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: 819–2086–10
2005 年 8 月, Revision A

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

Part No: 819-0703–10

Revision A

050805@12762

目次

はじめに　**9**

1　Sun Cluster データサービスの計画　**15**
Sun Cluster データサービス構成のガイドライン　16
データサービス固有の要件の確認　16
アプリケーションバイナリの格納先の決定　16
`nsswitch.conf` ファイルの内容の確認　17
クラスタファイルシステムの構成の計画　17
Sun Cluster の制御下で動作するよう Solaris SMF サービスを有効にする　18
リソースグループとディスクデバイスグループの関係　18
`HAStorage` と `HAStoragePlus` の概要　19
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　20
`HAStorage` または `HAStoragePlus` の選択　21
データサービスのインストールと構成に関する考慮事項　22
ノードリストプロパティー　22
`installed_nodes` プロパティー　22
`nodelist` プロパティー　23
`auxnodelist` プロパティー　23
インストールと構成プロセスの概要　23
インストールと構成の作業の流れ　24
フェイルオーバーデータサービスの構成例　25
データサービスリソースを管理するためのツール　25
SunPlex Manager グラフィカルユーザーインタフェース (GUI)　26
SPARC: Sun Management Center GUI 用の Sun Cluster モジュール　26
`scsetup` ユーティリティー　26

scrgadm コマンド　26
データサービスリソースを管理するためのツールの作業ごとの概要　27

2　データサービスリソースの管理　**29**
データサービスリソースの管理作業の概要　30
Sun Cluster データサービスの構成と管理　33
リソースタイプの登録　33
▼ リソースタイプを登録する　33
リソースタイプの更新　34
▼ 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　35
▼ 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　36
リソースタイプのダウングレード　41
▼ 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　41
リソースグループの作成　42
▼ フェイルオーバーリソースグループを作成する　43
▼ スケーラブルリソースグループを作成する　44
リソースグループへのリソースの追加　46
▼ 論理ホスト名リソースをリソースグループに追加する　46
▼ 共有アドレスリソースをリソースグループに追加する　48
▼ 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　50
▼ 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　52
リソースグループをオンラインにする　54
▼ リソースグループをオンラインにする　54
リソースモニターの無効化と有効化　56
▼ リソース障害モニターを無効にする　56
▼ リソース障害モニターを有効にする　57
リソースタイプの削除　57
▼ リソースタイプを削除する　58
リソースグループの削除　59
▼ リソースグループを削除する　59
リソースの削除　60
▼ リソースを削除する　61
リソースグループの主ノードの切り替え　61
▼ リソースグループの主ノードを切り替える　62
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　64
▼ リソースを無効にしてリソースグループを UNMANAGED 状態に移行する。　64

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　66
リソースタイプ、リソースグループ、リソースプロパティーの変更　67
▼ リソースタイププロパティーを変更する　67
▼ リソースグループプロパティーを変更する　69
▼ リソースプロパティーを変更する　70
▼ 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　71
リソースの STOP_FAILED エラーフラグの消去　72
▼ リソースの STOP_FAILED エラーフラグを消去する　72
事前登録されているリソースタイプのアップグレード　74
新しいリソースタイプバージョンの登録に関する情報　74
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　75
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　75
▼ 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　76
リソースグループへのノードの追加と削除　76
リソースグループにノードを追加する　77
▼ スケーラブルリソースグループにノードを追加する　77
▼ 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　78
リソースグループからノードを削除する　80
▼ スケーラブルリソースグループからノードを削除する　81
▼ 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　82
▼ 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　84
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　86
▼ 新しいリソース用に HAStorage リソースタイプを設定する　86
▼ 既存のリソース用に HAStorage リソースタイプを設定する　88
HAStorage から HAStoragePlus へのアップグレード　89
▼ 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　89
▼ 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　91
高可用性ローカルファイルシステムの有効化　92
高可用性ローカルファイルシステムの構成要件　92
ボリュームマネージャーを使用しないデバイスのデバイス名の形式　93
高可用性ローカルファイルシステムの /etc/vfstab のサンプルエントリ　93
▼ NFS エクスポートファイルシステム用に HAStoragePlus リソースタイプを設定する　94
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　96
▼ 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　97
▼ 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　99

▼ HAStoragePlus リソースの変更後に障害から回復する　102
HAStoragePlus リソースタイプのアップグレード　103
新しいリソースタイプバージョンの登録に関する情報　104
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　104
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　104
リソースグループのアフィニティー　105
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　107
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　108
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　109
重要なサービスに優先権を指定する　110
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　111
リソースグループ間のアフィニティーの組み合わせ　112
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　113
▼ RGOffload リソースを設定する　114
RGOffload 拡張プロパティーを構成する　116
障害モニター　117
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　118
▼ リソースグループ、リソースタイプ、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　118
▼ リソースグループ、リソースタイプ、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　119
Sun Cluster データサービス用に障害モニターを調整する　120
障害モニターの検証間隔の設定　121
障害モニターの検証タイムアウトの設定　122
継続的な障害とみなす基準の定義　122
リソースのフェイルオーバー動作を指定する　123

A 標準プロパティー　**125**
リソースタイププロパティー　125
リソースのプロパティー　133
リソースグループのプロパティー　149
リソースプロパティーの属性　158

B 有効な **RGM** 名と値　**161**
有効な RGM 名　161

命名規則 (リソースタイプ名を除く)　161
リソースタイプ名の形式　161
RGM の値　163

C　データサービス構成のワークシートと記入例　**165**
構成のワークシート　165
リソースタイプのワークシート　166
ネットワークリソースのワークシート　168
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　170
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　172
リソースグループ — フェイルオーバーのワークシート　174
リソースグループ — スケーラブルのワークシート　176

索引　**179**

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。

このマニュアルで説明されている作業手順を行うには、Solaris™ オペレーティングシステムに関する知識と、Sun Cluster と使用するボリューム管理ソフトウェアに関する専門知識が必要です。

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、UNIX® の基本的なコマンドや手順 (システムの停止、システムのブート、デバイスの構成など) については説明していません。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。	『コードマネージャー・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 3.0-3.1 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 のマニュアルの完全なリストについては、お使いの Sun Cluster のリリースノートを `http://docs.sun.com` で参照してください。

関連するサン以外の Web サイトの引用

このマニュアル内で引用するサン以外の URL では、補足的な関連情報が得られます。

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。こうしたサイトやリソース上の、またはこれらを通じて利用可能な、コンテンツ、広告、製品、その他の素材について、Sun は推奨しているわけではなく、Sun はいかなる責任も負いません。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発生した (あるいは発生したと主張される) 実際の (あるいは主張される) 損害や損失についても、Sun は一切の責任を負いません。

マニュアル、サポート、およびトレーニング

Sun のサービス	URL	内容
マニュアル	http://jp.sun.com/documentation/	PDF 文書および HTML 文書をダウンロードできます。
サポートおよびトレーニング	http://jp.sun.com/supportraining/	技術サポート、パッチのダウンロード、および Sun のトレーニングコース情報を提供します。

問い合わせについて

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)

ご購入先に連絡するときは、次のコマンドを使用して、システムの各ノードに関する情報を集めます。

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。

- 16 ページの「Sun Cluster データサービス構成のガイドライン」
- 18 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の関係」
- 19 ページの「HAStorage と HAStoragePlus の概要」
- 22 ページの「データサービスのインストールと構成に関する考慮事項」
- 22 ページの「ノードリストプロパティー」
- 23 ページの「インストールと構成プロセスの概要」
- 25 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」

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の詳細については、『Sun Cluster の概念 (Solaris OS 版)』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。

お使いのアプリケーションに Sun Cluster データサービスが提供されていない場合は、そのアプリケーション用のカスタムデータサービスの開発を検討してください。カスタムデータサービスを開発するには、Sun Cluster データサービス API を使用します。詳細については、『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。

注 – Sun Cluster は、`sendmail(1M)` サブシステム用のデータサービスは提供していません。`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可能ですが、`sendmail` の機能は高可用性ではありません。この制限は、`sendmail` のすべての機能 (メールの配信、メールの経路設定、待ち行列化、再試行) に適用されます。

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。

注 – 一部の Sun Cluster データサービスは、x86 ベースのクラスタでは使用できません。詳細は、`http://docs.sun.com` で、ご使用のリリースの Sun Cluster のリリースノートを参照してください。

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「広域デバイスとクラスタファイルシステムについての計画」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、アフィニティー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、92 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のマニュアルを参照してください。

Sun Cluster 環境用に `nsswitch.conf` ファイルを構成する方法の詳細については、『Sun Cluster ソフトウェアのインストール (Solaris OS 版)』の「Sun Cluster 環境の計画」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。特別な検討事項が適用され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。

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「広域デバイスとクラスタファイルシステムについての計画」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` リソースタイプを設定する方法については、92 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。

Sun Cluster の制御下で動作するよう Solaris SMF サービスを有効にする

Sun Cluster で、Solaris Service Management Facility (SMF) と統合されるアプリケーション (NFS または DNS 以外) を高可用性にする必要が生じる場合があります。障害発生後 Sun Cluster がアプリケーションを正しく再起動またはフェイルオーバーできるようにするには、次のように、アプリケーションの SMF サービスインスタンスを無効にする必要があります。

- NFS または DNS 以外のアプリケーションの場合、アプリケーションを表す Sun Cluster リソースのすべての潜在的主ノード上で SMF サービスインスタンスを無効にします。
- アプリケーションの複数のインスタンスが、Sun Cluster で監視する必要があるコンポーネントを共有している場合、そのアプリケーションのすべてのサービスインスタンスを無効にします。このようなコンポーネントの例としては、デーモン、ファイルシステム、デバイスなどがあります。

注 – アプリケーションの SMF サービスインスタンスを無効にしないと、Solaris SMF と Sun Cluster の両方がアプリケーションの起動とシャットダウンを制御しようとする場合があります。その結果、アプリケーションの動作が予測不可能になる場合があります。

詳細については、次のマニュアルを参照してください。

- 『Solaris のシステム管理 (基本編)』の「サービスインスタンスを無効にする方法」
- 『Sun Cluster Data Service for NFS ガイド (Solaris OS 版)』
- 『Sun Cluster Data Service for Domain Name Service (DNS) ガイド (Solaris OS 版)』

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。Sun Cluster はフェイルバックポリシーを使用して、次の条件のセットに対応する Sun Cluster の動作を決定します。

- 障害が発生しクラスタを離脱していたノードが、クラスタに再結合する。
- クラスタに再結合するノードが、現在の主ノードよりも先にノードリストに出現する。

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主ノードから、再結合したノードに切り替えられ、このノードが新しい主ノードになります。

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-2 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とリソースグループの関係の詳細については、『Sun Cluster の概要 (Solaris OS 版)』の「ディスクデバイスグループ」を参照してください。

ディスクデバイスグループの設定方法の詳細は、次のマニュアルを参照してください。

- 『Sun Cluster ソフトウェアのインストール (Solaris OS 版)』の「ディスクデバイスグループ」
- 『Sun Cluster ソフトウェアのインストール (Solaris OS 版)』の「デバイスグループとリソースグループの構成例」

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。

- ディスクデバイスとリソースグループの起動の順番を調整します。HAStorage または HAStoragePlus リソースを含むリソースグループのそのほかのリソースがオンラインになるのは、ディスクデバイスリソースが利用可能になったあとに限られます。
- 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配置します。このような配置により、ディスクに負荷をかけるデータサービスのパフォーマンスが向上します。

また、HAStoragePlus は、マウントされていない状態である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 の設定は効果がありません。リソースグループはデバイスグループととも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、86 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) と 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、92 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

次のタイプのデータサービスには、HAStorage または HAStoragePlus が必要です。

- 記憶装置に直接接続されていないノードを持つデータサービス
- ディスクに負荷をかけるデータサービス

記憶装置に直接接続されていないノードを持つデータサービス

データサービスのリソースグループのノードリストにあるノードの中には、記憶装置に直接接続されていないものがある場合があります。このような状況では、記憶装置とデータサービス間の起動の順番を調整する必要があります。この要件を満たすには、リソースグループを次のように構成します。

- リソースグループで HAStorage リソースまたは HAStoragePlus リソースを構成します。
- その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 リソースに設定します。

ディスクに負荷をかけるデータサービス

Sun Cluster HA for Oracle や Sun Cluster HA for NFS など、データサービスの中にはディスクに負荷をかけるものがあります。ディスクに負荷をかけるデータサービスの場合、リソースグループとディスクデバイスグループを同じノード上に配置します。この要件を満たすには、次の作業を行います。

- `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。

注 – 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。

データサービスの中にはディスクに負荷をかけないものもあります。たとえば Sun Cluster HA for DNS は、起動時にファイルをすべて読み込むため、ディスクに負荷をかけません。データサービスがディスクに負荷をかけない場合、`HAStorage` リソースタイプまたは `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 を使用している場合は、`HAStoragePlus` を使用してください。
- 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。

注 – フェイルオーバー用に構成された Sun Cluster 環境にローカルにファイルシステムを統合するには、`HAStoragePlus` リソースタイプを使用します。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。Sun Cluster 3.0 12/01 またはそれ以前を使用している場合は、まず Sun Cluster 3.0 5/02 または Sun Cluster 3.1 にアップグレードする必要があります。

データサービスのインストールと構成に関する考慮事項

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。

- ディスク障害時の入出力サブシステム内の再試行により、データサービスがディスクに負荷をかけるアプリケーションには、遅延が生じることがあります。ディスクに負荷をかけるデータサービスは入出力中心で、クラスタで多数のディスクを構成しています。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ー

データサービスを構成するときに、次のノードリストプロパティーを指定できます。

- `installed_nodes` プロパティー
- `nodelist` プロパティー
- `auxnodelist` プロパティー

`installed_nodes` プロパティー

`installed_nodes` プロパティーは、データサービスのリソースタイプのプロパティーです。このプロパティーには、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。

nodelist プロパティー

nodelist プロパティーは、リソースグループのプロパティーです。このプロパティーは、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リストを指定します。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。

auxnodelist プロパティー

auxnodelist プロパティーは、共有アドレスリソースのプロパティーです。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。

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 Cluster CD-ROM
  - Sun Cluster Agents CD-ROM
- 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、リソースグループマネージャー (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次の作業に関する手順が説明されています。

- 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
- ネットワークリソースが使用する Internet Protocol Network Multipathing (IP Networking Multipathing) グループの構成

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。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

次の表に、Sun Cluster データサービスのインストールと構成作業の概要を示します。作業手順の詳細が記載されている参照先も示します。

表 **1–1** Sun Cluster データサービスをインストールおよび構成するための作業

作業	参照先
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)』
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

フェイルオーバーデータサービスの構成例

この例では、Oracle アプリケーション用のフェイルオーバーデータサービスが必要とする、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。Oracle アプリケーション用のデータサービスを構成する手順の詳細については、『Sun Cluster Data Service for Oracle ガイド (Solaris OS 版)』を参照してください。

この例とスケーラブルデータサービスの例の主な相違点は、次のとおりです。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に加え、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用の独立したリソースグループ (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 を使用して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の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 に関する詳細が説明されている Sun Management Center ソフトウェアマニュアルのセットを参照するには、`http://docs.sun.com` にアクセスしてください。

`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「Sun Cluster HA for Oracle の登録と構成」の手順を参照してください。

`scrgadm` コマンドを使用してデータサービスリソースを管理する方法の詳細については、次のマニュアルを参照してください。

- 第 2 章
- `scrgadm(1M)` のマニュアルページ

データサービスリソースを管理するためのツールの作業ごとの概要

次の表に、データサービスリソースを管理するために使用できるツール (コマンド行以外) を作業ごとに示します。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を管理するためのツール

作業	**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	不可	Yes
リソースグループを作成	Yes	不可	Yes
リソースグループへソースを追加する	Yes	不可	Yes
リソースグループをオンラインにする	Yes	Yes	不可
リソースグループを削除	Yes	Yes	不可
リソースを削除する	Yes	Yes	不可
リソースグループの現在の主ノードを切り替える	Yes	不可	不可
リソースを使用不可にする	Yes	Yes	不可
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	不可	不可
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	不可
リソースプロパティを変更する	Yes	不可	不可
リソースの `STOP_FAILED` エラーフラグを消去する	Yes	不可	不可
ノードをリソースグループに追加する	Yes	不可	不可

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、クラスタ内でリソースや、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。そのほかのツールを使用して手順を完了できるかどうかを判断するには、25 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。

リソースタイプ、リソースグループ、およびリソースの概要については、第 1 章および『Sun Cluster の概念 (Solaris OS 版)』を参照してください。

この章の内容は次のとおりです。

- 30 ページの「データサービスリソースの管理作業の概要」
- 33 ページの「Sun Cluster データサービスの構成と管理」
- 33 ページの「リソースタイプの登録」
- 34 ページの「リソースタイプの更新」
- 41 ページの「リソースタイプのダウングレード」
- 42 ページの「リソースグループの作成」
- 46 ページの「リソースグループへのリソースの追加」
- 54 ページの「リソースグループをオンラインにする」
- 56 ページの「リソースモニターの無効化と有効化」
- 57 ページの「リソースタイプの削除」
- 59 ページの「リソースグループの削除」
- 60 ページの「リソースの削除」
- 61 ページの「リソースグループの主ノードの切り替え」
- 64 ページの「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」
- 66 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」
- 67 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ーの変更」
- 72 ページの「リソースの STOP_FAILED エラーフラグの消去」
- 74 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」
- 75 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の再登録」
- 76 ページの「リソースグループへのノードの追加と削除」
- 86 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」
- 89 ページの「HAStorage から HAStoragePlus へのアップグレード」

