削除したときに、RGM によって更新されます。

カテゴリ: 照会のみ

初期値: デフォルトなし

調整: いいえ

`RG_affinities` (`string`)
RGM は、別のリソースグループの現在のマスターであるノードにリソースグループを配置するか (肯定的なアフィニティの場合)、あるいは、別のリソースグループの現在のマスターでないノード上にリソースグループを配置するか (否定的なアフィニティの場合) しようとします。

`RG_affinities` には、次の文字列を設定できます。

- ++ (強い肯定的なアフィニティ)
- + (弱い肯定的なアフィニティ)
- - (弱い否定的なアフィニティ)
- -- (強い否定的なアフィニティ)
- +++ (フェイルオーバー委託付きの強い肯定的なアフィニティ) たとえば、`RG_affinities=+RG2,--RG3` は、このリソースグループが RG2 に対しては弱い肯定的なアフィニティを持っており、RG3 に対しては強い否定的なアフィニティを持っていることを示します。

`RG_affinities` の使用方法については、『*Sun Cluster* データサービスの計画と管理 *(Solaris* 編*)*』の「Administering Data Service Resources」を参照してください。

カテゴリ: オプション

初期値: 空の文字列

調整: Yes

`RG_dependencies` (`string_array`)
同じノード上の別のグループをオンライン/オフラインにするときの優先順位を示すリソースグループのリスト (任意)。すべての強い `RG_affinities` (肯定的と否定的) と `RG_dependencies` が一緒のグラフはサイクルを含むことが許されません。

たとえば、リソースグループ RG2 がリソースグループ RG1 の `RG_dependencies` リスト内に含まれると仮定します。言い換えると、RG1 が RG2 にリソースグループ依存関係を持っていると仮定します。次のリストに、このリソースグループ依存関係の効果を要約します。

- あるノードがクラスタに参加するとき、RG2 内のリソース上でそのノード上のすべての `Boot` メソッドが完了するまで、RG1 内のリソース上でそのノード上の `Boot` メソッドは動作しません。
- RG1 と RG2 が両方とも同じノード上で同時に `Pending_online` 状態である場合、RG2 内のすべてのリソースが自分の開始メソッドを完了するまで、RG1 内のどのリソースも自分の開始メソッド (`Prenet_start` または `Start`) を実行

できません。
- RG1 と RG2 が両方とも同じノード上で同時に Pending_offline 状態である場合、RG2 内のすべてのリソースが自分の停止メソッドを完了するまで、RG1 内のどのリソースも自分の停止メソッド (Stop または Postnet_stop) を実行できません。
- プライマリを切り替えたとき、任意のノード上で RG1 がオンラインのままになり、すべてのノード上で RG2 がオフラインになった場合、RG1 または RG2 のプライマリの切り替えは失敗します。詳細については、scswitch(1M) と scsetup(1M) を参照してください。
- RG2 上で Desired_primaries プロパティをゼロに設定した場合、RG1 上で Desired_primaries プロパティをゼロより大きな値に設定することは許可されません。
- RG2 上で Auto_start_on_new_cluster プロパティを FALSE に設定した場合、RG1 上で Auto_start_on_new_cluster プロパティを TRUE に設定することは許可されません。

カテゴリ: オプション

初期値: 空のリスト

調整: Yes

RG_description (string)
リソースグループの簡単な説明。

カテゴリ: オプション

初期値: 空の文字列

調整: Yes

RG_is_frozen (boolean)
あるリソースグループが依存している大域デバイスをスイッチオーバーするかどうかを表します。このプロパティが TRUE に設定された場合、大域デバイスはスイッチオーバーされます。このプロパティが FALSE に設定された場合、どの大域デバイスもスイッチオーバーされません。リソースグループが大域デバイスに依存するかどうかは、Global_resources_used プロパティの設定によります。

ユーザーは RG_is_frozen プロパティを直接設定できません。RG_is_frozen プロパティは、大域デバイスのステータスが変わったときに、RGM によって更新されます。

カテゴリ: オプション

初期値: デフォルトなし

調整: いいえ

RG_mode（列挙型）
リソースグループがフェイルオーバーグループかスケーラブルグループかを指定します。値が Failover の場合、RGM はグループの Maximum_primaries プロパティの値を 1 に設定し、リソースグループのマスターを単一のノードに制限します。

このプロパティの値が Scalable の場合、RGM は Maximum_primaries プロパティに 1 より大きい値を設定することを許可します。結果として、グループを複数のノードで同時にマスターできます。Failover プロパティの値が TRUE のリソースを、RG_mode の値が Scalable のリソースグループに追加することはできません。

