Oracle® Solaris Cluster 4.4 データサービスの計画と管理

Part No: E75663-01
2018 年 8 月

Part No: E75663-01

Copyright © 2000, 2018, Oracle and/or its affiliates. All rights reserved.

このソフトウェアおよび関連ドキュメントの使用と開示は、ライセンス契約の制約条件に従うものとし、知的財産に関する法律により保護されています。ライセンス契約で明示的に許諾されている場合もしくは法律によって認められている場合を除き、形式、手段に関係なく、いかなる部分も使用、複写、複製、翻訳、放送、修正、ライセンス供与、送信、配布、発表、実行、公開または表示することはできません。このソフトウェアのリバース・エンジニアリング、逆アセンブル、逆コンパイルは互換性のために法律によって規定されている場合を除き、禁止されています。

ここに記載された情報は予告なしに変更される場合があります。また、誤りが無いことの保証はいたしかねます。誤りを見つけた場合は、オラクルまでご連絡ください。

このソフトウェアまたは関連ドキュメントを、米国政府機関もしくは米国政府機関に代わってこのソフトウェアまたは関連ドキュメントをライセンスされた者に提供する場合は、次の通知が適用されます。

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

このソフトウェアまたはハードウェアは様々な情報管理アプリケーションでの一般的な使用のために開発されたものです。このソフトウェアまたはハードウェアは、危険が伴うアプリケーション(人的傷害を発生させる可能性があるアプリケーションを含む)への用途を目的として開発されていません。このソフトウェアまたはハードウェアを危険が伴うアプリケーションで使用する際、安全に使用するために、適切な安全装置、バックアップ、冗長性(redundancy)、その他の対策を講じることは使用者の責任となります。このソフトウェアまたはハードウェアを危険が伴うアプリケーションで使用したことに起因して損害が発生しても、Oracle Corporationおよびその関連会社は一切の責任を負いかねます。

OracleおよびJavaはオラクル およびその関連会社の登録商標です。その他の社名、商品名等は各社の商標または登録商標である場合があります。

Intel、Intel Xeonは、Intel Corporationの商標または登録商標です。すべてのSPARCの商標はライセンスをもとに使用し、SPARC International, Inc.の商標または登録商標です。AMD、Opteron、AMDロゴ、AMD Opteronロゴは、Advanced Micro Devices, Inc.の商標または登録商標です。UNIXは、The Open Groupの登録商標です。

このソフトウェアまたはハードウェア、そしてドキュメントは、第三者のコンテンツ、製品、サービスへのアクセス、あるいはそれらに関する情報を提供することがあります。適用されるお客様とOracle Corporationとの間の契約に別段の定めがある場合を除いて、Oracle Corporationおよびその関連会社は、第三者のコンテンツ、製品、サービスに関して一切の責任を負わず、いかなる保証もいたしません。適用されるお客様とOracle Corporationとの間の契約に定めがある場合を除いて、Oracle Corporationおよびその関連会社は、第三者のコンテンツ、製品、サービスへのアクセスまたは使用によって損失、費用、あるいは損害が発生しても一切の責任を負いかねます。

ドキュメントのアクセシビリティについて

オラクルのアクセシビリティについての詳細情報は、Oracle Accessibility ProgramのWeb サイト(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc)を参照してください。

Oracle Supportへのアクセス

サポートをご契約のお客様には、My Oracle Supportを通して電子支援サービスを提供しています。詳細情報は(http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info) か、聴覚に障害のあるお客様は (http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs)を参照してください。

目次

表目次

例目次

このドキュメントの使用法

- **概要** – データサービス構成を計画して管理する方法について説明します。
- **対象読者** – 技術者、システム管理者、および認定サービスプロバイダ
- **前提知識** – ハードウェアのトラブルシューティングや交換に関する豊富な経験

製品ドキュメントライブラリ

この製品および関連製品のドキュメントとリソースは http://www.oracle.com/pls/topic/lookup?ctx=E69294_01 で入手可能です。