- 92 ページの「高可用性ローカルファイルシステムの有効化」
- 96 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」
- 103 ページの「HAStoragePlus リソースタイプのアップグレード」
- 104 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」
- 113 ページの「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」
- 118 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」
- 120 ページの「Sun Cluster データサービス用に障害モニターを調整する」

データサービスリソースの管理作業の概要

次の表に、Sun Cluster データサービスのインストールと構成作業の概要を示します。作業手順の詳細が記載されている参照先も示します。

表 2–1 データサービスリソースを管理するための作業

タスク	参照先
リソースタイプを登録する	33 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	36 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」 35 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	43 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」 44 ページの「スケーラブルリソースグループを作成する」
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	46 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」 48 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」 50 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」 52 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」

表 2–1 データサービスリソースを管理するための作業　(続き)

タスク	参照先
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	54 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	56 ページの「リソース障害モニターを無効にする」 57 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	58 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	59 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	61 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	62 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	64 ページの「リソースを無効にしてリソースグループを UNMANAGED 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	66 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ーの変更	67 ページの「リソースタイププロパティーを変更する」 69 ページの「リソースグループプロパティーを変更する」 70 ページの「リソースプロパティーを変更する」
失敗した リソースグループマネージャー (RGM) プロセスのエラーフラグの消去	72 ページの「リソースの STOP_FAILED エラーフラグを消去する」
組み込みリソースタイプ LogicalHostname および SharedAddress の再登録	76 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	34 ページの「リソースタイプの更新」 74 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	77 ページの「リソースグループにノードを追加する」

表 2–1 データサービスリソースを管理するための作業　(続き)

タスク	参照先
リソースグループからノードを削除する	80 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	86 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	94 ページの「NFS エクスポートファイルシステム用に 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	96 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	34 ページの「リソースタイプの更新」 103 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	104 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	114 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	118 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	120 ページの「Sun Cluster データサービス用に障害モニターを調整する」

注 – この章の手順では、scrgadm(1M) コマンドを使用し、これらの作業を完了する方法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。このようなオプションの詳細については、25 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 データサービスの構成には次の作業が必要です。

- 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ーを変更するには、67 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ーの変更」に進みます。

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ー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。リソースタイプの詳細については、第 1 章を参照してください。

▼ リソースタイプを登録する

注 – この手順は、任意のクラスタノードから実行します。

始める前に　登録するリソースタイプに名前が付けられていることを確認します。リソースタイプの名前はデータサービス名の省略型です。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。

手順　**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 2–1 リソースタイプの登録

次の例では、Sun Cluster 構成の Sun Java System Web Server アプリケーションを表す SUNW.iws リソースタイプを登録します。

```
# 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
```

次の手順 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、42 ページの「リソースグループの作成」を参照してください。

参照 scrgadm(1M) のマニュアルページ。

リソースタイプの更新

リソースタイプをアップグレードすると、新しいバージョンのリソースタイプで導入された新機能を使用できるようになります。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ーのデフォルト設定が変更されている場合がある。
- リソースタイプの新しい拡張プロパティーが導入されている場合がある。
- リソースタイプの既存の拡張プロパティーがなくなっている場合がある。

- リソースタイプに対して宣言されている標準プロパティーのセットが変更される場合がある。
- min、 max、arraymin、arraymax、default、および tunability などのリソースプロパティーの属性が変更される場合がある。
- 宣言済みメソッドのセットが異なる場合がある。
- メソッドまたは障害モニターの実装が変更される場合がある。

リソースタイプのアップグレードには、次の各節で説明されている作業が必要です。

1. 35 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」
2. 36 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

次の手順では、scrgadm(1M) コマンドを使用してこの作業を実行する方法を説明します。ただし、この作業を行うには、scrgadm コマンドを使用する方法以外もあります。scrgadm コマンドを使用する代わりに、SunPlex Manager や、scsetup(1M) コマンドの Resource Group オプションを使用してこの作業を実行することもできます。

始める前に ノードにアップグレードパッケージをインストールする前にどのような作業を行わなければならないかを判断するには、リソースタイプのドキュメントを参照してください。次のリストのいずれかのアクションが必要です。

- 非クラスタモードでノードを再起動する。
- ノードをクラスタモードで動作させ続け、リソースタイプのすべてのインスタンスの監視をオフにする。
- ノードをクラスタモードで動作させ続け、リソースタイプのすべてのインスタンスに対して監視をオンのままにする。

非クラスタモードでノードを再起動する必要がある場合、ローリングアップグレードを実行することでサービスが失われるのを防止します。順次アップグレードでは、残りのノードをクラスタモードで動作させ続けながら、各ノードでパッケージを個別にインストールします。

手順 **1.** スーパーユーザーになるか、同等の役割になります。

2. リソースタイプのインスタンスをオンラインにするすべてのクラスタノード上で、リソースタイプアップグレード用のパッケージをインストールします。

3. 新しいバージョンのリソースタイプを登録します。

正しいバージョンのリソースタイプを登録するには、次の情報を指定する必要があります。

- リソースタイプ名
- リソースタイプを定義するリソースタイプ登録 (RTR) ファイル

```
# scrgadm -a -t resource-type-name -f path-to-new-rtr-file
```

リソースタイプ名の形式は次のとおりです。

vendor-id.*base-rt-name*:*rt-version*

この形式の詳細については、161 ページの「リソースタイプ名の形式」を参照してください。

4. 新しく登録されたリソースタイプを表示します。

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

5. 必要に応じて、**Installed_nodes** プロパティーを、リソースタイプアップグレード用のパッケージがインストールされるノードに設定します。

リソースタイプアップグレード用のパッケージが一部のクラスタノードでインストールされていない場合、この手順を実行する必要があります。

リソースタイプのインスタンスを含むすべてのリソースグループの `nodelist` プロパティーは、リソースタイプの `Installed_nodes` プロパティーのサブセットである必要があります。

```
# scrgadm -c -t resource-type -h installed-node-list
```

▼ 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

次の手順では、`scrgadm(1M)` コマンドを使用してこの作業を実行する方法を説明します。ただし、この作業を行うには、`scrgadm` コマンドを使用する方法以外もあります。`scrgadm` コマンドを使用する代わりに、SunPlex Manager や、`scsetup(1M)` コマンドの Resource Group オプションを使用してこの作業を実行することもできます。

始める前に　リソースを新しいバージョンのリソースタイプに移行できる時点を判断するには、リソースタイプをアップグレードするための手順を参照してください。

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

指示では、既存のバージョンのリソースをアップグレードできないことが規定されている場合があります。リソースを移行できない場合は、次の代替策を検討してください。

- リソースを削除し、アップグレードされたバージョンの新しいリソースに置き換える
- リソースタイプの古いバージョンでリソースから離脱する

手順 **1.** スーパーユーザーになるか、同等の役割になります。

2. 移行するリソースタイプの各リソースに関して、リソースまたはリソースグループの状態を適切な状態に変更します。

- 任意の時点でリソースを移行できる場合、アクションは必要ありません。
- リソースが監視されていない場合にのみリソースを移行できる場合は、次のコマンドを入力します。

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

- リソースがオフラインである場合にのみリソースを移行できる場合は、次のコマンドを入力します。

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

注 – 移行するリソースにそのほかのリソースが依存している場合、この手順は失敗します。このような場合は、出力されるエラーメッセージを参照して、依存しているリソースの名前を判別します。続いて、移行するリソースと依存するリソースを含むコンマ区切りリストを指定して、この手順を繰り返します。

- リソースが無効である場合にのみリソースを移行できる場合は、次のコマンドを入力します。

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

注 – 移行するリソースにそのほかのリソースが依存している場合、この手順は失敗します。このような場合は、出力されるエラーメッセージを参照して、依存しているリソースの名前を判別します。続いて、移行するリソースと依存するリソースを含むコンマ区切りリストを指定して、この手順を繰り返します。

- リソースグループが管理されていない場合にのみリソースを移行できる場合は、次のコマンドを入力します。

  # **scswitch -n -j** *resource-list*
  # **scswitch -F -g** *resource-group*
  # **scswitch -u -g** *resource-group*

上記コマンドの各項目の意味は次のとおりです。

resource-list 管理されていないリソースグループにあるすべてのリソースのコンマ区切りリストを指定します。

resource-group 管理されていないリソースグループを指定します。

注 – *resource-list* では任意の順序でリソースを指定できます。`scswitch` コマンドにより、*resource-list* での順序に関係なく、リソース間の依存関係を満たすのに必要な順序でリソースが無効になります。

---

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

   必要に応じて、同じコマンドで、同じリソースのそのほかのプロパティーを適切な値に設定します。これらのプロパティーを設定するには、コマンドで追加の `-x` オプションまたは `-y` オプションを指定します。

   そのほかのプロパティーを設定する必要があるかどうかを判別するには、リソースタイプをアップグレードするための手順を参照してください。次の理由により、そのほかのプロパティーを設定しなければならない場合があります。

   - 新しいバージョンのリソースタイプに拡張プロパティーが導入されている。
   - 既存のプロパティーのデフォルト値が、新しいバージョンのリソースタイプにおいて変更されている。

   ```
   # scrgadm -c -j resource -y Type_version=new-version \
   [-x extension-property=new-value] [ -y standard-property=new-value]
   ```

注 – 既存のバージョンのリソースタイプが、新しいバージョンへのアップグレードをサポートしていない場合、この手順は失敗します。

---

4. 手順 2で入力したコマンドを逆にすることで、リソースまたはリソースグループの以前の状態を回復します。

   - 任意の時点でリソースを移行できる場合、アクションは必要ありません。

注 – いつでも移行できるリソースを移行した後、リソースの検証により、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。

- リソースが監視されていない場合にのみリソースを移行できる場合は、次のコマンドを入力します。

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

- リソースがオフラインである場合にのみリソースを移行できる場合は、次のコマンドを入力します。

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

  注 – 手順 2で、移行するリソースに依存するそのほかのリソースを無効にした場合、依存するリソースも有効にします。

- リソースが無効である場合にのみリソースを移行できる場合は、次のコマンドを入力します。

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

  注 – 手順 2で、移行するリソースに依存するそのほかのリソースを無効にした場合、依存するリソースも有効にします。

- リソースグループが管理されていない場合にのみリソースを移行できる場合は、次のコマンドを入力します。

  ```
  # scswitch -e -j resource-list
  # scswitch -o -g resource-group
  # scswitch -z -g resource-group
  ```

例 2–2 オフラインである場合にのみ移行可能なリソースの移行

この例では、リソースがオフラインである場合にのみ移行可能なリソースの移行を示します。新しいリソースタイプパッケージには、新しいパスにあるメソッドが含まれています。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。

この例のリソースには次のような特徴があります。

- 新しいリソースタイプバージョンは 2.0 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。
- 移行されるリソースに対する依存関係は存在しない。

- 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。

この例では、サプライヤの指示に従って、アップグレードパッケージがすでにすべてのクラスタノードでインストールされていると仮定されています。

```
# 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 2–3 監視対象外である場合にのみ移行可能なリソースの移行

この例では、リソースが監視対象外である場合にのみ移行可能なリソースの移行を示します。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、アップグレードパッケージをインストールする前にリソースの監視を無効にする必要があります。

この例のリソースには次のような特徴があります。

- 新しいリソースタイプバージョンは 2.0 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

この例では、次の操作が実行されます。

1. アップグレードパッケージをインストールする前に、次のコマンドを実行してリソースの監視を無効にします。

   ```
   # scswitch -M -n -j myresource
   ```

2. サプライヤの指示に従って、アップグレードパッケージはすべてのクラスタノードにインストールされます。
3. 新しいバージョンのリソースタイプを登録するには、次のコマンドを実行します。

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

4. `Type_version` プロパティーを新しいバージョンに変更するには、次のコマンドを実行します。

   ```
   # scrgadm -c -j myresource -y Type_version=2.0
   ```

5. 移行後リソースの監視を有効にするには、次のコマンドを実行します。

   ```
   # scswitch -M -e -j myresource
   ```

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。リソースを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ための条件は、新しいバージョンのリソースタイプにアップグレードするための条件よりも厳しくなります。リソースを含むリソースグループを管理対象外にする必要があります。

▼ 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

次の手順では、`scrgadm(1M)` コマンドを使用してこの作業を実行する方法を説明します。ただし、この作業を行うには、`scrgadm` コマンドを使用する方法以外もあります。`scrgadm` コマンドを使用する代わりに、SunPlex Manager や、`scsetup(1M)` コマンドの Resource Group オプションを使用してこの作業を実行することもできます。

手順 **1.** スーパーユーザーになるか、同等の役割になります。

2. ダウングレードするリソースを含む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*

3. ダウングレードするリソースを含むリソースグループのすべてのリソースを無効にします。

`scswitch -n -j` *resource-list*

注 – *resource-list* では任意の順序でリソースを指定できます。`scswitch` コマンドにより、*resource-list* での順序に関係なく、リソース間の依存関係を満たすのに必要な順序でリソースが無効になります。

resource-list のリソースにほかのリソースが依存している場合、この手順は失敗します。このような場合は、出力されるエラーメッセージを参照して、依存しているリソースの名前を判別します。続いて、最初に指定したリソースと依存するリソースを含むコンマ区切りリストを指定して、この手順を繰り返します。

---

4. ダウングレードするリソースを含むリソースグループの管理を解除します。

`scswitch -u -g` *resource-group*

5. 必要に応じて、ダウングレード先の古いバージョンのリソースタイプを再登録します。

   ダウングレード先のバージョンがもう登録されていない場合にのみ、この手順を実行します。ダウングレード先のバージョンがまだ登録されている場合は、この手順を省略します。

   scrgadm -a -t *resource-type-name*

6. ダウングレードするリソースに対して、**Type_version** プロパティーをダウングレード先の古いバージョンに設定します。

   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ーに適切な値を設定します。

   scrgadm -c -j *resource-todowngrade* **-y Type_version=***old-version*

7. 手順 3で無効にしたすべてのリソースを有効にします。

   # **scswitch -e -j** *resource-list*

8. ダウングレードしたリソースを含むリソースグループを管理状態にします。

   # **scswitch -o -g** *resource-group*

9. ダウングレードしたリソースを含むリソースグループをオンラインにします。

   # **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。

リソースグループの概念については、第 1 章および『Sun Cluster の概念 (Solaris OS 版)』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には、次の種類のリソースが含まれています。

- ネットワークアドレスリソース (組み込みリソースタイプ `LogicalHostname` および `SharedAddress` のインスタンス)
- 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データサービスアプリケーションリソース)

ネットワークアドレスリソースと依存するデータサービスリソースは、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、クラスタノード間を移動します。

注 – この手順は、任意のクラスタノードから実行します。

手順 **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 2–4 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:
```

次の手順　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、46 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。

参照　`scrgadm(1M)` のマニュアルページ。

▼ 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。

注 – この手順は、任意のクラスタノードから実行します。

手順　**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 2–5 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:
```

次の手順 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、52 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。

参照 scrgadm(1M) のマニュアルページ。

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。

リソースの詳細については、『Sun Cluster の概念 (Solaris OS 版)』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

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ー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。詳細については、71 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。

注 – この手順は、任意のクラスタノードから実行します。

始める前に　次の情報を用意してください。

- 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

手順 **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`	コマンドの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 – 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
```

例 2–6 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
```

例 2–7 IP ネットワークマルチパスグループを識別する論理ホスト名リソースの追加

次に、次の論理ホスト名リソースをリソースグループ `nfs-fo-rg` に追加する例を示します。

- `cs23-rs` という名前のリソースが、ノード 1 および 2 上で IP ネットワークマルチパスグループ `sc_ipmp0` を識別します。
- `cs24-rs` という名前のリソースが、ノード 1 および 2 上で IP ネットワークマルチパスグループ `sc_ipmp1` を識別します。

```
# scrgadm -a -L -j cs23-rs -g nfs-fo-rg -l cs23-rs -n sc_ipmp0@1,sc_ipmp0@2
# scrgadm -a -L -j cs24-rs -g nfs-fo-rg -l cs24-rs -n sc_ipmp1@1,sc_ipmp1@2
```

次の手順 論理ホスト名リソースを追加したあと、54 ページの「リソースグループをオンラインにする」の手順を使用してそれらをオンラインにします。

注意事項 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。妥当性の検査に失敗すると、`scrgadm` コマンドはエラーメッセージを出力して終了します。妥当性の検査に失敗した理由を判別するには、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。

参照 `scrgadm(1M)` のマニュアルページ。

▼ 共有アドレスリソースをリソースグループに追加する

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ー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。詳細については、71 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。

注 – この手順は、任意のクラスタノードから実行します。

始める前に　次の情報を用意してください。

- 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。

手順　**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 – 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
```

例 2–8 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
```

次の手順 共有アドレスリソースを追加したあと、54 ページの「リソースグループをオンラインにする」の手順を使用してリソースを有効にします。

注意事項 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。妥当性の検査に失敗すると、scrgadm コマンドはエラーメッセージを出力して終了します。妥当性の検査に失敗した理由を判別するには、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。

参照 scrgadm(1M) のマニュアルページ。

▼ 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。

注 – この手順は、任意のクラスタノードから実行します。

始める前に 次の情報を用意してください。

- 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。

手順 **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*	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。このリソースグループはすでに存在している必要があります。
`-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, …*	リソース用に設定する拡張プロパティーのコンマ区切りリストを指定します。設定できる拡張プロパティーはリソースタイプに依存します。どの拡張プロパティーを設定するかを決定するには、リソースタイプのマニュアルを参照してください。
`-y` *standard-property =value, …*	リソース用に設定する標準プロパティーのコンマ区切りリストを指定します。設定できる標準プロパティーはリソースタイプに依存します。どの標準プロパティーを設定するかを決定するには、ソースタイプのマニュアルと付録 A 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
```

