



# 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: 820-3811-10  
2008 年 2 月, Revision A

Sun Microsystems, Inc. は、本文書に記載されている製品によって具現化されたテクノロジーに関する知的財産権を保有しています。特に、この知的財産権はひとつかそれ以上の米国における特許、あるいは米国およびその他の国において申請中の特許を含んでいることがありますが、それらに限定されるものではありません。

U.S. Government Rights – Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

この配布には、第三者が開発したソフトウェアが含まれている可能性があります。

本製品の一部は、カリフォルニア大学からライセンスされている Berkeley BSD システムに基づいていることがあります。UNIX は、X/Open Company, Ltd. が独占的にライセンスしている米国ならびに他の国における登録商標です。

Sun、Sun Microsystems、Sun のロゴマーク、Solaris のロゴマーク、Java Coffee Cup のロゴマーク、docs.sun.com、Java、および Solaris は、米国およびその他の国における米国 Sun Microsystems, Inc. (以下、米国 Sun Microsystems 社とします) の商標もしくは、登録商標です。すべての SPARC 商標は、米国 SPARC International, Inc. のライセンスを受けて使用している同社の米国およびその他の国における商標または登録商標です。SPARC 商標が付いた製品は、米国 Sun Microsystems 社が開発したアーキテクチャに基づくものです。

OPEN LOOK および Sun Graphical User Interface は、米国 Sun Microsystems 社が自社のユーザおよびライセンス実施権者向けに開発しました。米国 Sun Microsystems 社は、コンピュータ産業用のビジュアルまたはグラフィカル・ユーザインタフェースの概念の研究開発における米国 Xerox 社の先駆者としての成果を認めるものです。米国 Sun Microsystems 社は米国 Xerox 社から Xerox Graphical User Interface の非独占的ライセンスを取得しており、このライセンスは、OPEN LOOK のグラフィカル・ユーザインタフェースを実装するか、またはその他の方法で米国 Sun Microsystems 社との書面によるライセンス契約を遵守する、米国 Sun Microsystems 社のライセンス実施権者にも適用されます。

本書で言及されている製品や含まれている情報は、米国輸出規制法で規制されるものであり、その他の国の輸出入に関する法律の対象となることがあります。核、ミサイル、化学あるいは生物兵器、原子力の海洋輸送手段への使用は、直接および間接を問わず厳しく禁止されています。米国が禁輸の対象としている国や、限定はされませんが、取引禁止顧客や特別指定国民のリストを含む米国輸出排除リストで指定されているものへの輸出および再輸出は厳しく禁止されています。

本書は、「現状のまま」をベースとして提供され、商品性、特定目的への適合性または第三者の権利の非侵害の黙示の保証を含みそれに限定されない、明示的であるか黙示的であるかを問わない、なんらの保証も行われないものとします。

# 目次

---

はじめに .....	9
<b>1 Sun Cluster データサービスの計画 .....</b>	<b>15</b>
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バイスグループの関係 .....	19
HASStoragePlus の理解 .....	20
データサービスに HASStoragePlus が必要であるかどうかの判別 .....	20
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 .....	21
ノードリストプロパティ .....	22
installed_nodes プロパティ .....	22
nodelist プロパティ .....	22
auxnodelist プロパティ .....	22
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 .....	23
フェイルオーバーデータサービスの構成例 .....	24
データサービスリソースを管理するためのツール .....	25
Sun Cluster Manager グラフィカルユーザーインタフェース (GUI) .....	25
SPARC: Sun Management Center GUI 用の Sun Cluster モジュール .....	26
clsetup ユーティリティ .....	26
Sun Cluster 保守コマンド .....	27
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 .....	34
	リソースタイプの登録 .....	34
	▼リソースタイプを登録する .....	34
	リソースタイプの更新 .....	35
	▼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 .....	36
	▼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 .....	37
	リソースタイプのダウングレード .....	42
	▼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 .....	42
	リソースグループの作成 .....	43
	▼フェイルオーバーリソースグループを作成する .....	44
	▼スケラブルリソースグループを作成する .....	46
	リソースをリソースグループに追加するためのツール .....	48
	▼clsetup ユーティリティを使用して論理ホスト名リソースをリソースグループ に追加する .....	49
	▼コマンド行インタフェースを使用して論理ホスト名リソースをリソースグループ に追加する .....	52
	▼clsetup ユーティリティを使用して共有アドレスリソースをリソースグループ に追加する .....	55
	▼コマンド行インタフェースを使用して共有アドレスリソースをリソースグループ に追加する .....	57
	▼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 ...	60
	▼スケ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 .....	62
	リソースグループをオンラインにする .....	65
	▼リソースグループをオンラインにする .....	66
	リソースの有効化 .....	67
	▼リソースを有効にする .....	67
	リソースグループの休止 .....	68
	▼リソースグループを休止する .....	69
	▼ただちにリソースグループを休止する .....	69
	リソースグループの自動回復アクションの保存停止と再開 .....	69
	メソッドを終了することによる自動回復の即時保存停止 .....	70
	▼リソースグループの自動回復アクションを保存停止する .....	71
	▼リソースグループの自動回復アクションをただちに保存停止する .....	71
	▼リソースグループの自動回復アクションを再開する .....	72
	リソース 모니터の無効化と有効化 .....	72

▼リソース障害モニターを無効にする .....	72
▼リソース障害モニターを有効にする .....	73
リソースタイプの削除 .....	74
▼リソースタイプを削除する .....	74
リソースグループの削除 .....	75
▼リソースグループを削除する .....	76
リソースの削除 .....	77
▼リソースを削除する .....	77
リソースグループの主ノードの切り替え .....	78
▼リソースグループの主ノードを切り替える .....	78
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 .....	80
▼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 .....	80
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 .....	82
リソースタイプ、リソースグループ、リソースプロパティの変更 .....	83
▼リソースタイププロパティを変更する .....	83
▼リソースグループプロパティを変更する .....	85
▼リソースプロパティを変更する .....	86
▼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 .....	87
リソースの STOP_FAILED エラーフラグの消去 .....	88
▼リソースの STOP_FAILED エラーフラグを消去する .....	89
Start_failed リソース状態の消去 .....	90
▼リソースグループのスイッチオーバーにより Start_failed リソース状態を解除する .....	91
▼リソースグループの再起動により Start_failed リソース状態を解除する .....	93
▼リソースの無効化および有効化によりリソース状態 Start_failed を解除する ..	95
事前登録されているリソースタイプのアップグレード .....	96
新しいリソースタイプバージョンの登録に関する情報 .....	97
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 .....	97
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 .....	98
▼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 .....	98
リソースグループへのノードの追加と削除 .....	99
リソースグループにノードを追加する .....	99
▼スケラブルリソースグループにノードを追加する .....	100
▼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 .....	101
リソースグループからノードを削除する .....	103
▼スケラブルリソースグループからノードを削除する .....	104

▼ 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 .....	105
▼ 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 .....	107
大域ゾーンから非大域ゾーンへのアプリケーションの移行 .....	109
▼ 大域ゾーンから非大域ゾーンへアプリケーションを移行する .....	109
リソースグループとデバイスグループ間での起動の同期 .....	112
▼ 新しいリソース用に HASStoragePlus リソースタイプを設定する .....	113
▼ 既存のリソース用に HASStoragePlus リソースタイプを設定する .....	115
クラスタファイルシステム用の HASStoragePlus リソースの構成 .....	116
クラスタファイルシステム用の /etc/vfstab のサンプルエントリ .....	116
▼ クラスタファイルシステム用に HASStoragePlus リソースを設定する .....	117
▼ クラスタファイルシステム用の HASStoragePlus リソースタイプを削除する .....	117
高可用性ローカルファイルシステムの有効化 .....	118
高可用性ローカルファイルシステムの構成要件 .....	119
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 .....	120
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 .....	120
▼ clsetup ユーティリティを使用することで HASStoragePlus リソースタイプを設定する .....	121
▼ HASStoragePlus リソースタイプを設定し、ローカル Solaris ZFS を高可用性にする .....	125
▼ ローカル Solaris ZFS を高可用性にしている HASStoragePlus リソースを削除する .....	127
HASStorage から HASStoragePlus へのアップグレード .....	127
▼ デバイスグループまたは CFS を使用している場合に HASStorage から HASStoragePlus へアップグレードする .....	128
▼ CFS による HASStorage から高可用性ローカルファイルシステムによる HASStoragePlus へアップグレードする .....	129
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変更する .....	130
▼ Solaris ZFS 以外のファイルシステムをオンラインの HASStoragePlus リソースに追加する .....	131
▼ オンラインの HASStoragePlus リソースから Solaris ZFS 以外のファイルシステムを削除する .....	133
▼ Solaris ZFS ストレージプールをオンラインの HASStoragePlus リソースに追加する .....	136
▼ オンラインの HASStoragePlus リソースから Solaris ZFS ストレージプールを削除する .....	137
▼ HASStoragePlus リソースの FileSystemMountPoints プロパティを変更したあと障害から回復する .....	138

▼ HASToragePlus リソースの Zpools プロパティを変更したあと障害から回復する .....	139
HASToragePlus リソースの広域ファイルシステムからローカルファイルシステムへの変更 .....	140
▼ HASToragePlus リソースの広域ファイルシステムをローカルファイルシステムに変更する .....	140
HASToragePlus リソースタイプのアップグレード .....	141
新しいリソースタイプバージョンの登録に関する情報 .....	141
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 .....	142
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 .....	142
リソースグループのアフィニティ .....	143
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 .....	145
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 .....	146
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 .....	148
重要なサービスに優先権を指定する .....	149
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 .....	150
リソースグループ間のアフィニティの組み合わせ .....	151
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 .....	152
▼ 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 .....	153
▼ 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 .....	154
Sun Cluster 上で Solaris SMF サービスを有効にする .....	155
▼ SMF サービスのフェイルオーバープロキシリソース構成へのカプセル化 .....	157
▼ SMF サービスのマルチマスタープロキシリソース構成へのカプセル化 .....	159
▼ SMF サービスのスケラブルプロキシリソース構成へのカプセル化 .....	162
Sun Cluster データサービス用に障害モニターを調整する .....	167
障害モニターの検証間隔の設定 .....	167
障害モニターの検証タイムアウトの設定 .....	168
継続的な障害とみなす基準の定義 .....	168
リソースのフェイルオーバー動作を指定する .....	170
<b>A Sun Cluster オブジェクト指向コマンド .....</b>	<b>171</b>
オブジェクト指向コマンド名および別名 .....	171

---

オブジェクト指向コマンドセットの概要 .....	172
<b>B</b> 標準プロパティ .....	183
資源タイプのプロパティ .....	183
リソースのプロパティ .....	194
リソースグループのプロパティ .....	216
リソースプロパティの属性 .....	232
<b>C</b> 有効な RGM 名と値 .....	235
有効な RGM 名 .....	235
命名規則(リソースタイプ名を除く) .....	235
リソースタイプ名の形式 .....	236
RGM の値 .....	237
<b>D</b> データサービス構成のワークシートと記入例 .....	239
設定ワークシート .....	239
リソースタイプのワークシート .....	240
ネットワークリソースのワークシート .....	242
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 .....	244
アプリケーションリソース—スケラブルのワークシート .....	246
リソースグループ—フェイルオーバーのワークシート .....	248
リソースグループ—スケラブルのワークシート .....	250
索引 .....	253

# 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 Sun Cluster リリースでは、SPARC™ および x86 系列のプロセッサアーキテクチャ (UltraSPARC、SPARC64、および AMD64) を使用するシステムをサポートします。このマニュアルでは、x86 という用語は AMD64 系列のプロセッサアーキテクチャを使用するシステムを意味します。

---

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 (Solaris OS) に関する知識と、Sun Cluster ソフトウェア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で動作します。このマニュアルで説明する情報は、章、節、注、箇条書き、図、表、例、またはコード例において特に明記しない限り、両方のプラットフォームに該当します。

---

## 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% <b>su</b>  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 <i>filename</i> 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。
\	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% <b>grep '^#define \</b>  <b>XV_VERSION_STRING'</b>

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。

## 一般規則

---

注-\*\*\* 下記の規則に該当する場合は、該当する規則のみ表記する。どちらも該当しない場合は、このセクションごと削除。\*\*\*

---

- このマニュアルでは、英語環境での画面イメージを使っています。このため、実際に日本語環境で表示される画面イメージとこのマニュアルで使っている画面イメージが異なる場合があります。本文中で画面イメージを説明する場合には、日本語のメニュー、ボタン名などの項目名と英語の項目名が、適宜併記されています。

## 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。

項目	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
Concept	『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)』

項目	マニュアル
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 3.1 - 3.2 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 <http://docs.sun.com> で参照してください。

## 第三者の関連する Web サイトの参照

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

---

注- このマニュアル内で引用する第三者の Web サイトの可用性について Sun は責任を負いません。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 Web サイトでは、次のサービスに関する情報も提供しています。

- マニュアル (<http://jp.sun.com/documentation/>)
- サポート (<http://jp.sun.com/support/>)
- トレーニング (<http://jp.sun.com/training/>)

## 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
- システムのモデル番号とシリアル番号
- Solaris オペレーティングシステムのバージョン番号 (例: Solaris 10)
- Sun Cluster のバージョン番号 (例: Sun Cluster 3.2)

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
<code>prtconf -v</code>	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
<code>psrinfo -v</code>	プロセッサの情報を表示する
<code>showrev -p</code>	インストールされているパッチを報告する
<code>prtdiag -v</code>	システム診断情報を表示する
<code>/usr/cluster/bin/clnode show-rev</code>	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

また、`/var/adm/messages` ファイルも用意してください。



# 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」
- 19 ページの「リソースグループとデバイスグループの関係」
- 20 ページの「HASStoragePlus の理解」
- 21 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」
- 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。詳細については、<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にであれば、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「グローバルデバイス、デバイスグループ、およびクラスタファイルシステムの計画」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、118 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「グローバルデバイス、デバイスグループ、およびクラスタファイルシステムの計画」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、118 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

SMF (Service Management Facility) を使用すると、ノードの起動中またはサービス障害中に自動的に SMF サービスを起動および再起動することができます。この機能は、クラスタアプリケーションに高可用性とスケーラビリティを実現する、Sun Cluster Resource Group Manager (RGM) に似ています。SMF サービスと RGM の機能は相互に補完的です。

Sun Cluster には、3つの新しい SMF プロキシリソースタイプが含まれています。これらを使用すると、フェイルオーバー、マルチマスター、またはスケーラブル構成の Sun Cluster とともに SMF サービスが実行できるようになります。SMF プロキシリソースタイプを使用すると、相互関係のある SMF サービスのセットを1つのリソースにカプセル化し、SMF プロキシリソースを Sun Cluster で管理することができます。この機能では、SMF は1つのノード上の SMF サービスの可用性を管理します。Sun Cluster は、SMF サービスの、クラスタ全体にわたる高い可用性とスケーラビリティを提供します。

これらのサービスのカプセル化の詳細については、[155 ページの「Sun Cluster 上で Solaris SMF サービスを有効にする」](#)を参照してください。

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な主ノード上で 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。

- 『System Administration Guide: Basic Administration』の「How to Disable a Service Instance」

- 『Sun Cluster Data Service for NFS Guide for Solaris OS』
- 『Sun Cluster Data Service for DNS Guide for Solaris OS』
- 『Sun Cluster の概念 (Solaris OS 版)』

## リソースグループとデバイスグループの関係

Sun Cluster は、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またはゾーンが、クラスタに再結合する。
- クラスタに再結合するノードまたはゾーンが、現在の主ノードまたはゾーン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、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。このようにすることで、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またはゾーンになります。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れば、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している時に、スケーラブルサービスが利用できることです。

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「デバイスグループ」を参照してください。

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「デバイスグループ」

## HASStoragePlus の理解

HASStoragePlus リソースタイプは、次のオプションの構成をするために使用できません。

- 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。HASStoragePlus 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、HASStoragePlus はローカルおよび広域ファイルシステムをマウントすることができます。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 - HASStoragePlus 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、[112 ページの「リソースグループと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。

VxFS や Solaris ZFS (Zettabyte File System) 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手順については、[118 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.HASStoragePlus(5)のマニュアルページを参照してください。

## データサービスに HASStoragePlus が必要であるかどうかの判別

次のタイプのデータサービスには HASStoragePlus 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。

- HAStoragePlus リソースをリソースグループ内で構成します。
- そのほかのデータサービスリソースの依存性を 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。

- HAStoragePlus リソースをデータサービスリソースグループに追加します。
- HAStoragePlus リソースをオンラインに切り替えます。
- HAStoragePlus リソースにデータサービスリソースの依存性を設定します。
- 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、HAStoragePlus リソースタイプの構成はオプションです。

# 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では、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、クラスタ内で多数のディスクを構成しているものです。入出力サブシステムが再試行し、障害から回復するまで、数分かかることもあります。この遅延によって、最終的にディスクが自分自身で回復したとしても、Sun Cluster がアプリケーションを別のノードにフェイルオーバーすることがあります。このような場合のフェイルオーバーを回避するには、データサービスのデフォルトの検証タイムアウト値を増やしてみてください。データサービスのタイムアウトについての詳細や、タイムアウト値を増やす方法については、ご購入先にお問い合わせください。
- よりよいパフォーマンスを保つために、ストレージに直結されたクラスタノードにデータサービスをインストールし、構成してください。

- クラスタノード上で動作するクライアントアプリケーションは、HA データサービスの論理 IP アドレスにマッピングしてはなりません。フェイルオーバー後、このような論理 IP アドレスは存在しなくなり、クライアントが切断されたままになる可能性があります。

## ノードリストプロパティ

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

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

### `installed_nodes` プロパティ

`installed_nodes` プロパティは、データサービスのリソースタイプのプロパティです。このプロパティには、リソースタイプがインストールされ、実行が有効になるクラスタノード名の一覧が含まれます。

### `nodelist` プロパティ

`nodelist` プロパティは、リソースグループのプロパティです。このプロパティは、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またはゾーン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な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

`auxnodelist` プロパティは、共有アドレスリソースのプロパティです。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、スケラブルサービスにのみ適用されます。詳細は、`clressharedaddress(1CL)`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 (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、Sun Cluster 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、Sun Cluster 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 D
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)』 or the appropriate Sun Cluster data services book
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、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は、『Sun Cluster Data Service for Oracle Guide for Solaris OS』を参照してください。

この例とスケ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。

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

Sun Cluster 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

Sun Cluster Manager では、次のアプリケーションに対して Sun Cluster のデータサービスの構成を自動化するためのウィザードを用意しています。

- Apache Web Server
- NFS
- Oracle
- Oracle Real Application Clusters
- SAP Web Application Server

各ウィザードでは、データサービスが必要とする Sun Cluster リソースを構成できません。このウィザードでは、Sun Cluster 構成で実行するためのアプリケーションソフトウェアのインストールと構成は自動化されません。Sun Cluster 構成で実行できるようにアプリケーションソフトウェアのインストールと構成を実行するには、アプリケーションのユーティリティおよび Sun Cluster の保守コマンドを使用してください。詳細については、アプリケーションのドキュメントおよび Sun Cluster のドキュメントセットを参照してください。各ウィザードでサポートされているデータサービスの構成オプションは制限されています。ウィザードでサポートされていないオプションを構成するには、Sun Cluster Manager または Sun Cluster の保守コマンドを使用して、データサービスの構成を手作業で実行します。詳細は、Sun Cluster のマニュアルを参照してください。

Sun Cluster Manager には、次の Sun Cluster リソースの構成を自動化するウィザードがあります。

- 論理ホスト名リソース
- 共有アドレスリソース
- 高可用性ストレージリソース

ウィザードを使用して作成したリソースは、データサービスの構成にかかわらず、どのデータサービスでも使用できます。

Sun Cluster 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、Sun Cluster 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。

## clsetup ユーティリティ

clsetup(1CL) ユーティリティは、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、clsetup のメインメニューからオプション 2 を選択して、「リソースグループ」というサブメニューを起動してください。

## Sun Cluster 保守コマンド

Sun Cluster の保守コマンドを使用すると、データサービスリソースを登録および構成することができます。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 Guide for Solaris OS』の「Registering and Configuring Sun Cluster HA for Oracle」の手順を参照してください。

これらの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については、[第2章](#)を参照してください。

## 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の管理に使用できるツール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	Sun Cluster Manager	SPARC:Sun Management Center	clsetup ユーティリティ
リソースタイプを登録する	+	-	+
リソースグループを作成	+	-	+
リソースグループへソースを追加する	+	-	+
リソースグループの自動回復アクションを保存停止する	+	-	+
リソースグループの自動回復アクションを再開する	+	-	+
リソースグループをオンラインにする	+	+	+
リソースグループを削除	+	+	+
リソースを削除する	+	+	+
リソースグループの現在の主ノードを切り替える	+	-	+
リソースを使用可能にする	+	+	+
リソースを使用不可にする	+	+	+

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 (続き)

作業	Sun Cluster Manager	SPARC:Sun Management Center	clsetup ユーティリティ
リソースグループを非管理状態に移行する	+	-	+
リソースタイプ、リソースグループ、リソース構成情報を表示する	+	+	+
リソースプロパティを変更する	+	-	+
リソースの STOP_FAILED エラーフラグを消去する	+	-	+
リソースの START_FAILED リソース状態を解除する	+	-	+
ノードをリソースグループに追加する	+	-	+

## データサービスリソースの管理

---

この章では、Sun Cluster の保守コマンドを使って、クラスタ内のリソース、リソースグループ、およびリソースタイプを管理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」
- 34 ページの「Sun Cluster データサービスの構成と管理」
- 34 ページの「リソースタイプの登録」
- 35 ページの「リソースタイプの更新」
- 42 ページの「リソースタイプのダウングレード」
- 43 ページの「リソースグループの作成」
- 48 ページの「リソースをリソースグループに追加するためのツール」
- 65 ページの「リソースグループをオンラインにする」
- 67 ページの「リソースの有効化」
- 68 ページの「リソースグループの休止」
- 69 ページの「リソースグループの自動回復アクションの保存停止と再開」
- 72 ページの「リソースモニターの有効化と無効化」
- 74 ページの「リソースタイプの削除」
- 75 ページの「リソースグループの削除」
- 77 ページの「リソースの削除」
- 78 ページの「リソースグループの主ノードの切り替え」
- 80 ページの「リソースの有効化とリソースグループの UNMANAGED 状態への移行」
- 82 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」
- 83 ページの「リソースタイプ、リソースグループ、リソースプロパティの変更」
- 88 ページの「リソースの STOP\_FAILED エラーフラグの消去」
- 90 ページの「Start\_failed リソース状態の消去」

- 96 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」
- 98 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」
- 99 ページの「リソースグループへのノードの追加と削除」
- 109 ページの「大域ゾーンから非大域ゾーンへのアプリケーションの移行」
- 112 ページの「リソースグループとデバイスグループ間での起動の同期」
- 116 ページの「クラスタファイルシステム用の HAStoragePlus リソースの構成」
- 118 ページの「高可用性ローカルファイルシステムの有効化」
- 127 ページの「HAStorage から HAStoragePlus へのアップグレード」
- 130 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に変更する」
- 140 ページの「HAStoragePlus リソースの広域ファイルシステムからローカルファイルシステムへの変更」
- 141 ページの「HAStoragePlus リソースタイプのアップグレード」
- 142 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」
- 152 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」
- 155 ページの「Sun Cluster 上で Solaris SMF サービスを有効にする」
- 167 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

作業	参照先
リソースタイプを登録する	34 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	37 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」 36 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」
リソースタイプをダウングレードする	42 ページの「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」
フェイルオーバーリソースグループまたはスケラブルリソースグループの作成	44 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」 46 ページの「スケラブルリソースグループを作成する」

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)

作業	参照先
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	49 ページの「clsetup ユーティリティを使用して論理ホスト名リソースをリソースグループに追加する」 52 ページの「コマンド行インタフェースを使用して論理ホスト名リソースをリソースグループに追加する」 55 ページの「clsetup ユーティリティを使用して共有アドレスリソースをリソースグループに追加する」 57 ページの「コマンド行インタフェースを使用して共有アドレスリソースをリソースグループに追加する」 60 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」 62 ページの「スケラブルアプリケーションリソースをリソースグループに追加する」
リソースとリソースモニターを有効にし、リソースグループを管理し、リソースグループおよび関連するリソースをオンラインにする	67 ページの「リソースを有効にする」 66 ページの「リソースグループをオンラインにする」
リソースグループを休止する	69 ページの「リソースグループを休止する」 69 ページの「ただちにリソースグループを休止する」
リソースグループの自動回復アクションを保存停止および再開する	71 ページの「リソースグループの自動回復アクションを保存停止する」 71 ページの「リソースグループの自動回復アクションをただちに保存停止する」 72 ページの「リソースグループの自動回復アクションを再開する」
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	72 ページの「リソース障害モニターを無効にする」 73 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	74 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	76 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	77 ページの「リソースを削除する」
リソースグループの稼働系を切り替える	78 ページの「リソースグループの主ノードを切り替える」

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)

作業	参照先
リソースを無効にし、そのリソースグループを UNMANAGED 状態に移行する	80 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	82 ページの「リソースタイプ、リソースグループ、リソース構成情報の表示」
リソースタイプ、リソースグループ、リソースプロパティの変更	83 ページの「リソースタイププロパティを変更する」 85 ページの「リソースグループプロパティを変更する」 86 ページの「リソースプロパティを変更する」
失敗した リソースグループマネージャ (RGM) プロセスのエラーフラグの消去	89 ページの「リソースの STOP_FAILED エラーフラグを消去する」
Start_failed リソース状態の消去	91 ページの「リソースグループのスイッチオーバーにより Start_failed リソース状態を解除する」 93 ページの「リソースグループの再起動により Start_failed リソース状態を解除する」 95 ページの「リソースの無効化および有効化によりリソース状態 Start_failed を解除する」
組み込みリソースタイプ LogicalHostname および SharedAddress の再登録	98 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」
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	99 ページの「リソースグループにノードを追加する」
リソースグループからノードを削除する	103 ページの「リソースグループからノードを削除する」
大域ゾーンから非大域ゾーンへアプリケーションを移行する	109 ページの「大域ゾーンから非大域ゾーンへアプリケーションを移行する」
リソースグループと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 HASStoragePlus を設定する	113 ページの「新しいリソース用に HASStoragePlus リソースタイプを設定する」 115 ページの「既存のリソース用に HASStoragePlus リソースタイプを設定する」 117 ページの「クラスタファイルシステム用に HASStoragePlus リソースを設定する」 121 ページの「clsetup ユーティリティを使用することで HASStoragePlus リソースタイプを設定する」

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)

作業	参照先
HASStoragePlus を設定してローカル Solaris ZFS を高可用性にする	125 ページの「HASStoragePlus リソースタイプを設定し、ローカル Solaris ZFS を高可用性にする」
HASStorage から HASStoragePlus へアップグレードする	128 ページの「デバイスグループまたは CFS を使用している場合に HASStorage から HASStoragePlus へアップグレードする」 129 ページの「CFS による HASStorage から高可用性ローカルファイルシステムによる HASStoragePlus へアップグレードする」
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に変更する	130 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に変更する」
広域ファイルシステムを、HASStoragePlus リソースのローカルファイルシステムに変更する	140 ページの「HASStoragePlus リソースの広域ファイルシステムからローカルファイルシステムへの変更」
組み込みリソースタイプ LogicalHostname および SharedAddress のアップグレード	35 ページの「リソースタイプの更新」 96 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」
HASStoragePlus リソースタイプをアップグレードする	35 ページの「リソースタイプの更新」 141 ページの「HASStoragePlus 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クラスタノード間で分散する	142 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」
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	152 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 上で Solaris SMF サービスを有効にする	155 ページの「Sun Cluster 上で Solaris SMF サービスを有効にする」
Sun Cluster データベース用に障害モニターを調整する	167 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 - この章の各手順では Sun Cluster の保守コマンドを使ってこれらの作業を行います。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を変更するには、[83 ページの「リソースタイプ、リソースグループ、リソースプロパティの変更」](#)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。

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。
- 2 リソースタイプを登録します。

```
# clresourcetype register resource-type
```

`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。

- 登録されたリソースタイプを確認します。

```
# clresourcetype show
```

### 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.krb5 リソースタイプを登録します。

```
# clresourcetype register SUNW.krb5
# clresourcetype show SUNW.krb5
```

```
Resource Type:                SUNW.krb5
RT_description:                HA-Kerberos KDC server for Sun Cluster
RT_version:                    3.2
API_version:                   6
RT_basedir:                    /opt/SUNWsckrb5/bin
Single_instance:              False
Proxy:                          False
Init_nodes:                    All potential masters
Installed_nodes:               <All>
Failover:                       True
Pkglist:                        SUNWsckrb5
RT_system:                      False
```

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、[43 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 次のマニュアルページを参照してください。

- clresourcetype(1CL)
- clresourcegroup(1CL)
- clresource(1CL)

## 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. [36 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. [37 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、`clresource(1CL)` 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、`clresource` コマンドを使用する以外の方法もあります。`clresource` コマンドを使用する代わりに、`Sun Cluster Manager` や、`clsetup(1CL)` 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、順次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。

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。

- 2 対象となる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

```
# clresourcetype register -f path-to-new-rtr-file resource-type-name
```

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、[236 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。  

```
# clresourcetype show resource-type-name
```
- 5 必要に応じて、`Installed_nodes` プロパティを、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` プロパティは、リソースタイプの `Installed_nodes` プロパティのサブセットである必要があります。

```
# clresourcetype set -n installed-node-list resource-type
```

`-n 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。

## ▼ 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、`clresource(1CL)` 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、`clresource` コマンドを使用する以外の方法もあります。`clresource` コマンドを使用する代わりに、`Sun Cluster Manager` や、`clsetup(1CL)` 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。

- すべての時刻
- 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
  - リソースを古いバージョンのリソースタイプのままにする
- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC`の承認を提供する役割になります。
  - 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。  

```
# clresource unmonitor 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。  

```
# clresource disable 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。  

```
# clresource disable 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。

```
# clresource disable -gresource-group +
# clresourcegroup offline resource-group
# clresourcegroup unmanage resource-group
```

上記コマンドの各項目の意味は次のとおりです。

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

- 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を適切な値に設定します。これらのプロパティを設定するには、コマンドで `-p` オプションを指定します。

そのほかのプロパティ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が導入されている。
- 既存のプロパティのデフォルト値が、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。

```
# clresource set -p Type_version=new-version \
[-p extension-property=new-value] [-p standard-property=new-value] resource
```

---

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。

```
# clresource monitor 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。

```
# clresource enable 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。

```
# clresource enable 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。

```
# clresource enable -g resource-group +
```

```
# clresourcegroup manage resource-group
```

```
# clresourcegroup online 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。

```
# clresourcetype register -f /opt/XYZmyrt/etc/XYZ.myrt myrt
# clresource disable myresource
# clresource set -p Type_version=2.0 myresource
# clresource enable 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。

```
# clresource unmonitor 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。

```
# clresourcetype register -f /opt/XYZmyrt/etc/XYZ.myrt myrt
```

4. `Type_version` プロパティ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。

```
# clresource set -p Type_version=2.0 myresource
```

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。

```
# clresource monitor 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、`clresource(1CL)` 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、`clresource` コマンドを使用する以外の方法もあります。`clresource` コマンドを使用する代わりに、`Sun Cluster Manager` や、`clsetup(1CL)` 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。

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify` および `solaris.cluster.admin` **RBAC** の承認を提供する役割になります。
- 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。

```
clresourcegroup offline 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。

```
clresource disable -gresource-group +
```

- 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。

```
clresourcegroup unmanage 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。

```
clresourcetype register 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に適切な値を設定します。

```
clresource set -p Type_version=old-version resource-todowngrade
```

- 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。

```
# clresource enable -gresource-group +
```

- 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。

```
# clresourcegroup manage 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。

```
# clresourcegroup online 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ゾーン、あるいはノード群またはゾーン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。リソースグループは、非大域ゾーンで動作するように構成できます。

---

注-リソースグループのノードリストで指定されるゾーンは、リソースグループの作成時点では存在する必要はありません。ノードリストに指定されているゾーンがRGMにより検出されないと、警告メッセージが表示されますが、エラーにはなりません。

---

リソースグループには、フェイルオーバーとスケラブルの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つのノードまたはゾーンでのみです。一方、スケラブルリソースグループの場合、同時に複数のノードまたはゾーンでオンラインにできます。

次の手順では、`clresourcegroup(1CL)` コマンドを使用して、リソースグループを作成する方法を説明します。

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、クラスタノードまたはゾーン間を移動します。

---

注- この手順は、任意のクラスタノードから実行します。

---

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。
- 2 フェイルオーバーリソースグループを作成します。

```
# clresourcegroup create [-n node-zone-list] resource-group
```

`-n node-zone-list` このリソースグループをマスターできるゾーンの、コンマ区切りの順序付けされたリストを指定します。リスト内の各エントリの形式は `node:zone` です。この形式では、`node` はノード名を指定し、`zone` は非大域 Solaris ゾーンの名前を指定します。大域ゾーンを指定する、または非大域ゾーンを持たないノードを指定するには、`node` のみを指定します。

このリストはオプションです。このリストを省略すると、クラス内のすべてのノード上でリソースグループが作成されます。

---

注- 最高の可用性を実現するには、同一ノード上の異なるゾーンではなく、フェイルオーバーリソースグループのノードリストの異なるノード上でゾーンを指定します。

---

`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。

- リソースグループが作成されていることを確認します。

```
# clresourcegroup show resource-group
```

#### 例 2-4 フェイルオーバーリソースグループの作成

この例で、フェイルオーバーリソースグループ `resource-group-1` の作成を示します。ノード `phys-schost-1` および `phys-schost-2` の大域ゾーンは、このリソースグループをマスターできます。

```
# clresourcegroup create -n phys-schost1,phys-schost-2 resource-group-1
# clresourcegroup show -v resource-group-1
```

```
=== Resource Groups and Resources ===
```

```
Resource Group:                resource-group1
RG_description:                 <NULL>
RG_mode:                       Failover
RG_state:                      Unmanaged
RG_project_name:               default
RG_affinities:                 <NULL>
RG_SLM_type:                   manual
Auto_start_on_new_cluster:     True
Failback:                      False
Nodelist:                      phys-schost-1 phys-schost-2
Maximum primaries:             1
Desired primaries:             1
RG_dependencies:               <NULL>
Implicit network dependencies:  True
Global_resources_used:         <All>
Pingpong_interval:             3600
Pathprefix:                    <NULL>
RG_System:                     False
Suspend_automatic_recovery:    False
```

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、[48 ページの「リソースをリソースグループに追加するためのツール」](#)を参照してください。

参照 `clresourcegroup(1CL)` 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。スケーラブルリソースグループまたは共有アドレスリソースグループのノードリストには、同一ノードで複数の非大域ゾーンが含まれてはなりません。スケーラブルサービスの各インスタンスは、別々のクラスタノードで実行する必要があります。

また、スケーラブルリソースグループを非大域ゾーンで動作するように構成することもできます。スケーラブルリソースは、同一ノードの複数の非大域ゾーンで動作するようには構成しないでください。

---

注- この手順は、任意のクラスタノードから実行します。

---

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。
- 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。

```
# clresourcegroup create \-p Maximum primaries=m \-p Desired primaries=n \
-p RG_dependencies=depend-resource-group \
[-n node-zone-list] resource-group
```

```
-p 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。

```
-p 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。

```
-p 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。

```
-n node-zone-list
```