フィードバック

このドキュメントに関するフィードバックを http://www.oracle.com/goto/docfeedback からお聞かせください。

♦ ♦ ♦ 第 1 章

1 Oracle Solaris Cluster データサービスの計画

この章では、Oracle Solaris 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計画情報とガイドラインを提供します。この章で説明する内容は次のとおりです。

データサービスの概要は、『Concepts for Oracle Solaris Cluster 4.4』を参照してください。

Oracle Solaris Cluster ソフトウェアは、Oracle Solaris Cluster 製品に付属しているか、または Oracle Solaris Cluster データサービスのアプリケーションプログラミングインタフェース (API) を使用して作成されたデータサービスに対してのみサービスを提供できます。

使用しているアプリケーションに対して Oracle Solaris 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、Oracle Solaris Cluster データサービスの API を使用します。詳細は、『Developing Data Services』を参照してください。

注記 - Oracle Solaris Cluster は、`sendmail(8)` サブシステムのデータサービスを提供していません。`sendmail` サブシステムは個々のクラスタノード上で動作できますが、この `sendmail` 機能は高可用性ではありません。この制限は、メール配信やメールルーティング、キューイング、および再試行の機能を含む、すべての `sendmail` 機能に適用されます。

Oracle Solaris Cluster データサービスの構成ガイドライン

このセクションでは、Oracle Solaris Cluster データサービスの構成ガイドラインについて説明します。

データサービスに固有の要件の識別

Oracle Solaris OS と Oracle Solaris 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識別してください。そうしないと、Oracle Solaris OS や Oracle Solaris Cluster ソフトウェアを完全にインストールし直す必要があるインストールエラーが発生する可能性があります。

たとえば、Oracle Solaris Cluster Support for Oracle Real Application Clusters の Oracle Data Guard オプションには、クラスタ内で使用するホスト名に関する固有の要件があります。また、HA for SAP にも固有の要件があります。Oracle Solaris Cluster ソフトウェアのインストール後にホスト名を変更することはできないので、Oracle Solaris Cluster をインストールする前に、次の要件に対応する必要があります。

注記 - 一部の Oracle Solaris Cluster データサービスの x86 ベースのクラスタでの使用はサポートされていません。詳細は、使用しているリリースの Oracle Solaris Cluster のリリースノートを参照してください。

不変ゾーンの使用

不変ゾーンクラスタとして構成されているゾーンクラスタ内のアプリケーションをインストールまたは保守するとき、書き込み操作が確実に許可されるようにするために、インストールまたは保守の期間中はゾーンクラスタを書き込みモードでブートします。アプリケーションのインストールまたは保守が完了したら、ゾーンクラスタをリブートして不変ゾーンクラスタに戻すことができます。

1. ゾーンクラスタを不変ゾーンクラスタとして作成または構成します。
2. アプリケーションのインストールまたは保守を行うときに、`-w` オプションを指定してゾーンクラスタをリブートします。
3. アプリケーションをインストールし、保守を行い、Oracle Solaris Cluster データサービスを構成します。
4. 完了したら、ゾーンクラスタをリブートして不変ゾーンクラスタに戻します。

注記 - 必要に応じて、データセットまたはファイルシステムをゾーンクラスタに追加できます。`clzc configure add dataset` を使用して追加のデータセットが指定されたゾーンクラスタノードは、ゾーンクラスタが読み取り専用モードでブートされた場合でも、それらのデータセットに対する完全な制御権を引き続き持ちます。`clzc configure add fs` を使用して追加のファイルシステムが指定されたゾーンクラスタノードは、ファイルシステムが読み取り専用に設定されていないかぎり、ゾーンクラスタが読み取り専用モードでブートされた場合でも、それらのファイルシステムに対する完全な制御権を引き続き持ちます。『安装和配置 Oracle Solaris Cluster 4.4 环境』の「特定のゾーンクラスタノードにローカルファイルシステムを追加する」を参照してください。