例 2–9 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
```

次の手順　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、54 ページの「リソースグループをオンラインにする」の手順を使用してリソースを有効にします。

注意事項　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。妥当性の検査に失敗すると、`scrgadm` コマンドはエラーメッセージを出力して終了します。妥当性の検査に失敗した理由を判別するには、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。

参照　`scrgadm(1M)` のマニュアルページ。

▼ 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、共有アドレスリソースを使用するアプリケーションリソースです。共有アドレスリソースはフェイルオーバーリソースグループ内にあります。

注 – この手順は、任意のクラスタノードから実行します。

始める前に　次の情報を用意してください。

- 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。

手順　**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*, …
リソース用に設定する拡張プロパティーのコンマ区切りリストを指定します。設定できる拡張プロパティーはリソースタイプに依存します。どの拡張プロパティーを設定するかを決定するには、リソースタイプのマニュアルを参照してください。

-y *standard-property* =*value*, …
リソース用に設定する標準プロパティーのコンマ区切りリストを指定します。設定できる標準プロパティーはリソースタイプに依存します。スケーラブルサービスの場合、通常は Port_list、Load_balancing_weights、および Load_balancing_policy プロパティーを設定します。どの標準プロパティーを設定するかを決定するには、リソースタイプのマニュアルと付録 A を参照してください。

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
```

例 2–10 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
```

次の手順　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、54 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。

注意事項　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。妥当性の検査に失敗すると、`scrgadm` コマンドはエラーメッセージを出力して終了します。妥当性の検査に失敗した理由を判別するには、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。

参照　`scrgadm(1M)` のマニュアルページ。

リソースグループをオンラインにする

リソースを有効にして HA サービスの提供を開始するには、次の操作を実行する必要があります。

- リソースグループでのリソースの有効化
- リソースモニターの有効化
- リソースグループを管理対象にする
- リソースグループをオンラインにする

以上の作業は個別に行うことも、1 つのコマンドを使用して行うこともできます。

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。

▼ リソースグループをオンラインにする

この作業は、任意のクラスタノードから実行します。

手順　**1.** クラスタメンバーから、スーパーユーザーになるか、同等の役割を引き受けます。

2. コマンドを入力してリソースグループをオンラインにします。

- 無効のままでなければならないリソースまたは障害モニターを意図的に無効にしている場合は、次のコマンドを入力します。

  ```
  # scswitch -z -g rg-list
  ```

  `-z` リソースと障害モニターを有効にすることなく、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。

- リソースグループがオンラインになった時点でリソースと障害モニターを有効にする必要がある場合は、次のコマンドを入力します。

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

  `-Z` リソースと障害モニターを有効にしたあとで、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ーを宣言している場合、この操作は失敗します。詳細については、104 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. 手順 2で指定した各リソースグループがオンラインであることを確認します。

   このコマンドからの出力は、どのノードで各リソースグループがオンラインであるかを示します。

   ```
   # scstat -g
   ```

例 2–11 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。このリソースのすべてのリソースとその障害モニターも有効になります。

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

次の手順　リソースと障害モニターを有効にすることなくリソースグループをオンラインにした場合、有効にする必要があるリソースの障害モニターを有効にします。詳細については、57 ページの「リソース障害モニターを有効にする」を参照してください。

参照　scswitch(1M) マニュアルページ。

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

手順　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障害モニターを無効にします。
-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 2–12 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

手順 **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 2–13 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、リソースタイプを削除する場合は、次の手順に従います。

注 – この手順は、任意のクラスタノードから実行します。

▼ リソースタイプを削除する

リソースタイプを削除するには、リソースタイプを登録解除する前に、クラスタ内でそのタイプのすべてのリソースを無効にし、削除します。

始める前に　削除するリソースタイプのすべてのインスタンスを特定するには、次のコマンドを入力します。

```
# scrgadm -pv
```

手順　**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. リソースタイプの登録を解除します。

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

`-r`　指定したリソースタイプの登録を解除します。

`-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 2–14　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を登録解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
```

参照　次のマニュアルページを参照してください。

- scrgadm(1M)
- scswitch(1M)

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。

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

始める前に　削除するリソースタイプのすべてのリソースを特定するには、次のコマンドを入力します。

```
# scrgadm -pv
```

手順　**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。
   リソースごとに次のコマンドを入力します。

   ```
   # 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。

5. リソースグループの削除

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

   `-r` 指定したリソースグループを削除します。

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

6. 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 **2–15** 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
```

参照 次のマニュアルページを参照してください。

- `scrgadm(1M)`
- `scswitch(1M)`

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。

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

手順 **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 **2–16** 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
```

参照 次のマニュアルページを参照してください。

- `scrgadm(1M)`
- `scswitch(1M)`

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。

▼ リソースグループの主ノードを切り替える

注 – この手順は、任意のクラスタノードから実行します。

始める前に　次の条件が満たされていることを確認します。

- 次の情報を持っている。
  - 切り替えを行う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
- 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はクラスタノードである。
- これらのノードは、切り替えを行うリソースグループの潜在的マスターになるように設定されている。

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、次のコマンドを入力します。

```
# scrgadm -pv
```

手順　1. クラスタメンバーから、スーパーユーザーになるか、同等の役割を引き受けます。

2. リソースグループを、新しい主ノードのセットに切り替えます。

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

`-z`	指定したリソースグループを、新しい主ノードのセットに切り替えます。
`-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ー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。詳細については、104 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主ノードへ切り替えられていることを確認します。

このコマンドからの出力は、スイッチオーバーされたリソースグループの状態を示しています。

```
# scstat -g
```

例 2–17 リソースグループの新しい主ノードへの切り替え

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。

1. phys-schost-1 でリソースグループがオンラインであることを確認するには、次のコマンドを実行します。

```
phys-schost-1# scstat -g
...-- Resource Groups --

            Group Name          Node Name           State
            ----------          ---------           -----
     Group: resource-group-1    phys-schost-1       Online
     Group: resource-group-1    phys-schost-2       Offline...
```

2. 切り替えを実行するには、次のコマンドを実行します。

```
phys-schost-1# scswitch -z -g resource-group-1 -h phys-schost-2
```

3. phys-schost-2 でグループがオンラインに切り替わったことを確認するには、次のコマンドを実行します。

```
phys-schost-1# scstat -g
...-- Resource Groups --

            Group Name          Node Name           State
            ----------          ---------           -----
     Group: resource-group-1    phys-schost-1       Offline
     Group: resource-group-1    phys-schost-2       Online...
```

参照　次のマニュアルページを参照してください。

- scrgadm(1M)
- scswitch(1M)

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 UNMANAGED 状態に移行する。

注 – 共有アドレスリソースを無効にすると、そのリソースは依然として一部のホストから `ping(1M)` コマンドに応答できる場合があります。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。

始める前に　次の情報を用意してください。

- 無効にするリソースの名前
- UNMANAGED 状態に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、次のコマンドを入力します。

```
# scrgadm -pv
```

手順　**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. リソースグループのすべてのリソースを無効にします。

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

`-n`　リソースを無効にします。

-j *resource-list*　リソースグループのリソースのコンマ区切りリストを指定します。

注 – *resource-list* では任意の順序でリソースを指定できます。scswitch コマンドにより、*resource-list* での順序に関係なく、リソース間の依存関係を満たすのに必要な順序でリソースが無効になります。

---

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**UNMANAGED** 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 UNMANAGED 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 2–18　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

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 UNMANAGED 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
```

参照　次のマニュアルページを参照してください。

- scrgadm(1M)
- scswitch(1M)

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、これらのオブジェクトの現在の構成設定を表示します。

注 – 任意のクラスタノードから、リソース、リソースグループ、リソースタイプの構成設定を表示できます。

---

scrgadm コマンドは、構成状態に関する次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ー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ー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、拡張プロパティー、すべてのリソースとリソースグループのプロパティーを含む、詳細情報が表示されます。

表示したいオブジェクトの名前のあとに -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) オプションを使用することによって、特定のリソースタイプ、リソースグループ、および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ーの変更

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ーを定義します。これらの標準プロパティーについては、次の節を参照してください。

- 125 ページの「リソースタイププロパティー」
- 133 ページの「リソースのプロパティー」
- 149 ページの「 リソースグループのプロパティー」

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ー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ーについては、データサービスのマニュアルを参照してください。

プロパティーを変更できるかどうかを判断するには、そのプロパティーの説明において、プロパティー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ー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ーを変更する

注 – この手順は、任意のクラスタノードから実行します。

始める前に　次の情報を用意してください。

- 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ーの名前。リソースタイプの場合、特定のプロパティーだけを変更できます。プロパティーを変更できるかどうかを判断するには、125 ページの「リソースタイププロパティー」でプロパティーの Tunable エントリを参照してください。

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

手順 **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ーを変更します。

リソースタイプの場合、特定のプロパティーだけを変更できます。プロパティーを変更できるかどうかを判断するには、125 ページの「リソースタイププロパティー」でプロパティーの Tunable 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ーを変更します。
-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ーの名前と、そのプロパティーの新しい値を指定します。 Installed_nodes プロパティーは明示的には変更できません。このプロパティーを変更するには、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ーが変更されていることを確認します。

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

例 **2–19** リソースタイププロパティーの変更

次に、SUNW.apache プロパティーを変更し、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ーを変更する

この手順では、リソースグループプロパティーの変更方法について説明します。リソースグループパーティーの詳細については、149 ページの「 リソースグループのプロパティー」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。

始める前に 次の情報を用意してください。

- 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ーの名前とその新しいプロパティー値

手順

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ーを変更します。

   ```
   # scrgadm -c -g resource-group -y property=new-value
   ```

`-c`	指定したプロパティーを変更します。
`-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ーの名前を指定します。

3. リソースグループプロパティーが変更されていることを確認します。

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

例 2–20 リソースグループプロパティーの変更

次に、リソースグループ (`resource-group-1`) の `Failback` プロパティー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ーを変更する

この手順では、リソースの拡張プロパティーと標準プロパティーを変更する方法を説明します。

- 標準リソースプロパティーの詳細については、133 ページの「リソースのプロパティー」を参照してください。
- リソースの拡張プロパティーの詳細については、リソースのリソースタイプのマニュアル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。

始める前に　次の情報を用意してください。

- 変更するプロパティーを持つリソースの名前
- 変更するプロパティーの名前

手順　**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. 現在のリソースプロパティー設定を表示します。

```
# scrgadm -pvv -j resource
```

3. リソースプロパティーを変更します。

```
# scrgadm -c -j resource -y standard-property=new-value | -x extension-property=new-value
```

`-c`	指定したプロパティーを変更します。
`-j` *resource*	リソースの名前を指定します。
`-y` *standard-property* `=`*new-value*	変更する標準プロパティー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ーの名前を指定します。

4. リソースプロパティーが変更されていることを確認します。

```
# scrgadm pvv -j resource
```

例 2–21　標準リソースプロパティーの変更

次に、リソース (`resource-1`) のシステム定義プロパティー (`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 2–22 拡張リソースプロパティーの変更

次に、リソース (resource-1) の拡張プロパティー (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 拡張プロパティーを false に設定します。CheckNameService プロパティー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。詳細については、74 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。

手順 **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ーを変更します。

```
# 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` 拡張プロパティーを `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ーが `NONE` または `SOFT` に設定されている場合、リソースの `STOP` メソッドが失敗すると、次のような影響があります。

- 個々のリソースは `STOP_FAILED` 状態になります。
- リソースを含むリソースグループは `ERROR_STOP_FAILED` 状態になります。

このような状況では、次の操作を行うことができません。

- 任意のノードでリソースグループをオンラインにする
- リソースグループにリソースを追加する
- リソースグループからリソースを削除する
- リソースグループのプロパティーを変更する
- リソースグループのリソースのプロパティーを変更する

▼ リソースの STOP_FAILED エラーフラグを消去する

注 – この手順は、任意のクラスタノードから実行します。

始める前に 次の情報を用意してください。

- 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

手順

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で、リソースグループの状態を調べます。

   ```
   # scstat -g
   ```

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

   次の環境の組み合わせでは、リソースグループは `ERROR_STOP_FAILED` 状態のままになっています。

   - `STOP` メソッドの失敗が発生した時点でリソースグループがオフラインに切り替えられている。
   - 停止に失敗したリソースがリソースグループ内のそのほかのリソースに依存している。

6. リソースグループが **ERROR_STOP_FAILED** 状態のままである場合、次のようにエラーを修正します。

   a. 適切なノード上でリソースグループをオフラインにします。

      ```
      # 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。

   b. リソースグループをオンラインにします。

参照　`scswitch(1M)` マニュアルページ。

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、34 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、事前登録されている各リソースタイプバージョンと 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 **2–23** 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、アップグレード時に、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 プロパティーの値が 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場合は、リソースの CheckNameService 拡張プロパティーを false に設定します。

例 **2–24** 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

リソースタイプ SUNW.LogicalHostname および SUNW.SharedAddress は事前に登録されています。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、74 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。

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

手順 ● 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 2–25 事前登録されているリソースタイプを誤って削除したあとに再登録する

次に、`SUNW.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
```

参照 `scrgadm(1M)` のマニュアルページ。

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ー) 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手順には、次の手順が含まれます。

1. 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を繰り返す
2. 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

詳細は、`scrgadm(1M)` のマニュアルページを参照してください。

注 – いずれの手順も任意のクラスタノードから実行できます。

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。

- 77 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」
- 78 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

手順 **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 5 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ー**)** 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のコンマ区切りリストを指定します。

3. **(**省略可能**)** スケーラブルリソースの **Load_balancing_weights** プロパティーを更新し、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

手順 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。

3. **HAStorage** または **HAStoragePlus AffinityOn** 拡張プロパティーが **True** に等しい場合、適切なディスクセットまたはデバイスグループにノードを追加します。

   - **Solstice DiskSuite** または **Solaris Volume Manager** を使用している場合は、**metaset** コマンドを使用します。

     `#` **`metaset -s`** *disk-set-name* **`-a -h`** *node-name*

     `-s` *disk-set-name*　`metaset` コマンドの実行対象となるディスクセットの名前を指定します。

     `-a`　指定したディスクセットにドライブまたはホストを追加します。

     `-h` *node-name*　ディスクセットに追加するノードを指定します。

   - **SPARC:VERITAS Volume Manager** を使用している場合は **scsetup** ユーティリティーを使用します。

     a. アクティブなクラスタメンバー上で **scsetup** ユーティリティーを起動します。

        `#` **`scsetup`**

        メインメニューが表示されます。

     b. メインメニューで、デバイスグループおよびボリュームのオプションに対応する数字を入力します。

     c. 「**Device Groups**」メニューで、ノードを **VxVM** デバイスグループに追加するためのオプション対応する数字を入力します。

     d. プロンプトに応答し、**VxVM** デバイスグループにノードを追加します。

4. 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のコンマ区切りリストを指定します。

5. 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 2–26　リソースグループにノードを追加する

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 ネットワークマルチパス
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
# metaset -s red -a -h phys-schost-2
# 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。

- 81 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」
- 82 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」
- 84 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」

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の名前を指定します。このリソースグループは、削除するノード上でマスターされます。
`-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。

詳細は、`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` プロパティーは、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。このプロパティーの詳細については、付録 A を参照してください。

スケーラブルサービス構成の詳細については、『Sun Cluster の概念 (Solaris OS 版)』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。

手順 **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ー)** 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*	このリソースグループをマスターできるノードの名前のコンマ区切りリストを指定します。

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。

詳細については、84 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`** プロパティーを更新し、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。

参照 `scrgadm(1M)` のマニュアルページ。

▼ 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、84 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。

手順 **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*	このリソースグループをマスターできるノードの名前のコンマ区切りリストを指定します。

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 – 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、82 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。共有アドレスリソースを継続して使用するには、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。

手順 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 ネットワークマルチパス
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あと、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あと、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、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。データサービスが、まだオンラインになっていない広域デバイスまたはクラスタファイルシステムに依存する場合、START メソッドはタイムアウトします。この場合、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、手動でデータサービスを再起動する必要があります。

このような追加の管理作業を避けるには、HAStorage リソースタイプまたは HAStoragePlus リソースタイプを使用します。広域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 のインスタンスを追加します。このようなリソースタイプのインスタンスは、次の操作を実行します。

- 広域デバイスとクラスタファイルシステムを監視する
- 広域デバイスとクラスタファイルシステムが利用可能になるまで、同じリソースグループ内のほかのリソースの START メソッドを待機させる

どちらのリソースタイプを使用するかを決定するには、21 ページの「HAStorage または HAStoragePlus の選択」を参照してください。

HAStorage リソースを作成するには、86 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 リソースを作成するには、94 ページの「NFS エクスポートファイルシステム用に 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 ソフトウェア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 にアップグレードする手順については、89 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、86 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 リソースを作成するには、92 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。

手順 **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の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とも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
```

参考 アフィニティースイッチオーバー

HAStorage リソースタイプは、別の拡張プロパティー (AffinityOn) を含みます。この拡張プロパティーは、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。詳細については、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 ソフトウェア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 にアップグレードする手順については、89 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。

始める前に 86 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」を読んでください。

手順 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 ソフトウェア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には、次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

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 プロパティーは 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。

手順 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。

注 – HAStorage の ServicePaths プロパティーを使用する代わりに、HAStoragePlus の FilesystemMountPoints プロパティーまたは GlobalDevicePaths プロパティーを使用する必要があります。

- ファイルシステムのマウントポイントを指定するには、次のコマンドを入力します。

  FilesystemMountPoints 拡張プロパティーは、/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。