このリソースグループが使用可能となる、コンマ区切りの順序付けられたリストを指定します。リスト内の各エントリの形式は `node:zone` です。この形式では、`node` はノード名を指定し、`zone` は非大域 Solaris ゾーンの名前を指定しま

す。大域ゾーンを指定する、または非大域ゾーンを持たないノードを指定するには、*node*のみを指定します。

このリストはオプションです。このリストを省略すると、クラス内のすべてのノード上でリソースグループが作成されます。

スケーラブルリソースのノードリストは、共有アドレスリソースのノードリストと同じリストまたは *nodename:zonename* ペアのサブセットを含むことができます。

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。任意の名前の先頭文字はASCIIにする必要があります。

*resource-group*

- 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。

```
# clresourcegroup show resource-group
```

## 例 2-5 スケーラブルリソースグループの作成

この例では、スケーラブルリソースグループ *resource-group-1* の作成を示します。このリソースグループは、ノード *phys-schost-1* および *phys-schost-2* の大域ゾーンでホストされます。スケーラブルリソースグループは、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 *resource-group-2* に依存します。

```
# clresourcegroup create\  
-p Maximum primaries=2\  
-p Desired primaries=2\  
-p RG_dependencies=resource-group-2\  
-n phys-schost-1, phys-schost-2\  
resource-group-1
```

```
# clresourcegroup show resource-group-1
```

```
=== Resource Groups and Resources ===
```

Resource Group:	resource-group-1
RG_description:	<NULL>
RG_mode:	Scalable
RG_state:	Unmanaged
RG_project_name:	default
RG_affinities:	<NULL>

Auto_start_on_new_cluster:	True
Failback:	False
Nodelist:	phys-schost-1 phys-schost-2
Maximum primaries:	2
Desired primaries:	2
RG_dependencies:	resource-group2
Implicit_network_dependencies:	True
Global_resources_used:	<All>
Pingpong_interval:	3600
Pathprefix:	<NULL>
RG_System:	False
Suspend_automatic_recovery:	False

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、[62 ページの「スケ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 `clresourcegroup(1CL)` のマニュアルページ。

## リソースをリソースグループに追加するためのツール

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

Sun Cluster には、リソースをリソースグループに追加するための次のツールがあります。

- **Sun Cluster Manager**。詳細は、Sun Cluster Manager のオンラインヘルプを参照してください。
- `clsetup(1CL)` ユーティリティ。
- **Sun Cluster** の保守コマンド。

Sun Cluster Manager のウィザード、`clsetup` ユーティリティ、または Sun Cluster の保守コマンドを使用すると、論理ホスト名リソースと共有アドレスリソースをリソースグループに追加することができます。

Sun Cluster Manager および `clsetup` ユーティリティーを使用すると、対話形式でリソースをリソースグループに追加できます。これらのリソースを対話的に使うことにより、コマンドの構文エラーまたは脱落による設定エラーが起きる可能性が少なくなります。Sun Cluster Manager および `clsetup` ユーティリティーは、すべての必須リソースが作成され、リソース間のすべての必須依存関係が設定されるようにします。

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またはゾーン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。

---

注-DEPRECATED フラグは、論理ホスト名と共有アドレスリソースを非推奨アドレスにします。これらのアドレスは、アウトバウンド要求には適していません。これは、これらのアドレスがフェイルオーバーまたはスイッチオーバーにより、別のクラスタノードに移行する可能性があるためです。

---

リソースについては、『Sun Cluster の概念 (Solaris OS 版)』および第 1 章を参照してください。

## ▼ `clsetup` ユーティリティーを使用して論理ホスト名リソースをリソースグループに追加する

次の手順では、`clsetup` ユーティリティーを使用することにより、論理ホスト名リソースをリソースグループに追加する方法を説明します。この手順は1つのノードからのみ実行します。

この手順では、Sun Cluster の保守コマンドの長い形式を使用します。多くのコマンドには短縮形もあります。コマンド名の形式を除き、コマンドは同じです。コマンドのリストとその短縮形については、付録 A を参照してください。

始める前に 次の前提条件を満たしていることを確認します。

- リソースにより使用可能になる各論理ホスト名のエントリが、ネームサービスデータベースに追加されています。
- IP Networking Multipathing (IPMP) グループを使用する場合は、論理ホスト名リソースをオンライン状態にできるノード上で一連のグループを構成します。
- 該当リソースのマスターになることができる非大域ゾーンはすべて、クラスタノード上で構成済みです。

次の情報を用意してください。

- リソースグループに追加するホスト名

1 任意のクラスタノードでスーパーユーザーになります。

2 `clsetup`ユーティリティを起動します。

```
# clsetup
```

`clsetup`のメインメニューが表示されます。

3 データサービスのオプションに対応する番号を入力し、**Return**キーを押します。

「データサービス」メニューが表示されます。

4 論理ホスト名リソースを構成するためのオプションに対応する番号を入力し、**Return**キーを押します。

`clsetup`ユーティリティは、この作業を実行するための前提条件のリストを表示します。

5 前提条件が満たされていることを確認し、**Return**キーを押して続けます。

`clsetup`ユーティリティは、論理ホスト名リソースをオンラインにすることができるクラスタノードまたはゾーンのリストを表示します。

6 論理ホスト名リソースをオンラインにすることができるノードまたはゾーンを選択します。

- 任意の順序で並んでいる一覧表示されたすべてのノードのデフォルト選択をそのまま使用するには、**a**と入力し、**Return**キーを押します。

- 一覧表示されたノードまたはゾーンのサブセットを選択するには、ノードに対応する番号のコンマ区切りまたはスペース区切りのリストを入力します。続いて、**Return**キーを押します。

- 特定の順序ですべてのノードを選択するには、ノードに対応する番号のコンマ区切りまたはスペース区切りの順序付きリストを入力し、**Return**キーを押します。  
論理ホスト名リソースグループのノードリストにノードが表示される順序でノードが一覧表示されていることを確認します。リストの最初のノードは、このリソースグループの主ノードです。

7 ノードの選択を確認するには、**d**を入力して、**Return**キーを押します。

`clsetup`ユーティリティは、リソースが使用可能にする論理ホスト名をユーザーが選択できる画面を表示します。

- 8 このリソースが使用可能にする論理ホスト名を入力し、**Return** キーを押します。  
clsetup ユーティリティは、このユーティリティが作成する Sun Cluster オブジェクトの名前を表示します。
- 9 **Sun Cluster** オブジェクトに別の名前が必要である場合、次のように名前を変更します。
  - a. 変更する名前に対応する番号を入力し、**Return** キーを押します。  
clsetup ユーティリティは、新しい名前を指定できる画面を表示します。
  - b. 「新しい値」プロンプトで、新しい名前を入力し、**Return** キーを押します。  
clsetup ユーティリティは、このユーティリティが作成する Sun Cluster オブジェクトの名前のリストに戻ります。
- 10 **Sun Cluster** オブジェクト名の選択を確認するには、dを入力して、**Return** キーを押します。  
clsetup ユーティリティは、このユーティリティが作成する Sun Cluster の構成に関する情報を表示します。
- 11 構成を作成するには、cを入力し、**Return** キーを押します。  
clsetup ユーティリティは、構成を作成するためにこのユーティリティがコマンドを実行していることを示す進行状況のメッセージを表示します。構成が完了した時点で、clsetup ユーティリティは、構成を作成するためにユーティリティが実行したコマンドを表示します。
- 12 (省略可能) clsetup ユーティリティが終了するまで、繰り返し q を入力し、**Return** キーを押します。  
必要に応じて、ほかの必要な作業を実行している間、clsetup ユーティリティを動作させたままにし、そのあとでユーティリティを再度使用することができます。  
clsetup を終了する場合、ユーザーがユーティリティを再起動する際に、ユーティリティは既存の論理ホスト名リソースグループを認識します。
- 13 論理ホスト名リソースが作成されていることを確認します。  
このためには、clresource(1CL) ユーティリティを使用します。デフォルトでは、clsetup ユーティリティは、リソースグループに名前 `node_name-rg` を割り当てます。  

```
# clresource show node_name-rg
```

## ▼ コマンド行インタフェースを使用して論理ホスト

## 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、[87 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

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC`の承認を提供する役割になります。
- 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。

```
# clreslogicalhostname create -g resource-group -h hostnamelist, ... [-N netiflist] resource
```

`-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。

`-h 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。論理ホスト名リソースが、非大域ゾーンで動作するリソースグループに追加される場合、対応する IP アドレスはそのゾーン内で構成されます。これらの IP アドレスは、そのゾーンで動作するアプリケーションのみが使用できます。

完全修飾ホスト名が必要である場合は、`-h` オプションを使用して完全修飾名を指定します。

`-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。

*resource*                      リソース名を指定します (省略可能)。リソース名では完全修飾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。

```
# clresource show 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。

```
# clreslogicalhostname create -g resource-group-1 -h schost-1 resource-1
# clresource show resource-1
```

```
=== Resources ===
```

```
Resource:                               resource-1
Type:                                     SUNW.LogicalHostname:2
Type_version:                             2
Group:                                    resource-group-1
R_description:
Resource_project_name:                    default
Enabled{phats1}:                          True
Enabled{phats2}:                          True
Monitored{phats1}:                        True
Monitored{phats2}:                        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。

```
# clreslogicalhostname create -g nfs-fo-rg -h cs23-rs -N sc_ipmp0@1,sc_ipmp0@2 cs23-rs
# clreslogicalhostname create -g nfs-fo-rg -h cs24-rs -N sc_ipmp1@1,sc_ipmp1@2 cs24-rs
```

次の手順 論理ホスト名リソースを追加したあと、66 ページの「リソースグループをオンラインにする」を参照して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、`clreslogicalhostname` 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 `clreslogicalhostname` 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** `clreslogicalhostname(1CL)` のマニュアルページ。

## ▼ `clsetup` ユーティリティーを使用して共有アドレスリソースをリソースグループに追加する

次の手順では、`clsetup` ユーティリティーを使用することにより、共有アドレスリソースをリソースグループに追加する方法を説明します。この手順は、任意のクラスタノードから実行します。

この手順では、Sun Cluster の保守コマンドの長い形式を使用します。多くのコマンドには短縮形もあります。コマンド名の形式を除き、コマンドは同じです。コマンドのリストとその短縮形については、[付録 A](#) を参照してください。

**始める前に** 次の前提条件を満たしていることを確認します。

- リソースにより使用可能となる共有アドレスは、ネームサービスデータベース内にエントリを持っています。
- IP Networking Multipathing (IPMP) グループを使用する場合は、共有アドレスリソースをオンライン状態にできるノード上で一連のグループを構成します。
- 該当リソースのマスターになることができる非大域ゾーンはすべて、クラスタノード上で構成済みです。

次の情報を用意してください。

- リソースグループに追加するホスト名。

1 任意のクラスタノードでスーパーユーザーになります。

2 `clsetup` ユーティリティを起動します。

# `clsetup`

`clsetup` のメインメニューが表示されます。

3 データサービスのオプションに対応する番号を入力し、**Return** キーを押します。

「データサービス」メニューが表示されます。

4 共有アドレスリソースを構成するためのオプションに対応する番号を入力し、**Return** キーを押します。

`clsetup` ユーティリティは、この作業を実行するための前提条件のリストを表示します。

5 前提条件が満たされていることを確認し、**Return** キーを押して続けます。

`clsetup` ユーティリティは、共有アドレスリソースをオンラインにすることができるクラスタノードまたはゾーンのリストを表示します。

6 共有アドレスリソースをオンラインにすることができるノードまたはゾーンを選択します。

- 任意の順序で並んでいる一覧表示されたすべてのノードのデフォルト選択をそのまま使用するには、**a** と入力し、**Return** キーを押します。

- 一覧表示されたノードまたはゾーンのサブセットを選択するには、ノードに対応する番号のコンマ区切りまたはスペース区切りのリストを入力します。続いて、**Return** キーを押します。

- 特定の順序ですべてのノードを選択するには、ノードに対応する番号のコンマ区切りまたはスペース区切りの順序付きリストを入力し、**Return** キーを押します。

7 ノードの選択を確認するには、**d** を入力して、**Return** キーを押します。

`clsetup` ユーティリティは、リソースが使用可能にする共有アドレスをユーザーが指定できる画面を表示します。

8 このリソースが使用可能にする共有アドレスを入力し、**Return** キーを押します。

`clsetup` ユーティリティは、このユーティリティが作成する Sun Cluster オブジェクトの名前を表示します。

- 9 **Sun Cluster** オブジェクトに別の名前が必要である場合、次のように名前を変更します。
  - a. 変更する名前に対応する番号を入力し、**Return** キーを押します。  
clsetup ユーティリティは、新しい名前を指定できる画面を表示します。
  - b. 「新しい値」プロンプトで、新しい名前を入力し、**Return** キーを押します。  
clsetup ユーティリティは、このユーティリティが作成する Sun Cluster オブジェクトの名前のリストに戻ります。
- 10 **Sun Cluster** オブジェクト名の選択を確認するには、dを入力して、**Return** キーを押します。  
clsetup ユーティリティは、このユーティリティが作成する Sun Cluster の構成に関する情報を表示します。
- 11 構成を作成するには、cを入力し、**Return** キーを押します。  
clsetup ユーティリティは、構成を作成するためにこのユーティリティがコマンドを実行していることを示す進行状況のメッセージを表示します。構成が完了した時点で、clsetup ユーティリティは、構成を作成するためにユーティリティが実行したコマンドを表示します。
- 12 (省略可能) clsetup ユーティリティが終了するまで、繰り返し q を入力し、**Return** キーを押します。  
必要に応じて、ほかの必要な作業を実行している間、clsetup ユーティリティを動作させたままにし、そのあとでユーティリティを再度使用することができます。clsetup ユーティリティを終了する場合、ユーザーがユーティリティを再起動する際に、ユーティリティは既存の共有アドレスリソースグループを認識します。
- 13 共有アドレスリソースが作成されていることを確認します。  
このためには、clresource(1CL) ユーティリティを使用します。デフォルトでは、clsetup ユーティリティは、リソースグループに名前 *node\_name-rg* を割り当てます。  

```
# clresource show node_name-rg
```

## ▼ コマンド行インタフェースを使用して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、[87 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。

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC`の承認を提供する役割になります。
- 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。

```
# clressharedaddress create -g resource-group -h hostnamelist, ... \  
[-X auxnodelist] [-N netiflist] resource
```

`-g resource-group` リソースグループの名前を指定します。共有アドレスリソースのノードリストでは、同一ノード上では複数のゾーンを指定しないでください。共有アドレスリソースのノードリストは、同一ノード上で異なるゾーンを指定してはいけません。`nodename:zonename` のペアの同一リストをスケラブルリソースグループのノードリストとして指定します。

`-h 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主ノードとして使用されない)を識別する物理ノード名またはIDまたはゾーン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。補助ノードリストが明示的に指定されていない場合、リストのデフォルトは、共有アドレスリソースを含むリソースグループのノードリストには含まれていない、すべてのクラスタノード名のリストになります。

---

注-サービスをマスターするために作成されたすべての非大域ゾーン内でスケーラブルサービスを動作させるには、共有アドレスリソースグループのノードリスト、または共有アドレスリソースの `auxnodelist` にゾーンの完全なリストを含めます。すべてのゾーンがノードリスト内にある場合は、`auxnodelist` を省略できます。

---

`-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。

---

`resource`

リソース名を指定します (省略可能)。

- 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。

```
# clresource show 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。

```
# clressharedaddress create -g resource-group-1 -h schost-1 resource-1
# clresource show resource-1
```

```
=== Resources ===
```

```
Resource:                resource-1
Type:                    SUNW.SharedAddress:2
Type_version:            2
Group:                   resource-group-1
R_description:
Resource_project_name:   default
Enabled{phats1}:         False
Enabled{phats2}:         False
Monitored{phats1}:      True
Monitored{phats2}:      True
```

次の手順 共有アドレスリソースを追加したあと、66 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、`clressharedaddress` 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 `clressharedaddress` 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 `clressharedaddress(1CL)` 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。

---

注-この手順はプロキシリソースにも該当します。

---

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。
- 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。

```
# clresource create -g resource-group -t resource-type \
```

```
[-p "extension-property[{node-specifier}]=value, ...] [-p standard-property=value, ...] resource
```

```
-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。

-p "*extension-property*[*{node-specifier}*]"=*value*, ...

リソース用に設定する拡張プロパティ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はリソースタイプに依存します。どの拡張プロパティ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。

*node-specifier* は、-p オプションおよび -x オプションに対する「オプション」の修飾子です。この修飾子は、指定された1つまたは複数のノードまたはゾーン上でのみ、1つまたは複数の拡張プロパティがリソースの作成時に設定されることを示します。指定した拡張プロパティは、クラスタ内のほかのノードまたはゾーン上では、設定されません。*node-specifier* を指定しないと、クラスタ内のすべてのノードおよびゾーン上の指定された拡張プロパティが設定されます。*node-specifier* にはノード名またはノード識別子を指定できます。*node-specifier* の構文例を次に示します。

**-p "myprop{phys-schost-1}"**

中括弧({}) は、指定した拡張プロパティをノード `phys-schost-1` でのみ設定することを示します。大部分のシェルでは、二重引用符(")が必要です。

また次の構文を使用して、2つの異なるノード上の2つの異なるゾーン内で拡張プロパティを設定することもできます。

**-x "myprop{phys-schost-1:zoneA,phys-schost-2:zoneB}"**

---

注 - *node-specifier* を使用して指定する拡張プロパティは、ノードごとのプロパティとして RTR ファイルで宣言します。Per\_node リソースプロパティの属性の詳細は、[付録 B](#) を参照してください。

---

-p *standard-property*=*value*, ...

リソース用に設定する標準プロパティ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はリソースタイプに依存します。どの標準プロパティ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 B](#) を参照してください。

*resource*

追加するリソースの名前を指定します。

リソースは有効状態で作成されます。

- 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。

```
# clresource show 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。

```
# clresource create -g resource-group-1 -t resource-type-1 \
-p Network_resources_used=schost-1,schost2 resource-1\
# clresource show resource-1

=== Resources ===

Resource:                resource-1
Type:                    resource-type-1
Type_version:
Group:                   resource-group-1
R_description:
Resource_project_name:   default
Enabled{phats1}:         False
Enabled{phats2}:         False
Monitored{phats1}:      True
Monitored{phats2}:      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、[66 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、clresource 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 clresource 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** clresource(1CL) 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。

---

注- この手順はプロキシリソースにも該当します。

---

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC`の承認を提供する役割になります。
- 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。

```
# clresource create -g resource-group -t resource-type \
-p Network_resources_used=network-resource[,network-resource...] \
-p Scalable=True
[-p "extension-property[{node-specifier}]=value, ..." [-p standard-property=value, ...] resource
-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。
-p 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。
-p 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。
-p "extension-property[{node-specifier}]=value, ..."
  リソース用に設定する拡張プロパティ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はリソースタイプに依存します。どの拡張プロパティ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。
```

`node-specifier` は、`-p` オプションおよび `-x` オプションに対する「オプション」の修飾子です。この修飾子は、指定された1つまたは複数のノードまたはゾーン上でのみ、1つまたは複数の拡張プロパティがリソースの作成時に設定されることを示します。指定した拡張プロパティは、クラスタ内のほかのノードまたはゾーン上では、設定されません。`node-specifier` を指定しないと、クラスタ内のすべ

でのノードおよびゾーン上の指定された拡張プロパティが設定されます。  
*node-specifier*にはノード名またはノード識別子を指定できます。*node-specifier*の構文例を次に示します。

```
-p "myprop{phys-schost-1}"
```

中括弧({})は、指定した拡張プロパティをノード `phys-schost-1` でのみ設定することを示します。大部分のシェルでは、二重引用符(")が必要です。

また次の構文を使用して、2つの異なるノード上の2つの異なるゾーン内で拡張プロパティを設定することもできます。

```
-x "myprop{phys-schost-1:zoneA,phys-schost-2:zoneB}"
```

---

注 *-node-specifier* を使用して指定する拡張プロパティは、ノードごとのプロパティとして RTR ファイルで宣言します。Per\_node リソースプロパティの属性の詳細は、[付録 B](#) を参照してください。

---

```
-p standard-property=value, ...
```

リソース用に設定する標準プロパティ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はリソースタイプに依存します。スケラブルサービスの場合、通常は `Port_list`、`Load_balancing_weights`、および `Load_balancing_policy` プロパティを設定します。どの標準プロパティ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 B](#)を参照してください。

*resource*

追加するリソースの名前を指定します。

リソースは有効状態で作成されます。

- 3 スケ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。

```
# clresource show resource
```

## 例 2-10 スケ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。

```
# clresource create -g resource-group-1 -t resource-type-1 \
-p Network_resources_used=schost-1,schost-2 resource-1 \
-p Scalable=True
# clresource show resource-1

=== Resources ===

Resource:                resource-1
Type:                    resource-type-1
Type_version:
Group:                   resource-group-1
R_description:
Resource_project_name:   default
Enabled{phats1}:         False
Enabled{phats2}:         False
Monitored{phats1}:       True
Monitored{phats2}:       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、[66 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、clresource 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 clresource 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 clresource(1CL) 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となります。リソース、ノード、またはゾーンで障害が発生した場合は、RGM は別のノードまたはゾーン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。

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.admin RBAC`の承認を提供する役割になります。
- 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。

- 無効のままであ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。

```
# clresourcegroup online rg-list
```

*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。

*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。

```
# clresourcegroup online -emM rg-list
```

*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。

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

---

注-オンラインにしようとしている任意のリソースグループがほかのリソースグループに対して強いアフィニティを宣言している場合、この操作は失敗します。詳細については、[142 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、どのノードまたはゾーンで各リソースグループがオンラインであることを示します。

```
# clresourcegroup status
```

## 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。

```
# clresourcegroup online -emM resource-group-1
# clresourcegroup status
```

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、73 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 `clresourcegroup(1CL)` のマニュアルページ。

## リソースの有効化

リソースグループをオンラインにしたときに有効にしなかったリソースを有効にすることができます。

### ▼ リソースを有効にする

---

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 有効にするリソースを作成し、名前が付いていることを確認します。

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.admin RBAC` の承認を提供する役割になります。
- 2 リソースを有効にします。

```
# clresource enable [-n node-zone-list] resource
```

`-n node-zone-list` リソースを有効にするノードまたはゾーンの、コンマ区切りの順序付きリストを指定します。ゾーンを指定する場合、リストの各エントリの形式は `node:zone` です。この形式では、`node` はノード名を指定し、`zone` は非大域ゾーンの名前を指定します。大域ゾーンを指定する、または非大域ゾーンを持たないノードを指定するには、`node` のみを指定します。

このリストはオプションです。このリストを省略すると、そのリソースグループのノードリスト内のすべてのノード上でリソースが有効になります。

---

注--n オプションを使用して複数のノードまたはゾーンを指定した場合、1つのリソースのみを指定できます。

---

*resource*                   有効にするリソースの名前を指定します。

- 3 リソースが有効であることを確認します。

**# clresource status**

このコマンドからの出力は、有効にしたリソースの状態を示します。

参照 `clresource(1CL)` のマニュアルページ。

## リソースグループの休止

START または STOP メソッドに障害が発生した場合、あるノードまたはゾーンから別のノードまたはゾーンへリソースグループが継続的に切り替わるのを停止するには、そのリソースグループを休止状態にします。リソースグループを休止状態にするには、`clresourcegroup quiesce` コマンドを実行します。

リソースグループを休止する場合、実行中のリソースメソッドは、完了するまで実行することを許可されます。重大な問題が発生した場合は、ただちにリソースグループを休止する必要がある場合があります。このためには、次のメソッドを終了させる `-k` コマンドオプションを指定します。

- `Prenet_start`
- `Start`
- `Monitor_start`
- `Monitor_stop`
- `Stop`
- `Postnet_stop`

---

注- このコマンドオプションを指定した場合、`Init`、`Fini`、`Boot`、および `Update` メソッドは終了しません。

---

ただし、メソッドを終了することによりリソースグループをただちに休止した場合、リソースのいずれかを `Start_failed` や `Stop_failed` などのエラー状態のままにしてしまう場合があります。ユーザーは自分自身でこれらのエラー状態を消去する必要があります。

## ▼ リソースグループを休止する

- 1 スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。
- 2 リソースグループを休止します。

```
# clresourcegroup quiesce resource-group
```

## ▼ ただちにリソースグループを休止する

- 1 スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。
- 2 ただちにリソースグループを休止します。

```
# clresourcegroup quiesce -k resource-group
```

リソースグループと関連付けられている `Prenet_start`、`Start`、`Monitor_start`、`Monitor_stop`、`Stop`、および `Postnet_stop` メソッドはただちに終了します。リソースグループは休止状態になります。

`clresourcegroup quiesce -k` コマンドは、指定したリソースグループが休止状態になるまで作業をブロックします。

# リソースグループの自動回復アクションの保存停止と再開

リソースグループの自動回復アクションを一時的に中断することもできます。クラスタ内の問題を調べて修正するため、リソースグループの自動回復を中断しなければならない場合があります。または、リソースグループサービスの保守作業を実行しなければならない場合もあります。

リソースグループの自動回復アクションを保存停止するには、`clresourcegroup suspend` コマンドを実行します。自動回復アクションを再開するには、`clresourcegroup resume` コマンドを実行します。

リソースグループの自動回復アクションを保存停止した場合は、そのリソースグループを休止状態にすることもなります。

自動復旧を再開するコマンドを明示的に実行するまで、中断されたリソースグループが自動的に再開またはフェイルオーバーされることはありません。中断されたデータサービスは、オンラインかオフラインかにかかわらず、現在の状態のままとなります。指定したノードまたはゾーン上でリソースグループの状態を手作業で切り替えることもできます。また、リソースグループ内の個々のリソースも有効または無効にできます。

次のいずれかの状況でリソースグループの自動回復アクションを保存停止した場合、依存関係またはアフィニティーは保存停止され、行使されません。

- 別のリソースに対する再起動の依存関係を持つリソースを含む
- 別のリソースグループに対する強い肯定的または否定的なアフィニティーを宣言している

これらのカテゴリのリソースグループのいずれかを保存停止した場合、Sun Cluster は、依存関係またはアフィニティーも保存停止されるという警告を表示します。

---

注-RG\_system プロパティーを設定しても、リソースグループの自動回復アクションを保存停止または再開するユーザーの能力には影響しません。RG\_system プロパティーが TRUE に設定されているリソースグループを保存停止すると、警告メッセージが表示されます。RG\_system プロパティーは、リソースグループに重要なシステムサービスが含まれていることを指定します。TRUE に設定されている場合、RG\_system プロパティーは、リソースグループまたはそのリソースをユーザーが誤って停止、削除、または変更できないようにします。

---

## メソッドを終了することによる自動回復の即時保存停止

リソースグループの自動回復アクションを保存停止した場合、実行中のリソースメソッドは完了するまで実行を許可されます。重大な問題が発生した場合、リソースグループの自動回復アクションをただちに保存停止する必要がある場合があります。このためには、次のメソッドを終了させる -k コマンドオプションを指定します。

- Prenet\_start
- Start
- Monitor\_start
- Monitor\_stop
- Stop

## ■ Postnet\_stop

注- このコマンドオプションを指定した場合、Init、Fini、Boot、およびUpdate メソッドは終了しません。

ただし、メソッドを終了することにより自動回復アクションをただちに保存停止した場合、リソースのいずれかを Start\_failed や Stop\_failed などのエラー状態のままにしてしまう場合があります。ユーザーは自分自身でこれらのエラー状態を消去する必要があります。

## ▼ リソースグループの自動回復アクションを保存停止する

- 1 スーパーユーザーになるか、solaris.cluster.modify RBAC の承認を提供する役割になります。
- 2 リソースグループの自動回復アクションを保存停止します。

```
# clresourcegroup suspend resource-group
```

指定したリソースグループは、自動回復アクションを再開するまで、自動的に起動、再起動、またはフェイルオーバーされることはありません。72 ページの「リソースグループの自動回復アクションを再開する」を参照してください。

## ▼ リソースグループの自動回復アクションをただちに保存停止する

- 1 スーパーユーザーになるか、solaris.cluster.modify RBAC の承認を提供する役割になります。
- 2 リソースグループの自動回復アクションをただちに保存停止します。

```
# clresourcegroup suspend -k resource-group
```

リソースグループと関連付けられている Prenet\_start、Start、Monitor\_start、Monitor\_stop、Stop、および Postnet\_stop メソッドはただちに終了します。リソースグループの自動回復アクションは保存停止されます。リソースグループは、ユーザーが自動回復アクションを再開するまで、自動的に起動、再起動、またはフェイルオーバーされることはありません。72 ページの「リソースグループの自動回復アクションを再開する」を参照してください。

`clresourcegroup suspend -k` コマンドは、指定したリソースグループが休止状態になるまで作業をブロックします。

## ▼ リソースグループの自動回復アクションを再開する

- 1 スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。
- 2 リソースグループの自動回復アクションを再開します。

```
# clresourcegroup resume resource-group
```

指定したリソースグループは、自動的に起動、再起動、またはフェイルオーバーされます。

## リソースモニターの無効化と有効化

この節の手順では、リソース自体ではなく、リソース障害モニターを有効または無効にする方法を説明します。したがって、障害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開始されません。

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

---

注-これらの手順は、任意のクラスタノードから実行します。

---

## ▼ リソース障害モニターを無効にする

- 1 任意のクラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。
- 2 リソース障害モニターを無効にします。

```
# clresource unmonitor [-n node-zone-list] resource
```

`-n node-zone-list` リソースの監視を解除するノードまたはゾーンの、コンマ区切りの順序付きリストを指定します。ゾーンを指定する場合、リストの各エントリの形式は `node:zone` です。この形式では、`node` は

ノード名を指定し、*zone* は非大域ゾーンの名前を指定します。大域ゾーンを指定する、または非大域ゾーンを持たないノードを指定するには、*node* のみを指定します。

このリストはオプションです。このリストを省略すると、そのリソースグループのノードリスト内のすべてのノード上でリソースの監視が解除されます。

---

注 `--n` オプションを使用して複数のノードまたはゾーンを指定した場合、1つのリソースのみを指定できます。

---

*resource*            1つ以上のリソースの名前を指定します。

- 3 各クラスタノード上で `clresource` コマンドを実行し、監視対象フィールド (RS Monitored) をチェックし、リソース障害モニターが無効になったことを確認します。

```
# clresource show -v
```

## 例 2-12 リソース障害モニターを無効にする

```
# clresource unmonitor resource-1
# clresource show -v
...
RS Monitored: no...
```

## ▼ リソース障害モニターを有効にする

- 1 任意のクラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。
- 2 リソース障害モニターを有効にします。

```
# clresource monitor [-n node-zone-list] resource
```

`-n node-zone-list`    リソースを監視するノードまたはゾーンの、コンマ区切りの順序付きリストを指定します。ゾーンを指定する場合、リストの各エントリの形式は `node:zone` です。この形式では、*node* はノード名を指定し、*zone* は非大域ゾーンの名前を指定します。大域ゾーンを指定する、または非大域ゾーンを持たないノードを指定するには、*node* のみを指定します。

このリストはオプションです。このリストを省略すると、そのリソースグループのノードリスト内のすべてのノード上でリソースが監視されます。

---

注--n オプションを使用して複数のノードまたはゾーンを指定した場合、1つのリソースのみを指定できます。

---

*resource* 1つ以上のリソースの名前を指定します。

- 3 各クラスタノード上で `clresource` コマンドを実行し、監視対象フィールド (RS Monitored) をチェックし、リソース障害モニターが有効になったことを確認します。

```
# clresource show -v
```

### 例 2-13 リソース障害モニターを有効にする

```
# clresource monitor resource-1
# clresource show -v
...
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。

```
# clresourcetype show -v
```

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC`の承認を提供する役割になります。
- 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。  

```
# clresource disable resource
```

`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。  

```
# clresource delete resource
```

`resource`            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。  

```
# clresourcetype unregister resource-type
```

`resource-type`    登録を解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。  

```
# clresourcetype show
```

#### 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。

```
# clresource disable resource-1
# clresource delete resource-1
# clresourcetype unregister resource-type-1
```

参照 次のマニュアルページを参照してください。

- `clresource(1CL)`
- `clresourcetype(1CL)`

## 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。

```
# clresource show -v
```

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC`の承認を提供する役割になります。

- 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。

```
# clresourcegroup offline resource-group
```

`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。

```
# clresource disable resource
```

`resource` 無効にするリソースの名前を指定します。

- 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。

```
# clresource delete resource
```

`resource` 削除するリソースの名前を指定します。

- 5 リソースグループを削除します。

```
# clresourcegroup delete resource-group
```

`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。

```
# clresourcegroup show
```

### 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。

```
# clresourcegroup offline resource-group-1
```

```
# clresource disable resource-1
```

```
# clresource delete resource-1
```

```
# clresourcegroup delete resource-group-1
```

参照 次のマニュアルページを参照してください。

- `clresource(1CL)`
- `clresourcegroup(1CL)`

## 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

1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC`の承認を提供する役割になります。

2 削除するリソースを無効にします。

```
# clresource disable resource
```

*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。

```
# clresource delete resource
```

*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。

```
# clresource show
```

#### 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。

```
# clresource disable resource-1
# clresource delete resource-1
```

参照 `clresource(1CL)`

## 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またはゾーンに切り替え (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か確認します。

- 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またはゾーンの名前
- 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またはゾーンはクラスタ内にある。
- これらのノードまたはゾーンは、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な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。

```
# clresourcegroup show -v
```

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC`の承認を提供する役割になります。
- 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。

```
# clresourcegroup switch [-n node-zone-list] resource-group
```

`-n node-zone-list` このリソースグループをマスターできるゾーンの、コンマ区切りの順序付けされたリストを指定します。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。リスト内の各エントリの形式は `node:zone` です。この形式では、`node` はノード名を指定し、`zone` は非大域 Solaris ゾーンの名前を指定します。大域ゾーンを指定する、または非大域ゾーンを持たないノードを指定するには、`node` のみを指定します。

このリストはオプションです。このリストを省略すると、そのリソースグループのノードリスト内のすべてのノード上でリソースグループが切り替えられます。

`resource-group` 切り替えるリソースグループの名前を指定します。

---

注-切り替えようとしている任意のリソースグループが他のリソースグループに対して強いアフィニティを宣言している場合、その操作は失敗するか、委任されません。詳細については、[142 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。

---

- 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。

```
# clresourcegroup status
```

### 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、潜在的な主ノード (`phys-schost-2`) へ切り替える例を示します。

- `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# clresourcegroup status
```

```
=== Cluster Resource Groups ===
```

Group Name	Node Name	Suspended	Status
-----	-----	-----	-----
resource-group1	phys-schost-1	No	Online
	phys-schost-2	No	Offline

- 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# clresourcegroup switch -n phys-schost-2 resource-group-1
```

- `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# clresourcegroup status
```

```
=== Cluster Resource Groups ===
```

Group Name	Node Name	Suspended	Status
------------	-----------	-----------	--------

resource-group1	phys-schost-1	No	Offline
	phys-schost-2	No	Online

参照 `clresourcegroup(1CL)` のマニュアルページ。

## 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。

詳細は、`clresourcegroup(1CL)` 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しても、そのリソースは依然として一部のホストから `ping(1M)` コマンドに回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` コマンドに回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。

- 無効にする各リソースの名前
- 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。

```
# clresourcegroup show -v
```

- 1 任意のクラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.admin` RBACの承認を提供する役割になります。

- 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。