Maximum_primaries が 1 の場合、デフォルトは Failover です。
Maximum_primaries が 1 より大きい場合、デフォルトは Scalable です。

カテゴリ: オプション

初期値: Maximum_primaries の値に依存。

調整: いいえ

RG_name (string)
リソースグループの名前。この名前は、クラスタ内で一意である必要があります。

カテゴリ: 必須

初期値: デフォルトなし

調整: いいえ

RG_project_name (string)
リソースグループに関連付けられた Solaris プロジェクト名。このプロパティは、CPU の共有、クラスタデータサービスのリソースプールといった Solaris のリソース管理機能に適用できます。RGM は、リソースグループをオンラインにするとき、Resource_project_name プロパティセットを持たないリソースに対して、このプロジェクトで関連付けられたプロセスを起動します。プロジェクトデータベース内に存在するプロジェクト名を指定する必要があります。また、root ユーザーは、このプロジェクトのメンバーとして構成されている必要があります。

このプロパティは、Solaris 9 以降のバージョンでのみサポートされます。

注 – このプロパティの変更は、このリソースが次回起動されるときに有効になります。

カテゴリ: オプション

初期値: テキスト文字列 “default”。

調整: ANYTIME

各クラスタ上の RG_state (enum)
RGM によって各クラスタノード上のグループの状態を表す値に設定されます。設定される値は、Unmanaged 、Online、Offline、Pending_online、Pending_offline、Pending_online_blocked、Error_stop_failed、Online_faulted、または Pending_online_blocked です。

ユーザーはこのプロパティを構成できません。しかし、scswitch(1M) を呼び出すことによって、あるいは同等の scsetup(1M) か SunPlex Manager コマンドを使用して、このプロパティを間接的に設定することは可能です。

カテゴリ: 照会のみ

初期値: デフォルトなし

調整: いいえ

RG_system (boolean)

リソースグループの RG_system プロパティの値が TRUE の場合、そのリソースグループとそのリソースグループ内のリソースに関する特定の操作が制限されます。この制限は、重要なリソースグループやリソースを間違って変更または削除してしまうことを防ぐためにあります。このプロパティの影響を受けるのは、scrgadm(1M) コマンドと scswitch(1M) コマンドだけです。scha_control(1HA) コマンドと scha_control(3HA) コマンドの動作には影響ありません。

リソースグループ (またはリソースグループ内のリソース) の制限操作を実行する前には、まず、リソースグループの RG_system プロパティを FALSE に設定する必要があります。クラスタサービスをサポートするリソースグループ (または、リソースグループ内のリソース) を変更または削除するときには注意してください。

RG_system が TRUE に設定されているリソースグループのことを「システムリソースグループ」と呼びます。RG_system の現在の値に関わらず、 RG_system プロパティ自身を編集することは制限されません。これらの制限の詳細については、rg_properties(5) のマニュアルページを参照してください。

カテゴリ: オプション

初期値: FALSE

調整: Yes

リソースプロパティの属性

次の節では、システム定義のプロパティの変更または拡張プロパティの作成に使用できるリソースプロパティ属性について説明します。

注意 – boolean、enum、int タイプのデフォルト値に、Null または空の文字列 ("") は指定できません。

最初にプロパティ名が表示され、次に説明が表示されます。

`Array_maxsize`
  `stringarray` タイプの場合、設定できる配列要素の最大数。

`Array_minsize`
  `stringarray` タイプの場合、設定できる配列要素の最小数。

`Default`
  プロパティのデフォルト値を示します。

`Description`
  プロパティを簡潔に記述した注記 (文字列)。RTR ファイル内でシステム定義プロパティに対する `Description` 属性を設定することはできません。

`Enumlist`
  `enum` タイプの場合、プロパティに設定できる文字列値のセット。

`Extension`
  リソースタイプの実装によって定義された拡張プロパティが RTR ファイルのエントリで宣言されていることを示します。拡張プロパティが使用されていない場合、そのエントリはシステム定義プロパティです。