```
# 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、次のコマンド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
```

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

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 プロパティーは 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。

手順 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。

注 – HAStorage の ServicePaths プロパティーを使用する代わりに、HAStoragePlus の FilesystemMountPoints プロパティーまたは GlobalDevicePaths プロパティーを使用する必要があります。

- ファイルシステムのマウントポイントを指定するには、次のコマンドを入力します。

  FilesystemMountPoints 拡張プロパティーは、/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。

```
# 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、次のコマンド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
```

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

高可用性ローカルファイルシステムを使用すると、出入力負荷が高いデータサービスのパフォーマンスを改善できます。Sun Cluster 環境でローカルファイルシステムを高可用性にするには、HAStoragePlus リソースタイプを使用します。

出入力負荷が高い各 Sun Cluster データサービスの作業手順では、データサービスを構成して HAStoragePlus リソースタイプとともに動作させる方法が説明されています。詳細については、個別の Sun Cluster データサービスのガイドを参照してください。

NFS エクスポートファイルシステム用に HAStoragePlus リソースタイプを設定する手順については、94 ページの「NFS エクスポートファイルシステム用に HAStoragePlus リソースタイプを設定する」を参照してください。

注 – HAStoragePlus リソースタイプを使用してルートファイルシステムを高可用性にしないでください。

高可用性ローカルファイルシステムの構成要件

多重ホストディスク上のすべてのファイルシステムは、これらの多重ホストディスクに直接接続されたすべてのホストからアクセス可能である必要があります。この要件を満たすには、次のように、高可用性ローカルファイルシステムを構成します。

- ローカルファイルシステムのディスクパーティションが広域デバイス上に存在するようにします。
- これらの広域デバイスを指定する HAStoragePlus リソースの AffinityOn 拡張プロパティーを True に設定します。
- フェイルオーバーリソースグループに HAStoragePlus リソースを作成します。

- デバイスグループと、HAStoragePlus リソースを含むリソースグループのフェイルバック設定が同じであるようにします。

注 – 高可用性ローカルファイルシステム用の広域デバイスと、ボリュームマネージャーの併用は、任意に選択できます。

ボリュームマネージャーを使用しないデバイスのデバイス名の形式

ボリュームマネージャーを使用しない場合、基本のストレージデバイスの名前には適切な形式を使用します。使用する形式は、次のように、ストレージデバイスの種類に依存します。

- ーブロックデバイスの場合: /dev/global/dsk/d *D*s*S*
- raw デバイスの場合: /dev/global/rdsk/d *D*s*S*

これらの論理名の変数の意味は次のとおりです。

- *D* はデバイス ID (DID) インスタンス番号を指定する整数です。
- *S* はスライス番号を指定する整数です。

高可用性ローカルファイルシステムの /etc/vfstab のサンプルエントリ

次の例に、高可用性ローカルファイルシステムに使用される広域デバイスの /etc/vfstab ファイルにあるエントリを示します。

例 **2–27** ボリュームマネージャーのない広域デバイスの /etc/vfstab にあるエントリ

この例に、ボリュームマネージャーを使用しない物理ディスク上の広域デバイス用の /etc/vfstab ファイルにあるエントリを示します。

```
/dev/global/dsk/d1s0        /dev/global/rdsk/d1s0
/global/local-fs/nfs  ufs     5  no     logging
```

例 **2–28** Solaris ボリュームマネージャー を使用する広域デバイスの /etc/vfstab にあるエントリ

この例では、Solaris ボリュームマネージャー を使用する広域デバイス用の /etc/vfstab ファイルにあるエントリを示します。

```
/dev/md/kappa-1/dsk/d0   /dev/md/kappa-1/rdsk/d0
/global/local-fs/nfs ufs     5  no     logging
```

例 **2–29** VxVM を使用する広域デバイス用の /etc/vfstab にあるエントリ

この例では、VxVM を使用する広域デバイス用の /etc/vfstab ファイルにあるエントリを示します。

```
/dev/vx/dsk/kappa-1/appvol     /dev/vx/rdsk/kappa-1/appvol
/global/local-fs/nfs vxfs      5 no      log
```

▼ NFS エクスポートファイルシステム用に HAStoragePlus リソースタイプを設定する

HAStoragePlus リソースタイプには HAStorage と同じ機能があり、リソースグループとディスクデバイスグループとの間で起動の同期をとります。
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、92 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。

注 – 次では、UNIX ファイルシステムで HAStoragePlus リソースタイプを使用する方法を説明します。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 QFS のマニュアルを参照してください。

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エクスポートします。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 エントリは、広域オプションを省略し、「mount at boot」フラグを「no」に指定する必要があります。
- path-prefix ディレクトリは、マウントする同じファイルシステムのルートディレクトリ上に存在します (/global/local-fs/nfs など)。path-prefix ディレクトリは、HA-NFS が管理情報と状態情報を保持するために使用するディレクトリです。

手順 **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. **HAStoragePlus** リソースタイプと **SUNW.nfs** リソースタイプが登録されているかどうかを判別します。

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. 必要に応じて、**HAStoragePlus** リソースタイプと **SUNW.nfs** リソースタイプを登録します。

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

4. フェイルオーバーリソースグループ **nfs-rg** 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. 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 拡張プロパティーは、ファイルシステムの 1 つ以上のマウントポイントのリストを指定するために使用できます。このリストは、ローカルファイルシステムと広域ファイルシステムの両方のマウントポイントから構成されます。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。

---

7. 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。

   リソースグループがオンラインであるノードは、/global/local-fs/nfs ファイルシステムの基本となる広域デバイスパーティションの主ノードになります。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
```

8. 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。

   ファイル dfstab.nfs-rs が /global/local-fs/nfs/SUNW.nfs に作成される必要があります。

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

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

9. リソースグループ **nfs-rg** をオフラインにします。

```
# scswitch -F -g nfs-rg
```

10. **nfs-rg** グループをクラスタノード上でオンラインにします。

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

注意 – 切り替えは、リソースグループに限定します。デバイスグループは切り替えないでください。デバイスグループを切り替えようとすると、リソースグループとデバイスグループの状態に矛盾が生じ、リソースグループのフェイルオーバーが発生します。

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 拡張プロパティーが True の場合だけです。

AffinityOn 拡張プロパティーについては、86 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。

手順 **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、追加する各ファイルシステムのマウントポイント用に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** 拡張プロパティーを変更して、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 2–30 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
# 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 拡張プロパティーが True の場合だけです。

AffinityOn 拡張プロパティーについては、86 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。

手順 **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** 拡張プロパティーを変更して、**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 2–31 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
          -------------   ---------    -----     --------------
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 拡張プロパティー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。

手順 **1.** 変更が失敗した原因となる障害を特定します。

```
# 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** 拡張プロパティー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 2–32　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。

```
# 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、34 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 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` プロパティー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ー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ーを使用して、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ーを組み合わせて、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ー

リソースグループ間のアフィニティーは、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。各アフィニティーにおいて、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ーを宣言します。リソースグループ間にアフィニティーを作成するには、ソースの `RG_affinities` リソースグループプロパティーを次のように設定します。

`-y RG_affinities=`*affinity-list*

affinity-list ソースリソースグループとターゲットリソースグループ (複数可) の間のアフィニティーのコンマ区切りリストを指定します。リストでは 1 つまたは複数のアフィニティーを指定できます。

リストでは各アフィニティーを次のように指定します。

operator target-rg

注 – *operator* と *target-rg* の間には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ー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 作成しているアフィニティーのターゲットであるリソースグループを指定します。

表 **2–2** リソースグループ間のアフィニティーのタイプ

演算子	アフィニティーのタイプ	効果
+	弱い肯定的な	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	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	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	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	強い肯定的なアフィニティーと似ていますが、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。詳細については、111 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。

弱いアフィニティーは、`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、強いアフィニティーが成立しないことがあります。このような状況では、アフィニティー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、強いアフィニティーが成立できるようになると、アフィニティー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 – 複数のターゲットリソースグループを持つソースリソースグループに強いアフィニティーを宣言するときは、注意が必要です。宣言されたすべての強いアフィニティーが成立しない場合、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ーを宣言します。

```
# 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ー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ー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ー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ーを宣言する対象のリソースグループです。

強い肯定的なアフィニティー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ー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ー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。`rg1` 内のすべてのリソースが操作可能であるとしても、このフェイルオーバーは発生します。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` を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ー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。詳細については、111 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 **2–33** 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ーを宣言するためのコマンドを示します。このアフィニティーを宣言すると、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ーを宣言します。

```
# 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ー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ー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ー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ーを宣言する対象の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ー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ーのソースは、まず、そのアフィニティー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ーのソースは、そのアフィニティー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ーのソースは、そのアフィニティー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 **2–34** 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ーを宣言するためのコマンドを示します。このアフィニティーを宣言すると、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 はリソースグループが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ーを宣言します。

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

-g *source-rg*
弱い否定的なアフィニティー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否定的なアフィニティーを宣言するリソースグループです。

-y RG_affinities=*neg-affinity-list*
ソースリソースグループと、弱い否定的なアフィニティーのターゲットであるリソースグループの間の、弱い否定的なアフィニティーをコンマで区切って指定します。ターゲットリソースグループは、弱い肯定的なアフィニティーを宣言する対象の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ー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ーの最小数が違反されます。

例 **2–35** 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、クラスタで利用可能なノード間で均等に分配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ーを宣言します。

```
# 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ー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ー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ーのソースのリソースグループは、そのアフィニティーのターゲットのリソースグループから離れます。

例 **2–36** 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ーのソースリソースグループは、そのアフィニティー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ー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。このアフィニティーのターゲットがフェイルオーバーするとき、このアフィニティー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 – ++ 演算子で指定した強い肯定的なアフィニティー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、このアフィニティー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ーを宣言します。

```
# 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ー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ー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ーを宣言できます。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ー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ー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ーを宣言する場合、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 **2–37** 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ーを宣言するためのコマンドを示します。このアフィニティー関係の結果、`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ーの組み合わせ

複数のアフィニティー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ーを宣言します。
- 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ーを宣言します。

例 **2–38** リソースグループ間のアフィニティーの組み合わせ

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

例 2–38 リソースグループ間のアフィニティーの組み合わせ　(続き)

- 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。
- 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ー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ーを宣言します。
- 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ーを宣言します。

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ーを使用することです。詳細については、104 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 – プライオリティー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

手順 **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** リソースの読み込みが解除されるように、各リソースグループにおいて **Desired_primaries** プロパティー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、拡張プロパティーを設定します。
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。
RGOffload 拡張プロパティーの詳細については、116 ページの「RGOffload 拡張プロパティーを構成する」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ー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、プライオリティー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*

例 2–39 RGOffload リソースを構成する

この例では、RGOffload リソース rgofl を次のように構成する方法について説明します。

- 重要なリソースグループ oracle_rg には RGOffload リソースが含まれています。
- 重要なリソースは、oracle-server-rs です。
- 重要なリソースグループがオンラインになったときに、スケーラブルリソースグループ IWS-SC および IWS-SC-2 はオフロードされます。
- リソースグループ oracle_rg、IWS-SC 、および IWS-SC-2 は、クラスタ triped の任意のノード、つまり phys-triped-1、phys-triped-2 、または phys-triped-3 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 リソース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

[プライオリティーが高いリソースグループに RGOffload リソースを追加し、拡張プロパティーを設定する]
# 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
```

[プライオリティー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
```

[プライオリティー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 拡張プロパティーを構成する

この節では、RGOffload に対して構成可能な拡張プロパティーを示します。「調整可能」の欄には、そのプロパティーをいつ変更できるかが示されています。

通常、RGOffload リソースを作成するとき、拡張プロパティーを構成するには、コマンド行 scrgadm -x *parameter=value* を使用します。

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あとに、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指定します。

  このプロパティーは START メソッドだけが使用します。

  初期値: True

  調整: 任意の時点

max_offload_retry (整数型)
: クラスタ再構成またはリソースグループ再構成によりオフロードに障害が発生した場合の起動中に、リソースグループをオフロードしようとする回数を指定します。連続する再試行の間の間隔は 10 秒です。

  max_offload_retry が高すぎると、最大オフロード試行回数に到達する前に、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。この可能性を避けるために、次の式を使用して max_offload_retry を計算します。

  max-offload-retry < *start-timeout* / (*num-rg* × *offload-retry-interval*)

max-offload-retry	max_offload_retry 拡張プロパティーの値
start-timeout	RGOffload リソースの Start_timeout プロパティーの値
num-rg	オフロードされるリソースグループの数
offload-retry-interval	連続する再試行の間の間隔 (10 秒)

  このプロパティーは START メソッドだけが使用します。

初期値: 15

調整: 任意の時点

rg_to_offload (文字列型)
プライオリティー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されるリソースグループのコンマ区切りリストを指定します。このプロパティー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。

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。RGOffload は、rg_to_offload 拡張プロパティー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 が何らかの形で RG-A に依存する場合、両方のリソースグループが 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

障害モニター

RGOffload 障害モニターは、重要なリソースをマスターするノード上で、重要ではないリソースグループがオンラインになるのを防止します。障害モニターは、重要なリソースをマスターするノード上で、重要ではないリソースグループがオンラインであることを検出する場合があります。このような場合、障害モニターはそのほかのノードでリソースグループを起動しようとします。また障害モニターは、重要なリソースをマスターするノード上でリソースグループをオフラインにします。

重要でないリソースグループの desired_primaries はゼロに設定されているので、このあとで利用可能になったノード上では、オフロードされた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 の上限に到達するまで、可能な限り多くの主ノードで重要でないリソースグループを起動しようとします。ただし、障害モニターは、重要なリソースをマスターするノード上では、重要でないリソースグループをオフラインのままにします。

RGOffload は、リソースグループが 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グループを起動しようとします。リソースグループを UNMANAGED にするには、scswitch コマンドを使用します。

```
# scswitch -u -g resourcegroup
```

RGOffload リソースの Thorough_probe_interval プロパティーの値は、障害モニターの検証の間の間隔を指定します。

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、リソースタイプ、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、リソースタイプ、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、リソースタイプ、およびリソースの動的な状態を無視します。

▼ リソースグループ、リソースタイプ、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、リソースタイプ、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。

手順 **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 ツールの使用法の詳細については、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。リソースタイプ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タイプ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、リソースタイプ、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、リソースタイプ、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、リソースタイプ、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。

手順 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オンします。

   たとえば、node1 にログオ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、リソースタイプ、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 ツールの使用法の詳細については、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。リソースタイプ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タイプ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 データサービ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、当該リソースのシステムプロパティーと拡張プロパティー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ー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ーを必要な間隔 (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` 拡張プロパティー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ーとの関係

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、次のプロパティーの値を合計したものです。

- `Thorough_probe_interval` システムプロパティー
- `Probe_timeout` 拡張プロパティー

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ー

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ーを使用します。

- しきい値を設定するには、`Retry_count` システムプロパティー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ーに、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ーを設定します。このプロパティーに指定できる値については、133 ページの「リソースのプロパティー」における `Failover_mode` システムプロパティーの説明を参照してください。

付録 A

標準プロパティー

この付録では、標準リソースタイプ、リソース、リソースグループプロパティーについて説明します。また、システム定義プロパティーの変更および拡張プロパティーの作成に使用するリソースプロパティー属性についても説明します。

注 – リソースタイプ、リソース、リソースグループのプロパティー名は、大文字と小文字が区別されません。プロパティー名を指定する際には、大文字と小文字を任意に組み合わせることができます。

この付録の内容は、次のとおりです。

- 125 ページの「リソースタイププロパティー」
- 133 ページの「リソースのプロパティー」
- 149 ページの「 リソースグループのプロパティー」
- 158 ページの「リソースプロパティーの属性」

リソースタイププロパティー

以下に、Sun Cluster ソフトウェアにより定義されるリソースタイププロパティーを示します。プロパティー値は以下のように分類されます。

- 必須。プロパティー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、プロパティー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。RTR ファイル内に宣言を必要とするプロパティーです。宣言がない場合、RGM はこのプロパティーを作成しません。したがって、このプロパティーを管理ユーティリティーから利用することはできません。空白文字または空の文字列を値として指定できます。プロパティーが 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。RTR ファイル内に宣言と明示的な値を必要とするプロパティーです。宣言がない場合、RGM はこのプロパティーを作成しません。したがって、このプロパティーを管理ユーティリティーから利用することはできません。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。RTR ファイル内に宣言できるプロパティーです。プロパティーが RTR ファイル内で宣言されていない場合は、RGM がこれを作成し、デフォルト値を与えます。プロパティーが 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、プロパティーが 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。
- 照会のみ– 管理ツールから直接設定できません。

RTR ファイルでは宣言できないため、クラスタ管理者により設定される必要がある `Installed_nodes` と `RT_system` を除き、リソースタイププロパティーは管理ユーティリティーでは更新できません。

以下にプロパティー名とその説明を示します。

注 – `API_version` や `Boot` などのリソースタイププロパティー名では、大文字と小文字が区別されません。プロパティー名を指定する際には、大文字と小文字を任意に組み合わせることができます。

`API_version` (`integer`)
このリソースタイプの実装のサポートに必要なリソース管理 API の最小バージョン。

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`
3.1 3/05	`6`

RTR ファイルにおいて `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バージョン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` を宣言すると、このリソースタイプは、3.1 9/04 より前にリリースされた Sun Cluster のバージョンにはインストールされません。