例 1 不変ゾーンクラスタにローカル ZFS データセットを追加する

この例は、Oracle バイナリで使用するデータセットを、各ゾーンクラスタノードにローカルに追加する方法を示しています。これにより、ゾーンクラスタが読み取り専用モードでブートされた場合でも、ゾーンクラスタノードからファイルシステムに対する読み取り/書き込みアクセスが可能になります。

1. グローバルクラスタの各ノードで、ゾーンクラスタの各ノード内で使用する Oracle バイナリ用ファイルシステムを作成します。

```
# zfs create orapool/immutablezc-oracle
```

2. グローバルクラスタの 1 つのノードから、`clzc` を使用して、各ゾーンクラスタノードにデータセットを追加します。

```
root@gcnode1:~# clzc configure immutablezc
clzc:immutablezc> select node physical-host=gcnode2
clzc:immutablezc:node> add dataset
clzc:immutablezc:node:dataset> set name=orapool/immutablezc-oracle
clzc:immutablezc:node:dataset> end
clzc:immutablezc:node> end
clzc:immutablezc> select node physical-host=gcnode1
clzc:immutablezc:node> add dataset
clzc:immutablezc:node:dataset> set name=orapool/immutablezc-oracle
clzc:immutablezc:node:dataset> end
clzc:immutablezc:node> end
clzc:immutablezc> exit
root@gcnode1:~#
#  zfs list | grep immutablezc-oracle
orapool/immutablezc-oracle                                    31K  81.4G    31K
  /orapool/immutablezc-oracle
#
```

3. ゾーンクラスタを書き込みモードでリブートします。

```
# clzc reboot -w immutablezc
```

```
#  zfs list | grep immutablezc-oracle
orapool/immutablezc-oracle                                    31K  81.4G    31K
  /export/zones/immutablezc/root/immutablezc-oracle
#
```

4. グローバルクラスタの各ノードから、ゾーンクラスタにログインし、データセットのマウントを必要に応じて設定します。

```
# zlogin immutablezc
# zfs list | grep oracle
immutablezc-oracle                      31K  81.4G    31K  /immutablezc-oracle
#
# zfs set mountpoint=/u01 immutablezc-oracle
# zfs list | grep oracle
immutablezc-oracle                      31K  81.4G    31K  /u01
#
```

この時点以降、ゾーンクラスタを不変ゾーンクラスタとしてリブートでき、ゾーンクラスタノードから `/u01` への読み取り/書き込みアクセスが可能です。

アプリケーションバイナリの場所の決定

アプリケーションソフトウェアと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、サービスを停止することなく、あとでアプリケーションソフトウェアをアップグレードできるという利点が得られます。

  欠点は、保守および管理するソフトウェアと構成ファイルのコピーが複数になることです。
- **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配置した場合は、保守および管理するコピーが 1 つだけになります。ただし、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、クラスタ全体のデータサービスを停止する必要があります。アップグレードのための短期間のダウンタイムを許容できる場合は、アプリケーションと構成ファイルの 1 つのコピーをクラスタファイルシステムに配置してください。

  クラスタファイルシステムを作成する方法については、『安装和配置 Oracle Solaris Cluster 4.4 环境』 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。
- **高可用性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 Oracle Solaris Cluster 環境に統合して、そのローカルファイルシステムを高可用性にすることができます。`HAStoragePlus` は、Oracle Solaris Cluster がローカルファイルシステムをフェイルオーバーできるようにする、

チェック、マウント、アンマウントなどの追加のファイルシステム機能を提供します。フェイルオーバーするには、ローカルファイルシステムは、アフィニティースイッチオーバーが有効になっているグローバルディスクグループ上に存在する必要があります。

HAStoragePlus リソースタイプを使用する方法については、156 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のための構成ファイルです。このファイルによって、次の情報が決定されます。