```
# clresource disable [-n node-zone-list] -g resource-group +
```

`-n node-zone-list` リソースを無効にするノードまたはゾーンの、コンマ区切りの順序付きリストを指定します。ゾーンを指定する場合、リストの各エントリの形式は `node:zone` です。この形式では、`node` はノード名を指定し、`zone` は非大域ゾーンの名前を指定します。大域ゾーンを指定する、または非大域ゾーンを持たないノードを指定するには、`node` のみを指定します。

このリストはオプションです。このリストを省略すると、そのリソースグループのノードリスト内のすべてのノード上でリソースが無効になります。

---

注 `-n` オプションを使用して複数のノードまたはゾーンを指定した場合、1つのリソースのみを指定できます。

---

- 3 リソースグループをオフラインに切り替えます。

```
# clresourcegroup offline resource-group
```

`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。

```
# clresourcegroup unmanage resource-group
```

`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。

```
# clresourcegroup show resource-group
```

## 例 2-18 リソースを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。

```
# clresource disable resource-1
# clresourcegroup offline resource-group-1
# clresourcegroup unmanage resource-group-1
# clresourcegroup show resource-group-1
```

```

=== Resource Groups and Resources ===

Resource Group:                resource-group-1
RG_description:                <NULL>
RG_mode:                       Failover
RG_state:                      Unmanaged
Failback:                      False
Nodelist:                      phys-schost-1 phys-schost-2

--- Resources for Group resource-group-1 ---

Resource:                      resource-1
Type:                          SUNW.LogicalHostname:2
Type_version:                  2
Group:                         resource-group-1
R_description:
Resource_project_name:        default
Enabled{phys-schost-1}:       False
Enabled{phys-schost-2}:       False
Monitored{phys-schost-1}:     True
Monitored{phys-schost-2}:     True

```

参照 次のマニュアルページを参照してください。

- `clresource(1CL)`
- `clresourcegroup(1CL)`

## 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またはゾーンから、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。

---

また、`clresourcetype`、`clresourcegroup`、および`clresource` コマンドを使用して、特定のリソースタイプ、リソースグループ、およびリソースに関するステータス情報をチェックすることもできます。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。

```
# clresource show apache-1
```

詳細は、次のマニュアルページを参照してください。

- `clresourcetype(1CL)`
- `clresourcegroup(1CL)`
- `clresource(1CL)`

## リソースタイプ、リソースグループ、リソースプロパティの変更

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

- 183 ページの「資源タイプのプロパティ」
- 194 ページの「リソースのプロパティ」
- 216 ページの「リソースグループのプロパティ」

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

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

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

### ▼ リソースタイププロパティを変更する

---

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の名前。リソースタイプの場合、特定のプロパティだけを変更できます。プロパティを変更できるかどうかを判断するには、183 ページの「資源タイプのプロパティ」でプロパティの Tunable エントリを参照してください。

---

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

---

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。
- 2 `clresourcetype` 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。

```
# clresourcetype show -v
```

- 3 リソースタイププロパティを変更します。

リソースタイプの場合、特定のプロパティだけを変更できます。プロパティを変更できるかどうかを判断するには、[183 ページの「資源タイプのプロパティ」](#)でプロパティの Tunable エントリを参照してください。

```
# clresourcetype set -n installed-node-list \  
[-p property=new-value]resource-type
```

`-n 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。

`-p property=new-value` 変更する標準プロパティの名前と、そのプロパティの新しい値を指定します。

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

- 4 リソースタイププロパティが変更されていることを確認します。

```
# clresourcetype show resource-type
```

## 例 2-19 リソースタイププロパティの変更

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

```
# clresourcetype set -n phys-schost-1,phys-schost-2 SUNW.apache  
# clresourcetype show SUNW.apache
```

```
Resource Type:                SUNW.apache:4  
RT_description:               Apache Web Server on Sun Cluster  
RT_version:                   4
```

```

API_version:                2
RT_basedir:                  /opt/SUNWscapc/bin
Single_instance:            False
Proxy:                       False
Init_nodes:                  All potential masters
Installed_nodes:            All
Failover:                    False
Pkglist:                     SUNWscapc
RT_system:                   False

```

## ▼ リソースグループプロパティを変更する

この手順では、リソースグループプロパティの変更方法について説明します。リソースグループプロパティの詳細については、[216 ページの「リソースグループのプロパティ」](#)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の名前とその新しいプロパティ値

1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC`の承認を提供する役割になります。

2 リソースグループプロパティを変更します。

```
# clresourcegroup set -p property=new-value resource-group
```

`-p property` 変更するプロパティの名前を指定します。

`resource-group` リソースグループの名前を指定します。

3 リソースグループプロパティが変更されていることを確認します。

```
# clresourcegroup show resource-group
```

### 例 2-20 リソースグループプロパティの変更

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

```
# clresourcegroup set-p Failback=True resource-group-1
# clresourcegroup show resource-group-1
```

## ▼ リソースプロパティを変更する

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

- 標準リソースプロパティの詳細については、194 ページの「リソースのプロパティ」を参照してください。
- リソースの拡張プロパティの詳細については、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を持つリソースの名前
- 変更するプロパティの名前

1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC`の承認を提供する役割になります。

2 現在のリソースプロパティ設定を表示します。

```
# clresource show -v resource
```

3 リソースプロパティを変更します。

```
# clresource set -p standard-property=new-value | -p "extension-property[{node-specifier}]"=new-value resource
```

```
-p standard-property=new-value
```

変更する標準プロパティの名前を指定します。

```
-p "extension-property[{node-specifier}]"=new-value
```

変更する拡張プロパティの名前を指定します。

`node-specifier` は、`-p` オプションおよび `-x` オプションに対する「オプション」の修飾子です。この修飾子は、指定された1つまたは複数のノードまたはゾーン上でのみ、1つまたは複数の拡張プロパティがリソースの作成時に設定されることを示します。指定した拡張プロパティは、クラスタ内のほかのノードまたはゾーン上では、設定されません。`node-specifier` を指定しないと、クラスタ内のすべてのノードおよびゾーン上の指定された拡張プロパティが設定されます。`node-specifier` にはノード名またはノード識別子を指定できます。`node-specifier` の構文例を次に示します。

```
-p "myprop{phys-schost-1}"
```

中括弧({}) は、指定した拡張プロパティをノード `phys-schost-1` でのみ設定することを示します。大部分のシェルでは、二重引用符(")が必要です。

また次の構文を使用して、2つの異なるノード上の2つの異なるゾーン内で拡張プロパティを設定することもできます。

```
-x "myprop{phys-schost-1:zoneA,phys-schost-2:zoneB}"
```

---

注 - *node-specifier* を使用して指定する拡張プロパティは、ノードごとのプロパティとして RTR ファイルで宣言します。Per\_node リソースプロパティの属性の詳細は、[付録 B](#) を参照してください。

---

*resource*

リソースの名前を指定します。

- 4 リソースプロパティが変更されていることを確認します。

```
# clresource show -v resource
```

#### 例 2-21 標準リソースプロパティの変更

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

```
# clresource set -p start_timeout=30 resource-1
# clresource show -v resource-1
```

#### 例 2-22 拡張リソースプロパティの変更

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

```
# clresource set -p Log_level=3 resource-1
# clresource show -v resource-1
```

## ▼ 論理ホスト名リソースまたは共有アドレスリソースを変更する

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

---

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

---

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

---

注-リソースタイプのバージョンが2より前の場合、リソースを変更する前に、まず、リソースタイプをアップグレードする必要があります。詳細については、[96 ページの「事前登録されているリソースタイプのアップグレード」](#)を参照してください。

---

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。
- 2 リソースプロパティを変更します。

```
# clresource set -p CheckNameService=false resource
```

`-p CheckNameService=false`      リソースの `CheckNameService` 拡張プロパティを `false` に設定します。

`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。

## リソースの STOP\_FAILED エラーフラグの消去

`Failover_mode` リソースプロパティが `NONE` または `SOFT` に設定されている場合、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。

- 任意のノードまたはゾーン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を変更する
- リソースグループのリソースのプロパティを変更する

## ▼ 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 であるノードまたはゾーンの名前
- 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

1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。

2 STOP\_FAILED 状態のリソースと、どのノードまたはゾーンでこの状態なのかを確認します。

```
# clresource status
```

3 STOP\_FAILED 状態になっているノードまたはゾーン上で、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 リソースの STOP\_FAILED エラーフラグを消去します。

```
# clresource clear -f STOP_FAILED -n nodelist resource
```

-f STOP\_FAILED      フラグ名を指定します。

-n 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。

resource              リソースの名前を指定します。

5 **手順4**で STOP\_FAILED フラグを消去したノードまたはゾーンで、リソースグループの状態を調べます。

```
# clresourcegroup status
```

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他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. 適切なノードまたはゾーン上でリソースグループをオフラインにします。

```
# clresourcegroup offline resource-group
```

`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 次のマニュアルページを参照してください。

- `clresource(1CL)`
- `clresourcegroup(1CL)`

## Start\_failed リソース状態の消去

`Start_failed` リソース状態は、`Start` メソッドまたは `Pre-net_start` メソッドがリソース上で失敗またはタイムアウトしたが、そのリソースグループは結果的にオンラインになっていることを示します。リソースグループは、リソースが障害状態に置かれていてサービスを提供していなくても、オンライン状態になります。この状態は、リソースの `Failover_mode` プロパティに `None` またはリソースグループのフェイルオーバーを妨げる別の値が設定されている場合に発生することがあります。

`Stop_failed` リソース状態とは異なり、`Start_failed` リソース状態は、ユーザーや Sun Cluster ソフトウェアがリソースグループ上で操作を実行することを妨げません。該当リソースを再起動するコマンドを実行するだけで済みます。

この状態を消去するには、次のいずれかの手順を使用します。

## ▼ リソースグループのスイッチオーバーにより Start\_failed リソース状態を解除する

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 次の条件が満たされているか確認します。

- 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
    - リソースグループのスイッチオーバー先のノードまたはゾーンの名称
  - 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またはゾーンはクラスタ内にある。
- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。
  - 2 リソースグループを新しいノードまたはゾーンに切り替えます。

```
# clresourcegroup switch [-n node-zone-list] resource-group
```

`-n node-zone-list` このリソースグループをマスターできるノードまたはゾーンの、コマンド区切りの順序付けされたリストを指定します。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。リスト内の各エントリの形式は `node:zone` です。この形式では、`node` はノード名を指定し、`zone` は非大域 Solaris ゾーンの名前を指定します。大域ゾーンを指定する、または非大域ゾーンを持たないノードを指定するには、`node` のみを指定します。

このリストはオプションです。このリストを省略すると、そのリソースグループのノードリスト内のすべてのノード上でリソースグループが切り替えられます。

`resource-group` 切り替えるリソースグループの名前を指定します。

注- 切り替えようとしている任意のリソースグループが他のリソースグループに対して強いアフィニティを宣言している場合、その操作は失敗するか、委任されます。詳細については、[142 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。

- リソースグループが新しいノードまたはゾーンに切り替えられ、Start\_failed リソース状態が解除されたことを確認します。

```
# clresourcegroup status
```

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およびリソースグループの状態を示しています。

### 例 2-23 リソースグループのスイッチオーバーによる Start\_failed リソース状態の解除

次の例で、resource-group-1 リソースグループの rscon リソースで発生した Start\_failed リソース状態を解除する方法を示します。このコマンドは、リソースグループを大域ゾーン phys-schost-2 に切り替えることで、この状態を解除します。

- phys-schost-1 上でリソースが Start\_failed リソース状態であること確認するには、次のコマンドを実行します。

```
# clresource status
```

```
=== Cluster Resources ===
```

Resource Name	Node Name	Status	Message
rscon	phys-schost-1	Faulted	Faulted
	phys-schost-2	Offline	Offline
hastor	phys-schost-1	Online	Online
	phys-schost-2	Offline	Offline

- 切り替えを実行するには、次のコマンドを実行します。

```
# clresourcegroup switch -n phys-schost-2 resource-group-1
```

- リソースグループが phys-schost-2 上でオンラインに切り替えられ、Start\_failed リソース状態が解除されたことを確認するには、次のコマンドを実行します。

```
# clresource status
```

```
=== Cluster Resources ===
```

Resource Name	Node Name	Status	Message
rscon	phys-schost-1	Offline	Offline
	phys-schost-2	Online	Online
hastor	phys-schost-1	Online	Online

phys-schost-2

Offline

Offline

参照 `clresourcegroup(1CL)` のマニュアルページ。

## ▼ リソースグループの再起動により Start\_failed リソース状態を解除する

---

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 次の条件が満たされているか確認します。

- 次の情報を持っている。
  - 再起動するリソースグループの名前
  - リソースグループの再起動を行うノードの名前
- 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またはゾーンはクラスタノードである。

1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。

2 リソースグループを再起動します。

```
# clresourcegroup restart -n node resource-group
```

`-n node` 再起動するリソースグループのノードの名前を指定します。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。

`resource-group` 再起動するリソースグループの名前を指定します。

3 リソースグループが新しいノード上で再起動され、Start\_failed リソース状態が解除されたことを確認します。

```
# clresourcegroup status
```

このコマンドからの出力は、再起動されたリソースおよびリソースグループの状態を示しています。

## 例 2-24 リソースグループの再起動による Start\_failed リソース状態の解除

次の例で、resource-group-1 リソースグループの rscon リソースで発生した Start\_failed リソース状態を解除する方法を示します。このコマンドは、リソースグループを大域ゾーン phys-schost-2 上で再起動することで、この状態を解除します。

1. phys-schost-1 上でリソースが Start\_failed リソース状態であること確認するには、次のコマンドを実行します。

```
# clresource status

=== Cluster Resources ===

Resource Name          Node Name              Status                 Message
-----
rscon                  phys-schost-1          Faulted                Faulted
                      phys-schost-2          Offline                Offline

hastor                 phys-schost-1          Online                 Online
                      phys-schost-2          Offline                Offline
```

2. このリソースを再起動するには、次のコマンドを使用します。

```
# clresourcegroup restart -n phys-schost-1 -g resource-group-1
```

3. リソースグループが phys-schost-1 上で再起動され、Start\_failed リソース状態が解除されたことを確認するには、次のコマンドを実行します。

```
# clresource status

=== Cluster Resources ===

Resource Name          Node Name              Status                 Message
-----
rscon                  phys-schost-1          Offline                Offline
rscon                  phys-schost-2          Online                 Online

hastor                 phys-schost-1          Online                 Online
hastor                 phys-schost-2          Offline                Offline
```

参照 `clresourcegroup(1CL)` のマニュアルページ。

## ▼ リソースの無効化および有効化によりリソース状態 Start\_failed を解除する

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 有効または無効にするリソースの名前を確認します。

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC`の承認を提供する役割になります。
- 2 リソースを無効にしてから有効にします。

```
# clresource disable resource
# clresource enable resource
```

`resource` リソースの名前を指定します。

- 3 リソースが無効になってから有効になり、Start\_failed リソース状態が解除されたことを確認します。

```
# clresource status
```

このコマンドからの出力は、無効にしてから再度有効にしたリソースの状態を示します。

### 例 2-25 リソースの無効化および有効化によるリソース状態 Start\_failed の解除

次の例で、リソースを無効にしてから有効にすることで、`rscon` リソースで発生した Start\_failed リソース状態を解除する方法を示します。

1. リソースが Start\_failed リソース状態であることを確認するには、次のコマンドを実行します。

```
# clresource status
```

```
=== Cluster Resources ===
```

Resource Name	Node Name	Status	Message
rscon	phys-schost-1	Faulted	Faulted
	phys-schost-2	Offline	Offline
hastor	phys-schost-1	Online	Online
	phys-schost-2	Offline	Offline

- リソースを無効にしてから再度有効にするには、次のコマンドを実行します。

```
# clresource disable rscon
# clresource enable rscon
```

- リソースが再度有効にされ、Start\_failed リソース状態が解除されたこと確認するには、次のコマンドを実行します。

```
# clresource status
```

```
=== Cluster Resources ===
```

Resource Name	Node Name	Status	Message
rscon	phys-schost-1	Online	Online
	phys-schost-2	Offline	Offline
hastor	phys-schost-1	Online	Online
	phys-schost-2	Offline	Offline

参照 `clresource(1CL)` のマニュアルページ。

## 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、[35 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。

リソース型	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。

- `clresourcetype list`
- `clresourcetype list -v`

例 2-26 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。

```
# clresourcetype register SUNW.LogicalHostname:2
```

## リソースタイプの既存インスタンスの移行に関する情報

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

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

例2-27 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。

```
# clresource set -p CheckNameService=false -p Type_version=2 lhostrs
```

## 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、[96 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

- リソースタイプを再登録します。

```
# clresourcetype register SUNW.resource-type
```

*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2-28 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。

```
# clresourcetype register SUNW.LogicalHostname
```

参照 `clresourcetype(1CL)` 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またはゾーンを構成する
- リソースグループからのノードまたはゾーンの削除

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

スケ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、`clresourcegroup(1CL)` 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

ノードまたはゾーン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。

- 100 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」
- 101 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 ID およびゾーンの名前
- ノードまたはゾーンが追加されるリソースグループの名前
- すべてのノードまたはゾーン上のリソースグループによって使用されるネットワークリソースをホストする IP ネットワークマルチパスグループの名前

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

## ▼ スケーラブルリソースグループにノードを追加する

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

詳細は、以下の作業の[手順 1](#)から[手順 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またはゾーンのリスト (`nodelist` リソースグループプロパティ) に新しいノードまたはゾーンを追加します。

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。

```
# clresourcegroup set [-n node-zone-list] resource-group
```

`-n node-zone-list` このリソースグループをマスターできるノードまたはゾーンの、コマンド区切りの順序付けされたリストを指定します。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。リスト内の各エントリの形式は `node:zone` です。この形式では、`node` はノード名を指定し、`zone` は非大域 Solaris ゾーンの名前を指定します。大域ゾーンを指定する、または非大域ゾーンを持たないノードを指定するには、`node` のみを指定します。

このリストはオプションです。このリストを省略すると、`Nodelist` プロパティがクラスタ内のすべてのノードに対して設定されます。

`resource-group` ノードまたはゾーンが追加されるリソースグループの名前を指定します。

- 3 (省略可能) スケーラブルリソースの `Load_balancing_weights` プロパティを更新し、リソースグループに追加するノードまたはゾーン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、`clresourcegroup(1CL)` のマニュアルページを参照してください。

## ▼ フェイルオーバーリソースグループにノードを追加する

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

```
# clresourcegroup show -v resource-group | grep -i nodelist
# clresourcegroup show -v 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 `clnode show -v | grep -i node-id` を実行してください。

---

- 2 ノードまたはゾーン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。

```
# clresource set -p netiflist=netiflist network-resource
```

`-pnetiflist=netiflist`      各ノードまたはゾーン上の IP ネットワークマルチパスグループをコマンド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。

`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。

- 3 `HASStoragePlus AffinityOn` 拡張プロパティが `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 `clsetup` 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 `clsetup` ユーティリティーを起動します。

```
# clsetup
```

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. 「デバイスグループとボリューム」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. プロンプトに回答し、**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またはゾーンを含めるように、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またはゾーンをここに含める必要があります。

```
# clresourcegroup set [-n node-zone-list] resource-group
```

-n *node-zone-list* このリソースグループをマスターできるゾーンの、コンマ区切りの順序付けされたリストを指定します。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。リスト内の各エントリの形式は `node:zone` です。この形式では、`node` はノード名を指定し、`zone` は非大域 Solaris ゾーンの名前を指定します。大域ゾーンを指定する、または非大域ゾーンを持たないノードを指定するには、`node` のみを指定します。

このリストはオプションです。このリストを省略すると、`NodeList` プロパティーがクラスタ内のすべてのノードに対して設定されます。

*resource-group* ノードまたはゾーンが追加されるリソースグループの名前を指定します。

- 5 更新された情報を確認します。

```
# clresourcegroup show -vresource-group | grep -i nodelist
```

```
# clresourcegroup show -vresource-group | grep -i netiflist
```

## 例 2-29 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) に大域ゾーン (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。

```
# clresourcegroup show -v resource-group-1 | grep -i nodelist
( Nodelist:      phys-schost-1 phys-schost-3
# clresourcegroup show -v resource-group-1 | grep -i netiflist
( Res property name: NetIfList
  Res property class: extension
  List of IP ネットワークマルチパス
  interfaces on each node
  Res property type: stringarray
  Res property value: sc_ipmp0@1 sc_ipmp0@3

(Only nodes 1 and 3 have been assigned IP ネットワークマルチパス groups.
You must add an IP ネットワークマルチパス group for node 2.)

# clresource set -p netiflist=sc_ipmp0@1,sc_ipmp0@2,sc_ipmp0@3 schost-2
# metaset -s red -a -h phys-schost-2
# clresourcegroup set -n phys-schost-1,phys-schost-2,phys-schost-3 resource-group-1
# clresourcegroup show -v resource-group-1 | grep -i nodelist
  Nodelist:      phys-schost-1 phys-schost-2
                  phys-schost-3
# clresourcegroup show -v resource-group-1 | grep -i netiflist
  Res 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

ノードまたはゾーン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。

- 104 ページの「スケラブルリソースグループからノードを削除する」
- 105 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」
- 107 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

```
# clnode show -v | grep -i "Node ID"
```

- ノードまたはゾーンを削除する予定である1つまたは複数のリソースグループの名前

```
# clresourcegroup show | grep "NodeList"
```

- すべてのノードまたはゾーン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

```
# clresourcegroup show -v | grep "NetIfList.*value"
```

さらに、削除するノードまたはゾーン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。削除するノードまたはゾーン上でマスターされている場合は、`clresourcegroup` コマンドを実行し、そのノードまたはゾーン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 `clresourcegroup` コマンドは、指定されたノードまたはゾーン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` にこのノードまたはゾーンが含まれていてはなりません。

```
# clresourcegroup switch -n new-masters resource-group
```

`-n 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またはゾーンを指定します。

`resource-group` 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、削除するノードまたはゾーン上でマスターされます。

詳細は、`clresourcegroup(1CL)` 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またはゾーンを削除する場合で、スケラブルサービス構成を使用するときは、最初にスケラブルリソースグループからそのノードまたはゾーン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またはゾーンを削除してください。

## ▼ スケラブルリソースグループからノードを削除する

スケラブルサービスは、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、スケラブルサービスリソースを含むスケラブルグループです。
- もう1つは、スケ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。

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

スケーラブルサービス構成の詳細は、『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またはゾーンを削除すると、そのスケーラブルサービスはそのノードまたはゾーン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またはゾーンを削除するには、以下の作業を行なってください。

- 1 スケーラブルリソースグループをマスターできるノードまたはゾーンのリスト (nodelist リソースグループプロパティ) からノードまたはゾーンを削除します。

```
# clresourcegroup set [-n node-zone-list] scalable-resource-group
```

*-n node-zone-list*

このリソースグループをマスターできるゾーンの、コンマ区切りの順序付けされたリストを指定します。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。リスト内の各エントリの形式は *node:zone* です。この形式では、*node* はノード名を指定し、*zone* は非大域 Solaris ゾーンの名前を指定します。大域ゾーンを指定する、または非大域ゾーンを持たないノードを指定するには、*node* のみを指定します。

このリストはオプションです。このリストを省略すると、NodeList プロパティがクラスタ内のすべてのノードに対して設定されます。

*scalable-resource-group*

ノードまたはゾーンが削除されるリソースグループの名前を指定します。

- 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またはゾーンを削除します。

詳細については、107 ページの「共有アドレスリソースを含むフェイルオーバーリソースグループからノードを削除する」を参照してください。

- 3 (省略可能) スケーラブルリソースの *Load\_balancing\_weights* プロパティを更新し、リソースグループから削除するノードまたはゾーンのウェイトを削除します。

参照 `clresourcegroup(1CL)` 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またはゾーンを削除するには、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またはゾーンを削除する場合で、スケラブルサービス構成を使用するときは、最初にスケラブルリソースグループからそのノードまたはゾーン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またはゾーン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、107 ページの「共有アドレスリソースを含むフェイルオーバーリソースグループからノードを削除する」を参照してください。

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

この手順はノードまたはゾーン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またはゾーンをここに含める必要があります。

```
# clresourcegroup set [-n node-zone-list] failover-resource-group
```

*-n node-zone-list*

このリソースグループをマスターできるゾーンの、コンマ区切りの順序付けされたリストを指定します。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。リスト内の各エントリの形式は *node:zone* です。この形式では、*node* はノード名を指定し、*zone* は非大域 Solaris ゾーンの名前を指定します。大域ゾーンを指定する、または非大域ゾーンを持たないノードを指定するには、*node* のみを指定します。

このリストはオプションです。このリストを省略すると、NodeList プロパティがクラスタ内のすべてのノードに対して設定されます。

*failover-resource-group*

ノードまたはゾーンが削除されるリソースグループの名前を指定します。

- 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。

```
# clresourcegroup show -v failover-resource-group | grep -i netiflist
```

- 3 ノードまたはゾーン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。

```
# clresource set -p netiflist=netiflist network-resource
```

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 `clnode show -v | grep -i "Node ID"` を実行してください。

`-pnetiflist=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マンド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。

`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。

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。

```
# clresourcegroup show -vfailover-resource-group | grep -i nodelist
# clresourcegroup show -vfailover-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、ノードまたはゾーン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またはゾーンを削除するには、[105 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またはゾーンを削除すると、そのノードまたはゾーン上の共有アドレスリソースを継続して使用し、スケ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` にそのノードまたはゾーンを追加する必要があります。`auxnodeList` にノードまたはゾーンを追加するには、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` からノードまたはゾーンを削除するためにも使用できます。 `auxnodelist` からノードまたはゾーンを削除するには、共有アドレスリソースを削除して作成し直す必要があります。

---

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

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

```
# clressharedaddress create -g failover-resource-group \  
-X new-auxnodelist shared-address
```

`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

`new-auxnodelist` 妥当なノードまたはゾーンの追加または削除によって変更された新しい `auxnodelist`

`shared-address` 共有アドレスの名前

## 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。

```
# clresourcegroup show -v resource-group-1 | grep -i nodelist  
Nodelist:          phys-schost-1 phys-schost-2  
                  phys-schost-3  
  
# clresourcegroup set -n phys-schost-1,phys-schost-2 resource-group-1  
# clresourcegroup show -v resource-group-1 | grep -i netiflist  
( Res property name: NetIfList  
Res property class: extension  
( List of IP ネットワークマルチパス  
interfaces on each node  
( Res property type: stringarray  
  Res property value: sc_ipmp0@1 sc_ipmp0@2  
                    sc_ipmp0@3  
  
(sc_ipmp0@3 is the IP ネットワークマルチパス group to be removed.)  
  
# clresource set -p netiflist=sc_ipmp0@1,sc_ipmp0@2 schost-1  
# clresourcegroup show -v resource-group-1 | grep -i nodelist
```

```

Nodelist:      phys-schost-1 phys-schost-2
# clresourcegroup show -v resource-group-1 | grep -i netiflist
Res property value: sc_ipmp0@1 sc_ipmp0@2

```

## 大域ゾーンから非大域ゾーンへのアプリケーションの移行

アプリケーションリソースを大域ゾーンから非大域ゾーンに移行することができます。

---

注- 移行するデータサービスはスケラブルであり、また非大域ゾーンでサポートされる必要があります。

---

### ▼ 大域ゾーンから非大域ゾーンへアプリケーションを移行する

この手順では、3つの各ノード上に作成された非大域ゾーンを持つ3つのノードクラスがあると仮定しています。HAStoragePlus リソースを使用することで高可用性を実現している構成ディレクトリも、非大域ゾーンからアクセス可能であるべきです。

- 1 スケラブルリソースが使用する共有アドレスを保持する大域ゾーンを使用してフェイルオーバーリソースグループを作成します。

```
# clresourcegroup create -n node1,node2,node3 sa-resource-group
```

*sa-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。

- 2 共有アドレスリソースをフェイルオーバーリソースグループに追加します。

```
# clressharedaddress create -g sa-resource-group -h hostnamelist, ... \
[-X auxnodelist] -N netiflist network-resource
```

*-g sa-resource-group* リソースグループの名前を指定します。共有アドレスリソースのノードリストでは、同一ノード上では複数のゾーンを指定しないでください。共有アドレスリソースのノードリストは、同一ノード上で異なるゾーンを指定してはいけません。*nodename:zonename* のペアの同一リストをスケラブルリソースグループのノードリストとして指定します。

*-h 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主ノードとして使用されない)を識別する物理ノード名またはIDまたはゾーンをコマンド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。補助ノードリストが明示的に指定されていない場合、リストのデフォルトは、共有アドレスリソースを含むリソースグループのノードリストには含まれていない、すべてのクラスタノード名のリストになります。

---

注-サービスをマスターするために作成されたすべての非大域ゾーン内でスケーラブルサービスを動作させるには、共有アドレスリソースグループのノードリスト、または共有アドレスリソースの *auxnodelist* にゾーンの完全なリストを含めます。すべてのゾーンがノードリスト内にある場合は、*auxnodelist* を省略できます。

---

**-N *netiflist*** 各ノード上の IP ネットワークマルチパス グループをコマンド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。

---

*network-resource* リソース名を指定します(省略可能)。

### 3 スケーラブルリソースグループを作成します。

```
# clresourcegroup create \-p Maximum primaries=m \-p Desired primaries=n \
-n node1,node2,node3 \
-p RG_dependencies=sa-resource-group resource-group-1
```

**-p *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。

**-p *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。

*resource-group-1* 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。任意の名前の先頭文字は ASCII にする必要があります。

- 4 HASStoragePlus のリソース `hastorageplus-1` を作成し、ファイルシステムのマウントポイントを定義します。

```
# clresource create -g resource-group-1 -t SUNW.HASStoragePlus \
-p FilesystemMountPoints=/global/resource-group-1 hastorageplus-1
```

リソースは有効状態で作成されます。

- 5 アプリケーションのリソースタイプを登録します。

```
# clresourcetype register resource-type
```

`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。

- 6 アプリケーションリソースを `resource-group-1` に追加し、依存関係を `hastorageplus-1` に設定します。

```
# clresource create -g resource-group-1 -t SUNW.application \
[-p "extension-property[{node-specifier}]"=value, ...] -p Scalable=True \
-p Resource_dependencies=network-resource -p Port_list=port-number/protocol \
-p Resource_dependencies=hastorageplus-1 resource
```

- 7 フェイルオーバーリソースグループをオンラインにします。

```
# clresourcegroup online sa-resource-group
```

- 8 すべてのノード上でスケラブルリソースグループをオンラインにします。

```
# clresourcegroup online resource-group-1
```

- 9 各ノード上 (`node1,node2,node3`) で `zone1` をインストールし、起動します。

- 10 2つのノード (`node1,node2`) 上でアプリケーションリソースグループをオフラインにします。

---

注-共有アドレスを `node3` 上でオンラインにします。

---

```
# clresourcegroup switch -n node3 resource-group-1
```

`resource-group-1` 切り替えるリソースグループの名前を指定します。

- 11 フェイルオーバーリソースグループの `nodelist` プロパティを更新し、ノードリストから削除された対応するノードの非大域ゾーンを含めます。

```
# clresourcegroup set -n node1:zone1,node2:zone1,node3 sa-resource-group
```

- 12 アプリケーションリソースグループの `nodeList` プロパティを更新し、ノードリストから削除された対応するノードの非大域ゾーンを含めます。

```
# clresourcegroup set node1:zone1,node2:zone1,node3 resource-group-1
```

- 13 新しく追加されたゾーンでのみ、フェイルオーバーリソースグループとアプリケーションリソースグループをオンラインにします。

---

注 - 両方のリソースグループは、`node1:zone1` および `node2:zone1` のみでオンラインになります。

---

```
# clresourcegroup switch -n node1:zone1,node2:zone1 sa-resource-group
```

```
# clresourcegroup switch -n node1:zone1,node2:zone1 resource-group-1
```

- 14 広域ノード `node3` をリストから削除することで、両方のリソースグループの `nodeList` プロパティを更新し、`node3` の非大域ゾーンを含めます。

```
# clresourcegroup set node1:zone1,node2:zone1,node3:zone1 sa-resource-group
```

```
# clresourcegroup set node1:zone1,node2:zone1,node3:zone1 resource-group-1
```

- 15 すべての非大域ゾーン上で両方のリソースグループをオンラインにします。

```
# clresourcegroup switch -n node1:zone1,node2:zone1,node3:zone1 sa-resource-group
```

```
# clresourcegroup switch -n node1:zone1,node2:zone1,node3:zone1 resource-group-1
```

## リソースグループと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、広域デバイスとローカルおよび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、広域デバイスとローカルおよび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ローカルおよび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。

このような余分な管理作業を回避するため、`HAStoragePlus` リソースタイプを使用します。広域デバイスやローカルおよび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、HASStoragePlus 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。

- 広域デバイスとローカルおよび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

HASStoragePlus リソースを作成するには、[113 ページ](#)の「[新しいリソース用に HASStoragePlus 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 HASStoragePlus リソースタイプを設定する

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)

---

注 - Solaris ZFS (Zettabyte File System) を使用して HASStoragePlus リソースを高可用性ローカルファイルシステムとして作成するには、[125 ページ](#)の「[HASStoragePlus リソースタイプを設定し、ローカル Solaris ZFS を高可用性にする](#)」の節を参照してください。

---

新しいリソースに対し、HASStoragePlus リソースの `hastorageplus-1` を resource-group-1 に作成するには、[112 ページ](#)の「[リソースグループと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。

HASStoragePlus リソースを作成するには、[118 ページ](#)の「[高可用性ローカルファイルシステムの有効化](#)」を参照してください。

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify` および `solaris.cluster.admin` RBAC の承認を提供する役割になります。

- 2 リソースグループ resource-group-1 を作成します。

```
# clresourcegroup create 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。

```
# clresourcetype show | egrep Type
```

- 4 必要であれば、リソースタイプを登録します。

```
# clresourcetype register SUNW.HASStoragePlus
```

- 5 HASStoragePlus のリソース `hastorageplus-1` を作成し、ファイルシステムのマウントポイントと広域デバイスパスを定義します。

```
# clresource create -g resource-group-1 -t SUNW.HASStoragePlus \
-p GlobalDevicePaths=/dev/global/dsk/d5s2,dsk/d6 \
-p FilesystemMountPoints=/global/resource-group-1 hastorageplus-1
```

GlobalDevicePaths 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、dsk/d5)
- 広域デバイスへのパス (例: /dev/global/dsk/d1s2、 /dev/md/nfsdg/dsk/d10)

FilesystemMountPoints は、次の値を含むことができます。

- ローカルまたはクラスタファイルシステムのマウントポイント (例: /local-fs/nfs、/global/nfs)

---

注-HASStoragePlus は、ZFS ストレージプールの構成に使用する Zpools 拡張プロパティと、ZFS ストレージプールのデバイス検索の指定に使う、ZpoolsSearchDir 拡張プロパティを持っています。ZpoolsSearchDir 拡張プロパティのデフォルト値は、/dev/dsk です。ZpoolsSearchDir 拡張プロパティは、zpool(1M) コマンドの -d オプションの指定と似ています。

---

リソースは有効状態で作成されます。

- 6 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 `hastorageplus-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。

```
# clresource create -g resource-group-1 -t SUNW.iws \
-p Confdir_list=/global/iws/schost-1 -p Scalable=False \
-p Network_resources_used=schost-1 -p Port_list=80/tcp \
-p Resource_dependencies=hastorageplus-1 resource
```

リソースは有効状態で作成されます。

- 7 リソースの依存性を正しく構成したかを確認します。

```
# clresource show -v resource | egrep Resource_dependencies
```

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