注 – このプロパティーを宣言しないか、このプロパティーをデフォルト値 (2) に設定すると、データサービスは Sun Cluster 3.0 以降の Sun Cluster の任意のバージョンにインストールできます。

カテゴリ: 任意

デフォルト: 2

調整: NONE

`Boot` (`string`)
任意のコールバックメソッド。RGM がノード上で実行するプログラムのパスを指定します。このプログラムは、このリソースタイプ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を初期化します。

カテゴリ: 条件付/明示

デフォルト: デフォルトなし

調整: NONE

`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。

次の表に、このリソースタイププロパティーと `Scalable` リソースプロパティーを組み合わせて使用する方法を示します。

`Failover` リソースタイプの値	**`Scalable`** リソースの値	説明
`TRUE`	`TRUE`	この非論理的な組み合わせは指定しないでください。
`TRUE`	`FALSE`	この組み合わせは、フェイルオーバーサービスに対して指定します。
`FALSE`	`TRUE`	この組み合わせは、ネットワーク負荷分散に `SharedAddress` リソースを使用するスケーラブルサービスに指定します。 `SharedAddress` の詳細については、『Sun Cluster の概念 (Solaris OS 版)』を参照してください。
`FALSE`	`FALSE`	この組み合わせは一般的ではありませんが、ネットワーク負荷均衡を使用しないマルチマスターサービスを選択するときに使用できます。

Scalable の詳細については、r_properties(5) のマニュアルページと、『Sun Cluster の概念 (Solaris OS 版)』の第 3 章「重要な概念 - システム管理者とアプリケーション開発者」を参照してください。

カテゴリ: 任意

デフォルト: FALSE

調整: NONE

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に、RGM によって実行されるプログラムのパスです。

カテゴリ: 条件付/明示

デフォルト: デフォルトなし

調整: NONE

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に、RGM によって実行されるプログラムのパスです。

カテゴリ: 条件付/明示

デフォルト: デフォルトなし

調整: NONE

Init_nodes (enum)
RGM が Init、Fini、 Boot、Validate メソッドをコールするノードを示します。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 (このリソースタイプがインストールされる全てのノード) のいずれかです。

カテゴリ: 任意

デフォルト: RG_PRIMARIES

調整: NONE

Installed_nodes (string_array)
リソースタイプを実行できるクラスタノードの名前のリスト。このプロパティーは 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。

デフォルト: すべてのクラスタノード

調整: ANYTIME

Is_logical_hostname (boolean
TRUEは、このリソースタイプが、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 リソースタイプ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

デフォルト: デフォルトなし

調整: NONE

`Is_shared_address` (`boolean`)
`TRUE` は、このリソースタイプが、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共有アドレスリソースタイプ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

デフォルト: デフォルトなし

調整: NONE

`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タイプのフェイルオーバーを実行する前に、RGM によって実行されるプログラムのパスです。

カテゴリ: 条件付/明示

デフォルト: デフォルトなし

調整: NONE

`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 によって実行されるプログラムのパスです。

カテゴリ: 条件付/明示

デフォルト: デフォルトなし

調整: NONE

`Monitor_stop` (`string`)
`Monitor_start` が設定されている場合、必須のコールバックメソッドになります。この型のリソースの障害モニターを停止するために RGM によって実行されるプログラムのパスです。

カテゴリ: 条件付/明示

デフォルト: デフォルトなし

調整: NONE

`Pkglist` (`string_array`)
リソースタイプ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/明示

デフォルト: デフォルトなし

調整: NONE

`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 によって実行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あと、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/明示

デフォルト: デフォルトなし

調整: `NONE`

`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 によって実行されるプログラムのパスです。このメソッドは、ネットワークインタフェースが構成される前に必要な `Start` アクションを行います。

カテゴリ: 条件付/明示

デフォルト: デフォルトなし

調整: `NONE`

`Resource_list` (`string_array`)
リソースタイプの全リソースのリストです。クラスタ管理者はこのプロパティーを直接設定しません。ただし、クラスタ管理者がこの型のリソースをリソースグループに追加したり、リソースグループから削除した場合、RGM はこのプロパティーを更新します。

カテゴリ: 照会のみ

デフォルト: 空のリスト

調整: `NONE`

`Resource_type` (`string`)
リソースタイプの名前です。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*`:` *rt-version*

リソースタイプ名は RTR ファイル内に指定された 3 つのプロパティー *vendor-id*、*resource-type* 、*rt-version* で構成されます。scrgadm コマンドは、ピリオド (.コロン (:) の区切り文字を挿入します。リソースタイプの名前の最後の部分、rt-version には、RT_version プロパティーと同じ値が入ります。*vendor-id* が一意であることを保証するためには、リソースタイプを作成した会社の株式の略号を使用します。Sun Cluster 3.1 以前に登録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

デフォルト: 空の文字列

調整: NONE

RT_basedir (string)
コールバックメソッドの相対パス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ディレクトリに設定する必要があります。このパスには、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。

カテゴリ: 必須 (絶対パスでないメソッドパスがある場合)

デフォルト: デフォルトなし

調整: NONE

RT_description (string)
リソースタイプの簡単な説明です。

カテゴリ: 条件付き

デフォルト: 空の文字列

調整: NONE

RT_system (boolean)
リソースタイプの RT_system プロパティーが TRUE の場合、そのリソースタイプは削除できません (scrgadm -r -t *resource-type-name*)。このプロパティーは、LogicalHostname など、クラスタのインフラをサポートするリソースタイプを間違って削除してしまうことを防ぎます。しかし、RT_system プロパティーはどのリソースタイプにも適用できます。

RT_system プロパティーが TRUE に設定されたリソースタイプを削除するには、まず、このプロパティーを 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を持つリソースタイプを削除するときには注意してください。

カテゴリ: 任意

デフォルト: FALSE

調整: ANYTIME

`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` プロパティーは 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: 条件付き/明示または必須

デフォルト: デフォルトなし

調整: `NONE`

`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。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: 任意

デフォルト: `FALSE`

調整: `NONE`

`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 によって実行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

デフォルト: デフォルトなし

調整: `NONE`

`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 によって実行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

デフォルト: デフォルトなし

調整: `NONE`

`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ーが変更されたときに、RGM によって実行されるプログラムのパスです。

カテゴリ: 条件付/明示

デフォルト: デフォルトなし

調整: `NONE`

`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ー値を検査するために RGM により実行されるプログラムのパスです。

カテゴリ: 条件付/明示

デフォルト: デフォルトなし

調整: NONE

`Vendor_ID` (`string`)
`Resource_type` を参照してください。

カテゴリ: 条件付き

デフォルト: デフォルトなし

調整: NONE

リソースのプロパティー

この節では、Sun Cluster ソフトウェアで定義されているリソースプロパティーについて説明します。プロパティー値は以下のように分類されます。

- 必須。クラスタ管理者は、管理ユーティリティーを使ってリソースを作成するとき、必ず値を指定しなければなりません。
- 任意。クラスタ管理者がリソースグループの作成時に値を指定しないと、システムのデフォルト値が使用されます。
- 条件付。RGM は、RTR ファイル内にプロパティーが宣言されている場合にかぎりプロパティーを作成します。宣言されていない場合プロパティーは存在せず、クラスタ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ーは、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ーの説明を参照してください。
- 照会のみ。管理ツールで直接設定することはできません。

158 ページの「リソースプロパティーの属性」で説明されている `Tunable` 属性は、次のように、リソースプロパティー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。

`FALSE` または `NONE`	不可
`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

以下にプロパティー名とその説明を示します。

`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。

このプロパティーは、`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` に設定されている必要があります。

このプロパティーは、スケーラブルサービス専用です。

カテゴリ: 任意

デフォルト: デフォルトなし

調整: `ANYTIME`

各コールバックメソッドの `Boot_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)。特定のリソースタイプに関して、タイムアウトのプロパティーは RTR ファイルで宣言されているメソッドに対してのみ定義されます。

カテゴリ: 条件付き/任意

デフォルト: RTR ファイルにメソッド自体が宣言されている場合は `3600` (1 時間)

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)。このプロパティーは 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、クラスタ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、このプロパティーは任意です。

RTR ファイル内に `Tunable` 属性が指定されていない場合、このプロパティーの `Tunable` 値は `WHEN_DISABLED` になります。

カテゴリ: 条件付き

デフォルト: デフォルトなし

調整: `WHEN_DISABLED`

拡張プロパティー
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ー。リソースタイプの実装によって、これらのプロパティーを定義します。拡張プロパティーに設定可能な各属性については、158 ページの「リソースプロパティーの属性」を参照してください。

カテゴリ: 条件付き

デフォルト: デフォルトなし

調整: 特定のプロパティーに依存

Failover_mode (enum)
リソースが正常に開始または停止できなかった場合、またはリソースモニターが正常ではないリソースを検出し、その結果再起動またはフェイルオーバーを要求する場合に RGM が取る回復アクションを変更します。

NONE、SOFT、または HARD (メソッドの失敗)

これらの設定は、起動または停止メソッド (Prenet_start、Start、Monitor_stop、 Stop、Postnet_stop) が失敗した場合にのみフェイルオーバー動作に影響します。リソースが正常に起動すれば、NONE、SOFT、および HARD は、リソースモニターが scha_control コマンドまたは scha_control() 関数で開始したこれ以降の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。scha_control(1HA) および scha_control(3HA) のマニュアルページを参照してください。NONE は、前に示した起動または停止メソッドが失敗した場合、RGM は回復アクションを行わないことを示します。SOFT または HARD は、Start または Prenet_start メソッドが失敗した場合、RGM はリソースのグループを別のノードに再配置することを示します。Start または Prenet_start の失敗については、SOFT と HARD は同じになります。

停止メソッド (Monitor_stop、Stop、または Postnet_stop) の失敗の場合、SOFT は NONE と同じになります。これらの停止メソッドのいずれかが失敗した場合に、Failover_mode が HARD に設定されていれば、RGM はノードを再起動してリソースグループを強制的にオフラインにします。これにより RGM は別のノードでグループの起動を試みることが可能になります。

RESTART_ONLY または LOG_ONLY

起動メソッドまたは停止メソッドが失敗した場合にフェイルオーバー動作に影響する NONE、SOFT、および HARD とは異なり、RESTART_ONLY と LOG_ONLY はすべてのフェイルオーバー動作に影響します。フェイルオーバー動作には、モニター (scha_control) が開始するリソースおよびリソースグループの再起動と、リソースモニター (scha_control) により開始されるギブオーバーが含まれます。RESTART_ONLY は、モニターが scha_control を実行してリソースまたはリソースグループを再起動できることを示します。RGM では、Retry_interval の間に Retry_count 回数だけ再起動を試行できます。Retry_count の回数を超えると、それ以上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す。Failover_mode を LOG_ONLY に設定することは、Retry_count をゼロに設定した状態で Failover_mode を RESTART_ONLY に設定することと同じです。

RESTART_ONLY または LOG_ONLY (メソッドの失敗)

Prenet_start、Start、Monitor_stop 、Stop、または Postnet_stop メソッドが失敗した場合、 RESTART_ONLY と LOG_ONLY は NONE と同じになります。つまり、ノードのフェイルオーバーやリブートはどちらも行われません。

データサービスに対する Failover_mode 設定の影響

Failover_mode の各設定がデータサービスに及ぼす影響は、データサービスが監視されているかどうか、およびデータサービスが Data Services Development Library (DSDL) に基づいているかどうかによって決まります。

- データサービスが監視されるのは、データサービスが Monitor_start メソッドを実装し、リソースの監視が有効になっている場合です。RGM は、リソースそれ自体を起動した後で Monitor_start メソッドを実行することにより、リソースモニターを起動します。リソースモニターはリソースが正常であるかどうかを検証します。検証が失敗した場合、リソースモニターは、scha_control() 関数を呼び出すことで再起動またはフェイルオーバーを要求する場合があります。DSDL ベースのリソースの場合、検証によりデータサービスの部分的な障害 (機能低下) または完全な障害が明らかになる場合があります。部分的な障害が繰り返し蓄積されると、完全な障害になります。
- データサービスが監視されないのは、データサービスが Monitor_start メソッドを提供しないか、リソースの監視が無効になっている場合です。
- DSDL ベースのデータサービスには、Agent Builder や GDS により開発されたデータサービス、または DSDL を直接使用して開発されたデータサービスが含まれます。HA Oracle など一部のデータサービスは、DSDL を使用せずに開発されています。

NONE、SOFT、または HARD (検証の失敗)

Failover_mode が NONE、SOFT、または HARD に設定され、データサービスが監視対象の DSDL ベースのサービスであり、また検証が完全に失敗した場合、モニターは scha_control () 関数を呼び出してリソースの再起動を要求します。検証が失敗し続ける場合、リソースは Retry_interval 期間内の Retry_count の最大回数まで再起動されます。Retry_count の再起動数に到達した後も検証が再び失敗した場合、モニターは別のノードに対してリソースのグループのフェイルオーバーを要求します。

Failover_mode が NONE、SOFT、または HARD に設定され、データサービスが監視対象外の DSDL ベースのサービスである場合、検出される唯一の障害はリソースのプロセスツリーの故障のみです。リソースのプロセスツリーが故障すると、リソースが再起動されます。

データサービスが DSDL ベースのサービスではない場合、再起動またはフェイルオーバー動作は、リソースモニターがどのようにコード化されているかによって決まります。たとえば Oracle リソースモニターは、リソースまたはリソースグループを再起動するか、リソースグループのフェイルオーバーを行うことで回復します。

RESTART_ONLY (検証の失敗)

Failover_mode が RESTART_ONLY に設定され、データサービスが監視対象の DSDL ベースのサービスである場合、検証が完全に失敗すると、リソースは Retry_interval の期間内に Retry_count の回数再起動されます。ただし、Retry_count の回数を超えると、リソースモニターは終了し、リソースの状態を FAULTED に設定して、状態メッセージ「Application faulted, but not restarted. Probe quitting.」を生成します。この時点で監視はまだ有効ですが、リソースがクラスタ管理者により修復および再起動されるまで、リソースは事実上監視対象外になります。

Failover_mode が RESTART_ONLY に設定され、データサービスが監視対象外の DSDL ベースのサービスである場合、プロセスツリーが故障すると、リソースは再起動されません。

監視対象データサービスが DSDL ベースのデータサービスではない場合、回復動作はリソースモニターがどのようにコード化されているかに依存します。Failover_mode が RESTART_ONLY に設定されている場合、リソースまたはリソースグループは、Retry_interval の期間内に Retry_count の回数だけ scha_control() 関数の呼び出しにより再起動できます。リソースモニターが Retry_count を超えると、再起動の試みは失敗します。モニターが scha_control() 関数を呼び出してフェイルオーバーを要求する場合、その要求も同様に失敗します。

LOG_ONLY (検証の失敗)

Failover_mode がデータサービスに対して LOG_ONLY に設定されている場合、すべての scha_control() はリソースまたはリソースグループの再起動を要求するか、除外されているグループのフェイルオーバーを要求します。データサービスが DSDL ベースである場合、検証が完全に失敗した場合メッセージが記録されますが、リソースは再起動されません。検証が Retry_interval の期間内に Retry_count の回数以上完全に失敗した場合、リソースモニターは終了し、リソースの状態を FAULTED に設定して、状態メッセージ「Application faulted, but not restarted. Probe quitting.」を生成します。この時点で監視はまだ有効ですが、リソースがクラスタ管理者により修復および再起動されるまで、リソースは事実上監視対象外になります。

Failover_mode が LOG_ONLY に設定されていて、データサービスが監視対象外の DSDL ベースのサービスであり、プロセスツリーが故障した場合、メッセージが記録されますが、リソースは再起動されません。

監視対象データサービスが DSDL ベースのデータサービスではない場合、回復動作はリソースモニターがどのようにコード化されているかに依存します。Failover_mode が LOG_ONLY に設定されている場合、すべての scha_control() 要求はリソースまたはリソースグループを再起動するか、グループの障害をフェイルオーバーします。

カテゴリ: 任意

デフォルト: NONE

調整: ANYTIME

各コールバックメソッドの Fini_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)。特定のリソースタイプに関して、タイムアウトのプロパティーは RTR ファイルで宣言されているメソッドに対してのみ定義されます。

カテゴリ: 条件付き/任意

デフォルト: RTR ファイルにメソッド自体が宣言されている場合は 3600 (1 時間)

調整: ANYTIME

各コールバックメソッドの Init_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)。特定のリソースタイプに関して、タイムアウトのプロパティーは RTR ファイルで宣言されているメソッドに対してのみ定義されます。

カテゴリ: 条件付き/任意

デフォルト: RTR ファイルにメソッド自体が宣言されている場合は 3600 (1 時間)

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。このプロパティーは、スケーラブルサービス専用です。RTR ファイルに Scalable プロパティーが宣言されている場合、RGM は自動的にこのプロパティーを作成します。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 プロパティー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。ワイルドスティッキーサービスの IP アドレスに接続する Lb_sticky_wild で指定されたクライアントの IP アドレスは、IP アドレスが到着するポート番号とは無関係に、常に同じクラスタノードに送られます。

カテゴリ: 条件付き/任意

デフォルト: Lb_weighted

調整: AT_CREATION

Load_balancing_weights (string_array)
このプロパティーは、スケーラブルサービス専用です。RTR ファイルに Scalable プロパティーが宣言されている場合、RGM は自動的にこのプロパティーを作成します。形式は、「*weight@node, weight@node*」になります。*weight* は指定のノード (*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 が負荷の 3/4 を受け取ることを指定します。デフォルトの空の文字列 ("") は、一定の分散を指定します。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。