- ネームサービスの検索のために使用する Oracle Solaris 環境内のデータベース
- 各データベースが参照される順序

一部のデータサービスでは、group 検索を最初に files に対して行わせる必要があります。これらのデータサービスの場合は、nsswitch.conf ファイル内の group 行を files エントリが最初にリストされるように変更します。group 行を変更する必要があるかどうかを確認するには、構成する予定のデータサービスに関するドキュメントを参照してください。scinstall ユーティリティーは、nsswitch.conf ファイルを自動的に構成します。nsswitch.conf ファイルを手動で変更する場合は、新しい nsswitch 構成情報をエクスポートする必要があります。

クラスタファイルシステム構成の計画

データサービスによっては、Oracle Solaris Cluster の要件を満たすようにクラスタファイルシステムを構成しなければならないことがあります。何らかの特別な考慮事項が適用されるかどうかを確認するには、構成する予定のデータサービスに関するドキュメントを参照してください。

クラスタファイルシステムの計画については、『安装和配置 Oracle Solaris Cluster 4.4 环境』 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、Oracle Solaris Cluster 環境で、フェイルオーバー用に構成された高可用性ローカルファイルシステムを使用できます。HAStoragePlus リソースタイプの設定については、156 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。

Oracle Solaris SMF サービスを Oracle Solaris Cluster の制御下で実行できるようにする

サービス管理機能 (SMF) では、ノードのブート中やサービスの障害時に、SMF サービスを自動的に起動および再起動できます。この機能は、クラスタアプリケーションの高可用性とスケーラビリティーを促進する Oracle Solaris Cluster Resource Group Manager (RGM) と類似しています。SMF サービスと RGM は、互いに補完的な機能です。

Oracle Solaris Cluster には、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で SMF サービスを Oracle Solaris Cluster とともに実行できるようにするために使用可能な 3 つの SMF プロキシリソースタイプが含まれています。SMF プロキシリソースタイプを使用すると、互いに関連する一連の SMF サービスを 1 つのリソースである *SMF* プロキシリソースにカプセル化して Oracle Solaris Cluster によって管理されるようにすることができます。この機能では、SMF は、単一のノード上の SMF サービスの可用性を管理します。Oracle Solaris Cluster は、SMF サービスのクラスタ全体にわたる高可用性とスケーラビリティーを提供します。

これらのサービスをカプセル化する方法については、122 ページの「Oracle Solaris SMF サービスを Oracle Solaris Cluster とともに実行できるようにする」を参照してください。

Oracle Solaris Cluster で、Solaris サービス管理機能 (SMF) と統合された NFS または DNS 以外のアプリケーションを高可用性にすることが必要になる場合があります。Oracle Solaris Cluster が障害のあとに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、そのアプリケーションの SMF サービスインスタンスを次のように無効にする必要があります。

- NFS または DNS 以外のすべてのアプリケーションの場合、そのアプリケーションを表す Oracle Solaris Cluster リソースのすべての潜在的なプライマリノード上で SMF サービスインスタンスを無効にします。
- Oracle Solaris Cluster でモニターする必要があるいずれかのコンポーネントをアプリケーションの複数のインスタンスが共有している場合は、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には、デーモン、ファイルシステム、およびデバイスがあります。

注記 - 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、Oracle Solaris SMF と Oracle Solaris Cluster の両方がアプリケーションの起動と停止を制御しようとする可能性があります。その結果、アプリケーションの動作が予測できなくなることがあります。

詳細については、次のドキュメントを参照してください。

- 『Oracle Solaris 11.4 でのシステムサービスの管理』 の 「サービスインスタンスを無効にする方法」
- 『Oracle Solaris Cluster Data Service for NFS Guide』
- 『Concepts for Oracle Solaris Cluster 4.4』