`Max`
  `int` タイプの場合、プロパティに設定できる最大値。

`Maxlength`
  `string` および `stringarray` タイプの場合、設定できる文字列の最大。

`Min`
  `int` タイプの場合、プロパティに設定できる最小値。

`Minlength`
  `string` および `stringarray` タイプの場合、設定できる文字列の最小長。

`Property`
  リソースプロパティの名前。

`Tunable`
  クラスタ管理者がリソースのプロパティ値をいつ設定できるかを示します。管理者にプロパティの設定を許可しない場合は、`NONE` または `FALSE` に設定します。管理者にプロパティの調整を許可する属性値は、次のとおりです。`TRUE` または `ANYTIME` (任意の時点)、`AT_CREATION` (リソースの作成時のみ)、`WHEN_DISABLED` (リソースがオフラインのとき)。ほかの調整可能な条件 (「いつ監視を無効にするか」や「いつオフラインにするか」など) を設定するには、この属性を `ANYTIME` に設定して、`Validate` メソッドでリソースの状態を検証します。

  デフォルトは、標準リソースプロパティごとに異なります (次の節を参照)。RTR ファイルで特に指定していない限り、拡張プロパティを調整する設定のデフォルトは `TRUE` (`ANYTIME`) です。

プロパティのタイプ
  指定可能な型は、`string`、`boolean`、`int`、`enum`、`stringarray` です。RTR ファイル内で、システム定義プロパティの型の属性を設定することはできません。タイプは、RTR ファイルのエントリに登録できる、指定可能なプロパティ値とタイ

プ固有の属性を決定します。enum タイプは、文字列値のセットです。

付録 B

有効な RGM 名と値

この付録では、Resource Group Manager (RGM) の名前と値に指定できる文字の条件について説明します。

有効な RGM 名

RGM 名は、次のカテゴリに分類されます。

- リソースグループ名
- リソースタイプ名
- リソース名
- プロパティ名
- 列挙型リテラル名

名前の規則 (リソースタイプ名を除く)

リソースタイプ名を除き、他の名前はすべて次の規則に従う必要があります。

- ASCII であること。
- 先頭は必ず文字にする。
- 名前に使用できる文字は、英字の大文字と小文字、数字、ハイフン (-)、下線 (_)。
- 255 文字を超えないこと。

リソースタイプ名の書式

リソースタイプの完全な名前の書式は、次のように、リソースタイプによって異なります。

- リソースタイプのリソースタイプ登録 (RTR) ファイルに `#$upgrade` 指令が含まれる場合、書式は次のようになります。

  vendor-id.*base-rt-name*:*version*

- リソースタイプの RTR ファイルに `#$upgrade` 指令が含まれない場合、書式は次のようになります。

  vendor-id.*base-rt-name*

ピリオドは、*vendor-id* と *base-rt-name* を分離します。コロンは、 *base-rt-name* と *version* を分離します。

この書式における変数項目は次のようになります。

vendor-id ベンダー ID 接頭辞を指定します。ベンダー ID 接頭辞は、RTR ファイル内の `Vendor_id` リソースタイププロパティの値です。

base-rt-name ベースリソースタイプ名を指定します。ベースリソースタイプ名は、RTR ファイル内の `Resource_type` リソースタイププロパティの値です。

version バージョン接尾辞を指定します。バージョン接尾辞は、RTR ファイル内の `RT_version` リソースタイププロパティの値です。バージョン接尾辞は、RTR ファイルが `#$upgrade` 指令を含む場合、完全なリソースタイプ名の部分だけを示します。 `#$upgrade` 指令は、Sun Cluster のリリース 3.1 から導入されました。

注 – ベースリソースタイプ名が 1 つのバージョンだけ登録されている場合、`scrgadm(1M)` コマンドで完全な名前を使用する必要はありません。ベンダー ID 接頭辞、バージョン接尾辞、あるいは、その両方は省略できます。

リソースタイププロパティの詳細については、119 ページの「リソースタイププロパティ」を参照してください。

例 **B–1** `#$upgrade` 指令付きのリソースタイプ名の完全な名前

この例では、RTR ファイルで次のようなプロパティが設定されているリソースタイプの完全な名前を示します。

- `Vendor_id=SUNW`
- `Resource_type=sample`
- `RT_version=2.0`

例 **B–1** #$upgrade 指令付きのリソースタイプ名の完全な名前　(続き)

この RTR ファイルで定義されているリソースタイプの完全な名前は、次のとおりです。

```
SUNW.sample:2.0
```

例 **B–2** #$upgrade 指令なしのリソースタイプ名の完全な名前

この例では、RTR ファイルで次のようなプロパティが設定されているリソースタイプの完全な名前を示します。

- `Vendor_id=SUNW`
- `Resource_type=nfs`

この RTR ファイルで定義されているリソースタイプの完全な名前は、次のとおりです。

```
SUNW.nfs
```

RGM の値

RGM の値は、プロパティ値と記述値という 2 つのカテゴリに分類されます。property values and description values. どちらのカテゴリも規則は同じで、次のようになります。