RTR ファイル内に Tunable 属性が指定されていない場合、このプロパティーの Tunable 値は ANYTIME になります。このプロパティーを変更すると、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

デフォルト: 空の文字列 (“”)

調整: ANYTIME

各コールバックメソッドの Monitor_check_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)。特定のリソースタイプに関して、タイムアウトのプロパティーは RTR ファイルで宣言されているメソッドに対してのみ定義されます。

カテゴリ: 条件付き/任意

デフォルト: RTR ファイルにメソッド自体が宣言されている場合は 3600 (1 時間)

調整: ANYTIME

各コールバックメソッドの Monitor_start_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)。特定のリソースタイプに関して、タイムアウトのプロパティーは RTR ファイルで宣言されているメソッドに対してのみ定義されます。

カテゴリ: 条件付き/任意

デフォルト: RTR ファイルにメソッド自体が宣言されている場合は 3600 (1 時間)

調整: ANYTIME

各コールバックメソッドの Monitor_stop_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)。特定のリソースタイプに関して、タイムアウトのプロパティーは RTR ファイルで宣言されているメソッドに対してのみ定義されます。

カテゴリ: 条件付き/任意

デフォルト: RTR ファイルにメソッド自体が宣言されている場合は 3600 (1 時間)

調整: ANYTIME

Monitored_switch (enum)
クラスタ管理者が管理ユーティリィティーを使っ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 に設定されている場合、リソースの監視は停止されますが、リソースそれ自体はオンラインのままになります。監視が再度有効になるまで、Monitor_start メソッドは呼び出されません。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、このプロパティーは存在しません。

カテゴリ: 照会のみ

デフォルト: デフォルトなし

調整: NONE

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、このプロパティー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、このプロパティー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 プロパティーが宣言さ

れている場合、RGM は自動的にこのプロパティーを作成します。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。

RTR ファイル内に Tunable 属性が指定されていない場合、このプロパティーの Tunable 値は AT_CREATION になります。

注 – このプロパティーを 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

デフォルト: デフォルトなし

調整: AT_CREATION

各クラスタノード上の Num_resource_restarts (integer)
このプロパティーを直接設定することはできません。このプロパティーは、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ます。*n* はリソースの Retry_interval プロパティーの値です。このリソースが scha_control ギブオーバーを実行した場合は常に、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 (0) にリセットされます。

リソースタイプが Retry_interval プロパティーを宣言していない場合、この型のリソースに Num_resource_restarts プロパティーを使用できません。

カテゴリ: 照会のみ

デフォルト: デフォルトなし

調整: NONE

各クラスタノード上の Num_rg_restarts (integer)
このプロパティーを直接設定することはできません。このプロパティーは、RGM によって、リソースを含むリソースグループに対してリソースがこのノード上で過去 *n* 秒以内に行った scha_control Restart の呼び出し回数に設定されます。*n* は、リソースの Retry_interval プロパティーの値です。リソースタイプが Retry_interval プロパティーを宣言していない場合、このタイプのリソースには Num_resource_restarts プロパティーを使用できません。

カテゴリ: 説明を参照

デフォルト: デフォルトなし

調整: NONE

`On_off_switch` (enum)
クラスタ管理者が管理ユーティリィティーを使っ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。無効に設定されている場合、リソースはオフラインにされ、再度有効にされるまでコールバックは実行されません。

カテゴリ: 照会のみ

デフォルト: デフォルトなし

調整: `NONE`

`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`)。プロトコルには、次のものを指定できます。

- `tcp` (TCP IPv4)
- `tcp6` (TCP IPv6)
- `udp` (UDP IPv4)
- `udp6` (UDP IPv6)

`Scalable` プロパティーが 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、このプロパティーは 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。

Apache 用にこのプロパティーを設定する方法は、『Sun Cluster Data Service for Apache ガイド (Solaris OS 版)』を参照してください。

カテゴリ: 条件付き/必須

デフォルト: デフォルトなし

調整: `ANYTIME`

各コールバックメソッドの `Postnet_stop_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)。特定のリソースタイプに関して、タイムアウトのプロパティーは RTR ファイルで宣言されているメソッドに対してのみ定義されます。

カテゴリ: 条件付き/任意

デフォルト: RTR ファイルにメソッド自体が宣言されている場合は `3600` (1 時間)

調整: `ANYTIME`

各コールバックメソッドの `Prenet_start_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)。特定のリソースタイプに関して、タイムアウトのプロパティーは RTR ファイルで宣言されているメソッドに対してのみ定義されます。

カテゴリ: 条件付き/任意

デフォルト: RTR ファイルにメソッド自体が宣言されている場合は 3600 (1 時間)

調整: ANYTIME

R_description (string)
リソースの簡単な説明。

カテゴリ: 任意

デフォルト: 空の文字列

調整: ANYTIME

Resource_dependencies (string_array)
Resource_dependencies リソースが強い依存関係を持っている同じグループまたは異なるグループ内のリソースのリスト。このリソースを起動するためには、リストのすべてのリソースがオンラインになっていなければなりません。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、このリソースは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、START_FAILED 状態であるために、このリソースのリスト内のリソースが起動しない可能性があります。起動に失敗した異なるリソースグループのリソースに対する依存関係により、このリソースがオフラインのままである場合、このリソースのグループは PENDING_ONLINE_BLOCKED 状態に入ります。

このリソースが、リストのリソースと同時にオフラインにされる場合は、このリソースが停止されてから、リストのほかのリソースが停止されます。ただし、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、異なるリソースグループに属するリストのリソースは停止されます。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。

同じリソースグループ内では、デフォルトとして、アプリケーションリソースが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依存性を持っています。詳細については、149 ページの「 リソースグループのプロパティー」の Implicit_network_dependencies を参照してください。

同じリソースグループ内では、依存性の順序に従って Prenet_start メソッドが Start メソッドより先に実行されます。Postnet_stop メソッドは Stop メソッドよりあとに、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 を完了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グループが Stop および Postnet_stop を完了するまで待機してから、Stop を実行します。

カテゴリ: 任意

デフォルト: 空のリスト

調整: ANYTIME

Resource_dependencies_restart (string_array)
Resource_dependencies_restart リソースが再起動の依存関係を持っている、同じグループまたは異なるグループ内のリソースのリスト。

このプロパティーの動作は Resource_dependencies とよく似ていますが、1 点例外があります。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あと、RGM はこのリソースを再起動します。

カテゴリ: 任意

デフォルト: 空のリスト

調整: ANYTIME

Resource_dependencies_weak (string_array)
Resource_dependencies_weak リソースが弱い依存関係を持っている、同じグループ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、メソッド呼び出しの順序を決定します。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始動に失敗したりオフラインのままであっても、リソースは起動されます。

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。リストのすべてのリソースが起動した時点、または起動に失敗した時点で、このリソースは起動し、そのグループは再度 PENDING_ONLINE 状態になります。

このリソースが、リストのリソースと同時にオフラインにされる場合は、このリソースが停止されてから、リストのほかのリソースが停止されます。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、リストのリソースは停止されます。リストのリソースを無効にするためには、このリソースをまず無効にする必要があります。

同じリソースグループ内では、依存性の順序に従って Prenet_start メソッドが Start メソッドより先に実行されます。Postnet_stop メソッドは Stop メソッドよりあとに、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 を完了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グループが Stop および Postnet_stop を完了するまで待機してから、Stop を実行します。

カテゴリ: 任意

デフォルト: 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

デフォルト: デフォルトなし

調整: NONE

Resource_project_name (string)
リソースに関連付けられた Solaris プロジェクト名。このプロパティーは、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。このプロパティーが指定されなかった場合、リソースを含むリソースグループの RG_project_name プロパティーからプロジェクト名が決定されます (rg_properties(5) のマニュアルページを参照)。どちらのプロパティー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。指定されたプロジェクト名は、プロジェクトデータベースに存在する必要があります (projects(1) のマニュアルページ、および『Solaris のシステム管理 (Solaris コンテナ : 資源管理と Solaris ゾーン)』を参照)。

このプロパティーは Solaris 9 以降でサポートされます。

注 – このプロパティーへの変更は、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: 任意

デフォルト: Null

調整: ANYTIME

各クラスタノード上の 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ーを構成できません。

カテゴリ: 照会のみ

デフォルト: デフォルトなし

調整: NONE

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。Retry_count を超えると、特定のデータサービス、および Failover_mode プロパティーの設定に応じて、モニターは次のいずれかのアクションを実行します。

- リソースが障害状態であったとしても、リソースグループが現在の主ノード上にとどまることを許可する
- 別のノードへのリソースグループのフェイルオーバーを要求する

このプロパティーは RGM によって作成されます。このプロパティーは RTR ファイルに宣言されている場合にかぎり、クラスタ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、このプロパティーは任意です。

RTR ファイル内に Tunable 属性が指定されていない場合、このプロパティーの Tunable 値は WHEN_DISABLED になります。

注 – このプロパティーにマイナスの値を指定すると、モニターは無限回リソースの再起動を試みます。

カテゴリ: 条件付き

デフォルト: 上記を参照

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、このプロパティーと Retry_count を組み合わせて使用します。このプロパティーは 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、クラスタ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、このプロパティーは任意です。

RTR ファイル内に Tunable 属性が指定されていない場合、このプロパティーの Tunable 値は WHEN_DISABLED になります。

カテゴリ: 条件付き

デフォルト: デフォルトなし (上記を参照)

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 ソフトウェア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。

このプロパティーが 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ーを自動的に作成します。Affinity_timeout 、Load_balancing_policy、Load_balancing_weights、Network_resources_used、Port_list、

UDP_affinity、および Weak_affinity。これらのプロパティーは、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 ファイルで宣言されている場合、Scalable のデフォルトは TRUE です。

RTR ファイルにこのプロパティー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ー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、このプロパティー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ーをいっさい設定しません。ただし、Network_resources_used および Port_list プロパティーは、RTR ファイルで明示的に宣言できます。これらのプロパティーは、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す。

このリソースプロパティーを Failover リソースタイププロパティーと組み合わせて使用する方法の詳細については、r_properties(5) のマニュアルページを参照してください。

カテゴリ: 任意

デフォルト: デフォルトなし

調整: AT_CREATION

各コールバックメソッドの Start_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)。特定のリソースタイプに関して、タイムアウトのプロパティーは RTR ファイルで宣言されているメソッドに対してのみ定義されます。

カテゴリ: 条件付き/任意

デフォルト: RTR ファイルにメソッド自体が宣言されている場合は 3600 (1 時間)

調整: ANYTIME

各クラスタノード上の Status (enum)
リソースモニターにより scha_resource_setstatus コマンドまたは scha_resource_setstatus() 関数で設定されます。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

デフォルト: デフォルトなし

調整: NONE

各クラスタノード上の Status_msg (string)
リソースモニターによって、Status プロパティー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されると、RGM は自動的にこのプロパティーを空文字列でリセットします。ただし、このプロパティーがリソースのメソッドによって設定される場合を除きます。

カテゴリ: 照会のみ

デフォルト: デフォルトなし

調整: NONE

各コールバックメソッドの Stop_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)。特定のリソースタイプに関して、タイムアウトのプロパティーは RTR ファイルで宣言されているメソッドに対してのみ定義されます。

カテゴリ: 条件付き/任意

デフォルト: RTR ファイルにメソッド自体が宣言されている場合は 3600 (1 時間)

調整: ANYTIME

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)。このプロパティーは 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、クラスタ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、このプロパティーは任意です。

RTR ファイル内に Tunable 属性が指定されていない場合、このプロパティーの Tunable 値は WHEN_DISABLED になります。

カテゴリ: 条件付き

デフォルト: 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

デフォルト: デフォルトなし

調整: NONE

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タイプのバージョンを指定します。このプロパティーは 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。このプロパティーの値は、リソースタイプの RT_version プロパティーと等しくなります。リソースの作成時、Type_version プロパティー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 プロパティーが新しい値に変更されることがあります。

このプロパティーの調整については、次の情報から判断されます。

- 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

デフォルト: デフォルトなし

調整: 説明を参照

UDP_affinity (boolean)
このプロパティーが TRUE に設定されている場合、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すべての TCP トラフィックを処理している同じサーバーノードに送信されます。

このプロパティーは、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 に設定されている必要があります。

このプロパティーは、スケーラブルサービス専用です。

カテゴリ: 任意

デフォルト: デフォルトなし

調整: WHEN_DISABLED

各コールバックメソッドの Update_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)。特定のリソースタイプに関して、タイムアウトのプロパティーは RTR ファイルで宣言されているメソッドに対してのみ定義されます。

カテゴリ: 条件付き/任意

デフォルト: RTR ファイルにメソッド自体が宣言されている場合は 3600 (1 時間)

調整: ANYTIME

各コールバックメソッドの Validate_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)。特定のリソースタイプに関して、タイムアウトのプロパティーは RTR ファイルで宣言されているメソッドに対してのみ定義されます。

カテゴリ: 条件付き/任意

デフォルト: RTR ファイルにメソッド自体が宣言されている場合は 3600 (1 時間)

調整: ANYTIME

Weak_affinity (boolean)
このプロパティーが TRUE に設定されている場合、このプロパティーにより弱い形式のクライアントアフィニティーが有効になります。弱い形式のクライアントアフィニティーが有効になっている場合、特定のクライアントからの接続は、次の場合を除き、同じサーバーノードに送信されます。

- 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にサーバーのリスナーが起動する場合。
- クラスタ管理者により管理アクションが実行されたため、スケーラブルリソースの Load_balancing_weights が変更された場合。

弱いアフィニティー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。

このプロパティーは、Load_balancing_policy が Lb_sticky または Lb_sticky_wild の場合にかぎり有効です。

このプロパティーは、スケーラブルサービス専用です。

カテゴリ: 任意

デフォルト: 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ー

以下に、Sun Cluster ソフトウェアにより定義されるリソースグループのプロパティーを示します。プロパティー値は以下のように分類されます。

- 必須。クラスタ管理者は、管理ユーティリティーを使ってリソースグループを作成するとき、必ず値を指定しなければなりません。
- 任意。クラスタ管理者がリソースグループの作成時に値を指定しないと、システムのデフォルト値が使用されます。
- 照会のみ。管理ツールで直接設定することはできません。

以下にプロパティー名とその説明を示します。

Auto_start_on_new_cluster (boolean)
このプロパティーは、新しいクラスタの形成時にリソースグループマネージャー (RGM) が自動的にリソースグループを起動するかどうかを制御します。デフォルトは TRUE です。

TRUE に設定した場合、クラスタの全てのノードが同時に再起動すると、RGM はリソースグループを自動的に起動して Desired_primaries を取得しようとします。

FALSE に設定されていると、クラスタが再起動されたとき、リソースグループは自動的には起動しません。scswitch コマンドまたは同等の GUI 指令を使用して、最初にリソースグループが手動でオンラインに切り替えられるまで、リソースグループはオフラインのままになります。その後、このリソースグループは通常のフェイルオーバー動作を再開します。

カテゴリ: 任意

デフォルト: TRUE

調整: ANYTIME

`Desired_primaries` (integer)
グループが同時に実行できるノード数として望ましい値。

デフォルトは 1 です。`RG_mode` プロパティーが `Failover` である場合、このプロパティーの値は 1 以下である必要があります。`RG_mode` プロパティーが `Scalable` である場合、1 より大きな値に設定できます。

カテゴリ: 任意

デフォルト: `1`

調整: `ANYTIME`

`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: 任意

デフォルト: `FALSE`

調整: `ANYTIME`

`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。クラスタ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: 任意

デフォルト: すべてのグローバルリソース

調整: `ANYTIME`

`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。このとき、RGM は、すべてのネットワークアドレスリソースを起動してからその他のリソースを起動します。また、グループ内のその他のすべてのリソースを停止してからネットワークアドレスリソースを停止します。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タイプがあります。

スケーラブルなリソースグループ内では、このプロパティーの影響はありません。これは、スケーラブルなリソースグループにはネットワークアドレスリソースが含まれないからです。

カテゴリ: 任意

デフォルト: `TRUE`

調整: `ANYTIME`

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 プロパティーが Failover である場合、このプロパティーの値は 1 以下である必要があります。RG_mode プロパティーが Scalable である場合、1 より大きな値に設定できます。

カテゴリ: 任意

デフォルト: 1

調整: ANYTIME

Nodelist (string_array)
グループを優先度順にオンラインにできるクラスタノードのリストです。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。

カテゴリ: 任意

デフォルト: すべてのクラスタノードの順不同のリスト

調整: ANYTIME

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ーです。各リソースグループの Pathprefix は、一意にする必要があります。

カテゴリ: 任意

デフォルト: 空の文字列

調整: ANYTIME

Pingpong_interval (integer)
次の条件で、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)。

- 再構成が発生している場合
- scha_control GIVEOVER コマンドまたは関数の実行の結果として

再構成が発生したときは、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い場合があります。この障害が発生した原因は、リソースの Start 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。その結果、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と判断され、RGM は別のマスターを探します。

scha_control コマンドまたは scha_control GIVEOVER コマンドが特定のノード上でリソースによって実行され、それによりそのリソースグループが別のノードにフェイルオーバーした場合、Pingpong_interval 秒が経過するまで、(scha_control コマンドが実行された) 最初のノードは、同じリソースによる別の scha_control GIVEOVER の宛先になることはできません。

カテゴリ: 任意

デフォルト: 3600 (1 時間)

調整: ANYTIME

Resource_list (string_array)
グループ内に含まれるリソースのリストです。クラスタ管理者はこのプロパティーを直接設定しません。このプロパティーは、クラスタ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グループからリソースを削除したりすると、RGM により自動的に更新されます。