```
# clresourcegroup online -M resource-group-1
```

## 参考 アフィニティスイッチオーバー

HASStoragePlus リソースタイプには別の拡張プロパティーである AffinityOn が含まれます。これは、GlobalDevicePaths および FileSystemMountPoints 拡張プロパティーで定義されている広域デバイスのアフィニティスイッチオーバーを HASStoragePlus が実行する必要があるかどうかを指定するブール値です。詳細は、SUNW.HASStoragePlus(5) のマニュアルページを参照してください。

---

注-スケラブルサービスの場合、AffinityOn フラグの設定は無視されます。スケラブルリソースグループでアフィニティスイッチオーバーを実行することはできません。

---

## ▼ 既存のリソース用に HASStoragePlus リソースタイプを設定する

始める前に [112 ページの「リソースグループとデバイスグループ間での起動の同期」](#) を読んでください。

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

```
# clresourcetype show | egrep Type
```

- 2 必要であれば、リソースタイプを登録します。

```
# clresourcetype register SUNW.HASStoragePlus
```

- 3 HASStoragePlus のリソース hastorageplus-1 を作成します。

```
# clresource create -g resource-group \  
-t SUNW.HASStoragePlus -p GlobalDevicePaths= ... \  
-p FileSystemMountPoints=... -p AffinityOn=True hastorageplus-1
```

リソースは有効状態で作成されます。

- 4 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。

```
# clresource set -p Resource_Dependencies=hastorageplus-1 resource
```

- 5 リソースの依存性を正しく構成したかを確認します。

```
# clresource show -v resource | egrep Resource_dependencies
```

## クラスタファイルシステム用の HAStoragePlus リソースの構成

HAStoragePlus リソースがクラスタファイルシステム用に構成され、オンラインになると、これらのファイルシステムは使用可能になります。クラスタファイルシステムは UFS (Unix File System) と VxFS (Veritas File System) でサポートされています。データサービスの入出力負荷が高い場合は、HAStoragePlus とローカルファイルシステムを併用します。HAStoragePlus リソースのファイルシステムを変更する方法については、140 ページの「HAStoragePlus リソースの広域ファイルシステムをローカルファイルシステムに変更する」を参照してください。

## クラスタファイルシステム用の /etc/vfstab のサンプルエントリ

次の例に、クラスタ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。

---

注-クラスタファイルシステム用の /etc/vfstab ファイルのエントリには、マウントオプションに `global` キーワードが含まれているべきです。

---

例 2-30 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/rdisk/d0
/global/local-fs/nfs ufs      5 yes      logging,global
```

例 2-31 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 yes      log,global
```

## ▼ クラスタファイルシステム用に HASStoragePlus リソースを設定する

- 1 クラスタ内の任意のノードで、スーパーユーザーになるか、`solaris.cluster.modify RBAC`の承認を提供する役割になります。

- 2 フェイルオーバーリソースグループを作成します。

```
# clresourcegroup create resource-group-1
```

- 3 HASStoragePlus リソースタイプを登録します。

```
# clresourcetype register SUNW.HASStoragePlus
```

- 4 HASStoragePlus リソースを作成し、ファイルシステムのマウントポイントを定義します。

```
# clresource create -g resource-group -t SUNW.HASStoragePlus \  
-p FileSystemMountPoints="mount-point-list" hasp-resource
```

リソースは有効状態で作成されます。

- 5 `resource-group-1` にデータサービスリソースを追加し、`hasp-resource` に対する依存関係を設定します。

- 6 HASStoragePlus リソースを含むリソースグループをオンラインにし、管理状態にします。

```
# clresourcegroup online -M resource-group-1
```

## ▼ クラスタファイルシステム用の HASStoragePlus リソースタイプを削除する

- クラスタファイルシステム用に構成された HASStoragePlus リソースを無効にし、削除します。

```
# clresource delete -F -g resource-group -t SUNW.HASStoragePlus resource
```

## 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、HASStoragePlus リソースタイプを使用します。

グローバルまたはローカルのどちらのファイルシステムも指定できます。グローバルファイルシステムには、クラスタのすべてのノードまたはゾーンからアクセス可能です。ローカルファイルシステムには、1つのクラスタノードまたはゾーンからアクセス可能です。SUNW.HASStoragePlus リソースによって管理されているローカルファイルシステムは、1つのクラスタノードまたはゾーンにマウントされます。これらのローカルファイルシステムでは、配下のデバイスは Sun Cluster グローバルデバイスであることが必要です。

これらのファイルシステムのマウントポイントは、`paths[,...]` という書式で定義されます。非大域ゾーンのパスと大域ゾーンのパスの両方を次の書式で指定できます。

*Non-GlobalZonePath:GlobalZonePath*

大域ゾーンのパスは省略可能です。大域ゾーンパスを指定しない場合、Sun Cluster は非大域ゾーンのパスと大域ゾーンのパスが同じであることを前提とします。

*Non-GlobalZonePath:GlobalZonePath* のようにパスを指定する場合、大域ゾーンの `/etc/vfstab` にある *GlobalZonePath* を指定してください。

このプロパティのデフォルト設定は、空のリストです。

SUNW.HASStoragePlus リソースタイプを使用すると、ファイルシステムを非大域ゾーンで利用可能にすることができます。SUNW.HASStoragePlus リソースタイプを使用してこのようにするには、大域ゾーンと非大域ゾーンにマウントポイントを作成してください。ファイルシステムを非大域ゾーンで利用可能にするために、SUNW.HASStoragePlus リソースタイプは、まず大域ゾーンにあるファイルシステムをマウントします。このリソースタイプは、次に非大域ゾーンでループバックマウントを実行します。すべてのクラスタノードおよびすべての大域ゾーンにある `/etc/vfstab` には、各ファイルシステムのマウントポイントに対応する等価なエントリが存在するはずですが、SUNW.HASStoragePlus リソースタイプは、非大域ゾーンにある `/etc/vfstab` をチェックしません。

---

注 - ローカルファイルシステムには、Unix File System (UFS)、Quick File System (QFS)、Veritas File System (VxFS)、Solaris ZFS (Zettabyte File System) などがあります。

---

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。

---

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。

---

Sun Cluster には、HAStoragePlus リソースタイプを設定してローカルファイルシステムを高可用性にするための、次のツールがあります。

- **Sun Cluster Manager**。詳細は、Sun Cluster Manager のオンラインヘルプを参照してください。
- `clsetup(1CL)` ユーティリティ。
- **Sun Cluster** の保守コマンド。

Sun Cluster Manager および `clsetup` ユーティリティを使用すると、対話形式でリソースをリソースグループに追加できます。これらのリソースを対話的に使うことにより、コマンドの構文エラーまたは脱落による設定エラーが起きる可能性が少なくなります。Sun Cluster Manager および `clsetup` ユーティリティは、すべての必須リソースが作成され、リソース間のすべての必須依存関係が設定されるようにします。

## 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` 拡張プロパティを `True` に設定します。

HAStoragePlus リソースの `Zpools` 拡張プロパティは、`AffinityOn` 拡張プロパティを無視します。

- 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。

- ブロックデバイスの場合: /dev/global/dsk/dDsS
- raw デバイスの場合: /dev/global/rdisk/dDsS

これらのデバイス名の置換可能な要素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。

---

注 - Solaris ZFS (Zettabyte File System) は、/etc/vfstab ファイルを使用しません。

---

例 2-32 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 2-33 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。

例 2-33 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 (続き)

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

例 2-34 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
```

## ▼ clsetup ユーティリティーを使用することで HAStoragePlus リソースタイプを設定する

次の手順では、clsetup ユーティリティーを使用することで HAStoragePlus リソースタイプを設定する方法を説明します。この手順は、任意のクラスタノードから実行します。

この手順では、Sun Cluster の保守コマンドの長い形式を使用します。多くのコマンドには短縮形もあります。コマンド名の形式を除き、コマンドは同じです。コマンドのリストとその短縮形については、[付録 A](#) を参照してください。

始める前に 次の前提条件を満たしていることを確認します。

- 必要なボリューム、ディスクグループおよびファイルシステムが作成されている。

1 任意のクラスタノードでスーパーユーザーになります。

2 `clsetup` ユーティリティを起動します。

# `clsetup`

`clsetup` のメインメニューが表示されます。

3 データサービスのオプションに対応する番号を入力し、**Return** キーを押します。  
「データサービス」メニューが表示されます。

4 ファイルシステムを構成するためのオプションに対応する番号を入力し、**Return** キーを押します。

`clsetup` ユーティリティは、この作業を実行するための前提条件のリストを表示します。

5 前提条件が満たされていることを確認し、**Return** キーを押して続けます。

`clsetup` ユーティリティは、高可用性 `HASStoragePlus` リソースをマスターできるクラスタノードまたはゾーンのリストを表示します。

6 高可用性 `HASStoragePlus` リソースをマスターできるノードまたはゾーンを選択します。

- 任意の順序で並んでいる一覧表示されたすべてのノードのデフォルト選択をそのまま使用するには、`a` と入力し、**Return** キーを押します。

- 一覧表示されたノードまたはゾーンのサブセットを選択するには、ノードに対応する番号のコンマ区切りまたはスペース区切りのリストを入力します。続いて、**Return** キーを押します。

`HASStoragePlus` リソースグループのノードリストにノードが表示される順序でノードが一覧表示されていることを確認します。リストの最初のノードは、このリソースグループの主ノードです。

- 特定の順序ですべてのノードを選択するには、ノードに対応する番号のコンマ区切りまたはスペース区切りの順序付きリストを入力し、**Return** キーを押します。

7 ノードの選択を確認するには、`d` を入力して、**Return** キーを押します。

`clsetup` ユーティリティは、データが格納される共有ストレージタイプの種類のリストを表示します。

- 8 データの格納に使用する共有ストレージの種類に対応する番号を入力し、**Return** キーを押します。  
clsetup ユーティリティは、クラスタ内で構成されているファイルシステムのマウントポイントを表示します。既存のマウントポイントが存在しない場合は、clsetup ユーティリティで新しいマウントポイントを定義できます。
- 9 デフォルトのマウントディレクトリ、**raw** デバイスのパス、Global Mount オプション、および Check File System Periodically オプションを指定して、**Return** キーを押します。  
clsetup ユーティリティは、ユーティリティが作成するマウントポイントのプロパティを返します。
- 10 マウントポイントを作成するには、**d**を入力し、**Return** キーを押します。  
clsetup ユーティリティは、使用可能なファイルシステムのマウントポイントを表示します。

---

注-c オプションを使用すると、新しい別のマウントポイントを定義できます。

---
- 11 ファイルシステムのマウントポイントを選択します。
  - 任意の順序で並んでいる一覧表示されたすべてのファイルシステムのマウントポイントのデフォルト選択をそのまま使用するには、**a**と入力し、**Return** キーを押します。
  - 一覧表示されたファイルシステムのマウントポイントのサブセットを選択するには、ファイルシステムのマウントポイントに対応する番号の、コンマまたはスペースで区切られたリストを入力し、**Return** キーを押します。
- 12 ノードの選択を確認するには、**d**を入力して、**Return** キーを押します。  
clsetup ユーティリティは、クラスタ内で構成されている広域ディスクセットとデバイスグループを表示します。
- 13 広域デバイスグループを選択します。
  - 任意の順序で並んでいる一覧表示されたすべてのデバイスグループのデフォルト選択をそのまま使用するには、**a**と入力し、**Return** キーを押します。
  - 一覧表示されたデバイスグループのサブセットを選択するには、デバイスグループに対応する番号の、コンマまたはスペースで区切られたリストを入力し、**Return** キーを押します。

- 14 ノードの選択を確認するには、**d**を入力して、**Return** キーを押します。  
clsetup ユーティリティーは、このユーティリティーが作成する Sun Cluster オブジェクトの名前を表示します。
- 15 Sun Cluster オブジェクトに別の名前が必要である場合、次のように名前を変更します。
  - a. 変更する名前に対応する番号を入力し、**Return** キーを押します。  
clsetup ユーティリティーは、新しい名前を指定できる画面を表示します。
  - b. 「新しい値」プロンプトで、新しい名前を入力し、**Return** キーを押します。  
clsetup ユーティリティーは、このユーティリティーが作成する Sun Cluster オブジェクトの名前のリストに戻ります。
- 16 Sun Cluster オブジェクト名の選択を確認するには、**d**を入力して、**Return** キーを押します。  
clsetup ユーティリティーは、このユーティリティーが作成する Sun Cluster の構成に関する情報を表示します。
- 17 構成を作成するには、**c**を入力し、**Return** キーを押します。  
clsetup ユーティリティーは、構成を作成するためにこのユーティリティーがコマンドを実行していることを示す進行状況のメッセージを表示します。構成が完了した時点で、clsetup ユーティリティーは、構成を作成するためにユーティリティーが実行したコマンドを表示します。
- 18 (省略可能) clsetup ユーティリティーが終了するまで、繰り返し **q** を入力し、**Return** キーを押します。  
必要に応じて、ほかの必要な作業を実行している間、clsetup ユーティリティーを動作させたままにし、そのあとでユーティリティーを再度使用することができます。clsetup を終了する場合、ユーザーがユーティリティーを再起動する際に、ユーティリティーは既存のリソースグループを認識します。
- 19 HAStoragePlus リソースが作成されたことを確認します。  
このためには、clresource(1CL) ユーティリティーを使用します。デフォルトでは、clsetup ユーティリティーは、リソースグループに名前 `node_name-rg` を割り当てます。  

```
# clresource show node_name-rg
```

## ▼ HAStoragePlus リソースタイプを設定し、ローカル Solaris ZFS を高可用性にする

ローカル Solaris ZFS (Zettabyte File System) を高可用性にするには、主に次の作業を行います。

- ZFS ストレージプールを作成する。
- ZFS ストレージプール内に ZFS を作成する。
- ZFS ストレージプールを管理する HAStoragePlus リソースを設定する。

この節では両方の作業を完了する方法を説明します。

### 1 ZFS ストレージプールを作成する。



注意 - 構成済みの定足数デバイスは、ZFS ストレージプールに追加しないでください。構成済みの定足数デバイスをストレージプールに追加すると、ディスクは EFI ディスクとしてラベルが変更され、また定足数構成情報が失われ、ディスクはクラスタへの定足数投票を提供しなくなります。ディスクがストレージプールにある場合、そのディスクを定足数デバイスとして構成できます。または、ディスクの定足数デバイス構成を解除し、ディスクをストレージプールに追加した後に、そのディスクを定足数デバイスとして再構成することができます。

Sun Cluster 構成で ZFS ストレージプールを作成する際には、次の必要条件を確認します。

- ZFS ストレージプールの作成元であるすべてのデバイスが、クラスタ内のすべてのノードからアクセス可能であることを確認します。これらのノードは、HAStoragePlus リソースが属するリソースグループのノードリストで構成します。
- `zpool(1M)` コマンドに対して指定した Solaris デバイス識別子 (`/dev/dsk/c0t0d0` など) が `cldevice list -v` コマンドで認識できることを確認します。

注 - ZFS ストレージプールは、ディスク全体またはディスクスライスを使用して作成できます。ディスクの書き込みキャッシュを有効にして ZFS の性能が向上するように Solaris 論理デバイスを指定し、ディスク全体を使用した ZFS ストレージプールを作成することをお勧めします。完全なディスクが提供されている場合、ZFS は EFI を使用してディスクにラベルを付けます。

ZFS ストレージプールの作成方法の詳細は、「『Solaris ZFS Administration Guide』」の『「Creating a ZFS Storage Pool」』を参照してください。

- ### 2 作成した ZFS とストレージプール内で、ZFS を作成します。
- 同一の ZFS ストレージプール内で複数の ZFS を作成できます。

注-HAStoragePlusは、ZFS ボリューム上に作成されたファイルシステムをサポートしていません。

ZFSはFilesystemMountPoints 拡張プロパティには配置しないでください。

ZFS ストレージプール内での ZFS の作成方法の詳細は、『『Solaris ZFS Administration Guide』』の「『Creating a ZFS File System Hierarchy』」を参照してください。

- 3 クラスタ内の任意のノードで、スーパーユーザーになるか、`solaris.cluster.modify RBAC`の承認を提供する役割になります。

- 4 フェイルオーバーリソースグループを作成します。

```
# clresourcegroup create resource-group
```

- 5 HAStoragePlus リソースタイプを登録します。

```
# clresourcetype register SUNW.HAStoragePlus
```

- 6 ローカルZFS用のHAStoragePlus リソースを作成します。

```
# clresource create -g resource-group -t SUNW.HAStoragePlus \
-p Zpools=zpool -p ZpoolsSearchDir=/dev/did/dsk \
resource
```

ZFS ストレージプールのデバイスのデフォルトの検索場所は、`/dev/dsk`です。これは、`ZpoolsSearchDir` 拡張プロパティを使用して上書きできます。

リソースは有効状態で作成されます。

- 7 HAStoragePlus リソースを含むリソースグループをオンラインにし、管理状態にします。

```
# clresourcegroup online -M resource-group
```

### 例 2-35 HAStoragePlus リソースタイプを設定してローカルZFSを高可用性にする

次の例では、ローカルZFSを高可用性にするためのコマンドを示します。

```
phys-schost-1% su
Password:
# cldevice list -v
```

DID Device	Full Device Path
-----	-----
d1	phys-schost-1:/dev/rdisk/c0t0d0
d2	phys-schost-1:/dev/rdisk/c0t1d0

```
d3          phys-schost-1:/dev/rdisk/c1t8d0
d3          phys-schost-2:/dev/rdisk/c1t8d0
d4          phys-schost-1:/dev/rdisk/c1t9d0
d4          phys-schost-2:/dev/rdisk/c1t9d0
d5          phys-schost-1:/dev/rdisk/c1t10d0
d5          phys-schost-2:/dev/rdisk/c1t10d0
d6          phys-schost-1:/dev/rdisk/c1t11d0
d6          phys-schost-2:/dev/rdisk/c1t11d0
d7          phys-schost-2:/dev/rdisk/c0t0d0
d8          phys-schost-2:/dev/rdisk/c0t1d0
```

*you can create a ZFS storage pool using a disk slice by specifying a Solaris device identifier:*

```
# zpool create HAZpool c1t8d0s2
  or you can create a ZFS storage pool using disk slice by specifying a logical device identifier
# zpool create HAZpool /dev/did/dsk/d3s2
# zfs create HAZpool/export
# zfs create HAZpool/export/home
# clresourcegroup create hasp-rg
# clresourcetype register SUNW.HASStoragePlus
# clresource create -g hasp-rg -t SUNW.HASStoragePlus \
  -p Zpools=HAZpool hasp-rs
# clresourcegroup online -M hasp-rg
```

## ▼ ローカル **Solaris ZFS** を高可用性にしている HASStoragePlus リソースを削除する

- ローカル Solaris ZFS (Zettabyte File System) を高可用性にしている HASStoragePlus リソースを無効にし、削除します。

```
# clresource delete -F -g resource-group -t SUNW.HASStoragePlus resource
```

## HASStorage から HASStoragePlus へのアップグレード

HASStorage は、Sun Cluster ソフトウェアの現在のリリースではサポートされていません。同等の機能が HASStoragePlus でサポートされています。HASStorage から HASStoragePlus へアップグレードするには、次の節を参照してください。

---

注 - HASStorage がサポートされなくなったため、HASStorage リソースが構成されているリソースグループは STOP\_FAILED 状態になります。リソースの ERROR\_STOP\_FAILED フラグを消去し、HASStorage を HASStoragePlus にアップグレードするための手順に従ってください。

---

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

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

- 1 リソースグループ nfs1-rg をオフラインにします。  
# clresourcegroup offline nfs1-rg
- 2 HASStorage 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。  
# clresource set -p Resource\_Dependencies="" nfsserver-rs
- 3 HASStorage リソースを無効にします。  
# clresource disable nfs1storage-rs
- 4 アプリケーションリソースグループから HASStorage リソースを削除します。  
# clresource delete nfs1storage-rs
- 5 HASStorage リソースタイプの登録を解除します。  
# clresourcetype unregister SUNW.HASStorage
- 6 HASStoragePlus リソースタイプを登録します。  
# clresourcetype register SUNW.HASStoragePlus
- 7 HASStoragePlus リソースを作成します。

---

注 - HASStorage の ServicePaths プロパティを使用する代わりに、HASStoragePlus の FilesystemMountPoints プロパティまたは GlobalDevicePaths プロパティ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。  
# clresource create -g nfs1-rg -t \  
SUNW.HASStoragePlus -p FilesystemMountPoints=/global/nfsdata -p \  
AffinityOn=True nfs1-hastp-rs

- 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。

```
# clresource create -g nfs1-rg -t \
  SUNW.HASStoragePlus -p GlobalDevicePaths=nfsdg -p AffinityOn=True nfs1-hastp-rs
```

リソースは有効状態で作成されます。

- 8 アプリケーションサーバーリソースを無効にします。
 

```
# clresource disable nfsserver-rs
```
- 9 nfs1-rg グループをクラスタノード上でオンラインにします。
 

```
# clresourcegroup online nfs1-rg
```
- 10 アプリケーションサーバーとHASStoragePlus との間の依存性を設定します。
 

```
# clresource set -p Resource_dependencies=nfs1-hastp-rs nfsserver-rs
```
- 11 nfs1-rg グループをクラスタノード上でオンラインにします。
 

```
# clresourcegroup online -eM nfs1-rg
```

## ▼ CFS による HASStorage から高可用性ローカルファイルシステムによる HASStoragePlus へアップグレードする

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

- 1 HASStorage 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。
 

```
# clresource set -p Resource_Dependencies="" nfsserver-rs
```
- 2 HASStorage リソースを無効にします。
 

```
# clresource disable nfs1storage-rs
```
- 3 アプリケーションリソースグループから HASStorage リソースを削除します。
 

```
# clresource delete nfs1storage-rs
```
- 4 HASStorage リソースタイプの登録を解除します。
 

```
# clresourcetype unregister SUNW.HASStorage
```

- 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 プロパティを使用する代わりに、HAStoragePlus の FilesystemMountPoints プロパティまたは GlobalDevicePaths プロパティ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。

```
# clresource create -g nfs1-rg -t \  
SUNW.HAStoragePlus -p FilesystemMountPoints=/global/nfsdata -p \  
AffinityOn=True nfs1-hastp-rs
```

- 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。

```
# clresource create -g nfs1-rg -t \  
SUNW.HAStoragePlus -p GlobalDevicePaths=nfsdgm -p AffinityOn=True nfs1-hastp-rs
```

リソースは有効状態で作成されます。

- 7 アプリケーションサーバーリソースを無効にします。

```
# clresource disable nfsserver-rs
```

- 8 nfs1-rg グループをクラスタノード上でオンラインにします。

```
# clresourcegroup online nfs1-rg
```

- 9 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。

```
# clresource set -p Resource_dependencies=nfs1-hastp-rs nfsserver-rs
```

- 10 nfs1-rg グループをクラスタノード上でオンラインにします。

```
# clresourcegroup online -eM nfs1-rg
```

## 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に変更する

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に変更します。

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に変更するには、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 ソフトウェアでは、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。

---

## ▼ Solaris ZFS 以外のファイルシステムをオンラインの `HAStoragePlus` リソースに追加する

ローカルまたは広域ファイルシステムを `HAStoragePlus` リソースに追加する場合、`HAStoragePlus` リソースは自動的にファイルシステムをマウントします。

- 1 クラスタの1つのノードで、スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。
- 2 クラスタの各ノードの `/etc/vfstab` ファイルにおいて、追加しようとしている各ファイルシステムのマウントポイント用のエントリを追加します。  
エントリごとに、`mount at boot` フィールドと `mount options` フィールドを次のように設定します。
  - ローカルファイルシステムの場合
    - `mount at boot` フィールドを `no` に設定します。
    - `global` フラグを削除します。
  - クラスタファイルシステムの場合
    - 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 `HASStoragePlus` リソースを指定します。

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

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

- `HASStoragePlus` 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
- `HASStoragePlus` 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

```
# clresource set -p FileSystemMountPoints="mount-point-list" hasp-resource
```

```
-p FileSystemMountPoints="mount-point-list"
```

`HASStoragePlus` 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。リスト内の各エントリの形式は、`LocalZonePath:GlobalZonePath` です。この形式では、大域パスはオプションです。大域パスが指定されていない場合、大域パスはローカルパスと同じになります。

*hasp-resource*

ファイルシステムを追加する先の `HASStoragePlus` リソースを指定します。

- 5 `HASStoragePlus` 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 `HASStoragePlus` リソースを指定します。

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

- 6 `HASStoragePlus` 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。

`HASStoragePlus` 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、`HASStoragePlus` によるファイルシステムのマウントは失敗します。

```
# clresource status hasp-resource
```

**例 2-36** 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/fs 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/local-fs/fs です。

この例では、各クラスターノード上の /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/fs
# clresource set \
-p FileSystemMountPoints="/global/global-fs/fs,/global/local-fs/fs"
# scha_resource_get -O extension -R rshasp -G rghasp FileSystemMountPoints rshasp
STRINGARRAY
/global/global-fs/fs
/global/local-fs/fs
# clresource status rshasp
```

```
=== Cluster Resources ===
```

Resource Name	Node Name	Status	Message
rshasp	node46	Offline	Offline
	node47	Online	Online

## ▼ オンラインの HAStoragePlus リソースから **Solaris ZFS** 以外の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、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 リソースは広域ファイルシステムをアンマウントしません。



注意 - オンラインの HASStoragePlus 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 HASStoragePlus リソースからファイルシステムを削除すると、そのファイルシステムは強制的にアンマウントされます。アプリケーションが使用しているファイルシステムが強制的にアンマウントされると、そのアプリケーションは異常終了またはハングする可能性があります。

- 1 クラスタの1つのノードで、スーパーユーザーになるか、`solaris.cluster.modify RBAC`の承認を提供する役割になります。
- 2 HASStoragePlus 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 HASStoragePlus リソースを指定します。

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

- 3 HASStoragePlus リソースの `FileSystemMountPoints` 拡張プロパティを変更して、HASStoragePlus 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。

```
# clresource set -p FileSystemMountPoints="mount-point-list" hasp-resource
```

```
-p FileSystemMountPoints="mount-point-list"
```

HASStoragePlus 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はなりません。

```
hasp-resource
```

ファイルシステムを削除する元の HASStoragePlus リソースを指定します。

- 4 HASStoragePlus 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 HASStoragePlus リソースを指定します。

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

- 5 HASStoragePlus 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。

HASStoragePlus 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、HASStoragePlus によるファイルシステムのアンマウントは失敗します。

```
# clresource status hasp-resource
```

- 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 2-37 オンラインの HASStoragePlus 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 HASStoragePlus 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。

- HASStoragePlus 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 HASStoragePlus 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/fs
  - /global/local-fs/fs
- 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/local-fs/fs です。

```
# scha_resource_get -O extension -R rshasp -G rghasp FileSystemMountPoints
STRINGARRAY
/global/global-fs/fs
/global/local-fs/fs
# clresource set -p FileSystemMountPoints="/global/global-fs/fs"
# scha_resource_get -O extension -R rshasp -G rghasp FileSystemMountPoints rshasp
STRINGARRAY
/global/global-fs/fs
# clresource status rshasp
```

```
=== Cluster Resources ===
```

Resource Name	Node Name	Status	Message
rshasp	node46	Offline	Offline
	node47	Online	Online

## ▼ Solaris ZFS ストレージプールをオンラインの HAStoragePlus リソースに追加する

Solaris ZFS (Zettabyte File System) ストレージプールをオンラインの HAStoragePlus リソースに追加する場合、HAStoragePlus リソースは次の処理を行います。

- ZFS ストレージプールをインポートする。
- ZFS ストレージプール内のすべてのファイルシステムをマウントする。

- 1 クラスタ内の任意のノードで、スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。

- 2 HAStoragePlus リソースがすでに管理している ZFS ストレージプールを判別します。

```
# clresource show -g hasp-resource-group -p Zpools hasp-resource
```

`-g hasp-resource-group` HAStoragePlus リソースを含むリソースグループを指定します。

`hasp-resource` ZFS ストレージプールを追加する先の HAStoragePlus リソースを指定します。

- 3 HAStoragePlus リソースがすでに管理している ZFS ストレージプールの既存のリストに、新しい ZFS ストレージプールを追加します。

```
# clresource set -p Zpools="zpools-list" hasp-resource
```

`-p Zpools="zpools-list"` HAStoragePlus リソースがすでに管理している既存の ZFS ストレージプール名のコンマ区切りリストと、追加する新しい ZFS ストレージプール名を指定します。

`hasp-resource` ZFS ストレージプールを追加する先の HAStoragePlus リソースを指定します。

- 4 HAStoragePlus リソースが管理する ZFS ストレージプールの新しいリストと、[手順 2](#) で生成したリストを比較します。

```
# clresource show -g hasp-resource-group -p Zpools hasp-resource
```

`-g hasp-resource-group` HAStoragePlus リソースを含むリソースグループを指定します。

`hasp-resource` ZFS ストレージプールの追加先である HAStoragePlus リソースを指定します。

- 5 HASStoragePlus 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。

HASStoragePlus リソースがオンラインで障害が発生した場合は、リソースの検証自体は成功したことになります。ただし、HASStoragePlus リソースによる ZFS のインポートとマウントの試みは失敗しています。この場合、以前の一連の手順を繰り返す必要があります。

```
# clresourcegroup status hasp-resource
```

## ▼ オンラインの HASStoragePlus リソースから Solaris ZFS ストレージプールを削除する

オンラインの HASStoragePlus リソースから Solaris ZFS (Zettabyte File System) ストレージプールを削除する場合、HASStoragePlus リソースは次の処理を行います。

- ZFS ストレージプール内のファイルシステムをアンマウントする。
- ノードから ZFS ストレージプールをエクスポートする。

- 1 クラスタ内の任意のノードで、スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。

- 2 HASStoragePlus リソースがすでに管理している ZFS ストレージプールを判別します。

```
# clresource show -g hasp-resource-group -p Zpools hasp-resource
```

`-g hasp-resource-group` HASStoragePlus リソースを含むリソースグループを指定します。

`hasp-resource` ZFS ストレージプールの削除元である HASStoragePlus リソースを指定します。

- 3 HASStoragePlus リソースが現在管理している ZFS ストレージプールのリストから ZFS ストレージプールを削除します。

```
# clresource set -p Zpools="zpools-list" hasp-resource
```

`-p Zpools="zpools-list"` HASStoragePlus リソースが現在管理している ZFS ストレージプール名のコンマ区切りリストから、削除する ZFS ストレージプール名を除いたものを指定します。

`hasp-resource` ZFS ストレージプールの削除元である HASStoragePlus リソースを指定します。

- 4 HASStoragePlus リソースが現在管理する ZFS ストレージプールの新しいリストと、[手順 2](#) で生成したリストを比較します。

```
# clresource show -g hasp-resource-group -p Zpools hasp-resource
```

`-g hasp-resource-group`      HAStoragePlus リソースを含むリソースグループを指定します。

`hasp-resource`                ZFS ストレージプールの削除元である HAStoragePlus リソースを指定します。

- 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障害が発生した場合は、リソースの検証自体は成功したことになります。ただし、HAStoragePlus リソースによる ZFS のアンマウントとエクスポートの試みは失敗しています。この場合、以前の一連の手順を繰り返す必要があります。

```
# clresourcegroup status SUNW.HAStoragePlus +
```

## ▼ HAStoragePlus リソースの FileSystemMountPoints プロパティを変更したあと障害から回復する

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

- 1 変更が失敗した原因となる障害を特定します。

```
# clresource status hasp-resource
```

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

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

- 2 変更が失敗した原因となる障害を修正します。

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

```
# clresource set -p FileSystemMountPoints="mount-point-list" hasp-resource
```

```
-p 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マンドで区切って指定します。

```
hasp-resource
```

変更しようとしている HASToragePlus リソースを指定します。

- 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。

```
# clresource status
```

### 例 2-38 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。

```
# clresource status
```

```
=== Cluster Resources ===
```

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

## ▼ HASToragePlus リソースの zpools プロパティを変更したあと障害から回復する

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

- 1 変更が失敗した原因となる障害を特定します。

```
# clresource status hasp-resource
```

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。

- ZFS のプール *zpool* がインポートに失敗した。
- ZFS のプール *zpool* がエクスポートに失敗した。

- 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 zpools 拡張プロパティを変更する手順を繰り返します。

```
# clresource set -p Zpools="zpools-list" hasp-resource
```

```
-p Zpools="zpools-list"    HAStoragePlus が現在管理している ZFS ストレージプール名
                           のコンマ区切りリストから、削除する ZFS ストレージプー
                           ル名を除いたものを指定します。

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

```
# clresource status
```

### 例 2-39 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。ZFS のプール `zpool` がインポートに失敗したため、このリソースには障害が発生しています。

```
# clresource status hasp-resource
```

```
=== Cluster Resources ===
```

Resource Name	Node Name	Status	Status Message
hasp-resource	node46	Online	Faulted - Failed to import:hazpool
	node47	Offline	Offline

## HAStoragePlus リソースの広域ファイルシステムからローカルファイルシステムへの変更

HAStoragePlus リソースのファイルシステムを、広域ファイルシステムからローカルファイルシステムに変更できます。

### ▼ HAStoragePlus リソースの広域ファイルシステムをローカルファイルシステムに変更する

- 1 フェイルオーバーリソースグループをオフラインにします。

```
# clresourcegroup offline resource-group
```

- 2 HAStoragePlus リソースを表示します。

```
# clresource show -g resource-group -t SUNW.HAStoragePlus
```

- 3 各リソースのマウントポイントのリストを取得します。  

```
# clresource show -p FilesystemMountPoints hastorageplus-resource
```
- 4 広域ファイルシステムをアンマウントします。  

```
# umount mount-points
```
- 5 リソースグループのノードリストで構成されているすべてのノード上で、マウントポイントの /etc/vfstab エントリを変更します。

- マウントオプションから global キーワードを削除します。
- mount at boot オプションを yes から no に変更します。

リソースグループで構成されているすべての HAStoragePlus リソースのすべてのクラスタファイルシステムに対して手順を繰り返します。

- 6 リソースグループをオンラインにします。  

```
# clresourcegroup online -M resource-group
```

## 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に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、[35 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
4	3.2
6	3.2.2/08

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。

- `clresourcetype list`
- `clresourcetype list -v`

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

## リソースタイプの既存インスタンスの移行に関する情報

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

- 移行はいつでも実行できます。
- HAStoragePlus リソースタイプの新機能を使用する場合は、`Type_version` プロパティに設定する必要がある値は 4 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およびゾーン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、要求されている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およびゾーン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、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またはゾーンを制限します。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。

**-p `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。

---

<操作> 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

オペレータ	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またはゾーン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またはゾーン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またはゾーン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またはゾーン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またはゾーン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またはゾーン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またはゾーン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またはゾーン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任されます。詳細については、150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。

そのほかのリソースグループの現在の状態によっては、任意のノードまたはゾーン上で、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またはゾーン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またはゾーン上で動作する必要があります。

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

```
# clresourcegroup set|create -p RG_affinities=++target-rg source-rg
```

*source-rg*

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