リソースグループとデバイスグループの関係

Oracle Solaris Cluster は、デバイスグループとリソースグループに対して**ノードリスト**の概念を使用します。ノードリストは、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なマスターであるプライマリノードの順序付きリストです。Oracle Solaris Cluster は、次の一連の状態に対応する Oracle Solaris Cluster の動作を決定するために、**フェイルバックポリシー**を使用します。

- 障害が発生してクラスタから切り離されたノードがクラスタに再度参加した
- そのクラスタに再度参加したノードが、ノードリスト内で現在のプライマリノードより前に位置している

フェイルバックが `True` に設定されている場合は、デバイスグループまたはリソースグループを現在のプライマリから切り離して、再度参加したノードに切り替えることにより、再度参加したノードを新しいプライマリにします。

たとえば、ノードリスト内にノード `phys-schost-1` および `phys-schost-2` を含み、フェイルバックポリシーが `Enabled` に設定されたディスクデバイスグループ `disk-group-1` が存在するとします。また、`disk-group-1` を使用してアプリケーションデータを保持しているフェイルオーバーリソースグループ `resource-group-1` も存在するとします。`resource-group-1` を設定する場合は、その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には、そのスケーラブルリソー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。この設定により、ディスクに直接接続されたノードは確実に、スケーラブルリソースグループを実行できるノードにもなります。利点として、データに接続されたクラスタノードが少なくとも 1 つ起動されていれば、スケーラブルリソースグループがその同じノード上で実行され、それによりスケーラブルサービスも使用可能になる点が挙げられます。

デバイスグループとリソースグループの関係の詳細は、『Concepts for Oracle Solaris Cluster 4.4』 の 「Device Groups」を参照してください。デバイスグループを設定する方法については、『安装和配置 Oracle Solaris Cluster 4.4 环境』 の 「デバイスグループの計画」を参照してください。

HAStoragePlus について

HAStoragePlus リソースタイプを使用すると、次のオプションを構成できます。

- ディスクデバイスとリソースグループのブート順序を調整します。HAStoragePlus リソースを含むリソースグループ内のその他のリソースは、ディスクデバイスリソースが使用可能になったあとでのみオンラインになります。
- AffinityOn が True に設定されている場合は、リソースグループとデバイスグループの同じノード上での共用関係を適用します。この適用された共用関係によって、ディスク集中型のデータサービスのパフォーマンスが向上します。
- グローバルデバイス、ファイルシステム、ZFS ストレージプールなどの、HAStoragePlus リソースによって管理されているエンティティーをモニターします。

さらに、HAStoragePlus は、ローカルファイルシステムやグローバルファイルシステムもマウントできます。詳細は、25 ページの「クラスタファイルシステム構成の計画」を参照してください。

注記 - HAStoragePlus リソースがオンラインである間にデバイスグループが別のノードに切り替えられても、AffinityOn には影響しません。リソースグループがデバイスグループとともに移行することは**ありません**。ただし、リソースグループが別のノードに切り替えられた場合、AffinityOn を True に設定すると、デバイスグループはそのリソースグループに従って新しいノードに移行します。

デバイスグループとリソースグループの関係については、139 ページの「HAStoragePlus を使用したリソースグループとデバイスグループの間の起動の同期」を参照してください。

ZFS 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ための手順については、156 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。さらに詳細な説明については、SUNW.HAStoragePlus(7) 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 HAStoragePlus が必要かどうかの確認

次のタイプのデータサービスには HAStoragePlus が必要です。

- ストレージに直接接続されていないノードでのデータサービス
- ディスク集中型のデータサービス

ストレージに直接接続されていないノードでのデータサービス

データサービスのリソースグループのノードリスト内の一部のノードは、ストレージに直接接続されていない可能性があります。この状況では、ストレージとデータサービスの間のブート順序を調整する必要があります。この要件を満たすには、リソースグループを次のように構成します。

- リソースグループ内に `HAStoragePlus` リソースを構成します。
- その他のデータサービスリソースの `HAStoragePlus` リソースに対する依存関係を設定します。