- 値は ASCII であること。
- 値の最大長は 4M - 1 バイト (つまり、4,194,303 バイト) であること。
- `NULL`、
  - Null
  - 復帰改行
  - コンマ
  - セミコロン

付録 C

データサービス構成のワークシートと記入例

この付録では、クラスタ構成のリソース関連構成要素を計画する場合に使用するワークシートを提供します。参考のために、ワークシートの記入例も掲載しています。クラスタ構成内のその他のコンポーネントのワークシートは、『*Sun Cluster* ソフトウェアのインストール *(Solaris OS* 版*)*』の「Sun Cluster のインストールと構成のためのワークシート」を参照してください。

構成のワークシート

リソースに関連するコンポーネントがクラスタ構成に多数ある場合は、ワークシートを適宜コピーしてください。これらのワークシートを完成させるには、『*Sun Cluster* ソフトウェアのインストール *(Solaris OS* 版*)*』および 第 1 章の計画ガイドラインに従ってください。記入済みのワークシートを参照しながら、クラスタをインストールおよび構成します。

注 – ワークシートの記入例で使用されるデータはガイドとしてのみ提供されます。したがって、これらの例は、実際のクラスタの完全な構成を表しているわけではありません。

リソースタイプのワークシート

論理ホストまたは共有アドレス以外のリソースタイプにはこのワークシートを使用してください。

表 **C–1** リソースタイプのワークシート

リソースタイプ名	リソースタイプが動作するノード

例: リソースタイプのワークシート

表 **C–2** 例: リソースタイプのワークシート

リソースタイプ名	リソースタイプが動作するノード
SUNW.nshttp	**phys-schost-1, phys-schost-2**
SUNW.oracle_listener	**phys-schost-1, phys-schost-2**
SUNW.oracle_server	**phys-schost-1, phys-schost-2**

ネットワークリソースのワークシート

表 **C–3** ネットワークリソースのワークシート

コンポーネント	名前	
リソース名		
リソースグループ名		
リソースタイプ (1 つに丸を付けてください)	論理ホスト名 | 共有アドレス	
リソースタイプ名		
依存関係		
使用されているホスト名		
拡張プロパティ	名称	値

例: ネットワークリソース — 共有アドレスのワークシート

表 **C–4** 例: ネットワークリソース — 共有アドレスのワークシート

コンポーネント	名前	
リソース名	**sh-galileo**	
リソースグループ名	**rg-shared**	
リソースタイプ (1 つに丸を付けてください)	**Shared address**	
リソースタイプ名	**SUNW.SharedAddress**	
依存関係	**none**	
使用されているホスト名	**sh-galileo**	
拡張プロパティ	名称	値
	netiflist	**ipmp0@1, ipmp0@2**

例: ネットワークリソース — 論理ホスト名のワークシート

表 **C–5** 例: ネットワークリソース — 論理ホスト名のワークシート

コンポーネント	名前	
リソース名	**relo-galileo**	
リソースグループ名	**rg-oracle**	
リソースタイプ (1 つに丸を付けてください)	**Logical hostname**	
リソースタイプ名	**SUNW.LogicalHostname**	
依存関係	**none**	
使用されているホスト名	**relo-galileo**	
拡張プロパティ	名称	値
	netiflist	**ipmp0@1, ipmp0@2**

アプリケーションリソース — フェイルオーバーワークシート

表 **C–6** アプリケーションリソース — フェイルオーバーワークシート

コンポーネント	名前	
リソース名		
リソースグループ名		
リソースタイプ名		
依存関係		
拡張プロパティ	名前	値

例: アプリケーションリソース — フェイルオーバーワークシート

表 C–7 例: アプリケーションリソース — フェイルオーバーワークシート

コンポーネント	名前	
リソース名	**oracle-listener**	
リソースグループ名	**rg-oracle**	
リソースタイプ名	**SUNW.oracle_listener**	
依存関係	**hasp_resource**	
拡張プロパティ	名前	値
	ORACLE_HOME	**/global/oracle/orahome/**
	LISTENER_NAME	**lsnr1**

アプリケーションリソース — スケーラブルのワークシート

表 **C–8** アプリケーションリソース — スケーラブルのワークシート

コンポーネント	名前	
リソース名		
論理ホストのリソースグループ名		
共有アドレスのリソースグループ名		
論理ホストのリソースタイプ名		
共有アドレスのリソースタイプ名		
依存関係		
拡張プロパティ	名前	値