カテゴリ: 照会のみ

デフォルト: デフォルトなし

調整: NONE

RG_affinities (string)
RGM は、別の特定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ーの場合)、あるいは、特定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 (否定的なアフィニティーの場合) 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ー)
- + (弱い肯定的なアフィニティー)
- - (弱い否定的なアフィニティー)
- -- (強い否定的なアフィニティー)
- +++ (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ー)

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ーを持っており、RG3 に対しては強い否定的なアフィニティーを持っていることを示します。

RG_affinities の使用法については、第 2 章を参照してください。

カテゴリ: 任意

デフォルト: 空の文字列

調整: ANYTIME

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 が一緒のグループは、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。

- ノードがクラスタに結合されると、そのノードでは、RG2 のすべてのリソースに対する Boot メソッドが終わってから、 RG1 のリソースに対する Boot メソッドが実行されます。
- 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 内のどのリソースでも開始メソッド (Prenet_start または Start) は実行されません。
- 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、RG1 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、RG2 内のどのリソースでも停止メソッド(Stop または Postnet_stop) は実行されません。
- RG1 または RG2 の主ノードをスイッチする場合、それによって RG1 がいずれかのノードでオンラインに、RG2 がすべてのノードでオフラインになる場合は、このスイッチは失敗します。詳細については、scswitch(1M) および scsetup(1M) のマニュアルページを参照してください。
- RG2 上で Desired_primaries プロパティーをゼロに設定した場合、RG1 上で Desired_primaries プロパティー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 に対する Auto_start_on_new_cluster が FALSE に設定されている場合は、RG1 に対する Auto_start_on_new_cluster プロパティーを TRUE に設定することはできません。

カテゴリ: 任意

デフォルト: 空のリスト

調整: ANYTIME

RG_description (string)
リソースグループの簡単な説明です。

カテゴリ: 任意

デフォルト: 空の文字列

調整: ANYTIME

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。このプロパティーが TRUE に設定されている場合、大域デバイスはスイッチオーバーされます。このプロパティーが FALSE に設定されている場合、大域デバイスは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 プロパティーの設定によります。

RG_is_frozen プロパティーをユーザーが直接設定することはありません。RG_is_frozen プロパティーは、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: 任意

デフォルト: デフォルトなし

調整: NONE

RG_mode (enum)
リソースグループがフェイルオーバーグループなのか、スケーラブルグループなのかを指定します。この値が Failover であれば、 RGM はグループの Maximum_primaries プロパティーの値を 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。

このプロパティーの値が Scalable であれば、Maximum_primaries プロパティーは 1 より大きな値に設定されることがあります。その結果、このグループのマスターが同時に複数存在する可能性があります。Failover プロパティーの値が TRUE であるリソースを、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 である場合、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: 任意

デフォルト: Maximum_primaries の値によります。

調整: NONE

RG_name (string)
リソースグループの名前。これは必須プロパティーです。この値は、クラスタ内で一意でなければなりません。

カテゴリ: 必須

デフォルト: デフォルトなし

調整: NONE

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 (projects(1) のマニュアルページを参照)。このプロパティーは、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、Resource_project_name プロパティーセットを持たないリソース用として、このプロジェクト名下で関連プロセスを起動します (r_properties(5) のマニュアルページを参照)。指定されたプロジェクト名は、プロジェクトデータベースに存在する必要があります (projects(1) のマニュアルページ、および『Solaris のシステム管理 (Solaris コンテナ : 資源管理と Solaris ゾーン)』を参照)。

このプロパティーは Solaris 9 以降でサポートされます。

注 – このプロパティーへの変更は、リソースの次回起動時に有効になります。

カテゴリ: 任意

デフォルト: テキスト文字列「default」

調整: ANYTIME

各クラスタノード上の RG_state (enum)
RGM により UNMANAGED、ONLINE、OFFLINE、PENDING_ONLINE、PENDING_OFFLINE、ERROR_STOP_FAILED、ONLINE_FAULTED、または PENDING_ONLINE_BLOCKED に設定され、各クラスタノード上のグループの状態を表します。

ユーザーはこのプロパティーを構成できません。しかし、scswitch コマンドを実行することによって、あるいは同等の scsetup や SunPlex Manager コマンドを使用して、このプロパティーを間接的に設定することは可能です。RGM の制御下にないときは、グループは UNMANAGED 状態で存在することができます。

各状態の説明は次のとおりです。

注 – すべてのノードに適用される UNMANAGED 状態を除き、状態は個別のノードにのみ適用されます。たとえば、リソースグループがノード A では OFFLINE であり、ノード B では PENDING_ONLINE である場合があります。

UNMANAGED	新しく作成されたリソースグループの最初の状態や、前に管理されていたリソースグループの状態。そのグループのリソースに対して Init メソッドがまだ実行されていないか、そのグループのリソースに対して Fini メソッドがすでに実行されています。 このグループは RGM によって管理されていません。
ONLINE	リソースグループはノード上ですでに起動されています。つまり、各リソースに適用可能な起動メソッド Prenet_start、Start、および Monitor_start は、グループ内のすべての有効なリソースに対して正常に実行されました。
OFFLINE	リソースグループはノードですでに停止されています。つまり、各リソースに適用可能な停止メソッド Monitor_stop、Stop、および Postnet_stop はグループ内のすべての有効なリソースに対して正常に実行されました。さらに、リソースグループがノードで最初に起動されるまでは、グループにこの状態が適用されます。

PENDING_ONLINE
: リソースグループはノードで起動されようとしています。各リソースに適用可能な起動メソッド Prenet_start、Start、および Monitor_start はグループ内の有効なリソースに対して実行中です。

PENDING_OFFLINE
: リソースグループはノードで停止されようとしています。各リソースに適用可能な停止メソッド Monitor_stop、Stop、および Postnet_stop はグループ内の有効なリソースに対して実行中です。

ERROR_STOP_FAILED
: リソースグループ内の 1 つまたは複数のリソースが停止に失敗し、Stop_failed 状態になっています。グループのほかのリソースがオンラインまたはオフラインである可能性があります。ERROR_STOP_FAILED 状態がクリアされるまで、このリソースグループはノード上での起動が許可されません。

  scswitch - c などの管理コマンドを使用して Stop_failed リソースを手動で終了させ、その状態を OFFLINE に再設定する必要があります。

ONLINE_FAULTED
: リソースグループは PENDING_ONLINE で、このノード上での起動が完了しています。ただし、1 つまたは複数のリソースが Start_failed 状態または Faulted 状態で終了しています。

PENDING_ONLINE_BLOCKED
: リソースグループは、完全な起動を行うことに失敗しました。これは、リソースグループの 1 つまたは複数のリソースが、ほかのリソースグループのリソースに対して強いリソース依存性があり、それが満たされていないためです。このようなリソースは OFFLINE のままになります。リソースの依存関係が満たされている場合、リソースグループは自動的に PENDING_ONLINE 状態に戻ります。

カテゴリ: 照会のみ

デフォルト: デフォルトなし

調整: NONE

RG_system (boolean)
リソースグループの RG_system プロパティーの値が 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。このプロパティーの影響を受けるのは scrgadm コマンドと scswitch コマンドのみです。scha_control(1HA) および scha_control(3HA) に関する操作は影響を受けません。

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 プロパティーを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。

操作	サンプル
リソースグループを削除する	**scrgadm -r -g RG1**
リソースグループプロパティーを編集する (RG_system を除く)	**scrgadm -c -t RG1 -y nodelist=...**
リソースグループへソースを追加する	**scrgadm -a -j R1 -g RG1**
リソースグループからリソースを削除する	**scrgadm -r -j R1 -g RG1**
リソースグループに属するリソースのプロパティーを編集する	**scrgadm -c -j R1**
リソースグループをオフラインに切り替える	**scswitch -F -g RG1**
リソースグループを管理する	**scswitch -o -g RG1**
リソースグループを管理しない	**scswitch -u -g RG1**
リソースを使用可能にする	**scswitch -e -j R1**
リソースの監視を有効にする	**scswitch -e -M -j R1**
リソースを使用不可にする	**scswitch -n -j R1**
リソースの監視を無効にする	**scswitch -n -M -j R1**

リソースグループの RG_system プロパティーの値が TRUE の場合、そのリソースグループで編集可能な唯一のプロパティーは RG_system プロパティー自身です。つまり、RG_system プロパティーの編集は無制限です。

カテゴリ: 任意

デフォルト: FALSE

調整: ANYTIME

リソースプロパティーの属性

この節では、システム定義プロパティーの変更または拡張プロパティーの作成に使用できるリソースプロパティーの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。

以下にプロパティー名とその説明を示します。

`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`
  プロパティーのデフォルト値を示します。

`Description`
  プロパティー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ーに対する Description 属性を設定することはできません。

`Enumlist`
  enum タイプの場合、プロパティー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ーが 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。拡張プロパティー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ーです。

`Max`
  int タイプの場合、プロパティー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長さの最大値。

`Min`
  int タイプの場合、プロパティー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長さの最小値。

`Property`
  リソースプロパティーの名前。

`Tunable`
  クラスタ管理者がリソースのプロパティー値をいつ設定できるかを示します。クラスタ管理者にプロパティーの設定を許可しない場合は、NONE または FALSE に設定

します。クラスタ管理者にプロパティーの調整を許可する値には、TRUE または ANYTIME (任意の時点)、AT_CREATION (リソースの作成時のみ)、または WHEN_DISABLED (リソースが無効のとき) があります。「いつ監視を無効にするか」や「いつオフラインにするか」などのそのほかの条件を確立するには、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、次のエントリに示すように、標準リソースプロパティーごとに異なります。RTR ファイルで特に指定していない限り、拡張プロパティーを調整する設定のデフォルトは TRUE (ANYTIME) です。

プロパティーの型
: 指定可能な型は、string、boolean、integer、enum、stringarray です。RTR ファイル内で、システム定義プロパティー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ー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、リソースグループマネージャー (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
- プロパティー名
- 列挙型リテラル名

命名規則 (リソースタイプ名を除く)

リソースタイプ名を除き、すべての名前は次の規則に従う必要があります。

- 名前は ASCII である。
- 名前の先頭は文字である。
- 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 (_)。
- 名前に使用できる最大文字数は 255 である。

リソースタイプ名の形式

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*:*rt-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 *rt-version* を分離します。

この書式における変数要素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ーの値です。リソースタイプを開発する場合、会社の略号など、ベンダーを一意に識別するベンダー ID 接頭辞を選択します。たとえば、Sun Microsystems, Inc. により開発されるリソースタイプのベンダー ID 接頭辞は `SUNW` 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ーの値です。

rt-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ーの値です。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 製品のリリース 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` 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。

詳細については、125 ページの「リソースタイププロパティー」を参照してください。

例 **B–1** リソースタイプの完全な名前 (`#$upgrade` ディレクティブが指定されている場合)

この例では、RTR ファイルで次のようなプロパティー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`

RTR ファイルによって定義される完全なリソースタイプ名は次のようになります。

例 **B–1** リソースタイプの完全な名前 (#$upgrade ディレクティブが指定されている場合)　(続き)

```
SUNW.sample:2.0
```

例 **B–2** リソースタイプの完全な名前 (#$upgrade ディレクティブが指定されていない場合)

この例では、RTR ファイルで次のようなプロパティー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`

RTR ファイルによって定義される完全なリソースタイプ名は次のようになります。

```
SUNW.nfs
```

RGM の値

RGM の値は、プロパティー値と記述値という 2 つのカテゴリに分類されます。 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。
- 値に次の文字を含むことはできない。
  - 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。クラスタ構成のそのほかのコンポーネントのワークシートについては、『Sun Cluster ソフトウェアのインストール (Solaris OS 版)』の付録 A「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。

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。

構成のワークシート

この付録には次のワークシートが収録されています。

- 166 ページの「リソースタイプのワークシート」
- 168 ページの「ネットワークリソースのワークシート」
- 170 ページの「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」
- 172 ページの「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」
- 174 ページの「リソースグループ — フェイルオーバーのワークシート」
- 176 ページの「リソースグループ — スケーラブルのワークシート」

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。

リソースタイプ名	リソースタイプが動作するノード

例 **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
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

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 I 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		
拡張プロパティー	名称	値

例 **C–2** 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**	
拡張プロパティー	名称	値
	netiflist	**ipmp0@1, ipmp0@2**

例 **C–3** 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**	
拡張プロパティー	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

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		
拡張プロパティー	名称	値

例 **C–4** 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**	
拡張プロパティー	名称	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

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		
拡張プロパティー	名称	値

例 **C–5** 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		
拡張プロパティー	名称	値

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	記入欄	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 **C–6** 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	記入欄	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

コンポーネント	記入欄	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 **C–7** 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	記入欄	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。	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

数字・記号

#$upgrade 指令，162

A

Affinity_timeout, リソースプロパティー，134
API_version, リソースタイププロパティー，126
Array_maxsize, リソースプロパティー属性，158
Array_minsize, リソースプロパティー属性，158
Auto_start_on_new_cluster, リソースグループプロパティー，149
auxnodelist, ノードリストプロパティー，23

B

Boot, リソースタイププロパティー，127
Boot_timeout, リソースプロパティー，134

C

Cheap_probe_interval, リソースプロパティー，134
CheckNameService 拡張プロパティー，71
colocation, オンラインリソースグループに対する強制，107-108
continue_to_offload 拡張プロパティー，116

D

Default, リソースプロパティー属性，158
Description, リソースプロパティー属性，158
Desired_primaries, リソースグループプロパティー，149

E

Enumlist, リソースプロパティー属性，158
/etc/vfstab ファイル
  エントリの削除，101
  エントリの追加，97
Extension, リソースプロパティー属性，158

F

Failback, リソースグループプロパティー，150
Failover, リソースタイププロパティー，127
Failover_mode, リソースプロパティー，134
Failover_mode システムプロパティー，124
Fini, リソースタイププロパティー，128
Fini_timeout, リソースプロパティー，137

G

Global_resources_used, リソースグループプロパティー, 150

H

HAStoragePlus リソースタイプ
アップグレード, 103-104
インスタンスの変更, 96-103
インスタンスの変更の失敗, 102-103
概要, 19-21
使用基準, 20-21
注意事項, 100
対HAStorage リソースタイプ, 21
リソースタイプのバージョン, 104
HAStorage リソースタイプ
概要, 19-21
使用基準, 20-21
対HAStoragePlus リソースタイプ, 21

I

Implicit_network_dependencies, リソースグループプロパティー, 150
Init, リソースタイププロパティー, 128
Init_nodes, リソースタイププロパティー, 128
Init_timeout, リソースプロパティー, 137
installed_nodes, ノードリストプロパティー, 22
Installed_nodes, リソースタイププロパティー, 128
IP (インターネットプロトコル) アドレス, 制限, 22
Is_logical_hostname, リソースタイププロパティー, 128
Is_shared_address, リソースタイププロパティー, 129

L

Load_balancing_policy, リソースプロパティー, 138
Load_balancing_weights, リソースプロパティー, 138

M

Max, リソースプロパティー属性, 158
max_offload_retry 拡張プロパティー, 116
Maximum_primaries, リソースグループプロパティー, 150
Maxlength, リソースプロパティー属性, 158
Min, リソースプロパティー属性, 158
Minlength, リソースプロパティー属性, 158
Monitor_check, リソースタイププロパティー, 129
Monitor_check_timeout, リソースプロパティー, 138
Monitor_start, リソースタイププロパティー, 129
Monitor_start_timeout, リソースプロパティー, 139
Monitor_stop, リソースタイププロパティー, 129
Monitor_stop_timeout, リソースプロパティー, 139
Monitored_switch, リソースプロパティー, 139

N

Network_resources_used, リソースプロパティー, 139
nodelist, ノードリストプロパティー, 23
Nodelist, リソースグループプロパティー, 151
Nodelist リソースグループプロパティー, とアフィニティー, 106
nsswitch.conf, ファイルの内容の確認, 17
Num_resource_restarts, リソースプロパティー, 140
Num_rg_restarts, リソースプロパティー, 140

O

On_off_switch, リソースプロパティー, 140

P

Pathprefix, リソースグループプロパティー, 151
Pingpong_interval, リソースグループプロパティー, 151
ping コマンド, 無効にしたリソースからの応答, 64
Pkglist, リソースタイププロパティー, 129
Port_list, リソースプロパティー, 141
Postnet_stop, リソースタイププロパティー, 129
Postnet_stop_timeout, リソースプロパティー, 141
Prenet_start, リソースタイププロパティー, 130
Prenet_start_timeout, リソースプロパティー, 141
Probe_timeout 拡張プロパティー
  再起動時間への影響, 123
  調整, 122
Property, リソースプロパティー属性, 158
prtconf -v コマンド, 13
prtdiag -v コマンド, 13
psrinfo -v コマンド, 13