ディスク集中型のデータサービス

Oracle Solaris Cluster HA for Oracle や Oracle Solaris Cluster HA for NFS などの一部のデータサービスはディスク集中型です。データサービスがディスク集中型である場合は、リソースグループとデバイスグループが同じノード上で共用されるようにしてください。この要件を満たすには、次のタスクを実行します。

- データサービスリソースグループに `HAStoragePlus` リソースを追加する
- `HAStoragePlus` リソースをオンラインに切り替える
- データサービスリソースの `HAStoragePlus` リソースに対する依存関係を設定する
- `AffinityOn` を `True` に設定する

注記 - フェイルバックの設定が、リソースグループとデバイスグループの両方で同じである必要があります。

データサービスの中には、ディスク集中型でないものもあります。たとえば、(起動時にそのすべてのファイルを読み取る) HA for DNS はディスク集中型ではありません。データサービスがディスク集中型**でない**場合、`HAStoragePlus` リソースタイプの構成はオプションです。

データサービスをインストールおよび構成するための考慮事項

データサービスのインストールと構成を計画するには、このセクションの情報を使用してください。このセクションの情報は、行おうとしている決定がデータサービスのインストールと構成に与える影響について考慮する場合に役立ちます。データサービ

スに固有の考慮事項については、そのデータサービスに関するドキュメントを参照してください。

- ディスク障害中の I/O サブシステム内の再試行のために、ディスク集中型のデータサービスを含むアプリケーションで遅延が発生することがあります。ディスク集中型のデータサービスは I/O 集中型であり、クラスタ内で構成された多数のディスクを備えています。I/O サブシステムが再試行し、ディスク障害から復旧するには数分かかることがあります。ディスクが最終的には自力で復旧する可能性があったとしても、この遅延のために、Oracle Solaris Cluster がその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場合があります。

  この状況でのフェイルオーバーを回避するには、そのデータサービスのデフォルトの検証タイムアウトを増やすことを考慮してください。また、リソースがタイムアウト制限に近づいたときに通知するように `Timeout_threshold` プロパティーを設定することを検討してください。`Timeout_threshold` プロパティーを使用すると、偽のフェイルオーバーを回避するために役立ちます。データサービスのタイムアウトを増やすために詳細情報やサポートが必要な場合は、お近くのサポートエンジニアに問い合わせてください。
- パフォーマンスを向上させるために、データサービスのインストールと構成は、ストレージに直接接続されたクラスタノード上で行なってください。
- クラスタノード上で実行されるクライアントアプリケーションを HA データサービスの論理 IP アドレスにマップしないでください。フェイルオーバーのあとに、これらの論理 IP アドレスが存在しなくなり、クライアントの接続が失われたままになることがあります。

インストールおよび構成プロセスの概要

データサービスをインストールおよび構成するには、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。
- クラスタ環境内で実行するアプリケーションをインストールおよび構成します。
- 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します。データサービスを構成する場合は、Resource Group Manager (RGM) が管理するリソースタイプ、リソース、およびリソースグループを指定します。これらの手順は、個々のデータサービスのドキュメントで説明されています。

注記 - Oracle Solaris Cluster Manager を使用すると、一部のデータサービスを構成できます。

インストールおよび構成タスクのフロー

次の表に、Oracle Solaris 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タスクの要約を示します。この表にはまた、タスクを実行するための詳細な手順への相互参照も示されています。

表 1 Oracle Solaris 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タスク

タスク	参照先
Oracle Solaris および Oracle Solaris Cluster ソフトウェア (データサービスソフトウェアを含む) をインストールする	『安装和配置 Oracle Solaris Cluster 4.4 环境』
マルチホストディスクを設定する	『安装和配置 Oracle Solaris Cluster 4.4 环境』
リソースおよびリソースグループを計画する	53 ページの「リソースグループの作成」
アプリケーションバイナリの場所を決定し、`nsswitch.conf` ファイルを構成する	24 ページの「アプリケーションバイナリの場所の決定」 25 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をインストールおよび構成する	Oracle Solaris Cluster データサービスに関する適切な本
データサービスを登録および構成する	Oracle Solaris Cluster データサービスに関する適切な本