```
-p RG_affinities=++target-rg
```

強い肯定的なアフィニティのターゲットであるリソースグループを指定します。このリソースグループは、強い肯定的なアフィニティ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。ターゲットのリソースグループが別のノードに再配置された場合、ソースのリソースグループは自動的にターゲットと同じノードに切り替わります。しかし、強い肯定的なアフィニティを宣言しているソースのリソースグループは、ターゲットのリソースグループが動作していないノードまたはゾーン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またはゾーン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またはゾーンにフェイルオーバーします。

---

たとえば、リソースグループ `rg1` にリソースグループ `rg2` に対する強い肯定的なアフィニティが宣言されていると仮定します。`rg2` が別のノードまたはゾーンにフェイルオーバーすると、`rg1` もそのノードまたはゾーン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またはゾーンに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のターゲットをオンラインにしたときに、強い肯定的なアフィニティのソースがすべてのノード上でオフラインになっている場合があります。このような場合、強い肯定的なアフィニティのソースは、ターゲットと同じノード上で自動的にオンラインになります。

たとえば、リソースグループ rg1 にリソースグループ rg2 に対する強い肯定的なアフィニティが宣言されていると仮定します。最初は両方のリソースグループともすべてのノード上でオフラインです。管理者があるノード上で rg2 をオンラインにすると、rg1 は自動的に同じノード上でオンラインになります。

`clresourcegroup suspend` コマンドを使用すると、強いアフィニティまたはクラスタ再構成によりリソースグループが自動的にオンラインになるのを防止できます。

強い肯定的なアフィニティを宣言しているリソースグループをフェイルオーバーする必要がある場合、そのフェイルオーバーは委託する必要があります。詳細については、150 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 2-40** あるリソースグループと別のリソースグループを強制的に同じ場所に配置する

この例では、リソースグループ rg1 を変更して、リソースグループ rg2 に対する強い肯定的なアフィニティを宣言するためのコマンドを示します。このアフィニティを宣言すると、rg1 は rg2 が動作しているノードまたはゾーン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。

```
# clresourcegroup set -p RG_affinities=++rg2 rg1
```

## 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、同じノードまたはゾーン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またはゾーン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またはゾーン上で動作してもかまいません。なぜなら、リソースグループのフェイルオーバーの増加よりも効率の低下のほうが被害が小さいためです。

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

```
# clresourcegroup set|create -p RG_affinities==+target-rg source-rg
```

*source-rg*

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

```
-p RG_affinities==+target-rg
```

弱い肯定的なアフィニティのターゲットであるリソースグループを指定します。このリソースグループは、弱い肯定的なアフィニティを宣言する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またはゾーンで動作する確率が上がります。弱い肯定的なアフィニティのソースは、まず、そのアフィニティのターゲットがすでに動作しているノードまたはゾーン上でオンラインになろうとします。しかし、弱い肯定的なアフィニティのソースは、そのアフィニティのターゲットがリソースモニターによってフェイルオーバーされても、フェイルオーバーしません。同様に、弱い肯定的なアフィニティのソースは、そのアフィニティのターゲットがスイッチオーバーされても、フェイルオーバーしません。どちらの状況でも、ソースがすでに動作しているノードまたはゾーン上では、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またはゾーン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またはゾーン上で再起動されます。

---

例 2-41 あるリソースグループと別のリソースグループをできる限り同じ場所に配置する

この例では、リソースグループ `rg1` を変更して、リソースグループ `rg2` に対する弱い肯定的なアフィニティを宣言するためのコマンドを示します。このアフィニティを宣言すると、`rg1` と `rg2` はまず、同じノードまたはゾーン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またはゾーン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。

```
# clresourcegroup set -p RG_affinities==+rg2 rg1
```

## 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。

```
# clresourcegroup set|create -p RG_affinities=neg-affinity-list source-rg
```

*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。

```
-p 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、弱い否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 2-42 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。

```
# clresourcegroup set -p RG_affinities=-rg2,-rg3,-rg4 rg1
# clresourcegroup set -p RG_affinities=-rg1,-rg3,-rg4 rg2
# clresourcegroup set -p RG_affinities=-rg1,-rg2,-rg4 rg3
# clresourcegroup set -p RG_affinities=-rg1,-rg2,-rg3 rg4
```

## 重要なサービスに優先権を指定する

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

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

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

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

```
# clresourcegroup set|create -p RG_affinities=--critical-rg noncritical-rg
```

*noncritical-rg*

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

```
-p RG_affinities=--critical-rg
```

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

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

強い否定的なアフィニティのターゲットをオフラインにした場合、強い否定的なアフィニティのソースは、すべてのノード上でオフラインになる場合があります。このような状況では、強い否定的なアフィニティのソースは自動的にオンラインになります。通常、ノードリストのノードの順序および宣言されたアフィニティに基づいて、リソースグループは最も優先されるノード上でオンラインになります。

たとえば、リソースグループ `rg1` にリソースグループ `rg2` に対する強い否定的なアフィニティが宣言されていると仮定します。最初はリソースグループ `rg1` がすべてのノード上でオフラインになりますが、リソースグループ `rg2` は1つのノード上でオンラインになります。管理者が `rg2` をオフラインにすると、`rg1` は自動的にオンラインになります。

`clresourcegroup suspend` コマンドを使用すると、強いアフィニティーまたはクラスタ再構成により強い否定的なアフィニティーのソースが自動的にオンラインになるのを防止できます。

例2-43 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。

```
# clresourcegroup set -p RG_affinities=--mcdbrg ncrng1 ncrng2
```

## 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。

```
# clresourcegroup set|create source-rg -p RG_affinities=+++target-rg
```

*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。

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

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

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

**例 2-44** リソースグループのフェイルオーバーまたはスイッチオーバーを委託する

この例では、リソースグループ `rg1` を変更して、リソースグループ `rg2` に対するフェイルオーバー委託付きの強い肯定的なアフィニティを宣言するためのコマンドを示します。このアフィニティ関係の結果、`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。

```
# clresourcegroup set -p RG_affinities=+++rg2 rg1
```

## リソースグループ間のアフィニティの組み合わせ

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

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

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

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

#### 例 2-45 リソースグループ間のアフィニティの組み合わせ

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

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

この例では、リソースグループはアフィニティを次のように宣言します。

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

この例では、両方のリソースグループが存在していると仮定します。

```
# clresourcegroup set -p RG_affinities==rep-rg app-rg
# clresourcegroup set -p RG_affinities--app-rg rep-rg
```

## 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。

- 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 にログオ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。リソース型がローカルクラスタに存在しない場合、このスクリプトはエラーを書き込んで終了します。もう一度スクリプトを実行する前に、存在しないリソース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。

- 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。

- 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。
- このスクリプトをnode1から実行して、構成データをアップグレードします。

このスクリプトは、スクリプトが生成されたクラスタとローカルクラスタの特性を比較します。これらの特性が同じでない場合、このスクリプトはエラーを書き込んで終了します。次に、`-f` オプションを使用してスクリプトを実行し直すかどうかをたずねるメッセージが表示されます。`-f` オプションを使用した場合、上記のような特性の違いを無視して、スクリプトを強制的に実行します。`-f` オプションを使用した場合、クラスタ内に不整合がないことを確認します。

このスクリプトは、Sun Cluster リソースタイプがローカルクラスタ上に存在することを確認します。リソース型がローカルクラスタに存在しない場合、このスクリプトはエラーを書き込んで終了します。もう一度スクリプトを実行する前に、存在しないリソース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 上で Solaris SMF サービスを有効にする

SMF (Service Management Facility) を使用すると、ノードの起動中またはサービス障害中に自動的に SMF サービスを起動および再起動することができます。SMF は、単一ホスト上の SMF サービスに、ある程度の高可用性を実現します。この機能は、クラスタアプリケーションに高可用性とスケラビリティを実現する、Sun Cluster Resource Group Manager (RGM) に似ています。SMF サービスと RGM の機能は相互に補完的です。

Sun Cluster には、3つの新しい SMF プロキシリソースタイプが含まれています。これらを使用すると、フェイルオーバー、マルチマスター、またはスケラブル構成の Sun Cluster とともに SMF サービスが実行できるようになります。プロキシリソースタイプは次のとおりです。

- SUNW.Proxy\_SMF\_failover
- SUNW.Proxy\_SMF\_multimaster
- SUNW.Proxy\_SMF\_scalable

SMF プロキシリソースタイプを使用すると、相互関係のある SMF サービスのセットを1つのリソースにカプセル化し、SMF プロキシリソースを Sun Cluster で管理することができます。この機能では、SMF は1つのノード上の SMF サービスの可用性を管理します。Sun Cluster は、SMF サービスの、クラスタ全体にわたる高い可用性とスケラビリティを提供します。

SMF プロキシリソースタイプを使用すると、独自の SMF の制御によるサービスを Sun Cluster に統合できます。これらのサービスには、ユーザーがコールバックメソッドやサービスマニフェストを書き換えることなく、クラスタ全体のサービス可用性が与えられます。SMF サービスを SMF プロキシリソースに統合したあとは、SMF サービスはデフォルトの再起動プログラムにより管理されなくなります。Sun Cluster により委任された再起動プログラムが、SMF サービスを管理します。

SMF プロキシリソースはほかのリソースと同じで、使用法に制限はありません。たとえば、SMF プロキシリソースは、ほかのリソースとともにリソースグループにグループ化することができます。SMF プロキシリソースは、ほかのリソースと同じように作成、管理することができます。SMF プロキシリソースは、1 点のみほかのリソースとは異なります。SMF プロキシリソースタイプのリソースを作成する場合、拡張プロパティー `Proxied_service_instances` を指定する必要があります。SMF リソースによってプロキシされる SMF サービスに関する情報を含めます。拡張プロパティーの値は、プロキシされるすべての SMF サービスを含むファイルへのパスです。ファイル内の各行は 1 つの SMF サービス専用で、`svc_fmri` および対応するサービスマニフェストファイルのパスを指定します。

たとえば、リソースが 2 つのサービス、`restarter_svc_test_1:default` と `restarter_svc_test_2:default` を管理する必要がある場合、ファイルには次に示す 2 行が含まれているはずです。

```
<svc:/system/cluster/restarter_svc_test_1:default>,\n</var/svc/manifest/system/cluster/restarter_svc_test_1.xml>
```

```
<svc:/system/cluster/restarter_svc_test_2:default>,\n</var/svc/manifest/system/cluster/restarter_svc_test_2.xml>
```

SMF プロキシリソースの下でカプセル化されたサービスは、大域ゾーンまたは非大域ゾーンに存在できます。ただし、同じプロキシリソースの下のすべてのサービスは同じゾーン内に配置します。



注意 - SMF `svcadm` は、プロキシリソース内にカプセル化される SMF サービスを有効または無効にするためには使用しないでください。プロキシリソースにカプセル化される SMF サービス (SMF リポジトリ内) のプロパティーは変更しないでください。

- 157 ページの「SMF サービスのフェイルオーバープロキシリソース構成へのカプセル化」
- 159 ページの「SMF サービスのマルチマスタープロキシリソース構成へのカプセル化」
- 162 ページの「SMF サービスのスケラブルプロキシリソース構成へのカプセル化」

## ▼ SMF サービスのフェイルオーバープロキシリソース構成へのカプセル化

フェイルオーバー構成の詳細については、43 ページの「リソースグループの作成」を参照してください。

---

注- この手順は、任意のクラスタノードから実行します。

---

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC` の承認を提供する役割になります。

- 2 プロキシ SMF フェイルオーバーリソースタイプを登録します。

```
# clresourcetype register -f\  
/opt/SUNWscsmf/etc/SUNW.Proxy_SMF_failover SUNW.Proxy_SMF_failover
```

- 3 登録されたプロキシリソースタイプを確認します。

```
# clresourcetype show
```

- 4 SMF フェイルオーバーリソースグループを作成します。

```
# clresourcegroup create [-n node-zone-list] resource-group
```

`-n node-zone-list` このリソースグループをマスターできるゾーンの、コンマ区切りの順序付けされたリストを指定します。リスト内の各エントリの形式は `node:zone` です。この形式では、`node` はノード名を指定し、`zone` は非大域 Solaris ゾーンの名前を指定します。大域ゾーンを指定する、または非大域ゾーンを持たないノードを指定するには、`node` のみを指定します。

このリストはオプションです。このリストを省略すると、クラスタノードのすべての大域ゾーン上でリソースグループが構成されます。

---

注- 最高の可用性を実現するには、同一ノード上の異なるゾーンではなく、SMF フェイルオーバーリソースグループのノードリストの異なるノード上でゾーンを指定します。

---

`resource-group` 追加するスケラブル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。

- 5 SMF リソースグループが作成されていることを確認します。

```
# clresourcegroup status resource-group
```

- 6 **SMF** 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。

```
# clresource create -g resource-group -t SUNW.Proxy_SMF_failover \
[-p "extension-property[{node-specifier}]"=value, ...] [-p standard-property=value, ...] resource
```

リソースは有効状態で作成されます。

- 7 **SMF** 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。

```
# clresource show resource
```

- 8 フェイルオーバーリソースグループをオンラインにします。

```
# clresourcegroup online -M +
```

#### 例 2-46 SMF プロキシフェイルオーバーリソースタイプの登録

次の例では、SUNW.Proxy\_SMF\_failover リソースタイプを登録します。

```
# clresourcetype register SUNW.Proxy_SMF_failover
# clresourcetype show SUNW.Proxy_SMF_failover
```

Resource Type:	SUNW.Proxy_SMF_failover
RT_description:	Resource type for proxying failover SMF services
RT_version:	3.2
API_version:	6
RT_basedir:	/opt/SUNWscsmf/bin
Single_instance:	False
Proxy:	False
Init_nodes:	All potential masters
Installed_nodes:	<All>
Failover:	True
Pkglist:	SUNWscsmf
RT_system:	False
Global_zone:	False

#### 例 2-47 SMF プロキシ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

次の例に、プロキシリソースタイプ SUNW.Proxy\_SMF\_failover のリソースグループ resource-group-1 への追加を示します。

```
# clresource create -g resource-group-1 -t SUNW.Proxy_SMF_failover
-x proxied_service_instances=/var/tmp/svslist.txt resource-1
# clresource show resource-1
```

```
=== Resources ===
```

```
Resource:                resource-1
Type:                    SUNW.Proxy_SMF_failover
Type_version:            3.2
Group:                   resource-group-1
R_description:
Resource_project_name:   default
Enabled{phats1}:         True
Monitored{phats1}:       True
```

## ▼ SMF サービスのマルチマスタープロキシリソース構成へのカプセル化

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC`の承認を提供する役割になります。

- 2 SMF プロキシマルチマスターリソースタイプを登録します。

```
# clresourcetype register -f\
/opt/SUNWscsmf/etc/SUNW.Proxy_SMF_multimaster SUNW.Proxy_SMF_multimaster
```

- 3 SMF マルチマスターリソースグループを作成します。

```
# clresourcegroup create\-p Maximum primaries=m\-p Desired primaries=n\
[-n node-zone-list]\
resource-group
```

`-p 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。

`-p 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。

`-n node-zone-list` このリソースグループが使用可能となる、コンマ区切りの順序付けられたリストを指定します。リスト内の各エントリの形式は `node:zone` です。この形式では、`node` はノード名を指定し、`zone` は非大域 Solaris ゾーンの名前を指定します。大域ゾーンを指定する、または非大域ゾーンを持たないノードを指定するには、`node` のみを指定します。

このリストはオプションです。このリストを省略すると、クラスタノードの大域ゾーン上でリソースグループが構成されます。

*resource-group* 追加するスケラブル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。

- 4 **SMF** プロキシマルチマスターリソースグループが作成されたことを確認します。

```
# clresourcegroup show resource-group
```

- 5 **SMF** プロキシマルチマスターリソースをリソースグループに追加します。

```
# clresource create -g resource-group -t SUNW.Proxy_SMF_multimaster\  
[-p "extension-property[{node-specifier}]"=value, ...] [-p standard-property=value, ...] resource
```

```
-g resource-group
```

以前に作成したスケラブルサービスリソースグループの名前を指定します。

```
-p "extension-property[{node-specifier}]"=value, ...
```

リソース用に設定する拡張プロパティ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はリソースタイプに依存します。どの拡張プロパティ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。

*node-specifier* は、`-p` オプションおよび `-x` オプションに対する「オプション」の修飾子です。この修飾子は、指定された1つまたは複数のノードまたはゾーン上でのみ、1つまたは複数の拡張プロパティがリソースの作成時に設定されることを示します。指定した拡張プロパティは、クラスタ内のほかのノードまたはゾーン上では、設定されません。*node-specifier* を指定しないと、クラスタ内のすべてのノードおよびゾーン上の指定された拡張プロパティが設定されます。*node-specifier* にはノード名またはノード識別子を指定できます。*node-specifier* の構文例を次に示します。

```
-p "myprop{phys-schost-1}"
```

中括弧 ({} ) は、指定した拡張プロパティをノード `phys-schost-1` でのみ設定することを示します。大部分のシェルでは、二重引用符 (") が必要です。

また次の構文を使用して、2つの異なるノード上の2つの異なるゾーン内で拡張プロパティを設定することもできます。

```
-x "myprop{phys-schost-1:zoneA,phys-schost-2:zoneB}"
```

```
-p standard-property=value, ...
```

リソース用に設定する標準プロパティ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はリソースタイプに依存します。スケラブルサービスの場合、通常は `Port_list`、`Load_balancing_weights`、および `Load_balancing_policy` プロパティを設定します。どの標準プロパティ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 B を参照してください。

*resource*

追加するリソースの名前を指定します。

リソースは有効状態で作成されます。

- 6 **SMF** プロキシマルチマスタ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。

```
# clresource show resource
```

- 7 マルチマスターリソースグループをオンラインにします。

```
# clresourcegroup online -M +
```

## 例 2-48 SMF プロキシマルチマスターリソースタイプの登録

次の例では、SUNW.Proxy\_SMF\_multimaster リソースタイプを登録します。

```
# clresourcetype register SUNW.Proxy_SMF_multimaster
# clresourcetype show SUNW.Proxy_SMF_multimaster
```

```
Resource Type:          SUNW.Proxy_SMF_multimaster
RT_description:         Resource type for proxying multimastered SMF services
RT_version:             3.2
API_version:            6
RT_basedir:             /opt/SUNWscsmf/bin
Single_instance:        False
Proxy:                  False
Init_nodes:             All potential masters
Installed_nodes:        <All>
Failover:               True
Pkglist:                SUNWscsmf
RT_system:              False
Global_zone:            False
```

## 例 2-49 SMF プロキシマルチマスターアプリケーションリソースの作成とリソースグループへの追加

次の例に、マルチマスタープロキシリソースタイプ SUN.Proxy\_SMF\_multimaster の作成とリソースグループ resource-group-1 への追加を示します。

```
# clresourcegroup create\
-p Maximum primaries=2\
-p Desired primaries=2\
-n phys-schost-1, phys-schost-2\
resource-group-1
# clresourcegroup show resource-group-1
```

```

=== Resource Groups and Resources ===

Resource Group:                resource-group-1
RG_description:                <NULL>
RG_mode:                       multimastered
RG_state:                      Unmanaged
RG_project_name:              default
RG_affinities:                <NULL>
Auto_start_on_new_cluster:    True
Failback:                     False
Nodelist:                     phys-schost-1 phys-schost-2
Maximum primaries:            2
Desired primaries:            2
Implicit_network_dependencies: True
Global_resources_used:        <All>
Pingpong_interval:            3600
Pathprefix:                   <NULL>
RG_System:                    False
Suspend_automatic_recovery:    False

# clresource create -g resource-group-1 -t SUNW.Proxy_SMF_multimaster
-x proxied_service_instances=/var/tmp/svslist.txt resource-1
# clresource show resource-1

=== Resources ===

Resource:                      resource-1
Type:                          SUNW.Proxy_SMF_multimaster
Type_version:                  3.2
Group:                         resource-group-1
R_description:
Resource_project_name:        default
Enabled{phats1}:              True
Monitored{phats1}:            True

```

## ▼ SMF サービスのスケラブルプロキシリソース構成へのカプセル化

スケラブル構成の詳細については、[46 ページ](#)の「[スケラブルリソースグループを作成する](#)」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。

---

- 1 クラスタメンバーで、スーパーユーザーになるか、`solaris.cluster.modify RBAC`の承認を提供する役割になります。

- 2 SMF プロキシスケラブルリソースタイプを登録します。

```
# clresourcetype register -f\  
/opt/SUNWscsmf/etc/SUNW.Proxy_SMF_scalable SUNW.Proxy_SMF_scalable
```

- 3 スケラブルリソースグループが使用する共有アドレスを保持する SMF フェイルオーバーリソースグループを作成します。フェイルオーバーリソースグループの作成については、44 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」を参照してください。

- 4 SMF プロキシスケラブルリソースグループを作成します。

```
# clresourcegroup create\ -p Maximum primaries=m\ -p Desired primaries=n\  
-p RG_dependencies=depend-resource-group\  
[-n node-zone-list\  
resource-group
```

`-p 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。

`-p 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。

`-p 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。

`-n node-zone-list`

このリソースグループが使用可能となる、コンマ区切りの順序付けられたリストを指定します。リスト内の各エントリの形式は `node:zone` です。この形式では、`node` はノード名を指定し、`zone` は非大域 Solaris ゾーンの名前を指定します。大域ゾーンを指定する、または非大域ゾーンを持たないノードを指定するには、`node` のみを指定します。

このリストはオプションです。このリストを省略すると、クラス内のすべてのノード上でリソースグループが作成されます。

スケラブルリソースのノードリストは、共有アドレスリソースのノードリストと同じリストまたは

*nodename:zonename* ペアのサブセットを含むことができます。

*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。任意の名前の先頭文字は ASCII にする必要があります。

- 5 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。

```
# clresourcegroup show resource-group
```

- 6 SMF プロキシスケーラブルリソースをリソースグループに追加します。

```
# clresource create-g resource-group -t SUNW.Proxy_SMF_scalable \  
-p Network_resources_used=network-resource[,network-resource...] \  
-p Scalable=True \  
[-p "extension-property[{node-specifier}]"=value, ...] [-p standard-property=value, ...] resource
```

```
-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。

```
-p 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。

```
-p 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。

```
-p "extension-property[{node-specifier}]"=value, ...
```

リソースの拡張プロパティを設定していることを指定します。どの拡張プロパティ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。

*node-specifier* は、`-p` オプションおよび `-x` オプションに対する「オプション」の修飾子です。この修飾子は、指定された1つまたは複数のノードまたはゾーン上でのみ、1つまたは複数の拡張プロパティがリソースの作成時に設定されることを示します。指定した拡張プロパティは、クラスタ内のほかのノードまたはゾーン上では、設定されません。*node-specifier* を指定しないと、クラスタ内のすべてのノードおよびゾーン上の指定された拡張プロパティが設定されます。*node-specifier* にはノード名またはノード識別子を指定できます。*node-specifier* の構文例を次に示します。

```
-p "myprop{phys-schost-1}"
```

中括弧 ({} ) は、指定した拡張プロパティをノード `phys-schost-1` でのみ設定することを示します。大部分のシェルでは、二重引用符 (") が必要です。

また次の構文を使用して、2つの異なるノード上の2つの異なるゾーン内で拡張プロパティを設定することもできます。

```
-x "myprop{phys-schost-1:zoneA,phys-schost-2:zoneB}"
```

```
-p standard-property=value,...
```

リソース用に設定する標準プロパティ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はリソースタイプに依存します。スケラブルサービスの場合、通常は `Port_list`、`Load_balancing_weights`、および `Load_balancing_policy` プロパティを設定します。どの標準プロパティ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 B を参照してください。

*resource*

追加するリソースの名前を指定します。

リソースは有効状態で作成されます。

- 7 **SMF** プロキシスケ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。

```
# clresource show resource
```

- 8 **SMF** プロキシスケラブルリソースグループをオンラインにします。

```
# clresourcegroup online -M +
```

## 例 2-50 SMF プロキシスケラブルリソースタイプの登録

次の例では、`SUNW.Proxy_SMF_scalable` リソースタイプを登録します。

```
# clresourcetype register SUNW.Proxy_SMF_scalable
# clresourcetype show SUNW.Proxy_SMF_scalable
```

```
Resource Type:          SUNW.Proxy_SMF_scalable
RT_description:         Resource type for proxying scalable SMF services
RT_version:             3.2
API_version:            6
RT_basedir:             /opt/SUNWscsmf/bin
Single_instance:       False
Proxy:                  False
Init_nodes:             All potential masters
Installed_nodes:       <All>
Failover:               True
Pkglist:                SUNWscsmf
RT_system:              False
Global_zone:           False
```

**例 2-51** SMF プロキシスケーラブルアプリケーションリソースの作成とリソースグループへの追加

この例では、スケーラブルプロキシリソースタイプ `SUN.Proxy_SMF_scalable` の作成と `resource-group-1` への追加を示します。

```
# clresourcegroup create\  
-p Maximum_primaries=2\  
-p Desired_primaries=2\  
-p RG_dependencies=resource-group-2\  
-n phys-schost-1, phys-schost-2\  
resource-group-1  
# clresourcegroup show resource-group-1  
  
=== Resource Groups and Resources ===  
  
Resource Group:                resource-group-1  
RG_description:                 <NULL>  
RG_mode:                       Scalable  
RG_state:                      Unmanaged  
RG_project_name:               default  
RG_affinities:                 <NULL>  
Auto_start_on_new_cluster:     True  
Failback:                     False  
Nodelist:                      phys-schost-1 phys-schost-2  
Maximum_primaries:             2  
Desired_primaries:             2  
RG_dependencies:               resource-group2  
Implicit_network_dependencies:  True  
Global_resources_used:         <All>  
Pingpong_interval:            3600  
Pathprefix:                   <NULL>  
RG_System:                    False  
Suspend_automatic_recovery:    False  
  
# clresource create -g resource-group-1 -t SUNW.Proxy_SMF_scalable  
-x proxied_service_instances=/var/tmp/svslist.txt resource-1  
# clresource show resource-1  
  