R

R_description, リソースプロパティー, 142
Resource_dependencies, リソースプロパティー, 142
Resource_dependencies_restart, リソースプロパティー, 142
Resource_dependencies_weak, リソースプロパティー, 143
Resource_list
  リソースグループプロパティー, 152
  リソースタイププロパティー, 130
Resource_name, リソースプロパティー, 144
Resource_project_name, リソースプロパティー, 144
Resource_state, リソースプロパティー, 144
Resource_type, リソースタイププロパティー, 130
resources, 構成データの取得、複製、またはアップグレード, 118
Retry_count, リソースプロパティー, 144
Retry_count システムプロパティー, 123
Retry_interval, リソースプロパティー, 145
Retry_interval システムプロパティー, 123
RG_affinities, リソースグループプロパティー, 152
RG_affinities リソースグループプロパティー, 105-107
RG_dependencies, リソースグループプロパティー, 152
RG_description, リソースグループプロパティー, 153
RG_is_frozen, リソースグループプロパティー, 153
RG_mode, リソースグループプロパティー, 153
RG_name, リソースグループプロパティー, 154
RG_project_name, リソースグループプロパティー, 154
RG_state, リソースグループプロパティー, 154
RG_system, リソースグループプロパティー, 156
rg_to_offload 拡張プロパティー, 117
RGM (Resource Group Manager)
  値, 163
  有効な名前, 161
RGOffload リソースタイプ
  拡張プロパティー, 116-117
  構成, 114-116
  障害モニター, 117
RT_basedir, リソースタイププロパティー, 131
RT_description, リソースタイププロパティー, 131
RT_system, リソースタイププロパティー, 131
RT_version, リソースタイププロパティー, 131
RTR (リソースタイプ登録) ファイル, 104

S

Scalable, リソースプロパティー, 145
scinstall -pv コマンド, 13
scrgadm コマンド, 26-27
scsetup ユーティリティー, 26
scsnapshot ユーティリティー, 118
Service Management Facility (SMF), 18
showrev -p コマンド, 13

Single_instance, リソースタイププロパティー, 132
SMF (Service Management Facility), 18
Start, リソースタイププロパティー, 132
Start_timeout, リソースプロパティー, 146
Status, リソースプロパティー, 146
Status_msg, リソースプロパティー, 146
Stop, リソースタイププロパティー, 132
STOP_FAILED エラーフラグ, 72-73
Stop_timeout, リソースプロパティー, 147
Sun Management Center GUI, 26
SunPlex Manager GUI, 26
Sun StorEdge QFS ファイルシステム, 94
SUNW.LogicalHostname リソースタイプ
  アップグレード, 74-75
  誤って削除したあとの再登録, 75-76
  リソースタイプバージョン, 74
SUNW.SharedAddress リソースタイプ
  アップグレード, 74-75
  誤って削除したあとの再登録, 75-76
  リソースタイプバージョン, 74

T

Thorough_probe_interval, リソースプロパティー, 147
Thorough_probe_interval システムプロパティー
  再起動時間への影響, 123
  調整, 121
Tunable, リソースプロパティー属性, 158
Type, リソースプロパティー, 147
Type_version, リソースプロパティー, 147
Type_version プロパティー, 75, 104

U

UDP_affinity, リソースプロパティー, 148
Update, リソースタイププロパティー, 132
Update_timeout, リソースプロパティー, 148

V

Validate, リソースタイププロパティー, 132
Validate_timeout, リソースプロパティー, 148
Vendor_ID, リソースタイププロパティー, 133
vfstab ファイル
  エントリの追加, 97
  エントリの追加、削除, 101

W

Weak_affinity, リソースプロパティー, 148

あ

値, RGM (Resource Group Manager), 163
新しいリソースタイプバージョンへの移行, 36-40
アップグレード
  HAStoragePlus リソースタイプ, 103-104
  構成データ, 119
  事前登録されているリソースタイプ, 74-75
  リソースタイプ, 35-36
アフィニティー, リソースグループ, 105-107
アプリケーションバイナリ, 格納先の決定, 16-17
アンマウント, ファイルシステム, 99

い

移行
  HAStoragePlus リソース, 104
  共有アドレスリソース, 75
  論理ホスト名リソース, 75
委託, 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, 111-112
インストール, 概要, 23-25
インターネットプロトコル (IP) アドレス, 制限, 22

え
エラーフラグ, STOP_FAILED, 72-73
エラーメッセージ, ファイルシステムの変更の失敗, 102

お
オンラインにする, リソースグループ, 54-56

か
回復, ファイルシステムの変更の失敗から, 102-103
拡張, リソースプロパティー, 134
拡張プロパティー
  Probe_timeout
    再起動時間への影響, 123
    調整, 122
  RGOffload リソースタイプ, 116-117
確認
  HAStoragePlus リソースからのファイルシステムの削除, 100
  HAStoragePlus リソースへのファイルシステムの追加, 98
  nsswitch.conf ファイルの内容, 17
型, リソースプロパティーの属性, 159
間隔, 障害モニター検証, 121
完全な障害, 122-123

き
記述値, 規則, 163
規則
  記述値, 163
  プロパティー値, 163
  プロパティー名, 161
  リソースグループ名, 161
  リソース名, 161
  列挙型リテラル名, 161
起動の同期, リソースグループとディスクデバイスグループ, 86-89
共有アドレスリソース
  変更, 71-72
  無効にしたときにホストから分離, 64
  リソースグループへの追加, 48-50
均衡, クラスタノードの負荷, 109-110

く
組み合わせ, リソースグループ間のアフィニティー, 112-113

け
計画
  クラスタファイルシステム, 17
  データサービス, 15-27
継続的な障害, 定義, 122-123
現在の主ノードの切り替え, リソースグループ, 62-63

こ
高可用性ファイルシステム
  注意事項, 100
  ファイルシステムの削除, 99-102
  ファイルシステムの追加, 97-99
  変更, 96-103
  変更の失敗, 102-103
  有効化, 92-96
構成
  ガイドライン, 16-18
  概要, 23-25
  クラスタファイルシステムの計画, 17
構成と管理, Sun Cluster データサービス, 33
構文
  記述値, 163
  プロパティー値, 163
  プロパティー名, 161
  リソースグループ名, 161
  リソースタイプ名, 161
  リソース名, 161
  列挙型リテラル名, 161
考慮事項, 22
コマンド, ノード情報, 13

さ
再起動, 許可される最大, 122

再試行間隔，122
最大値, 再起動，122
削除
  HAStoragePlus リソースからのファイルシステム，99-102
  リソース，61
  リソースグループ，59-60
  リソースグループからのノード
    概要，80
    共有アドレスを使用したフェイルオーバー，84-85
    スケーラブル，81-82
    フェイルオーバー，82-84
  リソースタイプ，58-59
作成
  共有アドレスリソース，48-50
  スケーラブルアプリケーションリソース，52-54
  フェイルオーバーアプリケーションリソース，50-52
  リソースグループ
    スケーラブル，44-45
    フェイルオーバー，43-44
  論理ホスト名リソース，46-48

し

システムプロパティー
  「プロパティー」も参照
  「拡張プロパティー」も参照
  Failover_mode，124
  Retry_count，123
  Retry_interval，123
  Thorough_probe_interval
    再起動時間への影響，123
    調整，121
  障害モニターへの影響，121
事前登録されたリソースタイプ, 誤って削除したあとの再登録，75-76
事前登録されたリソースタイプの再登録，75-76
事前登録されているリソースタイプ, アップグレード，74-75
重要でないサービス, オフロード，110-111
重要でないリソースグループのオフロード
  RGOffload リソースタイプ，113-117
  アフィニティー，110-111
重要なサービス，110-111
取得, リソースグループ、リソースタイプ、およびリソースについての構成データ，119
障害
  継続的な，122-123
  ファイルシステムの変更，102-103
  への対応，123-124
障害追跡, ファイルシステムの変更，102-103
障害モニター
  RGOffload リソースタイプ，117
  検証間隔，121
  検証タイムアウト，122
  障害への対応，123-124
  調整，120-124
  による障害の検出，123-124
  無効化，56-57
  有効化，57
消去, STOP_FAILED エラーフラグ，72-73
書式, リソースタイプ名，161
指令, #$upgrade，162

す

スイッチオーバー, リソースグループの委託，111-112
スケーラブルアプリケーションリソース, リソースグループへの追加，52-54

せ

制限，22
性能
  重要なサービス用に最適化，110-111
  への検証間隔の影響，121
設定
  HAStoragePlus リソースタイプ，92-96
  HAStorage リソースタイプ
    新しいリソース，86-88
    既存のリソース，88-89
  RGOffload リソースタイプ，113-117

そ

属性, リソースプロパティー，158

た

対応, 障害への，123-124
タイムアウト
　障害モニター
　　設定の指針，122
ダウングレード, リソースタイプ，41-42

ち

注意事項, ファイルシステムの削除，100
調整, 障害モニター，120-124

つ

追加
　HAStoragePlus リソースへのファイルシステム，97-99
　リソースグループへのノード
　　概要，77
　　スケーラブル，77-78
　　フェイルオーバー，78-80
　リソースグループへのリソース
　　概要，46-54
　　共有アドレス，48-50
　　スケーラブルアプリケーション，52-54
　　フェイルオーバーアプリケーション，50-52
　　論理ホスト名，46-48
ツール
　scrgadm コマンド，26-27
　scsetup ユーティリティー，26
　Sun Management Center GUI，26
　SunPlex Manager GUI，26
強い肯定的なアフィニティー
　使用例，107-108
　定義，106
強い否定的なアフィニティー
　使用例，110-111
　定義，106

て

定義, 継続的な障害，122-123
ディスクデバイスグループ
　リソースグループとの関係，18-19
ディスクデバイスグループ (続き)
　リソースグループとの起動の同期，86-89
データサービス
　計画，15-27
　考慮事項，22
　特殊な要件，16

と

登録
　HAStoragePlus リソースタイプ
　　アップグレード中，104
　SUNW.LogicalHostname リソースタイプ
　　アップグレード中，74-75
　　誤って削除したあと，75-76
　SUNW.SharedAddress リソースタイプ
　　アップグレード中，74-75
　　誤って削除したあと，75-76
　事前登録されたリソースタイプ，75-76
　リソースタイプ，33-34
登録解除, リソースタイプ，58-59
特殊な要件, 確認，16

ね

ネームサービス, バイパス，71-72
ネットワーク, 制限，22

の

ノード
　重要でないサービスのオフロード，110-111
　負荷均衡，109-110
　リソースグループからの削除
　　概要，80
　　共有アドレスを使用したフェイルオーバー，84-85
　　スケーラブル，81-82
　　フェイルオーバー，82-84
　リソースグループの分散，104-113
　リソースグループへの追加
　　概要，77
　　スケーラブル，77-78
　　フェイルオーバー，78-80
ノードリストプロパティー，22-23

ノードリソースの解放
  RGOffload リソースタイプ，113-117
  アフィニティー，110-111

は

バージョン
  HAStoragePlus リソースタイプ，104
  SUNW.LogicalHostname リソースタイプ，74
  SUNW.SharedAddress リソースタイプ，74
配置, オンラインリソースグループに対する優先，108-109
バイパス, ネームサービス，71-72

ひ

表示, リソースタイプ、リソースグループ、リソース構成，66-67

ふ

ファイル
  /etc/vfstab
    エントリの削除，101
    エントリの追加，97
  RTR，104
ファイルシステム
  HAStoragePlus リソースからの削除，99-102
  HAStoragePlus リソースへの追加，97-99
  アンマウント，99
  高可用性
    変更，96-103
    有効化，92-96
  注意事項，100
  変更の失敗，102-103
  マウント，97
フェイルオーバー
  オンラインリソースグループの分散の維持，104-113
  リソースグループの委託，111-112
フェイルオーバーアプリケーションリソース, リソースグループへの追加，50-52
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ー
  使用例，111-112
  定義，106
負荷均衡，109-110
複製, 構成データ，118
部分的な障害，122-123
プロパティー
  「拡張プロパティー」も参照
  Type_version，75, 104
  リソース，133
  リソースグループ，149
  リソースタイプ，125
プロパティー属性, リソース，158
プロパティー値, 規則，163
プロパティー名, 規則，161
分散, オンラインリソースグループ，104-113

へ

変更
  共有アドレスリソース，71-72
  リソースグループプロパティー，69
  リソースタイププロパティー，67-69
  リソースプロパティー，70-71
  論理ホスト名リソース，71-72
編集
  HAStoragePlus リソース，104
  共有アドレスリソース，75
  論理ホスト名リソース，75

ほ

ボリュームマネージャー, 高可用性ファイルシステム，93

ま

マウント, ファイルシステム，97

む

無効化
  SMF インスタンス，18

無効化 (続き)
リソース，64-66
リソース障害モニター，56-57
無効にしたリソース, 予期せぬ動作，64

ゆ
有効化, リソース障害モニター，57
有効な名前, RGM (Resource Group Manager)，161

よ
要件, データサービス，16
弱い肯定的なアフィニティー
使用例，108-109
定義，106
弱い否定的なアフィニティー
使用例，109-110
定義，106

り
リソース
STOP_FAILED エラーフラグの消去，72-73
共有アドレス
変更，71-72
無効にしたときにホストから分離，64
リソースグループへの追加，48-50
構成情報の表示，66-67
削除，61
障害モニターの無効化，56-57
障害モニターの有効化，57
スケーラブルアプリケーション
リソースグループへの追加，52-54
フェイルオーバーアプリケーション
リソースグループへの追加，50-52
プロパティーの変更，70-71
無効化，64-66
リソースグループへの追加，46-54
リソースタイプの削除，58-59
論理ホスト名
変更，71-72
リソースグループへの追加，46-48
リソースグループ
UNMANAGED 状態への移行，64-66
アフィニティー，105-107
オンラインにする，54-56
強制的に同じ場所に配置，107-108
強制的に分離，110-111
共有アドレスを使用したフェイルオーバー
ノードの削除，84-85
均等分配，109-110
現在の主ノードの切り替え，62-63
構成情報の表示，66-67
構成データの取得、複製、またはアップグレード，118
削除，59-60
作成
スケーラブル，44-45
フェイルオーバー，43-44
スケーラブル
ノードの削除，81-82
ノードの追加，77-78
ディスクデバイスグループとの関係，18-19
ディスクデバイスグループとの起動の同期，86-89
できる限り同じ場所に配置，108-109
できる限り分離，109-110
ノード間で分散，104-113
ノードの削除，80
ノードの追加，77
フェイルオーバー
ノードの削除，82-84
ノードの追加，78-80
フェイルオーバーまたはスイッチオーバーの委託，111-112
プロパティーの変更，69
リソースの追加，46-54
共有アドレス，48-50
スケーラブルアプリケーション，52-54
フェイルオーバーアプリケーション，50-52
論理ホスト名，46-48
リソースグループプロパティー，149
Auto_start_on_new_cluster，149
Desired_primaries，149
Failback，150
Global_resources_used，150
Implicit_network_dependencies，150
Maximum_primaries，150
Nodelist，151

リソースグループプロパティー (続き)
Pathprefix, 151
Pingpong_interval, 151
Resource_list, 152
RG_affinities, 152
RG_dependencies, 152
RG_description, 153
RG_is_frozen, 153
RG_mode, 153
RG_name, 154
RG_project_name, 154
RG_state, 154
RG_system, 156
リソースグループ名, 規則, 161
リソース障害モニター, 56-57
リソースタイプ
HAStorage
新しいリソース, 86-88
既存のリソース, 88-89
HAStoragePlus
インスタンスの移行, 104
LogicalHostname
インスタンスの移行, 75
RGOffload, 113-117
SharedAddress
インスタンスの移行, 75
新しいリソースタイプバージョンへの移行, 36-40
アップグレード, 35-36
構成情報の表示, 66-67
構成データの取得、複製、またはアップグレード, 118
削除, 58-59
事前登録された
誤って削除したあとの再登録, 75-76
事前登録されている
アップグレード, 74-75
ダウングレード, 41-42
登録, 33-34
登録解除, 58-59
プロパティーの変更, 67-69
リソースタイプ登録 (RTR) ファイル, 104
リソースタイププロパティー, 125
API_version, 126
Boot, 127
Failover, 127
Fini, 128
Init, 128
リソースタイププロパティー (続き)
Init_nodes, 128
Installed_nodes, 128
Is_logical_hostname, 128
Is_shared_address, 129
Monitor_check, 129
Monitor_start, 129
Monitor_stop, 129
Pkglist, 129
Postnet_stop, 129
Prenet_start, 130
Resource_list, 130
Resource_type, 130
RT_basedir, 131
RT_description, 131
RT_system, 131
RT_version, 131
Single_instance, 132
Start, 132
Stop, 132
Update, 132
Validate, 132
Vendor_ID, 133
リソースタイプ名, 規則, 161
リソースプロパティー, 133
Affinity_timeout, 134
Boot_timeout, 134
Cheap_probe_interval, 134
Failover_mode, 134
Fini_timeout, 137
Init_timeout, 137
Load_balancing_policy, 138
Load_balancing_weights, 138
Monitor_check_timeout, 138
Monitor_start_timeout, 139
Monitor_stop_timeout, 139
Monitored_switch, 139
Network_resources_used, 139
Num_resource_restarts, 140
Num_rg_restarts, 140
On_off_switch, 140
Port_list, 141
Postnet_stop_timeout, 141
Prenet_start_timeout, 141
R_description, 142
Resource_dependencies, 142
Resource_dependencies_restart, 142
Resource_dependencies_weak, 143

リソースプロパティー (続き)
Resource_name, 144
Resource_project_name, 144
Resource_state, 144
Retry_count, 144
Retry_interval, 145
Scalable, 145
Start_timeout, 146
Status, 146
Status_msg, 146
Stop_timeout, 147
Thorough_probe_interval, 147
Type, 147
Type_version, 147
UDP_affinity, 148
Update_timeout, 148
Validate_timeout, 148
Weak_affinity, 148
拡張, 134
リソースプロパティーの属性, 158
Array_maxsize, 158
Array_minsize, 158
Default, 158
Description, 158
Enumlist, 158
Extension, 158
Max, 158
Maxlength, 158
Min, 158
Minlength, 158
Property, 158
Tunable, 158
型, 159
リソース名, 規則, 161

れ

列挙型リテラル名, 規則, 161

ろ

論理ホスト名リソース
変更, 71-72
リソースグループへの追加, 46-48