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 Database アプリケーション向けのフェイルオーバーデータサービスで必要とするリソースタイプ、リソース、およびリソースグループを設定する方法の概要を示します。Oracle Database アプリケーション向けのデータサービスを構成するための完全な手順については、『Oracle Solaris Cluster データサービス (Oracle Database 用）』を参照してください。

この例とスケーラブルデータサービスの例の主な違いは次のとおりです。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て、スケーラブルデータサービスには、アプリケーションリソースのための個別のリソースグループ (スケーラブルリソースグループ) が必要です。

Oracle Database アプリケーションには、サーバーとリスナーの 2 つのコンポーネントがあります。Oracle は Oracle Solaris Cluster HA for Oracle Database 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 Oracle Solaris Cluster リソースタイプにマップされています。この両方のリソースタイプがリソースおよびリソースグループに関連付けられています。

この例はフェイルオーバーデータサービスであるため、プライマリノードからセカンダリノードにフェイルオーバーする IP アドレスである、論理ホスト名のネットワークリソースを使用しています。この論理ホスト名リソースをフェイルオーバーリソース

グループに配置したあと、Oracle サーバーリソースおよびリスナーリソースを同じリソースグループに配置します。この順序付けにより、すべてのリソースのグループとしてのフェイルオーバーが可能になります。

Oracle Solaris Cluster HA for Oracle Database をクラスタで実行するには、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すでに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 – Oracle Solaris Cluster HA for Oracle Database では、データベースサーバーとリスナーの 2 つの Oracle リソースタイプが定義されています。
- 論理ホスト名リソース - これらのリソースは、ノード障害で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 - Oracle Solaris Cluster HA for Oracle Database の 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 - このコンテナは、グループとしてフェイルオーバーする Oracle サーバーおよびリスナーと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の管理のためのツール

このセクションでは、インストールおよび構成タスクを実行するために使用できるツールについて説明します。

Oracle Solaris Cluster Manager ブラウザインタフェース

Oracle Solaris Cluster Manager ブラウザインタフェースは、多くの構成および管理タスクを実行できる Web ベースのツールです。これらのタスクには、グローバルクラスタまたはゾーンクラスタの管理、リソースとリソースグループの作成と構成、および Geographic Edition パートナーシップの作成と管理が含まれます。

Oracle Solaris Cluster Manager には、特定のアプリケーションでの Oracle Solaris Cluster データサービスの構成を自動化するためのウィザードも用意されています。これらのウィザードを使用すると、データサービスに必要な Oracle Solaris Cluster リソースを構成できます。このウィザードでは、Oracle Solaris Cluster 構成内で実行する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。Oracle Solaris Cluster 構成内で実行するアプリケーションソフトウェアをインストールおよび構成するには、各アプリケーションのユーティリティーおよび Oracle Solaris Cluster の保守コマンドを使用します。詳細は、各アプリケーションのドキュメントおよび Oracle Solaris Cluster のドキュメントセットを参照してください。

各データサービスウィザードは、データサービスの構成オプションのうちの制限されたサブセットのみをサポートしています。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、Oracle Solaris Cluster Manager または Oracle Solaris Cluster の保守コマンドを使用してデータサービスを手動で構成します。詳細は、Oracle Solaris Cluster のドキュメントを参照してください。

Oracle Solaris Cluster Manager には、グローバルクラスタまたはゾーンクラスタ内の論理ホスト名リソースと高可用性ストレージリソースの構成を自動化するためのウィザードも用意されています。

Oracle Solaris Cluster Manager を使用して実行できる代替手順に、その情報とそのタスクへの移動手順が記載されています。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。