=== Resources ===  
  
Resource:                      resource-1  
Type:                          SUNW.Proxy_SMF_scalable  
Type_version:                  3.2  
Group:                         resource-group-1  
R_description:                   
Resource_project_name:         default  
Enabled{phats1}:              True
```

Monitored{phats1}:

True

## 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、このリソースのシステムプロパティと拡張プロパティによって制御されます。事前に設定された障害モニターの動作は、これらのプロパティのデフォルト値に基づいています。現在の動作は、ほとんどの Sun Cluster システムに適しているはずですが、したがって、障害モニターを調整するのは、事前に設定されたこの動作を変更したい場合「だけに」留めるべきです。

障害モニターを調整するには、次の作業が含まれます。

- 障害モニターの検証間隔を設定する。
- 障害モニターの検証タイムアウトを設定する。
- 継続的な障害とみなす基準を定義する。
- リソースのフェイルオーバー動作を指定する

これらの作業は、データサービスの登録と構成の際に行います。詳細は、データサービスのマニュアルを参照してください。

---

注-リソースの障害モニターは、そのリソースを含むリソースグループをオンラインにしたときに起動されます。障害モニターを明示的に起動する必要はありません。

---

## 障害モニターの検証間隔の設定

リソースが正しく動作しているかどうかを判断するには、障害モニターで当該リソースを定期的に検証します。障害モニターの検証間隔は、リソースの可用性とシステムの性能に次のような影響を及ぼします。

- 障害モニターの検証間隔は、障害の検出とその障害への対応にどの程度の時間がかかるかに影響を与えます。したがって、障害モニターの検証間隔を短くすると、障害の検出とその障害への対応にかかる時間も短くなります。このような時間の短縮は、リソースの可用性が向上することを意味します。

- 障害モニターの検証では、プロセッササイクルやメモリなどのシステムリソースが使用されます。したがって、障害モニターの検証間隔を短くすると、システムの性能は低下します。

さらに、障害モニターの最適な検証間隔は、リソースの障害への対応にどの程度の時間が必要かによって異なります。この時間は、リソースの複雑さが、リソースの再起動などの操作にかかる時間にどのような影響を及ぼすかに依存します。

障害モニターの検証間隔を設定するには、リソースの `Thorough_probe_interval` システムプロパティを必要な間隔 (秒単位) に設定します。

## 障害モニターの検証タイムアウトの設定

障害モニターの検証タイムアウトでは、検証に対するリソースからの応答にどのくらいの時間を許すかを指定します。このタイムアウト内にリソースからの応答がないと、障害モニターは、このリソースに障害があるものとみなします。障害モニターの検証に対するリソースの応答にどの程度の時間がかかるかは、障害モニターがこの検証に使用する操作によって異なります。データサービスの障害モニターがリソースを検証するために実行する操作については、データサービスのマニュアルを参照してください。

リソースの応答に要する時間は、障害モニターやアプリケーションとは関係のない次のような要素にも依存します。

- システム構成
- クラスタ構成
- システム負荷
- ネットワークトラフィックの量

障害モニターの検証タイムアウトを設定する場合は、必要なタイムアウト値をリソースの `Probe_timeout` 拡張プロパティに秒単位で指定します。

## 継続的な障害とみなす基準の定義

一時的な障害による中断を最小限に抑えるために、障害モニターは、このような障害が発生するとこのリソースを再起動します。継続的な障害の場合は、リソースの再起動よりも複雑なアクションをとる必要があります。

- フェイルオーバーリソースの場合は、障害モニターがこのリソースを別のノードにフェイルオーバーします。
- スケーラブルリソースの場合は、障害モニターがこのリソースをオフラインにします。

障害モニターは、指定された再試行間隔の中で、リソースの完全な障害の回数が、指定されたしきい値を超えると障害を継続的であるとみなします。ユーザーは、継続的な障害とみなす基準を定義することによって、可用性要件とクラスタの性能特性を満たすしきい値や再試行間隔を設定できます。

## リソースの完全な障害と部分的な障害

障害モニターは、いくつかの障害を、リソースの「完全な障害」としてみなします。完全な障害は通常、サービスの完全な損失を引き起こします。次に、完全な障害の例を示します。

- データサービスサーバーのプロセスの予期せぬ終了
- 障害モニターがデータサービスサーバーに接続できない

完全な障害が発生すると、障害モニターは再試行間隔内の完全な障害の回数を1つ増やします。

障害モニターは、それ以外の障害を、リソースの「部分的な障害」とみなします。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っ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。

$$\text{retry\_interval} \geq 2 \times \text{threshold} \times (\text{thorough\_probe\_interval} + \text{probe\_timeout})$$

係数2は、ただちにリソースをフェイルオーバーしたりオフラインにすることはない部分的な検証障害を考慮したものです。

## しきい値と再試行間隔を設定するシステムプロパティ

しきい値と再試行間隔を設定するには、リソースの次のようなシステムプロパティを使用します。

- しきい値を設定するには、`Retry_count` システムプロパティを完全な障害の最大値に設定します。
- 再試行間隔を設定する場合には、`Retry_interval` システムプロパティに、必要な間隔を秒数で指定します。

## リソースのフェイルオーバー動作を指定する

リソースのフェイルオーバー動作は、次の障害に対して RGM がどのように応答するかを決定します。

- リソースの起動の失敗
- リソースの停止の失敗
- リソースの障害モニターの停止の失敗

リソースのフェイルオーバー動作を指定するには、リソースの `Failover_mode` システムプロパティを設定します。このプロパティに指定できる値については、[194 ページの「リソースのプロパティ」](#)における `Failover_mode` システムプロパティの説明を参照してください。

# Sun Cluster オブジェクト指向コマンド

---

この付録では、オブジェクト指向コマンド、その短縮形、およびそのサブコマンドの概要を説明します。

## オブジェクト指向コマンド名および別名

多くの Sun Cluster コマンドには、長い説明的な形式以外にも、ユーザーの入力量を大幅に減らす、短縮形つまり別名もあります。次の表に、コマンドとその短い別名を示します。

表 A-1 オブジェクト指向コマンドと別名 (短縮名)

完全なコマンド	別名	目的
<code>claccess</code>	なし	Sun Cluster のアクセスポリシーの管理
<code>cldevice</code>	<code>cldev</code>	Sun Cluster デバイスの管理
<code>cldevicegroup</code>	<code>cldg</code>	Sun Cluster デバイスグループの管理
<code>clinterconnect</code>	<code>clintr</code>	Sun Cluster インターコネクットの管理
<code>clnasdevice</code>	<code>clnas</code>	Sun Cluster の NAS デバイスへのアクセスの管理
<code>clnode</code>	なし	Sun Cluster ノードの管理
<code>clquorum</code>	<code>clq</code>	Sun Cluster 定足数の管理
<code>clquorumserver</code>	<code>clqs</code>	定足数サーバーホスト上での定足数サーバープロセスの構成と管理
<code>clreslogicalhostname</code>	<code>clrslh</code>	論理ホスト名のための Sun Cluster リソースの管理
<code>clresource</code>	<code>clrs</code>	Sun Cluster データサービスのリソースの管理

表 A-1 オブジェクト指向コマンドと別名(短縮名) (続き)

完全なコマンド	別名	目的
clresourcegroup	clrg	Sun Cluster データサービスのリソースグループの管理
clresourcetype	clrt	Sun Cluster データサービスのリソースタイプの管理
clrssharedaddress	clrssa	共有アドレスのための Sun Cluster リソースの管理
clsetup	なし	Sun Cluster の対話型での構成。このコマンドにはサブコマンドはありません。
clsnmphost	なし	Sun Cluster SNMP ホストの管理
clsnmpmib	なし	Sun Cluster SNMP MIB の管理
clsnmpuser	なし	Sun Cluster SNMP ユーザーの管理
cltelemetryattribute	clta	システムリソース監視の構成
cluster	なし	Sun Cluster の広域構成と状態の管理
clvsvm	なし	Veritas Volume Manager for Sun Cluster の構成

## オブジェクト指向コマンドセットの概要

次の表に、オブジェクト指向コマンドセットのコマンドと各コマンドで使用可能なサブコマンドのリストを示します。

表 A-2 claccess: ノード用の Sun Cluster アクセスポリシーの管理

サブコマンド	目的
allow	指定されたマシン (1 つまたは複数) がクラスタ構成にアクセスすることを許可します。
allow-all	すべてのノードがクラスタ構成にアクセスすることを許可します。
deny	指定されたマシン (1 つまたは複数) がクラスタ構成にアクセスすることを禁止します。
deny-all	すべてのノードがクラスタ構成にアクセスすることを禁止します。
list	クラスタ構成へのアクセス権を持っているマシンの名前を表示します。
set	承認プロトコルを <code>-a</code> オプションで指定した値に設定します。
show	クラスタ構成へのアクセス権を持っているマシンの名前を表示します。

表 A-3 cldevice、cldev:Sun Cluster デバイスの管理

サブコマンド	目的
check	デバイスの物理デバイスに対する整合性検査を、カーネル表現と比較して実行します。
clear	現在のノードから排除されたデバイスに関して、すべての DID 参照を削除するよう指定します。
combine	指定された DID インスタンスを新しい宛先インスタンスに結合します。
export	クラスタデバイスの構成情報をエクスポートします。
list	すべてのデバイスパスを表示します。
monitor	指定したディスクパスの監視をオンにします。
populate	広域デバイス名前空間を生成します。
refresh	クラスタノード上にある現在のデバイスツリーに対してデバイス構成情報を更新します。
rename	指定された DID インスタンスを新しい DID インスタンスに移動します。
repair	指定されたデバイスインスタンスに対して修復手順を実行します。
replicate	コントローラベースの複製で使用する DID デバイスを構成します。
set	指定されたデバイスのプロパティを設定します。
show	指定されたすべてのデバイスパスの構成レポートを表示します。
status	コマンドに対するオペランドとして指定されたディスクパスの状態を表示します。
unmonitor	コマンドのオペランドとして指定されたディスクパスの監視をオフにします。

表 A-4 cldevicegroup、cldg:Sun Cluster デバイスグループの管理

サブコマンド	目的
add-device	新しいメンバーディスクデバイスを既存の raw ディスクデバイスグループに追加します。
add-node	新しいノードを既存のデバイスグループに追加します。
create	新しいデバイスグループを作成します。
delete	デバイスグループを削除します。
disable	オフラインのデバイスグループを無効にします。
enable	デバイスグループを有効にします。
export	デバイスグループ構成情報をエクスポートします。

表 A-4 cldevicegroup、cldg: Sun Cluster デバイスグループの管理 (続き)

サブコマンド	目的
list	デバイスグループのリストを表示します。
offline	デバイスグループをオフラインにします。
online	指定されたノードでデバイスグループをオンラインにします。
remove-device	メンバーディスクデバイスを raw ディスクデバイスグループから削除します。
remove-node	既存のデバイスグループからノードを削除します。
set	デバイスグループに関連付けられている属性を設定します。
show	デバイスグループの構成レポートを作成します。
status	デバイスグループのステータスレポートを作成します。
switch	Sun Cluster 構成内の、ある主ノードから別のノードにデバイスグループを転送します。
sync	クラスタリングソフトウェアとデバイスグループ情報の同期をとります。

表 A-5 clinterconnect、clintr: Sun Cluster インターコネクットの管理

サブコマンド	目的
add	コマンドへのオペランドとして指定された新しいクラスタインターコネクットコンポーネントを追加します。
disable	コマンドへのオペランドとして指定されたインターコネクットコンポーネントを無効にします。
enable	コマンドへのオペランドとして指定されたインターコネクットコンポーネントを有効にします。
export	クラスタインターコネクットの構成情報をエクスポートします。
remove	コマンドへのオペランドとして提供されたクラスタインターコネクットコンポーネントを削除します。
show	インターコネクットコンポーネントの構成を表示します。
status	インターコネクットパスのステータスを表示します。

表 A-6 clnasdevice、clnas: Sun Cluster の NAS デバイスへのアクセスの管理

サブコマンド	目的
add	NAS デバイスを Sun Cluster 構成に追加します。
add-dir	すでに構成されている NAS デバイスの指定されたディレクトリをクラスタ構成に追加します。

表 A-6 `clnasdevice`、`clnas`: Sun Cluster の NAS デバイスへのアクセスの管理 (続き)

サブコマンド	目的
<code>export</code>	クラスタ NAS デバイス構成情報をエクスポートします。
<code>list</code>	クラスタに構成されている NAS デバイス構成を表示します。
<code>remove</code>	指定された NAS デバイス (1 つまたは複数) を Sun Cluster 構成から削除します。
<code>remove-dir</code>	指定された NAS ディレクトリ (1 つまたは複数) を Sun Cluster 構成から削除します。
<code>set</code>	特定の NAS デバイスの指定されたプロパティを設定します。
<code>show</code>	クラスタ内の NAS デバイスの構成情報を表示します。

表 A-7 `clnode`: Sun Cluster ノードの管理

サブコマンド	目的
<code>add</code>	ノードをクラスタに構成および追加します。
<code>add-farm</code>	ファームノードをクラスタに追加します。
<code>clear</code>	Sun Cluster ソフトウェア構成からノードを削除します。
<code>evacuate</code>	指定されたノードから新しい主ノードに、すべてのリソースグループおよびデバイスグループを切り替えます。
<code>export</code>	ノードまたはファーム構成情報をファイルまたは標準出力 (stdout) にエクスポートします。
<code>list</code>	クラスタまたはファームで構成されているノードの名前を表示します。
<code>remove</code>	ノードをクラスタから削除します。
<code>remove-farm</code>	ファームノードをクラスタから削除します。
<code>set</code>	指定したノードに関連するプロパティを設定します。
<code>show</code>	指定されたノード (1 つまたは複数) の構成を表示します。
<code>show-rev</code>	ノードにインストールされている Sun Cluster パッケージの名前と、そのノードについてのリリース情報を表示します。
<code>status</code>	指定したノード (1 つまたは複数) のステータスを表示します。

表 A-8 `clquorum`、`clq`: Sun Cluster の定足数構成の管理

サブコマンド	目的
<code>add</code>	指定した共有デバイスを定足数デバイスとして追加します。
<code>disable</code>	定足数デバイスまたはノードを定足数保守状態に置きます。

表 A-8 clquorum、clq: Sun Cluster の定足数構成の管理 (続き)

サブコマンド	目的
enable	定足数デバイスまたはノードを定足数保守状態から解除します。
export	クラスタ定足数の構成情報をエクスポートします。
list	クラスタ内で設定されている定足数デバイスの名前を表示します。
remove	指定された定足数デバイス (1 つまたは複数) を、Sun Cluster 定足数構成から削除します。
reset	定足数構成全体をリセットし、デフォルトの投票数にします。
show	定足数デバイスのプロパティを表示します。
status	定足数デバイスの状態と投票数を表示します。

表 A-9 clquorumserver、clqs: 定足数サーバーの管理

サブコマンド	目的
clear	期限切れのクラスタ情報を定足数サーバーから削除します。
show	定足数サーバーについての構成情報を表示します。
start	ホストマシン上で定足数サーバープロセスを起動します。
stop	定足数サーバープロセスを停止します。

表 A-10 clreslogicalhostname、clrslh: Sun Cluster 論理ホスト名のリソースの管理

サブコマンド	目的
create	新しい論理ホスト名リソースを作成します。
delete	論理ホスト名リソースを削除します。
disable	論理ホスト名リソースを無効にします。
enable	論理ホスト名リソースを有効にします。
export	論理ホスト名のリソース構成をエクスポートします。
list	論理ホスト名リソースのリストを表示します。
list-props	論理ホスト名リソースのプロパティのリストを表示します。
monitor	論理ホスト名リソースに対する監視をオンにします。
reset	論理ホスト名リソースと関連するエラーフラグをクリアします。
set	論理ホスト名リソースの指定されたプロパティを設定します。
show	論理ホスト名リソースの構成を表示します。

表 A-10 clreslogicalhostname、clrs1h: Sun Cluster 論理ホスト名のリソースの管理 (続き)

サブコマンド	目的
status	論理ホスト名リソースのステータスを表示します。
unmonitor	論理ホスト名リソースに対する監視をオフにします。

表 A-11 clresource、clrs: Sun Cluster データサービスのリソースの管理

サブコマンド	目的
create	コマンドに対するオペランドとして指定されたリソースを作成します。
delete	コマンドに対するオペランドとして指定されたリソースを削除します。
disable	リソースを無効にします。
enable	リソースを有効にします。
export	クラスタリソース構成をエクスポートします。
list	クラスタリソースのリストを表示します。
list-props	リソースプロパティのリストを表示します。
monitor	リソースの監視をオンにします。
reset	クラスタリソースと関連しているエラーフラグをクリアします。
set	リソースプロパティを設定します。
show	リソース構成を表示します。
status	リソースのステータスを表示します。
unmonitor	リソースの監視をオフにします。

表 A-12 clresourcegroup、clrg: Sun Cluster データサービスのリソースグループの管理

サブコマンド	目的
add-node	ノードをリソースグループの <code>NodeList</code> プロパティの最後に追加します。
create	新しいリソースグループを作成します。
delete	リソースグループを削除します。
evacuate	-n オプションで指定したノード上のすべてのリソースグループをオフラインにします。
export	リソースグループの構成情報をファイルまたは標準出力 <code>stdout</code> に書き込みます。
list	リソースグループのリストを表示します。

表 A-12 clresourcegroup、clrg:Sun Cluster データサービスのリソースグループの管理 (続き)

サブコマンド	目的
manage	指定したリソースグループを管理状態にします。
offline	指定したリソースグループをオフライン状態にします。
online	指定したリソースグループをオンライン状態にします。
quiesce	指定されたリソースグループを休止状態にします。
remaster	指定したリソースグループを、最も優先されるノードに切り替えます。
remove-node	ノードをリソースグループの NodeList プロパティから削除します。
restart	もともとリソースグループをホストしていた主ノードの同じセット上でリソースグループをオフラインにしてからオンラインに戻します。
resume	保存停止にある指定されたソースグループの保存停止状態をクリアします。
set	指定したリソースグループに関連付けられているプロパティを設定します。
show	指定したリソースグループの構成レポートを生成します。
status	指定したリソースグループのステータスレポートを生成します。
suspend	指定したリソースグループにより管理されているすべてのアプリケーションに対して、Resource Group Manager (RGM) の制御を保存停止します。
switch	指定したリソースグループをマスターするノードまたはノードのセットを変更します。
unmanage	指定したリソースグループを管理されない状態にします。

表 A-13 clresourcetype、clrt:Sun Cluster データサービスのリソースタイプの管理

サブコマンド	目的
add-node	指定されたノードを、リソースタイプのノードリストに追加します。
export	クラスタリソースタイプ構成をエクスポートします。
list	リソースタイプのリストを表示します。
list-props	リソースタイプのリソース拡張プロパティまたはリソースタイププロパティのリストを表示します。
register	リソースタイプを登録します。
remove-node	オペランドリスト内のリソースタイプが登録されるノードのリストからノードを削除します。
set	リソースタイプのプロパティを設定します。

表 A-13 clresourcetype、clrt:Sun Cluster データサービスのリソースタイプの管理 (続き)

サブコマンド	目的
show	クラスタ内に登録されているリソースタイプについての構成情報を表示します。
unregister	リソースタイプを登録解除します。

表 A-14 clressharedaddress、clrssa: 共有アドレスの Sun Cluster リソースの管理

サブコマンド	目的
create	共有アドレスリソースを作成します。
delete	共有アドレスリソースを削除します。
disable	共有アドレスリソースを無効にします。
enable	共有アドレスリソースを有効にします。
export	共有アドレスリソース構成をエクスポートします。
list	共有アドレスリソースのリストを表示します。
list-props	共有アドレスリソースのプロパティのリストを表示します。
monitor	共有アドレスリソースの監視をオンにします。
reset	共有アドレスリソースと関連付けられたエラーフラグをクリアします。
set	共有アドレスリソースの指定されたプロパティを設定します。
show	共有アドレスリソースの構成を表示します。
status	共有アドレスリソースのステータスを表示します。
unmonitor	共有アドレスリソースの監視をオフにします。

表 A-15 clsnmphot: Sun Cluster SNMP ホストのリストの管理

サブコマンド	目的
add	SNMP ホストを、指定されたノード構成に追加します。
export	指定されたノードから SNMP ホスト情報をエクスポートします。
list	指定されたノード上で構成されている SNMP ホストを一覧表示します。
remove	SNMP ホストをノード構成から削除します。
show	指定されたノード上の SNMP ホスト構成情報を表示します。

表 A-16 `clsnmpmib`: Sun Cluster SNMP MIB の管理

サブコマンド	目的
<code>disable</code>	指定されたノード上の1つ以上のクラスタの MIB を無効にします。
<code>enable</code>	指定されたノード上にある1つ以上のクラスタの MIB を有効にします。
<code>export</code>	クラスタの MIB の構成情報をエクスポートします。
<code>list</code>	指定されたノード上のクラスタの MIB のリストを表示します。
<code>set</code>	1つまたは複数の MIB で使用されている SNMP プロトコル設定を設定します。
<code>show</code>	指定されたノード上の MIB の構成情報を表示します。

表 A-17 `clsnmpuser`: Sun Cluster SNMP ユーザーの管理

サブコマンド	目的
<code>create</code>	指定されたノード上の SNMP ユーザー構成にユーザーを追加します。
<code>delete</code>	SNMPv3 ユーザーを指定されたノードから削除します。
<code>export</code>	SNMP ユーザー情報を指定されたノードからエクスポートします。
<code>list</code>	指定されたノードで構成されている SNMPv3 ユーザーのリストを出力します。
<code>set</code>	指定されたノード上のユーザーの構成を設定します。
<code>set-default</code>	SNMPv3 を使用してトラップを送信する際に使用する、デフォルトのユーザーおよびセキュリティレベルを設定します。
<code>show</code>	指定されたノード上のユーザーについての情報を表示します。

表 A-18 `cltelemetryattribute`、`clta`: システムリソースの監視の構成

サブコマンド	目的
<code>disable</code>	指定されたオブジェクトタイプの指定されたテレメトリ属性を無効にします。
<code>enable</code>	指定されたオブジェクトタイプの指定されたテレメトリ属性のデータ収集を有効にします。
<code>export</code>	オブジェクトタイプおよびオブジェクトインスタンスのテレメトリ属性の構成をファイルまたは標準出力 <code>stdout</code> にエクスポートします。
<code>list</code>	指定されたオブジェクトタイプに対して構成されているテレメトリ属性を表示します。
<code>print</code>	指定されたオブジェクトインスタンスまたはオブジェクトタイプに対して有効な、指定されているテレメトリ属性のシステムリソースの使用状況を表示します。

表 A-18 cltelemetryattribute、clta: システムリソースの監視の構成 (続き)

サブコマンド	目的
set-threshold	ノード上の指定されたオブジェクトの指定されたテレメトリ属性のしきい値の設定を変更します。
show	オブジェクトタイプまたはオブジェクトインスタンスのテレメトリ属性に対して設定されているプロパティを表示します。

表 A-19 cluster: クラスタの広域構成とステータスの管理

サブコマンド	目的
create	clconfigfile ファイルに格納されている構成情報を使用してクラスタを作成します。
export	クラスタ構成ファイルの構成情報をエクスポートします。
list	クラスタコマンドの発行対象であるクラスタの名前を表示します。
list-cmds	使用可能なすべての Sun Cluster コマンドのリストを出力します。
rename	クラスタコマンドの発行対象であるクラスタの名前を変更します。
restore-netprops	クラスタコマンドの発行対象であるクラスタの、クラスタプライベートネットワーク設定を修復します。
set	クラスタコマンドの発行対象であるクラスタのプロパティを設定します。
set-netprops	クラスタプライベートネットワークアドレスのプロパティを設定します。
show	指定されたクラスタのクラスタコンポーネントに関する詳細な構成情報を表示します。
show-netprops	プライベートネットワークアドレスの設定を表示します。
shutdown	クラスタコマンドの発行対象であるクラスタを適切な順序で停止します。
status	指定されたクラスタのクラスタコンポーネントのステータスを表示します。

表 A-20 clvxdm: Sun Cluster の VERITAS Volume Manager の構成

サブコマンド	目的
encapsulate	ルートディスクをカプセル化し、ほかの Sun Cluster 固有のタスクを実行します。
initialize	VxVM を初期化し、その他の Sun Cluster 固有のタスクを実行します。



## 標準プロパティ

---

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

---

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名は、大文字と小文字が区別されません。プロパティ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。

- 183 ページの「資源タイプのプロパティ」
- 194 ページの「リソースのプロパティ」
- 216 ページの「リソースグループのプロパティ」
- 232 ページの「リソースプロパティの属性」

### 資源タイプのプロパティ

以下に、Sun Cluster ソフトウェアによって定義されるリソースタイププロパティを示します。

プロパティ値は以下のように分類されます。

- 必須。プロパティはリソースタイプ登録 (RTR) ファイルに明示的な値を必要とします。そうでない場合、プロパティが属するオブジェクトは作成できません。空白文字または空の文字列を値として指定することはできません。
- 条件付。RTR ファイル内に宣言を必要とするプロパティです。宣言がない場合、RGM はこのプロパティを作成しません。したがって、このプロパティを管理ユーティリティ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です。宣言がない場合、RGM はこのプロパティを作成しません。したがって、このプロパティを管理ユーティリティ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です。プロパティが 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。プロパティが RTR ファイル内で宣言されており、値が指定されていない場合は、RGM は、プロパティが RTR ファイル内で宣言されないときのデフォルト値と同じ値を使用します。
- 照会のみ-管理ツールから直接設定できません。

Installed\_nodes と RT\_system 以外のリソースタイププロパティは、管理ユーティリティで更新を行うことはできません。また、Installed\_nodes は RTR ファイル内に宣言できないため、クラスタ管理者のみが設定できます。RT\_system には RTR ファイル内で初期値を割り当てることができ、またクラスタ管理者が設定することもできます。

以下にプロパティ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名では、大文字と小文字が区別されません。プロパティ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 (整数型)

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 8/05	6
3.2	7
3.2 2/08	8

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=7` を宣言すると、このリソースタイプは、3.2 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- このプロパティを宣言しないか、このプロパティをデフォルト値 (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 (文字列型)

任意のコールバックメソッド。RGM がノードまたはゾーン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 (ブール型)

このプロパティを TRUE に設定した場合、このタイプのリソースは、複数のノードまたはゾーンで同時にオンラインになる可能性があるどのグループ内でも構成できません。

このリソースタイプのプロパティは、次のように Scalable リソースのプロパティと一緒に使用します。

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 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は、非大域ゾーンで動作するように構成できます。ただし、同一ノードの複数の非大域ゾーンで動作するようには構成しないでください。
FALSE	FALSE	この組み合わせは、ネットワーク負荷分散を使用しない複数マスターサービスを選択する場合に使用します。  このタイプのスケーラブルサービスはゾーン内で使用可能です。

Scalable 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 (文字列型)

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。

Fini メソッドは、通常、Init メソッドにより実行された初期化を元に戻します。

次のような状態が生じた場合にリソースが管理対象外となる各ノードまたはゾーンで、RGMはFiniを実行します。

- リソースを含むリソースグループが管理対象外状態に切り替わる。この場合、ノードリストのすべてのノードおよびゾーン上で、RGMはFiniメソッドを実行します。
- 管理されているリソースグループからリソースが削除される。この場合、ノードリストのすべてのノードおよびゾーン上で、RGMはFiniメソッドを実行します。
- リソースを含むリソースグループのノードリストからノードまたはゾーンが削除される。この場合、削除されたノードまたはゾーン上でのみ、RGMはFiniメソッドを実行します。

「ノードリスト」はリソースグループのNodelistまたはリソースタイプのInstalled\_nodes リストのいずれかです。「ノードリスト」がリソースグループのNodelistとリソースタイプのInstalled\_nodes リストのどちらを指すかは、リソースタイプのInit\_nodes プロパティの設定に依存します。Init\_nodes プロパティはRG\_nodelistまたはRT\_installed\_nodesに設定できます。大部分のリソースタイプでは、Init\_nodesはデフォルトであるRG\_nodelistに設定されます。この場合、InitメソッドとFiniメソッドは両方とも、リソースグループのNodelistで指定されているノードおよびゾーン上で実行されます。

Initメソッドが実行する初期化の種類は、次のように、ユーザーが実装したFiniメソッドが実行する必要があるクリーンアップの種類を定義します。

- ノード固有の構成のクリーンアップ。
- クラスタ全体の構成のクリーンアップ。

カテゴリ: 条件付/明示

デフォルト: デフォルトなし

調整: NONE

#### Global\_zone (ブール型)

RTRファイルで宣言されている場合、このリソースタイプのメソッドが大域ゾーンで実行されるかどうかを示すブール値。このプロパティにTRUEが設定されている場合、リソースを含むリソースグループが非大域ゾーンで動作しているときでも、メソッドは大域ゾーンで実行されます。このプロパティにTRUEを設定するのは、ネットワークアドレスやファイルシステムなど、大域ゾーンから管理できるサービスに対してだけです。



注意 - 信頼できる既知のソースであるリソースタイプを除いて、`Global_zone` プロパティに `TRUE` が設定されているリソースタイプは登録しないでください。このプロパティに `TRUE` を設定したリソースタイプは、ゾーン分離をすり抜け、危険があります。

カテゴリ: 任意  
 デフォルト: `FALSE`  
 調整: `ANYTIME`

#### Init (文字列型)

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 (列挙型)

RGM が `Init`、`Fini`、`Boot`、`Validate` メソッドをコールするノードまたはゾーンを示します。指定できる値は、リソースをマスターできるノードまたはゾーンのみを指定する `RG_PRIMARYES`、またはこのリソース型がインストールされるすべてのノードまたはゾーンを指定する `RT_INSTALLED_NODES` のいずれかです。

カテゴリ: 任意  
 デフォルト: `RG_PRIMARYES`  
 調整: `NONE`

#### Installed\_nodes (文字配列型)

リソースタイプを実行できるクラスタノードまたはゾーンの名前のリスト。このプロパティは RGM によって自動的に作成されます。クラスタ管理者は値を設定できます。RTR ファイル内には宣言できません。

カテゴリ: このプロパティはクラスタ管理者が構成できます。  
 デフォルト: すべてのクラスタノードおよびゾーン  
 調整: `ANYTIME`

#### Is\_logical\_hostname (ブール型)

`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 (ブール型)

TRUE は、このリソースタイプが、共有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 (文字列型)

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。ノードまたはゾーン上でモニターチェックプログラムがゼロ以外の値とともに終了した場合、GIVEOVER タグ付きで `scha_control` を呼び出した結果としてのそのノードまたはゾーンへのフェイルオーバーの試みは阻止されます。

カテゴリ: 条件付/明示

デフォルト: デフォルトなし

調整: NONE

#### Monitor\_start (文字列型)

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 (文字列型)

`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 (文字配列型)

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 (文字列型)

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 (文字列型)

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

#### Proxy (ブール型)

このタイプのリソースがプロキシリソースかどうかを示すブール値です。

「プロキシリソース」は、リソースの状態を Oracle Cluster Ready Services (CRS) などの別のクラスタフレームワークからインポートする Sun Cluster リソースです。Oracle クラスタウェア CRS として現在知られている Oracle CRS は、クラスタ環境向けのプラットフォームに依存しないシステムサービスセットです。

プロキシリソースタイプは、Prenet\_start メソッドを使用して、外部のプロキシリソースの状態を監視するデーモンを起動します。Postnet\_stop メソッドは、この監視デーモンを停止します。この監視デーモンは、CHANGE\_STATE\_ONLINE または CHANGE\_STATE\_OFFLINE タグとともに scha\_control コマンドを実行し、プロキシリソースの状態をそれぞれ Online または Offline に設定します。scha\_control() 関数も同じように SCHA\_CHANGE\_STATE\_ONLINE および SCHA\_CHANGE\_STATE\_OFFLINE タグを使用します。詳細は、scha\_control(1HA) と scha\_control(3HA) のマニュアルページを参照してください。

TRUE に設定されている場合、リソースはプロキシリソースです。

カテゴリ: 任意

デフォルト: FALSE

調整: NEVER

#### Resource\_list (文字配列型)

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 (文字配列型)

リソースタイプの名前。現在登録されているリソースタイプ名を表示するには、次のコマンドを使用します。

##### **resourcetype show +**

Sun Cluster 3.1 および Sun Cluster 3.2 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 *vendor-id*、*resource-type*、*rt-version* で構成されます。**resourcetype** コマンドは、ピリオド (.) とコロン (:) のプロパティの間に挿入します。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、リソース型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: 必要

デフォルト: 空の文字列

調整: NONE

#### RT\_basedir (文字列型)

コールバックメソッドの相対パスのを補完するディレクトリパスです。このパスは、リソースタイプパッケージのインストール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 (文字列型)

リソース型の簡単な説明です。

カテゴリ: 条件付き

デフォルト: 空の文字列

調整: NONE

RT\_system (ブール型)

リソースタイプの RT\_system プロパティが TRUE の場合、そのリソースタイプは削除できません (**resourcetype unregister resource-type-name**)。このプロパティは、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 プロパティはどのリソース型にも適用できます。

RT\_system プロパティが TRUE に設定されたリソース型を削除するには、まず、このプロパティを FALSE に設定する必要があります。クラスタサービスをサポートするリソースを持つリソース型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 (文字配列型)

Sun Cluster 3.1 リリース以降では、このリソースタイプの実装を特定する必須バージョン文字列。Sun Cluster 3.0 ではこのプロパティは任意でした。RT\_version は完全なリソースタイプ名のサフィックスコンポーネントです。

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 (ブール型)

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。

カテゴリ: 任意

デフォルト: FALSE

調整: NONE

Start (文字配列型)

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 (文字配列型)

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 (文字配列型)

任意のコールバックメソッド。この型の実行中のリソースのプロパティが変更されたときに、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 (文字配列型)

任意のコールバックメソッド。この型のリソースのプロパティ値を検査するために 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 (文字配列型)

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

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について説明します。

プロパティ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が宣言されている場合にかぎりプロパティを作成します。宣言されていない場合プロパティは存在せず、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は、デフォルト値が RTR ファイル内で指定されているかどうかによって、必須または任意になります。詳細については、各条件付きプロパティの説明を参照してください。
- 照会のみ。管理ツールで直接設定することはできません。

232 ページの「リソースプロパティの属性」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	しない
TRUE または ANYTIME	すべての時刻
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またはゾーンに送信されます。

このプロパティは、Load\_balancing\_policy が Lb\_sticky または Lb\_sticky\_wild の場合にかぎり有効です。さらに、Weak\_affinity が FALSE に設定されている必要があります。

このプロパティは、スケーラブルサービス専用です。

カテゴリ:	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

そのリソースのタイプのRTRファイルで宣言される拡張プロパティ。リソースタイプの実装によって、これらのプロパティを定義します。拡張プロパティに設定可能な各属性については、[232 ページの「リソースプロパティの属性」](#)を参照してください。

カテゴリ: 条件付き

デフォルト: デフォルトなし

調整: 特定のプロパティに依存

`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)

これらの設定が影響するのは、起動または停止メソッド(`Prenet_start`、`Start`、`Monitor_stop`、`Stop`、`Postnet_stop`)が失敗した場合のフェイルオーバー動作のみです。RESTART\_ONLY設定とLOG\_ONLY設定は、リソースモニターが`scha_control`コマンドまたは`scha_control()`関数の実行を開始できるかどうかにも影響します。`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、前述の起動メソッドまたは停止メソッドが失敗する場合に RGM が何の復旧処理も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またはゾーンに再配置することを示します。Start または Prenet\_start の失敗に関しては、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) の失敗に関しては、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ときに Failover\_mode が HARD に設定されている場合、RGM はノードまたはゾーンをリブートして、強制的にリソースグループをオフライン状態にします。これにより RGM は別のノードまたはゾーンでグループの起動を試みるのが可能になります。

#### 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する場合にフェイルオーバー動作に影響を与える NONE、SOFT、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を与えます。フェイルオーバー動作には、モニター起動 (scha\_control) によるリソースやリソースグループの再起動や、リソースモニター (scha\_control) によって開始されるギブオーバーなどがあります。

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意味します。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。

---

注-Retry\_count の負の値は、リソースタイプによっては適用できませんが、リソースを無制限に再起動できることを指定します。より確実に無制限の再起動を指定するには、次の手順を実行します。

- Retry\_interval に 1 や 0 などの小さい値を指定します。
- Retry\_count に 1000 などの大きい値を指定します。

リソースタイプが Retry\_count および Retry\_interval プロパティを宣言しない場合は、リソースは回数の制限なく再起動できます。

---

Failover\_mode が LOG\_ONLY に設定されている場合、リソースの再起動またはギブオーバーは許可されません。Failover\_mode に LOG\_ONLY を設定するのは、Failover\_mode に RESTART\_ONLY を設定し、Retry\_count にゼロを設定するのと同じことです。

#### 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 と同じことです。つまり、ノードまたはゾーン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の対象となるのは、そのサービスが Monitor\_start メソッドを実装しており、かつ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あとで 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により、再起動またはフェイルオーバーを要求できます。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あとも検証が再び失敗した場合、モニターは別のノードまたはゾーン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る場合、検出される障害はリソースのプロセスツリーの終了のみです。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` 内で `Retry_count` の回数 `scha_control()` 関数を呼び出すことで再起動できます。リソースグループが `Retry_count` を超過すると、再起動の試行が失敗します。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。プローブが `Retry_interval` 内で `Retry_count` の回数よりも多く完全に失敗した場合、リソースモニターは終了し、リソースのステータスを `FAULTED` に設定し、ステータスメッセージ「`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 `Finis_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。このプロパティは、スケーラブルサービス専用です。RTRファイルに `Scalable` プロパティが宣言されている場合、RGMは自動的にこのプロパティを作成します。`Load_balancing_policy`には次の値を設定できます。

`Lb_weighted`(デフォルト)。`Load_balancing_weights` プロパティに設定されている重みにより、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)`

このプロパティは、スケーラブルサービス専用です。RTRファイルに `Scalable` プロパティが宣言されている場合、RGMは自動的にこのプロパティを作成します。形式は、「`weight@node, weight@node`」になります。`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、このプロパティの `Tunable` 値は `ANYTIME` になります。このプロパティを変更すると、新しい接続時にのみ分散が変更されます。

カテゴリ: 条件付き/任意

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ティ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)**

このリソースが依存関係を持ってい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。このリストには、プロパティ Resource\_dependencies、Resource\_dependencies\_weak、Resource\_dependencies\_restart、またはResource\_dependencies\_offline\_restartに現れるすべてのネットワークアドレスリソースが含まれます。

RTR ファイルに Scalable プロパティが宣言されている場合、RGMは自動的にこのプロパティを作成します。ScalableがRTRファイルで宣言されていない場合、Network\_resources\_usedはRTRファイルで明示的に宣言されていない限り使用できません。

このプロパティは、リソース依存関係プロパティの設定に基づいて、RGMにより自動的に更新されます。このプロパティを直接設定する必要はありません。しかし、このプロパティにリソース名を追加する場合、そのリソース名は自動的にResource\_dependenciesプロパティに追加されます。また、このプロパティからリソース名を削除する場合、そのリソース名は自動的に、そのリソースが現れるすべてのリソース依存関係プロパティから削除されます。

カテゴリ: 条件付き任意

デフォルト: 空のリスト

調整: ANYTIME

**各クラスタノードまたはゾーン上の Num\_resource\_restarts (integer)**

過去  $n$  秒以内にこのリソースで発生した再起動要求の数。 $n$ は、Retry\_intervalプロパティの値です。

再起動要求は、次に示す呼び出しのいずれかです。

- RESOURCE\_RESTART 引数を持つ scha\_control(1HA) コマンド。
- SCHA\_RESOURCE\_RESTART 引数を持つ scha\_control(3HA) 関数。
- RESOURCE\_IS\_RESTARTED 引数を持つ scha\_control コマンド。
- SCHA\_RESOURCE\_IS\_RESTARTED 引数を持つ scha\_control 関数。

リソースが次に示す処理のいずれかを実行した場合、RGMは、ある特定のノードまたはゾーン上にある特定のリソースに対して再起動カウンタを必ず0にリセットします。

- GIVEOVER 引数を持つ `scha_control` コマンド。
- SCHA\_GIVEOVER 引数を持つ `scha_control` 関数。

カウンタは、ギブオーバーの試行が成功した場合でも失敗した場合でもリセットされます。

リソース型が `Retry_interval` プロパティを宣言していない場合、この型のリソースに `Num_resource_restarts` プロパティを使用できません。

カテゴリ: 照会のみ

デフォルト: デフォルトなし

調整: 説明を参照

各クラスタノードまたはゾーン上の `Num_rg_restarts` (integer)

過去  $n$  秒以内にこのリソースに対して発生したりソースグループ再起動要求の数。  $n$  は、`Retry_interval` プロパティの値です。

リソースグループ再起動要求は、次に示す呼び出しのいずれかです。

- RESTART 引数を持つ `scha_control(1HA)` コマンド。
- SCHA\_RESTART 引数を持つ `scha_control(3HA)` 関数。

リソース型が `Retry_interval` プロパティを宣言していない場合、この型のリソースに `Num_resource_restarts` プロパティを使用できません。

カテゴリ: 照会のみ

デフォルト: デフォルトなし

調整: 説明を参照

`On_off_switch` (enum)

クラスタ管理者が管理ユーティリティ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。ポート番号には、スラッシュ (/) と、そのポートで使用されるプロトコルが付加されます(たとえば、`Port_list=80/tcp` や `Port_list=80/tcp6,40/udp6` など)。

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 プロパティが RTR ファイルで宣言されている場合、RGM は自動的に Port\_list を作成します。それ以外の場合、このプロパティは RTR ファイルで明示的に宣言されていないかぎり使用できません。

Apache 用にこのプロパティを設定する方法は、『Sun Cluster Data Service for Apache Guide for Solaris OS』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

Proxied\_service\_instances

リソースによってプロキシされる SMF サービスに関する情報を含みます。値はプロキシされるすべての SMF サービスを含むファイルのパスです。ファイル内の各行は 1 つの SMF サービス専用で、svc fmri および対応するサービスマニフェストファイルのパスを指定します。

たとえば、リソースが2つのサービス、`restarter_svc_test_1:default` と `restarter_svc_test_2:default` を管理する必要がある場合、ファイルには次に示す2行が含まれているはずで

```
<svc:/system/cluster/restarter_svc_test_1:default>,svc:/system/cluster/\
restarter_svc_test_1:default>,</var/svc/manifest/system/cluster/\
restarter_svc_test_1.xml>
```