例: アプリケーションリソース — スケーラブルのワークシート

表 **C–9** 例: アプリケーションリソース — スケーラブルのワークシート

コンポーネント	名前	
リソース名	**sh-galileo**	
論理ホストのリソースグループ名		
共有アドレスのリソースグループ名	**rg-shared**	
論理ホストのリソースタイプ名		
共有アドレスのリソースタイプ名		
依存関係		
拡張プロパティ	名前	値

リソースグループ — フェイルオーバーのワークシート

表 **C–10** リソースグループ — フェイルオーバーのワークシート

コンポーネント	注記	名前
リソースグループ名	この名前は、クラスタ内で一意のものでなければなりません。	
機能	このリソースグループの機能について記述してください。	
フェイルバック機能があるか(1 つに丸を付けてください)	主ノードが停止して復旧したあと、このリソースグループを主ノードに戻すかどうかを選択してください。	戻す | 戻さない
ノードリスト	このリソースグループのホストになりえるクラスタノードを指定してください。このリストでは、最初のノードとして稼動系を指定し、続いて待機系を指定する必要があります。二次ノードの順序は、主ノードになる優先順位を示します。	
依存しているディスクデバイスグループ	このリソースグループが依存しているディスクデバイスグループを指定してください。	
構成ディレクトリ	管理作業のためにこのリソースグループ内のリソースがファイルを作成する必要がある場合、それらのリソースが使用するサブディレクトリを含めてください。	

例: リソースグループ — フェイルオーバーのワークシート

表 **C–11** 例: リソースグループ — フェイルオーバーのワークシート

コンポーネント	注記	名前
リソースグループ名	この名前は、クラスタ内で一意のものでなければなりません。	**rg-oracle**
機能	このリソースグループの機能について記述してください。	Oracle リソースを含む
フェイルバック機能があるか(1 つに丸を付けてください)	主ノードが停止して復旧したあと、このリソースグループを主ノードに戻すかどうかを選択してください。	戻さない
ノードリスト	このリソースグループのホストになりえるクラスタノードを指定してください。このリストでは、最初のノードとして稼動系を指定し、続いて待機系を指定する必要があります。二次ノードの順序は、主ノードになる優先順位を示します。	**1) phys-schost-1** **2) phys-schost-2**
依存しているディスクデバイスグループ	このリソースグループが依存しているディスクデバイスグループを指定してください。	**schost1-dg**
構成ディレクトリ	管理作業のためにこのリソースグループ内のリソースがファイルを作成する必要がある場合、それらのリソースが使用するサブディレクトリを含めてください。	

リソースグループ — スケーラブルのワークシート

表 **C–12** リソースグループ — スケーラブルのワークシート

コンポーネント	注記	名前
リソースグループ名	この名前は、クラスタ内で一意のものでなければなりません。	
機能		
稼動系の最大数		
主ノードの適切な数		
フェイルバック機能があるか(1 つに丸を付けてください)	稼動系が停止したあと、このリソースグループを稼動系に戻すかどうかを選択してください。	戻す | 戻さない
ノードリスト	このリソースグループのホストになりえるクラスタノードを指定してください。このリストでは、最初のノードとして稼動系を指定し、続いて待機系を指定する必要があります。二次ノードの順序は、主ノードになる優先順位を示します。	
依存関係	このリソースが依存するリソースグループをすべて挙げてください。	

例: リソースグループ — スケーラブルのワークシート

表 **C–13** 例: リソースグループ — スケーラブルのワークシート

コンポーネント	注記	名前
リソースグループ名	この名前は、クラスタ内で一意のものでなければなりません。	**rg-http**
機能		Web サーバーリソースを含む
稼動系の最大数		**2**
主ノードの適切な数		**2**
フェイルバック機能があるか(1 つに丸を付けてください)	稼動系が停止したあと、このリソースグループを稼動系に戻すかどうかを選択してください。	**No**
ノードリスト	このリソースグループのホストになりえるクラスタノードを指定してください。このリストでは、最初のノードとして稼動系を指定し、続いて待機系を指定する必要があります。二次ノードの順序は、主ノードになる優先順位を示します。	**1) phys-schost-1** **2) phys-schost-2**
依存関係	このリソースが依存するリソースグループをすべて挙げてください。	**rg-shared**

索引

Q

R

S

れ

ろ