```
<svc:/system/cluster/restarter_svc_test_2:default>,</var/svc/manifest/\
system/cluster/restarter_svc_test_2.xml>
```

デフォルト:""

調整:When\_disabled

**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)**

リソースが強い依存関係を持っているリソースのリスト。強い依存関係は、メソッド呼び出しの順序を決定します。

リソースの依存関係を有するリソースは依存しているリソースと呼ばれ、依存されているリソースと呼ばれるリスト内のリソースがオンラインでないと依存しているリソースを起動することはできません。依存しているリソースと、リスト内のいずれかの依存されているリソースが同時に起動した場合、RGM は、リスト内の依存されているリソースが起動するまで依存しているリソースの起動を待ちます。依存されているリソースが起動しないと、依存しているリソースはオフラインのままになります。依存されているリソースが起動しない場合があるのは、リスト内の依存されている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` 状態であるためです。異なるリソースグループ内の依存されているリソースが起動に失敗したり、無効またはオフラインになったりしていることが原因で、依存しているリソースがオフラインのままになっている場合、依存しているリソースのグループは `Pending_online_blocked` 状態になります。起動に失敗した、無効である、またはオフラインである同じリソースグループ内の依存されているリソースに、依存しているリソースが依存関係を持っている場合、リソースグループは `Pending_online_blocked` 状態にはなりません。

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、[216 ページの「リソースグループのプロパティ」](#)の `Implicit_network_dependencies` を参照してください。

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。同様に、`Postnet_stop` メソッドは `Stop` メソッドよりあとに実行されます。異なるリソースグループ内では、依存されるリソースが `Prenet_start` と `Start` を実行してから、依存するリソースが `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。

依存関係の範囲を指定するには、このプロパティを指定するときに、次の修飾子の中括弧 `{}` を含めてリソース名に付加します。

`{LOCAL_NODE}` 指定される依存関係をノード単位またはゾーン単位に限定します。依存関係の動作は、同じノードまたはゾーン上でのみ依存先のリソースに影響されます。依存しているリソースは、依存先のリソースが同じノードまたはゾーンで起動されるまで待機します。この状況は停止と再起動、および有効化と無効化に似ています。

`{ANY_NODE}` 指定された依存関係を任意のノードまたはゾーンに拡張します。依存関係の動作は、どのノードまたはゾーンでも依存先のリソースに影響されます。依存しているリソースは、自分が起動する前に依存されているリソースが少なくとも1つの主ノードまたはゾーンで起動するまで待機します。この状況は停止と再起動、および有効化と無効化に似ています。

依存しているリソースのリソースグループが依存されているリソースのリソースグループに対して肯定的なアフィニティを持っている場合であっても、依存関係は `ANY_NODE` のままになります。

`{FROM_RG_AFFINITIES}` リソースのリソースグループの `RG_affinities` 関係に基づいて、依存関係が `LOCAL_NODE` または `ANY_NODE` であるかを指定します。

依存しているリソースのグループが依存されているリソースのグループに対して肯定的なアフィニティを持っていて、リソースグループが同一ノード上で起動または停止する場合、依存関係は `LOCAL_NODE` であるとみなされます。そのような肯定的なアフィニティが存在しない場合、または異なるノード上でグループが起動する場合、依存関係は `ANY_NODE` であるとみなされます。

修飾子を指定しない場合は、`FROM_RG_AFFINITIES` 修飾子がデフォルトで使用されます。

同じリソースグループ内の2つのリソース間のリソース依存関係は、常に LOCAL\_NODE です。

カテゴリ: 任意

デフォルト: 空のリスト

調整: ANYTIME

#### Resource\_dependencies\_offline\_restart (string\_array)

リソースがオフライン再起動の依存関係を持っているリソースのリスト。オフライン再起動の依存関係は、メソッド呼び出しの順序を決定します。

このプロパティの動作は Resource\_dependencies とよく似ていますが、1点例外があります。依存されているリソースと呼ばれる、オフライン再起動の依存関係リストのリソースがオフラインになると、RGMは、依存しているリソースと呼ばれる、リソースの依存関係を有するリソースの再起動をトリガーします。依存しているリソースはただちに停止し、依存されているリソースが再起動されるまでオフラインのままになります。リスト内の依存されているリソースがオンラインに戻ると、RGMは依存しているリソースを再起動します。このような再起動動作が発生するのは、依存しているリソースと依存されているリソースを含むリソースグループがオンラインのままである場合です。

依存されているリソースがオンラインでない場合、依存しているリソースを起動することはできません。依存しているリソースと、リスト内のいずれかの依存されているリソースが同時に起動した場合、RGMは、リスト内の依存されているリソースが起動するまで依存しているリソースの起動を待ちます。依存されているリソースが起動しないと、依存しているリソースはオフラインのままになります。依存されているリソースが起動しない場合があるのは、リスト内の依存されている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 状態であるためです。異なるリソースグループ内の依存されているリソースが起動に失敗したり、無効またはオフラインになったりしていることが原因で、依存しているリソースがオフラインのままになっている場合、依存しているリソースのグループは Pending\_online\_blocked 状態になります。同じリソースグループ内の依存されているリソースが起動に失敗した、無効である、またはオフラインである場合、リソースグループは Pending\_online\_blocked 状態にはなりません。

依存関係の範囲を指定するには、このプロパティを指定するときに、次の修飾子を中括弧 {} を含めてリソース名に付加します。

{LOCAL\_NODE}

指定される依存関係をノード単位またはゾーン単位に限定します。依存関係の動作は、同じノードまたはゾーン上でのみ依存先のリソースに影響されます。依存しているリソースは、依存先のリソースが同じノードまたはゾーンで起動されるまで待機します。この状況は停止と再起動、および有効化と無効化に似ています。

**{ANY\_NODE}** 指定された依存関係を任意のノードまたはゾーンに拡張します。依存関係の動作は、どのノードまたはゾーンでも依存先のリソースに影響されます。依存しているリソースは、自分が起動する前に依存されているリソースが少なくとも1つの主ノードまたはゾーンで起動するまで待機します。この状況は停止と再起動、および有効化と無効化に似ています。

依存しているリソースのリソースグループが依存されているリソースのリソースグループに対して肯定的なアフィニティを持っている場合であっても、依存関係は **ANY\_NODE** のままになります。

**{FROM\_RG\_AFFINITIES}** リソースのリソースグループの **RG\_affinities** 関係に基づいて、依存関係が **LOCAL\_NODE** または **ANY\_NODE** であるかを指定します。

依存しているリソースのグループが依存されているリソースのグループに対して肯定的なアフィニティを持っていて、リソースグループが同一ノード上で起動または停止する場合、依存関係は **LOCAL\_NODE** であるとみなされます。そのような肯定的なアフィニティが存在しない場合、または異なるノード上でグループが起動する場合、依存関係は **ANY\_NODE** であるとみなされます。

修飾子を指定しない場合は、**FROM\_RG\_AFFINITIES** 修飾子がデフォルトで使用されます。

同じリソースグループ内の2つのリソース間のリソース依存関係は、常に **LOCAL\_NODE** です。

カテゴリ: 任意

デフォルト: 空のリスト

調整: **ANYTIME**

#### **Resource\_dependencies\_restart(string\_array)**

リソースが再起動の依存関係を持っているリソースのリスト。再起動の依存関係は、メソッド呼び出しの順序を決定します。

このプロパティの動作は **Resource\_dependencies** とよく似ていますが、1点例外があります。依存されているリソースと呼ばれる、再起動の依存関係リストのリソースが再起動すると、依存しているリソースと呼ばれるリソースの依存関係を有するリソースが再起動します。リスト内の依存されているリソースがオンラインに戻ったあと、RGMは依存しているリソースを停止し、再起動します。このような再起動動作が発生するのは、依存しているリソースと依存されているリソースを含むリソースグループがオンラインのままである場合です。

リソースの依存関係を有するリソースは依存しているリソースと呼ばれ、依存されているリソースと呼ばれるリスト内のリソースがオンラインでないとは依存しているリソースを起動することはできません。依存しているリソースと、リスト内のいずれかの依存されているリソースが同時に起動した場合、RGMは、リスト内の依存されているリソースが起動するまで依存しているリソースの起動を待ちます。依存されているリソースが起動しないと、依存しているリソースはオフラインのままになります。依存されているリソースが起動しない場合があるのは、リスト内の依存されている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` 状態であるためです。異なるリソースグループ内の依存されているリソースが起動に失敗したり、無効またはオフラインになったりしていることが原因で、依存しているリソースがオフラインのままになっている場合、依存しているリソースのグループは `Pending_online_blocked` 状態になります。起動に失敗した、無効である、またはオフラインである同じリソースグループ内の依存されているリソースに、依存しているリソースが依存関係を持っている場合、リソースグループは `Pending_online_blocked` 状態にはなりません。

依存関係の範囲を指定するには、このプロパティを指定するときに、次の修飾子の中括弧 {} を含めてリソース名に付加します。

`{LOCAL_NODE}` 指定される依存関係をノード単位またはゾーン単位に限定します。依存関係の動作は、同じノードまたはゾーン上でのみ依存先のリソースに影響されます。依存しているリソースは、依存先のリソースが同じノードまたはゾーンで起動されるまで待機します。この状況は停止と再起動、および有効化と無効化に似ています。

`{ANY_NODE}` 指定された依存関係を任意のノードまたはゾーンに拡張します。依存関係の動作は、どのノードまたはゾーンでも依存先のリソースに影響されます。依存しているリソースは、自分が起動する前に依存されているリソースが少なくとも1つの主ノードまたはゾーンで起動するまで待機します。この状況は停止と再起動、および有効化と無効化に似ています。

依存しているリソースのリソースグループが依存されているリソースのリソースグループに対して肯定的なアフィニティを持っている場合であっても、依存関係は `ANY_NODE` のままになります。

`{FROM_RG_AFFINITIES}` リソースのリソースグループの `RG_affinities` 関係に基づいて、依存関係が `LOCAL_NODE` または `ANY_NODE` であるかを指定します。

依存しているリソースのグループが依存されているリソースのグループに対して肯定的なアフィニティを持っていて、リソースグループが同一ノード上で起動ま

たは停止する場合、依存関係は LOCAL\_NODE であるとみなされます。そのような肯定的なアフィニティーが存在しない場合、または異なるノード上でグループが起動する場合、依存関係は ANY\_NODE であるとみなされます。

修飾子を指定しない場合は、FROM\_RG\_AFFINITIES 修飾子がデフォルトで使用されます。

同じリソースグループ内の2つのリソース間のリソース依存関係は、常に LOCAL\_NODE です。

カテゴリ: 任意

デフォルト: 空のリスト

調整: ANYTIME

#### Resource\_dependencies\_weak(string\_array)

リソースが弱い依存関係を持っているリソースのリスト。弱い依存関係は、メソッド呼び出しの順序を決定します。

依存しているリソースと呼ばれる、リソースの依存関係を有するリソースの Start メソッドの前に、RGM は、依存されているリソースと呼ばれるこのリスト内のリソースの Start メソッドを呼び出します。RGM は、依存されているリソースの Stop メソッドの前に、依存しているリソースの Stop メソッドを呼び出します。依存されているリソースが起動に失敗したり、オフラインのままであっても、依存しているリソースは依然として起動することができます。

Resource\_dependencies\_weak リストの依存しているリソースと依存されているリソースが同時に起動した場合、RGM は、リスト内の依存されているリソースが起動するまで、依存しているリソースの起動を待機します。リスト内の依存されているリソースが起動しない場合でも (たとえば、リスト内の依存されている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、リスト内の依存されているリソースが Start\_failed 状態である場合)、依存しているリソースは起動します。依存しているリソースの Resource\_dependencies\_weak リストのリソースが起動する際に、依存しているリソースのリソースグループが一時的に Pending\_online\_blocked 状態に入ることがあります。リストのすべての依存されているリソースが起動した時点、または起動に失敗した時点で、依存しているリソースは起動し、そのグループは再度 Pending\_online 状態になります。

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。同様に、Postnet\_stop メソッドは Stop メソッドよりあとに実行されます。異なるリソースグループ内では、依存されるリソースが Prenet\_start と Start を実行してから、依存するリソースが 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。

依存関係の範囲を指定するには、このプロパティを指定するときに、次の修飾子の中括弧 {} を含めてリソース名に付加します。

{LOCAL_NODE}	指定される依存関係をノード単位またはゾーン単位に限定します。依存関係の動作は、同じノードまたはゾーン上でのみ依存先のリソースに影響されます。依存しているリソースは、依存先のリソースが同じノードまたはゾーンで起動されるまで待機します。この状況は停止と再起動、および有効化と無効化に似ています。
{ANY_NODE}	指定された依存関係を任意のノードまたはゾーンに拡張します。依存関係の動作は、どのノードまたはゾーンでも依存先のリソースに影響されます。依存しているリソースは、自分が起動する前に依存されているリソースが少なくとも1つの主ノードまたはゾーンで起動するまで待機します。この状況は停止と再起動、および有効化と無効化に似ています。
{FROM_RG_AFFINITIES}	<p>依存しているリソースのリソースグループが依存されているリソースのリソースグループに対して肯定的なアフィニティを持っている場合であっても、依存関係は ANY_NODE のままになります。</p> <p>リソースのリソースグループの RG_affinities 関係に基づいて、依存関係が LOCAL_NODE または ANY_NODE であるかを指定します。</p> <p>依存しているリソースのグループが依存されているリソースのグループに対して肯定的なアフィニティを持っていて、リソースグループが同一ノード上で起動または停止する場合、依存関係は LOCAL_NODE であるとみなされます。そのような肯定的なアフィニティが存在しない場合、または異なるノード上でグループが起動する場合、依存関係は ANY_NODE であるとみなされます。</p>

修飾子を指定しない場合は、FROM\_RG\_AFFINITIES 修飾子がデフォルトで使用されます。

同じリソースグループ内の2つのリソース間のリソース依存関係は、常に LOCAL\_NODE です。

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: 必要  
 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。このプロパティは、CPU の共有、クラスタデータサービスのリソースプールといった Solaris のリソース管理機能に適用できます。RGM は、リソースをオンラインにすると、このプロジェクト名を持つ関連プロセスを起動します。このプロパティが指定されていない場合、プロジェクト名は、リソースを含むリソースグループの `RG_project_name` プロパティから取得されます (`rg_properties(5)` のマニュアルページを参照)。どちらのプロパティも指定されなかった場合、RGM は事前定義済みのプロジェクト名 `default` を使用します。プロジェクトデータベース内に存在するプロジェクト名を指定する必要があります (`projects(1)` のマニュアルページと『System Administration Guide: Solaris Containers-Resource Management and Solaris Zones』を参照)。

このプロパティは Solaris 9 OS からサポートされるようになりました。

---

注- このプロパティへの変更は、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

**各クラスタノードまたはゾーン上の Resource\_state (enum)**

RGM が判断した各クラスタノードまたはゾーン上のリソースの状態。この状態には、`Online`、`Offline`、`Start_failed`、`Stop_failed`、`Monitor_failed`、`Online_not_monitored`、`Starting`、`Stopping` があります。

ユーザーはこのプロパティを構成できません。

カテゴリ: 照会のみ  
 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またはゾーン上にとどまることを許可する
- 別のノードまたはゾーンへのリソースグループのフェイルオーバーを要求します。

このプロパティは RGM によって作成されます。RTR ファイルに宣言されている場合、クラスタ管理者のみ使用を許可されます。RTR ファイル内でデフォルト値が指定されている場合、このプロパティは任意です。

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

---

注- このプロパティ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を再起動を試みます。

ただし、一部のリソースタイプでは、Retry\_count に負の値を設定できません。より確実に無制限の再起動を指定するには、次の手順を実行します。

- Retry\_interval に 1 や 0 などの小さい値を指定します。
  - Retry\_count に 1000 などの大きい値を指定します。
- 

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、このプロパティと Retry\_count を組み合わせて使用します。このプロパティは RGM によって作成されます。RTR ファイルに宣言されている場合にかぎり、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ク負荷分散機能を使用するかどうかを表します。

---

注- スケーラブルなリソースグループ(ネットワーク負荷分散を使用)を、非大域ゾーンで動作するよう構成することができます。ただし、そのようなスケーラブルなリソースグループを実行できるのは、物理ノードごとに1つのゾーン内だけです。

---

このプロパティがRTRファイルで宣言されている場合は、そのタイプのリソースに対して、RGMは、次のスケーラブルサービスプロパティを自動的に作成します。Affinity\_timeout、Load\_balancing\_policy、Load\_balancing\_weights、Network\_resources\_used、Port\_list、UDP\_affinity、Weak\_affinity。これらのプロパティは、RTRファイル内で明示的に宣言されない限り、デフォルト値を持ちます。RTRファイル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が宣言されている場合、AT\_CREATION以外のTunable属性の割り当ては許可されません。

RTRファイルにこのプロパティが宣言されていない場合、このリソースはスケーラブルではないため、このプロパティを調整することはできません。RGMは、スケーラブルサービスプロパティをいっさい設定しません。ただし、Network\_resources\_usedおよびPort\_listプロパティは、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とFailoverリソースタイププロパティの併用については、r\_properties(5)のマニュアルページで詳しく説明されています。

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

各クラスタノードまたはゾーン上のStatus (enum)

scha\_resource\_setstatus コマンドまたはscha\_resource\_setstatus() 関数またはscha\_resource\_setstatus\_zone() 関数を使用してリソースモニターにより設定され

ます。取り得る値は 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

各クラスターノードまたはゾーン上の Status\_msg (string)

リソースモニターによって、Status プロパティと同時に設定されます。リソースがオンラインまたはオフラインにされると、RGM は自動的にこのプロパティを空文字列でリセットします。ただし、このプロパティ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、クラスター管理者は使用を許可されます。RTR ファイル内でデフォルト値が指定されている場合、このプロパティは任意です。

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: 必要  
 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。このプロパティはRTRファイル内に宣言できません。したがって、RGMによって自動的に作成されます。このプロパティの値は、リソースタイプのRT\_versionプロパティと等しくなります。リソースの作成時、Type\_versionプロパティはリソースタイプ名の接尾辞として表示されるだけで、明示的には指定されません。リソースを編集すると、Type\_versionプロパティ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)

このプロパティがTRUEに設定されている場合、このプロパティにより弱い形式のクライアントアフィニティが有効になります。

弱い形式のクライアントアフィニティ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。

弱いアフィニティはメモリーの消費とプロセッササイクルの点で、デフォルトの形式よりもオーバーヘッドを低く抑えられます。

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

このプロパティは、スケーラブルサービス専用です。

カテゴリ: 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

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を示します。

プロパティ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でリソースグループを作成するときに、必ず値を指定する必要があります。
- 任意。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場合、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。

以下にプロパティ名とその説明を示します。

#### Auto\_start\_on\_new\_cluster (boolean)

このプロパティは、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 (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 に設定した場合、クラスタの再起動時にリソースグループが自動的に再起動することはありません。 `clresourcegroup online` 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またはゾーン数として望ましい値。

デフォルトは 1 です。 `Desired_primaries` プロパティの値は、 `Maximum_primaries` プロパティの値以下にしてください。

カテゴリ: 任意

デフォルト: 1

調整: ANYTIME

#### Failback (boolean)

ノードまたはゾーンがクラスタに結合した場合、グループがオンラインとなるノード群またはゾーン群を再計算するかどうかを示すブール値。再計算により、RGM は優先度の低いノードまたはゾーンをオフラインにし、優先度の高いノードまたはゾーン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型があります。

スケラブルリソースグループの場合、ネットワークアドレスリソースを含んでいないため、このプロパティの影響はありません。

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またはゾーンの最大数です。

RG\_mode プロパティが Failover である場合、このプロパティの値は 1 以下である必要があります。RG\_mode プロパティが 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)

リソース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またはゾーンのリストです。これらのノードまたはゾーンは、リソースグループの潜在的な主ノードまたはマスターになります。

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です。Pathprefixの値はリソースグループごとに固有の値を指定します。

カテゴリ: 任意

デフォルト: 空の文字列

調整: ANYTIME

#### Pingpong\_interval(integer)

負数ではない整数値(秒)。次のような状況においてRGMは、この値を使って、リソースグループをどこでオンラインにするかを決めます。

- 再構成が発生している場合。
- GIVEOVER 引数付きで `scha_control` コマンドを実行した、または `SCHA_GIVEOVER` 引数付きで `scha_control()` 関数を実行した結果として。

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またはゾーン上で複数回、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、そのノードまたはゾーン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 -O GIVEOVER` コマンドが特定のノードまたはゾーン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またはゾーン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)最初のノードまたはゾーンは、同じリソースによる別の `scha_control -O 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を直接設定しません。このプロパティは、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によって更新されます。

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またはゾーン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またはゾーン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に対して弱いポジティブアフィニティーを、RG3に対して強いネガティブアフィニティーをもつことを表しています。

RG\_affinities プロパティ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)

同じノードまたはゾーン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(ポジティブおよびネガティブ)とRG\_dependenciesの関係図式の中に循環が含まれていてはなりません。

たとえば、リソースグループRG1のRG\_dependenciesリストにリソースグループRG2がリストされている、つまりRG1がRG2に対してリソースグループの依存関係を持っているとします。

次のリストに、リソースグループ依存関係の影響を要約します。

- ノードまたはゾーンがクラスタに結合されると、そのノードまたはゾーンでは、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またはゾーン上で同時に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またはゾーン上で同時に `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 の主ノードまたはゾーン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またはゾーンでオンラインに、RG2 がすべてのノードまたはゾーンでオフラインになる場合は、このスイッチは失敗します。詳細は、`clresourcegroup(1CL)` および `clsetup(1CL)` のマニュアルページを参照してください。
- RG2 に対する `Desired primaries` がゼロに設定されている場合は、RG1 に対する `Desired primaries` プロパティをゼロより大きい値に設定することはできません。
- 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グローバルデバイスをスイッチオーバーするかどうかを表します。このプロパティが `TRUE` に設定されている場合、大域デバイスはスイッチオーバーされます。このプロパティが `FALSE` に設定されている場合、グローバル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` プロパティの設定によります。

`RG_is_frozen` プロパティをユーザーが直接設定することはありません。

`RG_is_frozen` プロパティは、大域デバイスのステータスが変化したときに、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またはゾーンに制限します。

このプロパティの値が **Scalable** に設定されていれば、**Maximum primaries** プロパティは 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ことが可能です。**Failover** プロパティの値が **TRUE** のリソースを、**RG\_mode** の値が **Scalable** のリソースグループに追加することはできません。

**Maximum primaries** が 1 の場合、デフォルトは **Failover** です。**Maximum primaries** が 1 より大きい場合、デフォルトは **Scalable** です。

カテゴリ: 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: 必要

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びもと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)。プロジェクトデータベース内に存在するプロジェクト名を指定する必要があります (**projects(1)** のマニュアルページと『**System Administration Guide: Solaris Containers-Resource Management and Solaris Zones**』を参照)。

このプロパティは Solaris 9 OS からサポートされるようになりました。

---

注- このプロパティへの変更は、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

RG\_slm\_cpu (decimal number)

RG\_slm\_type プロパティが AUTOMATED に設定されている場合、この数は CPU シェアの数およびプロセッサセットのサイズの計算の基準になります。

---

注 - RG\_slm\_cpu プロパティを使用できるのは、RG\_slm\_type が AUTOMATED に設定されている場合のみです。詳細は、「RG\_slm\_type プロパティ」を参照してください。

---

RG\_slm\_cpu プロパティの最大値は 655 です。小数点のあとには 2 桁を含めることができます。RG\_slm\_cpu プロパティには 0 を指定しないでください。シェアの値を 0 に設定すると、CPU 負荷が高い場合に、公平配分スケジューラ (FFS) によりリソースをスケジューリングできない場合があります。

リソースグループがオンラインである間に RG\_slm\_cpu プロパティに対して行う変更は、動的に考慮されます。

RG\_slm\_type プロパティは AUTOMATED に設定されているため、Sun Cluster は SCSLM\_resourcegroupname という名前のプロジェクトを作成します。resourcegroupname は、ユーザーがリソースグループに割り当てる実際の名前を表します。リソースグループに属するリソースの各メソッドは、このプロジェクトで実行されます。Solaris 10 から、これらのプロジェクトは、リソースグループのゾーンが大域ゾーンであれ非大域ゾーンであれ、リソースグループのゾーン内に作成されます。project(4) のマニュアルページを参照してください。

プロジェクト SCSLM\_resourcegroupname の project.cpu-shares 値は、RG\_slm\_cpu のプロパティ値の 100 倍です。RG\_slm\_cpu プロパティが設定されていない場合、このプロジェクトは値 1 の project.cpu-shares を使用して作成されます。RG\_slm\_cpu プロパティのデフォルト値は 0.01 です。

Solaris 10 OS から、RG\_slm\_pset\_type プロパティが DEDICATED\_STRONG または DEDICATED\_WEAK に設定されている場合、プロセッサセットのサイズの計算には RG\_slm\_cpu プロパティが使用されます。また、RG\_slm\_cpu プロパティは zone.cpu-shares の値の計算にも使用されます。

プロセッサセットについては、『System Administration Guide: Solaris Containers-Resource Management and Solaris Zones』を参照してください。

カテゴリ: 任意

デフォルト: 0.01

調整: ANYTIME

**RG\_slm\_cpu\_min** (decimal number)

アプリケーションが動作できるプロセッサの最小数を決定します。

このプロパティは、次に示す要因がすべて真の場合だけ使用できます。

- **RG\_slm\_type** プロパティが **AUTOMATED** に設定されている
- **RG\_slm\_pset\_type** プロパティが **DEDICATED\_STRONG** または **DEDICATED\_WEAK** に設定されている
- **RG\_slm\_cpu** プロパティが、**RG\_slm\_cpu\_min** プロパティに対して設定されている値以上の値に設定されている
- Solaris 10 OS を使用している

**RG\_slm\_cpu\_min** プロパティの最大値は 655 です。小数点のあとには 2 桁を含めることができます。**RG\_slm\_cpu\_min** プロパティには 0 を指定しないでください。

**RG\_slm\_cpu\_min** および **RG\_slm\_cpu** プロパティは、それぞれ、Sun Cluster が生成するプロセッサセットに対して **pset.min** および **pset.max** の値を決定します。

リソースグループがオンラインである間にユーザーが **RG\_slm\_cpu** および **RG\_slm\_cpu\_min** プロパティに対して行う変更は、動的に考慮されます。**RG\_slm\_pset\_type** プロパティが **DEDICATED\_STRONG** に設定され、使用できる CPU が十分でない場合、**RG\_slm\_cpu\_min** プロパティに対してユーザーが要求した変更は無視されます。この場合は、警告メッセージが表示されます。次のスイッチオーバー時に、**RG\_slm\_cpu\_min** プロパティが使用できる CPU が十分でない場合、CPU の不足によるエラーが発生する可能性があります。

プロセッサセットについては、『System Administration Guide: Solaris Containers-Resource Management and Solaris Zones』を参照してください。

カテゴリ: 任意

デフォルト: 0.01

調整: ANYTIME

**RG\_slm\_type** (string)

システムリソースの使用状況を管理できるようにし、システムリソース管理用に Solaris オペレーティングシステムを設定する手順の一部を自動化します。

**RG\_SLM\_type** が取り得る値は **AUTOMATED** と **MANUAL** です。

**RG\_slm\_type** プロパティを **AUTOMATED** に設定した場合、リソースグループは CPU 使用率の制御とともに起動します。

その結果、Sun Cluster は次の処理を行います。

- `SCSLM_resourcegroupname` という名前のプロジェクトを作成します。このリソースグループ内のリソースのすべてのメソッドは、このプロジェクト内で実行されます。このプロジェクトは、このリソースグループ内のリソースのメソッドがノードまたはゾーンで初めて実行されるときに作成されます。
- プロジェクトと関連付けられている `project.cpu_shares` の値を、`RG_slm_cpu` プロパティの 100 倍の値に設定します。デフォルトでは、`project.cpu_shares` の値は 1 です。
- Solaris 10 OS からは、`zone.cpu_shares` を、すべてのオンラインリソースグループの `RG_slm_cpu` プロパティの合計の 100 倍に設定します。またこのプロパティは、当該ゾーン内で `RG_slm_type` を `AUTOMATED` に設定します。ゾーンは大域または非大域の場合があります。非大域ゾーンは、Sun Cluster が生成するプールにバインドされます。オプションで、`RG_slm_pset_type` プロパティが `DEDICATED_WEAK` または `DEDICATED_STRONG` に設定されている場合、Sun Cluster の生成されたプールは、Sun Cluster の生成されたプロセッサセットと関連付けられます。専用のプロセッサセットについては、`RG_slm_pset_type` プロパティの説明を参照してください。`RG_slm_type` プロパティを `AUTOMATED` に設定した場合、実行されるすべての処理はログに記録されます。

`RG_slm_type` プロパティを `MANUAL` に設定した場合、`RG_project_name` プロパティにより指定されているプロジェクト内でリソースグループが実行されます。

リソースプールとプロセッサセットについては、『System Administration Guide: Solaris Containers-Resource Management and Solaris Zones』を参照してください。

---

注-

- 58 文字を超えるリソースグループ名は指定しないでください。リソースグループ名が 58 文字を超える場合、CPU 制御を構成できなくなる、つまり、`RG_slm_type` プロパティに `AUTOMATED` を設定できなくなります。
  - リソースグループ名にはダッシュ (-) を含めないでください。Sun Cluster ソフトウェアは、プロジェクトの作成時に、リソースグループ名にあるすべてのダッシュを下線 () に置き換えます。たとえば、Sun Cluster が `rg-dev` というリソースグループに対して `SCSLM_rg_dev` というプロジェクトを作成する場合があります。Sun Cluster がリソースグループ `rg-dev` に対してプロジェクトを作成しようとするとき、`rg_dev` という名前のリソースグループがすでに存在する場合、衝突が発生します。
- 

カテゴリ: 任意

デフォルト: `manual`

調整: `ANYTIME`

**RG\_slm\_pset\_type(string)**

専用のプロセッサセットの作成を可能にします。

このプロパティは、次に示す要因がすべて真の場合だけ使用できます。

- RG\_slm\_type プロパティが **AUTOMATED** に設定されている
- Solaris 10 OS を使用している
- リソースグループが非大域ゾーンで実行される

RG\_slm\_pset\_type の取り得る値は **DEFAULT**、**DEDICATED\_STRONG**、および **DEDICATED\_WEAK** です。

リソースグループが **DEDICATED\_STRONG** または **DEDICATED\_WEAK** として実行される場合は、そのリソースグループのノードリストには非大域ゾーンだけが存在するようにリソースグループを設定してください。

非大域ゾーンは、デフォルトプールである **POOL\_DEFAULT** 以外のプールに対して設定しないでください。ゾーン構成の詳細は、**zonecfg(1M)** のマニュアルページを参照してください。非大域ゾーンは、デフォルトプール以外のプールに動的にバインドしないでください。プールバインディングの詳細は、**poolbind(1M)** のマニュアルページを参照してください。バインドされた2つのプールの状態は、リソースグループ内のリソースのメソッドが起動されている場合だけ、確認されません。

**DEDICATED\_STRONG** と **DEDICATED\_WEAK** の値は、ノードリストに同じゾーンを持つリソースグループと相互に排他的です。同じゾーン内では、一部のリソースグループの **RG\_slm\_pset\_type** が **DEDICATED\_STRONG** に設定され、ほかのリソースグループの **RG\_slm\_pset\_type** が **DEDICATED\_WEAK** に設定されるように、リソースグループを構成することはできません。

**RG\_slm\_pset\_type** プロパティを **DEDICATED\_STRONG** に設定した場合、Sun Cluster は、**RG\_slm\_type** プロパティが **AUTOMATED** に設定されている場合に **RG\_slm\_type** プロパティにより実行されるアクション以外にも、次の処理を行います。

- プールを作成し、リソースグループが **PRENET\_START** メソッドと **START** メソッドの一方または両方に対して起動する非大域ゾーンにそのプールを動的にバインドする。
- 次の合計の間のサイズを持つプロセッサセットを作成する。
  - 当該リソースグループが起動するゾーンでオンラインであるすべてのリソースグループ内の **RG\_slm\_cpu\_min** プロパティの合計。
  - 当該ゾーンで実行中であるリソースグループ内の **RG\_slm\_cpu** プロパティの合計。

**STOP** メソッドまたは **POSTNET\_STOP** メソッドのいずれかが実行中である場合、Sun Cluster の生成されたプロセッサセットは破棄されます。リソースグループがゾーン内でオンラインでなくなった場合、そのプールは破棄され、非大域ゾーンはデフォルトのプール (**POOL\_DEFAULT**) にバインドされます。

- プロセッサセットをプールに関連付けます。
- ゾーンを実行しているすべてのリソースグループの `RG_slm_cpu` プロパティの合計の 100 倍に `zone.cpu_shares` を設定します。

ユーザーが `RG_slm_pset_type` プロパティを `DEDICATED_WEAK` に設定した場合、リソースグループの動作は、`RG_slm_pset_type` が `DEDICATED_STRONG` に設定されている場合と同じようになります。しかし、プロセッサセットの作成に十分なプロセッサを使用できない場合、プールはデフォルトのプロセッサセットに関連付けられます。

ユーザーが `RG_slm_pset_type` プロパティを `DEDICATED_STRONG` に設定し、またプロセッサセットの作成に十分なプロセッサを使用できない場合、エラーが発生します。その結果、リソースグループは当該ノードまたはゾーン上では起動しません。

CPU が割り当てられている場合、`DEFAULTPSETMIN` 最小サイズは `DEDICATED_STRONG` よりも優先されます。`DEDICATED_STRONG` は `DEDICATED_WEAK` よりも優先されます。ただし、`clnode` コマンドを使用してデフォルトのプロセッサのサイズを大きくし、また十分なプロセッサが使用できない場合、この優先順位は無視されます。`DEFAULTPSETMIN` プロパティの詳細は、`clnode(1CL)` のマニュアルページを参照してください。

`clnode` コマンドは、デフォルトのプロセッサセットに最小限の CPU を動的に割り当てます。ユーザーが指定した CPU の数が使用できない場合、Sun Cluster は定期的にこの数の CPU を割り当てようとします。それに失敗すると、CPU の最小数が割り当てられるまで、Sun Cluster はデフォルトのプロセッサセットにより少ない数の CPU を割り当てようとします。このアクションは一部の `DEDICATED_WEAK` プロセッサセットを破棄する場合がありますが、`DEDICATED_STRONG` プロセッサセットを破棄することはありません。

ユーザーが `RG_slm_pset_type` プロパティを `DEDICATED_STRONG` に設定したりリソースグループを起動する場合、`DEDICATED_WEAK` プロセッサセットと関連付けられたプロセッサセットが破棄される場合があります。このリソースグループがこのような動作を行う場合があるのは、両方のプロセッサセットのノードまたはゾーン上で十分な CPU が使用できない場合です。この場合、`DEDICATED_WEAK` プロセッサセットで動作しているリソースグループのプロセスは、デフォルトのプロセッサセットに関連付けられます。

`DEDICATED_STRONG` または `DEDICATED_WEAK` の間で `RG_slm_pset_type` プロパティの値を交換するには、まずその値をデフォルトに設定します。

CPU 制御に対して構成されたリソースグループが非大域ゾーンでオンラインではない場合、CPU シェアの値はそのゾーンの `zone.cpu-shares` に設定されます。デフォルトでは、`zone.cpu-shares` は 1 に設定されています。ゾーン構成の詳細は、`zonestat(1M)` のマニュアルページを参照してください。

ユーザーが `RG_slm_pset_type` プロパティを `DEFAULT` に設定すると、Sun Cluster は `SCSLM_pool_zonename` という名前のプールを作成しますが、プロセッサセットは作成しません。この場合、`SCSLM_pool_zonename` はデフォルトのプロセッサセットに関連付けられます。ゾーンに割り当てられるシェアは、そのゾーン内のすべてのリソースグループの `RG_slm_cpu` の値の合計と等しくなります。

リソースプールとプロセッサセットについては、『System Administration Guide: Solaris Containers-Resource Management and Solaris Zones』を参照してください。

カテゴリ: 任意

デフォルト: `default`

調整: `ANYTIME`

各クラスタノードまたはゾーン上の `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、各クラスタノードまたはゾーン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を構成できません。しかし、`clresourcegroup` コマンドを実行することによって、あるいは同等の `clsetup` や Sun Cluster Manager コマンドを使用して、このプロパティを間接的に設定することは可能です。RGM の制御下にはないときは、グループは管理されていない状態で存在可能です。

各状態の説明は次のとおりです。

---

注 - すべてのノードまたはゾーンに適用される `UNMANAGED` 状態を除き、状態は個別のノードまたはゾーン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 のゾーン 1 では `OFFLINE` であり、ノード B のゾーン 2 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、過去には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またはゾーン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

	<p>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。</p>
OFFLINE	<p>リソースグループはノードまたはゾーンですでに停止されています。つまり、グループ内の有効なリソースすべてに対して、停止メソッド Monitor_stop、Stop、および Postnet_stop が (各リソースに合わせて) 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またはゾーン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。</p>
PENDING_ONLINE	<p>リソースグループはノードまたはゾーン上で起動中です。グループ内の有効なリソースに対して、起動メソッド Prenet_start、Start、および Monitor_start が (各リソースに合わせて) 実行されようとしています。</p>
PENDING_OFFLINE	<p>リソースグループはノードまたはゾーン上で停止中です。グループ内の有効なリソースに対して、停止メソッド Monitor_stop、Stop、および Postnet_stop が (各リソースに合わせて) 実行されようとしています。</p>
ERROR_STOP_FAILED	<p>リソースグループ内の 1 つ以上のリソースが正常に停止できず、Stop_failed 状態にあります。グループ内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またはゾーン上での起動が許可されません。</p> <p>clresource clear 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 に再設定します。</p>
ONLINE_FAULTED	<p>リソースグループは PENDING_ONLINE で、このノードまたはゾーン上での起動が完了しています。ただし、1 つまたは複数の</p>

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。リソースの依存性が満たされると、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

`Suspend_automatic_recovery` (boolean)

リソースグループの自動復旧が中断されるかどうかを指定するブール値です。クラスタ管理者が自動復旧を再開するコマンドを明示的に実行するまで、中断されたリソースグループが自動的に再開またはフェイルオーバーされることはありません。中断されたデータサービスは、オンラインかオフラインにかかわらず、現在の状態のままとなります。指定したノードまたはゾーン上でリソースグループの状態を手作業で切り替えることもできます。また、リソースグループ内の個々のリソースも有効または無効にできます。

`Suspend_automatic_recovery` プロパティに `TRUE` が設定されると、リソースグループの自動復旧は中断されます。このプロパティが `FALSE` に設定されると、リソースグループの自動復旧が再開され、アクティブになります。

このプロパティを直接設定することはありません。RGM は、クラスタ管理者がリソースグループの自動復旧を中断または再開したときに

`Suspend_automatic_recovery` プロパティの値を変更します。クラスタ管理者は、`clresourcegroup suspend` コマンドで自動復旧を中断します。クラスタ管理者は、`clresourcegroup resume` コマンドで自動復旧を再開します。RG\_system プロパティの設定にかかわらず、リソースグループは中断または再開できます。

カテゴリ: 照会のみ

デフォルト: FALSE

調整: NONE

`RG_system` (boolean)

リソースグループの `RG_system` プロパティの値が `TRUE` の場合、そのリソースグループとそのリソースグループ内のリソースに関する特定の操作が制限されま

す。この制限は、重要なリソースグループやリソースを間違え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。このプロパティにより影響を受けるのは `clresourcegroup` コマンドのみです。 `scha_control(1HA)` と `scha_control(3HA)` の操作には影響を与えません。

リソースグループ(またはリソースグループ内のリソース)の制限操作を実行する前には、まず、リソースグループの `RG_system` プロパティを `FALSE` に設定する必要があります。クラスタサービスをサポートするリソースグループ(または、リソースグループ内のリソース)を変更または削除するときには注意してください。

操作	例
リソースグループを削除する	<code>clresourcegroup delete RG1</code>
リソースグループプロパティを編集する ( <code>RG_system</code> を除く)	<code>clresourcegroup set -p RG_description=... +</code>
リソースグループへソースを追加する	<code>clresource create -g RG1 -t SUNW.nfs R1</code> リソースは作成後に有効な状態になり、リソース監視も有効になります。
リソースグループからリソースを削除する	<code>clresource delete R1</code>
リソースグループに属するリソースのプロパティを編集する	<code>clresource set -g RG1 -t SUNW.nfs -p r_description="HA-NFS res" R1</code>
リソースグループをオフラインに切り替える	<code>clresourcegroup offline RG1</code>
リソースグループを管理する	<code>clresourcegroup manage RG1</code>
リソースグループを管理しない	<code>clresourcegroup unmanage RG1</code>
リソースグループのリソースを有効にする	<code>clresource enable R1</code>
リソースグループのリソースに対する監視を有効にする	<code>clresource monitor R1</code>
リソースグループのリソースを無効にする	<code>clresource disable R1</code>
リソースの監視を無効にする	<code>clresource unmonitor R1</code>

リソースグループの `RG_system` プロパティの値が `TRUE` の場合、そのリソースグループで編集できるプロパティは `RG_system` プロパティ自体だけです。つまり、`RG_system` プロパティ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の属性

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

プロパティのデフォルト値を示します。

### Description

プロパティを簡潔に記述した注記(文字列)。RTR ファイル内でシステム定義プロパティに対する Description 属性を設定することはできません。

### Enumlist

enum タイプの場合、プロパティに設定できる文字列値のセット。

### Extension

リソースタイプの実装によって定義された拡張プロパティが RTR ファイルのエントリで宣言されていることを示します。拡張プロパティが使用されていない場合、そのエントリはシステム定義プロパティです。

### Max

int タイプの場合、プロパティに設定できる最大値。

### Maxlength

string および stringarray タイプの場合、設定できる文字列の長さの最大値。

### Min

int タイプの場合、プロパティに設定できる最小値。

### Minlength

string および stringarray タイプの場合、設定できる文字列の長さの最小値。

### Per\_node

使用された場合、拡張プロパティがノード単位またはゾーン単位で設定できることを示します。

Per\_node プロパティ属性をタイプ定義で指定する場合は、Default プロパティ属性でデフォルト値も指定してください。デフォルト値を指定すると、明示的な

値が割り当てられていないノードまたはゾーン上でノード単位またはゾーン単位のプロパティをユーザーが要求した場合に、値が返されることが保証されません。

タイプ `stringarray` のプロパティには `Per_node` プロパティ属性を指定できません。

#### Property

リソースプロパティの名前。

#### Tunable

クラスタ管理者がリソースのプロパティ値をいつ設定できるかを示します。クラスタ管理者にプロパティの設定を許可しない場合は、`NONE` または `FALSE` に設定します。クラスタ管理者にプロパティ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。ほかの条件 (「監視をいつ無効にするか」や「いつオフラインにするか」など) を設定する場合は、この値を `ANYTIME` に設定し、`Validate` メソッドを使って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、拡張プロパティを調整する設定のデフォルトは `TRUE (ANYTIME)` です。

#### プロパティの型

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の型の属性を設定することはできません。タイプは、RTR ファイルのエントリに登録できる、指定可能なプロパティ値とタイプ固有の属性を決定します。`enum` タイプは、文字列値のセットです。



## 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。

この付録の内容は次のとおりです。

- 235 ページの「有効な RGM 名」
- 237 ページの「RGM の値」

### 有効な RGM 名

RGM 名は、次のカテゴリに分類されます。

- リソースグループ名
- リソースタイプ名
- リソース名
- プロパティ名
- 列挙型リテラル名

### 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。

<i>vendor-id</i>	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 <code>Vendor_id</code> リソースタイププロパティの値です。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 で開発されるリソースタイプのベンダー ID 接頭辞は <code>SUNW</code> です。
<i>base-rt-name</i>	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 <code>Resource_type</code> リソースタイププロパティの値です。
<i>rt-version</i>	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 <code>RT_version</code> リソースタイププロパティの値です。バージョン接尾辞は、RTR ファイルが <code>#\$upgrade</code> 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。 <code>#\$upgrade</code> 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、管理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。

---

詳細は、[183 ページの「資源タイプのプロパティ」](#)を参照してください。

例 C-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。

```
SUNW.sample:2.0
```

例 C-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
  - セミコロン (;)



# 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。

---

## 設定ワークシート

この付録には次のワークシートが収録されています。

- 240 ページの「リソースタイプのワークシート」
- 242 ページの「ネットワークリソースのワークシート」
- 244 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」
- 246 ページの「アプリケーションリソース—スケラブルのワークシート」
- 248 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」
- 250 ページの「リソースグループ—スケラブルのワークシート」



例D-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
<b>SUNW.nshttp</b>	<b>phys-schost-1, phys-schost-2</b>
<b>SUNW.oracle_listener</b>	<b>phys-schost-1, phys-schost-2</b>
<b>SUNW.oracle_server</b>	<b>phys-schost-1, phys-schost-2</b>

# ネットワークリソースのワークシート

コンポーネント	名前	
リソース名		
リソースグループ名		
リソースタイプ (1つに丸を付けてください)	論理ホスト名   共有アドレス	
リソースタイプ名		
依存性		
使用されているホスト名		
拡張プロパティ	名称	値

## 例 D-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	<b>sh-galileo</b>	
リソースグループ名	<b>rg-shared</b>	
リソースタイプ (1つに丸を付けてください)	<b>Shared address</b>	
リソースタイプ名	<b>SUNW.SharedAddress</b>	
依存性	<b>none</b>	
使用されているホスト名	<b>sh-galileo</b>	
拡張プロパティ	名称	値
	<b>netiflist</b>	<b>ipmp0@1, ipmp0@2</b>

## 例 D-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	<b>relo-galileo</b>	
リソースグループ名	<b>rg-oracle</b>	
リソースタイプ (1つに丸を付けてください)	<b>Logical hostname</b>	
リソースタイプ名	<b>SUNW.LogicalHostname</b>	
依存性	<b>none</b>	
使用されているホスト名	<b>relo-galileo</b>	
拡張プロパティ	名称	値
	<b>netiflist</b>	<b>ipmp0@1, ipmp0@2</b>

# 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		
依存性		
拡張プロパティ	名称	値

## 例D-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	<b>oracle-listener</b>	
リソースグループ名	<b>rg-oracle</b>	
リソースタイプ名	<b>SUNW.oracle_listener</b>	
依存性	<b>hasp_resource</b>	
拡張プロパティ	名称	値
	<b>ORACLE_HOME</b>	<b>/global/oracle/orahome/</b>
	<b>LISTENER_NAME</b>	<b>lsnr1</b>

# アプリケーションリソースー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		
依存性		
拡張プロパティ	名称	値

例D-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	<b>sh-galileo</b>	
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	<b>rg-shared</b>	
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		
依存性		
拡張プロパティ	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	注	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 D-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	注	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。	<b>rg-oracle</b>
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働系を指定し、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。	<b>1) phys-schost-1</b> <b>2) phys-schost-2</b>
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。	<b>schost1-dg</b>
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	注	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		
稼働系の最大数		
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)	稼働系が停止したあと、このリソースグループを稼働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働系を指定し、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。	
依存性	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 D-7 例:リソースグループースケーラブルのワークシート

コンポーネント	注	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。	<b>rg-http</b>
機能		Web サーバーリソースを含む
稼働系の最大数		<b>2</b>
主ノードの適切な数		<b>2</b>
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)	稼働系が停止したあと、このリソースグループを稼働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働系を指定し、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。	<b>1) phys-schost-1</b> <b>2) phys-schost-2</b>
依存性	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。	<b>rg-shared</b>



# 索引

---

## A

Affinity\_timeout, リソースプロパティ, 194  
API\_version, リソースタイププロパティ, 184  
Array\_maxsize, リソースプロパティ属性, 232  
Array\_minsize, リソースプロパティ属性, 232  
Auto\_start\_on\_new\_cluster, リソースグループプロパティ, 217  
auxnodelist, ノードリストプロパティ, 22-23

## B

Boot, リソースタイププロパティ, 185  
Boot\_timeout, リソースプロパティ, 194

## C

Cheap\_probe\_interval, リソースプロパティ, 195  
CheckNameService 拡張プロパティ, 88  
claccess, 172  
cldev, 173  
cldevice, 173  
cldevicegroup, 174  
cldg, 174  
clearing, Start\_failed リソース状態, 91-93  
clinterconnect, 174  
clintr, 174  
clnas, 175  
clnasdevice, 175  
clnode, 175

clnode コマンド, 13  
clq, 176  
clqs, 176  
clquorum, 176  
clquorumserver, 176  
clreslogicalhostname, 177  
clresource, 177  
clresourcegroup, 178  
clresourcetype, 179  
clressharedaddress, 179  
clrg, 178  
clrs, 177  
clrslh, 177  
clrssa, 179  
clrt, 179  
clsetup ユーティリティ, 26  
clsetup ユーティリティ  
共有アドレス  
リソースグループに追加, 55-57  
論理ホスト名  
リソースグループに追加, 49-52  
clsnmphot, 179  
clsnmpmib, 180  
clsnmpuser, 180  
clta, 181  
cltelemetryattribute, 181  
cluster, 181  
clvxvm, 181  
colocation, オンラインリソースグループに対する  
強制, 145-146

**D**

Default, リソースプロパティ属性, 232  
Description, リソースプロパティ属性, 232  
Desired\_primaries, リソースグループプロパティ, 217

**E**

Enumlist, リソースプロパティ属性, 232  
/etc/vfstab ファイル  
    エントリの削除, 135  
    エントリの追加, 131

**F**

Failback, リソースグループプロパティ, 217  
Failover, リソースタイププロパティ, 185  
Failover\_mode, リソースプロパティ, 195  
Failover\_mode システムプロパティ, 170  
Fini, リソースタイププロパティ, 186  
Fini\_timeout, リソースプロパティ, 199

**G**

Global\_resources\_used, リソースグループプロパティ, 218  
Global\_zone, リソースタイプのプロパティ, 187

**H**

HASStoragePlus, アップグレードHASStorage, 127-130  
HASStoragePlus リソース  
    クラスタファイルシステム  
        構成, 116-117  
        ローカルファイルシステムからの変更, 140-141  
HASStoragePlus リソースタイプ  
    アップグレード, 141-142  
    インスタンスの変更, 130-140  
    インスタンスの変更の失敗, 138-139, 139-140  
    概要, 20-21

HASStoragePlus リソースタイプ (続き)  
    使用基準, 20-21  
    注意事項, 134  
    リソースタイプのバージョン, 141

**I**

Implicit\_network\_dependencies, リソースグループプロパティ, 218  
Init, リソースタイププロパティ, 188  
Init\_nodes, リソースタイププロパティ, 188  
Init\_timeout, リソースプロパティ, 199  
installed\_nodes, ノードリストプロパティ, 22  
Installed\_nodes, リソースタイププロパティ, 188  
Is\_logical\_hostname, リソースタイププロパティ, 188  
Is\_shared\_address, リソースタイププロパティ, 189

**L**

Load\_balancing\_policy, リソースプロパティ, 199  
Load\_balancing\_weights, リソースプロパティ, 199

**M**

Max, リソースプロパティ属性, 232  
Maximum\_primaries, リソースグループプロパティ, 218  
Maxlength, リソースプロパティ属性, 232  
messages ファイル, 13  
Min, リソースプロパティ属性, 232  
Minlength, リソースプロパティ属性, 232  
Monitor\_check, リソースタイププロパティ, 189  
Monitor\_check\_timeout, リソースプロパティ, 200  
Monitor\_start, リソースタイププロパティ, 189  
Monitor\_start\_timeout, リソースプロパティ, 200

Monitor\_stop, リソースタイププロパティ, 189  
 Monitor\_stop\_timeout, リソースプロパティ  
 , 200  
 Monitored\_switch, リソースプロパティ, 200

## N

Network\_resources\_used, リソースプロパティ  
 , 201  
 nodelist, ノードリストプロパティ, 22  
 Nodelist, リソースグループプロパティ, 218  
 Nodelist リソースグループプロパティ, とア  
 フィニティ, 144  
 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  
 , 201  
 Num\_rg\_restarts, リソースプロパティ, 202

## O

On\_off\_switch, リソースプロパティ, 202

## P

Pathprefix, リソースグループプロパティ, 219  
 Per\_node, リソースプロパティの属性, 232  
 Pingpong\_interval, リソースグループプロパ  
 ティ, 219  
 ping コマンド, 無効にしたリソースからの応  
 答, 80  
 Pkglist, リソースタイププロパティ, 189  
 Port\_list, リソースプロパティ, 202  
 Postnet\_stop, リソースタイププロパティ, 190  
 Postnet\_stop\_timeout, リソースプロパティ  
 , 203  
 Prenet\_start, リソースタイププロパティ, 190  
 Prenet\_start\_timeout, リソースプロパティ  
 , 203  
 Probe\_timeout 拡張プロパティ  
 再起動時間への影響, 169  
 調整, 168  
 Property, リソースプロパティ属性, 233

Proxy, リソースタイプのプロパティ, 190  
 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, 204  
 Resource\_dependencies, リソースプロパティ  
 , 204  
 Resource\_dependencies\_offline\_restart, リソース  
 プロパティ, 206  
 Resource\_dependencies\_restart, リソースプロパ  
 ティ, 207  
 Resource\_dependencies\_weak, リソースプロパ  
 ティ, 209  
 Resource\_list  
 リソースグループプロパティ, 219  
 リソースタイププロパティ, 191  
 Resource\_name, リソースプロパティ, 210  
 Resource\_project\_name, リソースプロパティ  
 , 211  
 Resource\_state, リソースプロパティ, 211  
 Resource\_type, リソースタイプのプロパティ  
 , 191  
 resources, 構成データの取得、複製、またはアップ  
 グレード, 152  
 Retry\_count, リソースプロパティ, 211  
 Retry\_count システムプロパティ, 170  
 Retry\_interval, リソースプロパティ, 212  
 Retry\_interval システムプロパティ, 170  
 RG\_affinities, リソースグループプロパティ  
 , 220  
 RG\_affinities リソースグループプロパティ  
 , 143-145  
 RG\_dependencies, リソースグループプロパティ  
 , 220  
 RG\_description, リソースグループプロパティ  
 , 221  
 RG\_is\_frozen, リソースグループプロパティ  
 , 221  
 RG\_mode, リソースグループプロパティ, 221  
 RG\_name, リソースグループプロパティ, 222

RG\_project\_name, リソースグループプロパティ  
 , 222  
RG\_slm\_cpu, リソースグループプロパティ, 223  
RG\_slm\_cpu\_min, リソースグループプロパティ  
 , 223  
RG\_slm\_pset\_type, リソースグループプロパティ  
 , 225  
RG\_slm\_type, リソースグループプロパティ, 224  
RG\_state, リソースグループプロパティ, 228  
RG\_system, リソースグループプロパティ, 230  
RGM (Resource Group Manager)  
 値, 237  
 有効な名前, 235  
RT\_basedir, リソースタイププロパティ, 191  
RT\_description, リソースタイププロパティ  
 , 192  
RT\_system, リソースタイプのプロパティ, 192  
RT\_version, リソースタイプのプロパティ, 192

## S

Scalable, リソースプロパティ, 212  
scsnapshot ユーティリティ, 152  
Service Management Facility (SMF), 18-19  
 有効化, 155-167  
show-rev サブコマンド, 13  
showrev -p コマンド, 13  
Single\_instance, リソースタイププロパティ  
 , 192  
SMF (Service Management Facility), 18-19  
Start, リソースタイプのプロパティ, 192  
Start\_failed リソース状態  
 clearing, 91-93  
 解除, 93-94, 95-96  
Start\_timeout, リソースプロパティ, 213  
Status, リソースプロパティ, 213  
Status\_msg, リソースプロパティ, 214  
Stop, リソースタイプのプロパティ, 193  
STOP\_FAILED エラーフラグ, 88-90  
Stop\_timeout, リソースプロパティ, 214  
Sun Cluster Manager GUI, 25-26  
Sun Management Center GUI, 26  
Sun Cluster の管理コマンド, 27

SUNW.LogicalHostname リソースタイプ  
 アップグレード, 96-98  
 誤って削除したあとの再登録, 98-99  
 リソースタイプバージョン, 97  
SUNW.SharedAddress リソースタイプ  
 アップグレード, 96-98  
 誤って削除したあとの再登録, 98-99  
 リソースタイプバージョン, 97  
Suspend\_automatic\_recovery, リソースグループプ  
 ロパティ, 230

## T

Thorough\_probe\_interval, リソースプロパティ  
 , 214  
Thorough\_probe\_interval システムプロパティ  
 再起動時間への影響, 169  
 調整, 168  
Tunable, リソースプロパティ属性, 233  
Type, リソースプロパティ, 214  
Type\_version, リソースプロパティ, 215  
Type\_version プロパティ, 97, 142

## U

UDP\_affinity, リソースプロパティ, 215  
Update, リソースタイプのプロパティ, 193  
Update\_timeout, リソースプロパティ, 215  
upgrade 指令, 236

## V

Validate, リソースタイプのプロパティ, 193  
Validate\_timeout, リソースプロパティ, 216  
 /var/adm/messages ファイル, 13  
Vendor\_ID, リソースタイプのプロパティ, 193  
vfstab ファイル  
 エントリの削除, 135  
 エントリの追加, 131

**W**

Weak\_affinity, リソースプロパティ, 216

**あ**

値, RGM (Resource Group Manager), 237  
 新しいリソースタイプバージョンへの移行, 37-42  
 アップグレード  
   HASStoragePlus リソースタイプ, 141-142  
   構成データ, 154  
   事前登録されているリソースタイプ, 96-98  
 アフィニティ, リソースグループ, 143-145  
 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, 133

**い****移行**

  HASStoragePlus リソース, 142  
   共有アドレスリソース, 97-98  
   論理ホスト名リソース, 97-98  
 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, 150-151  
 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, 88-90  
 エラーメッセージ  
   クラスタ, 13  
   ファイルシステムの変更の失敗, 138, 139

**お**

オンラインにする, リソースグループ, 66-67

**か****解除**

  Start\_failed リソース状態, 93-94, 95-96  
 回復  
   ファイルシステムの変更の失敗から, 138-139, 139-140  
 回復アクション  
   自動の再開, 69-72  
   自動の保存停止, 69-72  
 拡張プロパティ  
   Probe\_timeout  
     再起動時間への影響, 169  
     調整, 168  
   リソースタイプ, 195  
   リソースプロパティ属性, 232

**確認**

  HASStoragePlus リソースからのファイルシステムの削除, 134  
   HASStoragePlus リソースへのファイルシステムの追加, 132  
   nsswitch.conf ファイルの内容, 17  
 型, リソースプロパティ属性の属性, 233  
 間隔, 障害モニター検証, 167-168  
 完全な障害, 169

**き**

技術サポート, 13

記述値, 規則, 237

**規則**

  記述値, 237  
   プロパティ値, 237  
   プロパティ名, 235  
   リソースグループ名, 235  
   リソース名, 235  
   列挙型リテラル名, 235  
 起動の同期, リソースグループとデバイスグループ, 112-115  
 共有アドレスリソース  
   変更, 87-88  
   無効にしたときにホストから分離, 80  
   リソースグループに追加, 57-60  
     clsetup ユーティリティの使用, 55-57  
 切り替え, リソースグループ, 91-93

均衡, クラスタノードの負荷, 148

## く

組み合わせ, リソースグループ間のアフィニティ, 151-152

## け

### 計画

クラスタファイルシステム, 17

データサービス, 15-28

継続的な障害, 定義, 168-170

現在の主ノードの切り替え, リソースグループ, 78-80

## こ

高可用性ファイルシステム

注意事項, 134

ファイルシステムから削除, 133-135

ファイルシステムの追加, 131-133

変更, 130-140

変更の失敗, 138-139, 139-140

有効化, 118-127

更新, リソースタイプ, 36-37

### 構成

ガイドライン, 16-19

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, 34

### 構文

記述値, 237

プロパティ値, 237

プロパティ名, 235

リソースグループ名, 235

リソースタイプ名, 236

リソース名, 235

列挙型リテラル名, 235

考慮事項, 21-22

コマンド, 171-181

ノード情報, 13

コマンド行インタフェース

共有アドレス

リソースグループに追加, 57-60

論理ホスト名

リソースグループに追加, 53-55

## さ

### 再起動

許可される最大, 169

リソースグループ, 93-94

再試行間隔, 169

最大値, 再起動, 169

### 削除

HASStoragePlus リソースからファイルシステムを, 133-135

リソース, 77

リソースグループ, 76-77

リソースグループからのノード

概要, 103

共有アドレスを使用したフェイルオーバー, 107-108

スケラブル, 104-105

フェイルオーバー, 105-107

リソースタイプ, 74-75

### 作成

共有アドレスリソース, 55-57

CLIの使用, 57-60

スケラブルアプリケーションリソース, 62-65

フェイルオーバーアプリケーションリソース, 60-62

リソースグループ

スケラブル, 46-48

フェイルオーバー, 44-45

論理ホスト名リソース, 49-52, 53-55

サブコマンド, 171-181

## し

システムプロパティ, 障害モニターへの影響, 167

システムプロパティ

「プロパティ」も参照

システムプロパティ (続き)  
「拡張プロパティ」も参照  
Failover\_mode, 170  
Retry\_count, 170  
Retry\_interval, 170  
Thorough\_probe\_interval  
再起動時間への影響, 169  
調整, 168  
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, 98-99  
事前登録されたリソースタイプの再登録, 98-99  
事前登録されているリソースタイプ, アップグ  
レード, 96-98  
失敗  
ファイルシステムの変更, 138-139, 139-140  
自動回復アクション, 69-72  
重要でないサービス, オフロード, 149-150  
重要でないリソースグループのオフロード, ア  
フィニティ, 149-150  
重要なサービス, 149-150  
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, 153  
障害  
継続的, 168-170  
への対応, 170  
障害追跡  
ファイルシステムの変更, 138-139, 139-140  
障害モニター  
検証間隔, 167-168  
検証タイムアウト, 168  
障害への対応, 170  
調整, 167-170  
による障害の検出, 170  
無効化, 72-73  
有効化, 73-74  
リソース, 72-74  
除去, STOP\_FAILED エラーフラグ, 88-90  
書式, リソースタイプ名, 236  
指令, #upgrade, 236

## す

スイッチオーバー, リソースグループの委  
託, 150-151

スケラブルアプリケーションリソース, リソー  
スグループに追加, 62-65

## せ

制限, 22

## 性能

重要なサービス用に最適化, 149-150  
への検証間隔の影響, 168

## 設定

HASStoragePlus リソースタイプ, 118-127  
新しいリソース, 113-115  
既存のリソース, 115

## そ

属性, リソースプロパティ, 232

## た

対応, 障害への, 170

## タイムアウト

## 障害モニター

設定の指針, 168

ダウングレード, リソースタイプ, 42-43

## ち

注意事項, ファイルシステムの削除, 134

調整, 障害モニター, 167-170

## つ

## 追加

HASStoragePlus リソースにファイルシステ  
ムを, 131-133

リソースグループにリソースを

スケラブルアプリケーション, 62-65

フェイルオーバーアプリケーション, 60-62

## 追加 (続き)

- リソースグループへのノード
  - 概要, 99
  - スケラブル, 100-101
  - フェイルオーバー, 101-103
- リソースグループへのリソース
  - 概要, 48-65
- リソースをリソースグループに共有アドレス, 55-57, 57-60
- 論理ホスト名, 49-52, 53-55

## ツール

- clsetup ユーティリティー, 26
- Sun Cluster Manager GUI, 25-26
- Sun Management Center GUI, 26
- Sun Cluster の管理コマンド, 27

## 強い肯定的なアフィニティー

- 使用例, 145-146
- 定義, 144

## 強い否定的なアフィニティー

- 使用例, 149-150
- 定義, 144

## て

定義, 継続的な障害, 168-170

## データサービス

- 計画, 15-28
- 考慮事項, 21-22
- 特殊な要件, 16

## デバイスグループ

- リソースグループとの関係, 19
- リソースグループとの起動の同期, 112-115

## と

問い合わせ, 13

## 登録

- HASStoragePlus リソースタイプ
  - アップグレード中, 141-142
- SUNW.LogicalHostname リソースタイプ
  - アップグレード中, 97
  - 誤って削除したあと, 98-99

## 登録 (続き)

- SUNW.SharedAddress リソースタイプ
  - アップグレード中, 97
  - 誤って削除したあと, 98-99
  - 事前登録されたリソースタイプ, 98-99
  - リソースタイプ, 34-35
- 登録解除, リソースタイプ, 74-75
- 特殊な要件, 確認, 16

## ね

- ネームサービス, バイパス, 87-88
- 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, 149-150
- 負荷均衡, 148
- リソースグループからの削除
  - 概要, 103
  - 共有アドレスを使用したフェイルオーバー, 107-108
  - スケラブル, 104-105
  - フェイルオーバー, 105-107
- リソースグループの分散, 142-152
- リソースグループへの追加
  - 概要, 99
  - スケラブル, 100-101
  - フェイルオーバー, 101-103
- ノードリストプロパティ, 22-23
- ノードリソースの解放, 149-150

## は

## バージョン

- HASStoragePlus リソースタイプ, 141
- SUNW.LogicalHostname リソースタイプ, 97
- SUNW.SharedAddress リソースタイプ, 97
- 配置, オンラインリソースグループに対する優先, 146-147
- バイパス, ネームサービス, 87-88

## ひ

表示, リソースタイプ、リソースグループ、リソース構成, 82-83

## ふ

ファイル

/etc/vfstab

エントリの削除, 135

エントリの追加, 131

RTR, 142

ファイルシステム

HASStoragePlus リソースから削除, 133-135

HASStoragePlus リソースに追加, 131-133

アンマウント, 133

高可用性

変更, 130-140

有効化, 118-127

注意事項, 134

変更の失敗, 138-139, 139-140

フェイルオーバー

オンラインのリソースグループの分散の保持, 142-152

リソースグループの委託, 150-151

フェイルオーバーアプリケーションリソース, リソースグループに追加, 60-62

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, 使用例, 150-151

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, 定義, 144

負荷均衡, 148

複製, 構成データ, 153

部分的な障害, 169

プロパティ

「拡張プロパティ」も参照

Type\_version, 97, 142

リソース, 194

リソースグループ, 216

リソースタイプ, 183

プロパティ属性, リソース, 232

プロパティ値, 規則, 237

プロパティ名, 規則, 235

分散, オンラインのリソースグループ, 142-152

## へ

別名, 171-181

変更

リソースグループプロパティ, 85

リソースタイププロパティ, 83-85

リソースプロパティ, 86-87

編集

HASStoragePlus リソース, 142

共有アドレスリソース, 97-98

論理ホスト名リソース, 97-98

## ほ

ボリュームマネージャー, 高可用性ファイルシステム, 120

## み

短いコマンド, 171-181

## む

無効化

SMF インスタンス, 18-19

リソース, 80-82, 95-96

リソース障害モニター, 72-73

無効にしたリソース, 予期せぬ動作, 80

## ゆ

有効化

Solaris SMF サービス, 155-167

リソース, 67-68, 95-96

リソース障害モニター, 73-74

有効な名前, RGM (Resource Group Manager), 235

## 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, 146-147

定義, 144

## 弱い否定的なアフィニティー

使用例, 148

定義, 144

## リ

## リソース

STOP\_FAILED エラーフラグの消去, 88-90

共有アドレス

変更, 87-88

無効にしたときにホストから分離, 80

リソースグループに追加, 55-57, 57-60

構成情報の表示, 82-83

削除, 77

障害モニターの無効化, 72-73

障害モニターの有効化, 73-74

スケーラブルアプリケーション

リソースグループに追加, 62-65

フェイルオーバーアプリケーション

リソースグループに追加, 60-62

プロパティの変更, 86-87

無効化, 80-82, 95-96

有効化, 67-68, 95-96

リソースグループへの追加, 48-65

リソースタイプの削除, 74-75

論理ホスト名

変更, 87-88

リソースグループに追加, 49-52, 53-55

## リソースグループ

UNMANAGED 状態への移行, 80-82

アフィニティー, 143-145

オンラインにする, 66-67

休止, 68-69

強制的に同じ場所に配置, 145-146

強制的に分離, 149-150

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, 107-108

均等分配, 148

現在の主ノードの切り替え, 78-80

構成情報の表示, 82-83

## リソースグループ (続き)

構成データの取得、複製、またはアップグレード, 152

再起動, 93-94

削除, 76-77

作成

スケーラブル, 46-48

フェイルオーバー, 44-45

自動回復アクションの再開, 69-72

自動回復アクションの保存停止, 69-72

スイッチオーバー, 91-93

スケーラブル

ノードの削除, 104-105

ノードの追加, 100-101

できる限り同じ場所に配置, 146-147

できる限り分離, 148

デバイスグループとの関係, 19

デバイスグループとの起動の同期, 112-115

ノード間で分散, 142-152

ノードの削除, 103

ノードの追加, 99

フェイルオーバーまたはスイッチオーバーの委託, 150-151

フェイルオーバー

ノードの削除, 105-107

ノードの追加, 101-103

プロパティの変更, 85

リソースグループの追加

共有アドレス, 55-57

リソースの追加, 48-65

共有アドレス, 57-60

スケーラブルアプリケーション, 62-65

フェイルオーバーアプリケーション, 60-62

論理ホスト名, 49-52, 53-55

リソースグループの休止, 68-69

リソースグループプロパティ, 216

Auto\_start\_on\_new\_cluster, 217

Desired primaries, 217

Failback, 217

Global\_resources\_used, 218

Implicit\_network\_dependencies, 218

Maximum primaries, 218

Nodelist, 218

Pathprefix, 219

## リソースグループプロパティ (続き)

Pingpong\_interval, 219  
 Resource\_list, 219  
 RG\_affinities, 220  
 RG\_dependencies, 220  
 RG\_description, 221  
 RG\_is\_frozen, 221  
 RG\_mode, 221  
 RG\_name, 222  
 RG\_project\_name, 222  
 RG\_slm\_cpu, 223  
 RG\_slm\_cpu\_min, 223  
 RG\_slm\_pset\_type, 225  
 RG\_slm\_type, 224  
 RG\_state, 228  
 RG\_system, 230  
 Suspend\_automatic\_recovery, 230

## リソースグループ名, 規則, 235

## リソース障害モニター, 72-74

## リソースタイプ

## HASStoragePlus

新しいリソース, 113-115  
 インスタンスの移行, 142  
 既存のリソース, 115

## LogicalHostname

インスタンスの移行, 97-98

## SharedAddress

インスタンスの移行, 97-98

## 新しいリソースタイプバージョンへの移行, 37-42

## 更新, 36-37

## 構成情報の表示, 82-83

## 構成データの取得、複製、またはアップグレード, 152

## 削除, 74-75

## 事前登録

アップグレード, 96-98

## 事前登録された

誤って削除したあとの再登録, 98-99

## ダウングレード, 42-43

## 登録, 34-35

## 登録解除, 74-75

## プロパティの変更, 83-85

## リソースタイプ登録 (RTR) ファイル, 142

## リソースタイプのプロパティ

Global\_zone, 187  
 Proxy, 190  
 Resource\_type, 191  
 RT\_system, 192  
 RT\_version, 192  
 Start, 192  
 Stop, 193  
 Update, 193  
 Validate, 193  
 Vendor\_ID, 193

## リソースタイププロパティ, 183

API\_version, 184  
 Boot, 185  
 Failover, 185  
 Fini, 186  
 Init, 188  
 Init\_nodes, 188  
 Installed\_nodes, 188  
 Is\_logical\_hostname, 188  
 Is\_shared\_address, 189  
 Monitor\_check, 189  
 Monitor\_start, 189  
 Monitor\_stop, 189  
 Pkglist, 189  
 Postnet\_stop, 190  
 Prenet\_start, 190  
 Resource\_list, 191  
 RT\_basedir, 191  
 RT\_description, 192  
 Single\_instance, 192

## リソースタイプ名, 規則, 236

## リソースの変更, 87-88

## リソースプロパティ, 194

Affinity\_timeout, 194  
 Boot\_timeout, 194  
 Cheap\_probe\_interval, 195  
 Failover\_mode, 195  
 Fini\_timeout, 199  
 Init\_timeout, 199  
 Load\_balancing\_policy, 199  
 Load\_balancing\_weights, 199  
 Monitor\_check\_timeout, 200  
 Monitor\_start\_timeout, 200

## リソースプロパティ (続き)

- Monitor\_stop\_timeout, 200
- Monitored\_switch, 200
- Network\_resources\_used, 201
- Num\_resource\_restarts, 201
- Num\_rg\_restarts, 202
- On\_off\_switch, 202
- Port\_list, 202
- Postnet\_stop\_timeout, 203
- Prenet\_start\_timeout, 203
- R\_description, 204
- Resource\_dependencies, 204
- Resource\_dependencies\_offline\_restart, 206
- Resource\_dependencies\_restart, 207
- Resource\_dependencies\_weak, 209
- Resource\_name, 210
- Resource\_project\_name, 211
- Resource\_state, 211
- Retry\_count, 211
- Retry\_interval, 212
- Scalable, 212
- Start\_timeout, 213
- Status, 213
- Status\_msg, 214
- Stop\_timeout, 214
- Thorough\_probe\_interval, 214
- Type, 214
- Type\_version, 215
- UDP\_affinity, 215
- Update\_timeout, 215
- Validate\_timeout, 216
- Weak\_affinity, 216
- 拡張, 195

## リソースプロパティの属性, 232

- Array\_maxsize, 232
- Array\_minsize, 232
- Default, 232
- Description, 232
- Enumlist, 232
- Extension, 232
- Max, 232
- Maxlength, 232
- Min, 232
- Minlength, 232

## リソースプロパティの属性 (続き)

- Per\_node, 232
- Property, 233
- Tunable, 233
- 型, 233

リソース名, 規則, 235

## れ

列挙型リテラル名, 規則, 235

## ろ

論理ホスト名リソース

変更, 87-88

リソースグループに追加

CLIの使用, 53-55

clsetup ユーティリティの使用, 49-52