`clsetup` ユーティリティー

`clsetup` ユーティリティーは、Oracle Solaris Cluster の一般的な管理に使用できるメニュー駆動型のインタフェースです。また、このユーティリティーを使用すると、データサービスリソースやリソースグループを構成することもできます。「リソースグループ」メニューを起動するには、`clsetup` のメインメニューからオプション 2 を選択します。詳細は、`clsetup(8CL)` のマニュアルページを参照してください。

Oracle Solaris Cluster の保守コマンド

Oracle Solaris 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。データサービスを登録および構成する方法については、そのデータサービスに関する本の手順を参照してください。たとえば、Oracle Solaris Cluster HA for Oracle を使用している場合は、『Oracle Solaris Cluster データサービス (Oracle Database 用）』 の 「Registering and Configuring HA for Oracle Database」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コマンドを使用する方法の詳細は、第2章「データサービスリソースの管理」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タスクごとのサマリー

次の表に、`clsetup` ユーティリティーと Oracle Solaris Cluster Manager がデータサービスリソースを管理するために実行できるタスクの要約を示します。これらのタスク

の詳細、およびコマンド行を使用して関連する手順を完了する方法の詳細は、第2章「データサービスリソースの管理」を参照してください。

表 2 データサービスリソースを管理するためのタスク

タスク
リソースタイプを登録する
リソースグループを作成する
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復旧アクションを中断する
リソースグループの自動復旧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する
リソースを削除する
リソースグループの現在のプライマリを切り替える
リソースを有効にする
リソースを無効にする
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、およびリソース構成情報を表示する
リソースプロパティーを変更する
リソース依存関係を設定する
リソースに関する `STOP_FAILED` エラーフラグをクリアする
リソースの `START_FAILED` リソース状態をクリアする
リソースグループにノードを追加する

標準プロパティー

データサービスを構成する場合は、次のクラスタ、リソースタイプ、リソース、リソースグループの各標準プロパティーを指定できます。また、システム定義プロパティーの変更や拡張プロパティーの作成のために、リソースプロパティー属性も使用できます。

このセクションでは、次のプロパティーについて説明します。

- クラスタプロパティー
- リソースタイププロパティー
- リソースプロパティー
- リソースグループプロパティー
- リソースプロパティー属性

クラスタプロパティー

クラスタプロパティーは、データサービスを管理するために使用されます。クラスタプロパティーについては、cluster(8CL) のマニュアルページを参照してください。

リソースタイププロパティー

リソースタイププロパティーは、Installed_nodes と RT_system を除き、管理ユーティリティーでは更新できません。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。

Oracle Solaris Cluster ソフトウェアで定義される各リソースタイププロパティーについては、rt_properties(7) のマニュアルページを参照してください。

リソースプロパティー

Oracle Solaris Cluster ソフトウェアで定義される各リソースプロパティーについては、r_properties(7) のマニュアルページを参照してください。

リソースグループプロパティー

Oracle Solaris Cluster ソフトウェアで定義される各リソースグループプロパティーについては、rg_properties(7) のマニュアルページを参照してください。

リソースプロパティー属性

リソースプロパティー属性を使用すると、システム定義プロパティーを変更したり、拡張プロパティーを作成したりできます。各プロパティーについては、property_attributes(7) のマニュアルページを参照してください。

ノードリストのプロパティー

データサービスを構成する場合は、次のノードリストのプロパティーを指定できます。

- `Installed_nodes` プロパティー - 詳細は、`rt_properties(7)` のマニュアルページを参照してください。
- `Nodelist` プロパティー – 詳細は、`rg_properties(7)` のマニュアルページを参照してください。
- `Auxnodelist` プロパティー – 詳細は、`clressharedaddress(8CL)` のマニュアルページを参照してください。

有効な RGM 名

このセクションでは、Resource Group Manager (RGM) の名前と値に指定できる有効な文字の要件について説明します。

このセクションの内容は次のとおりです。

- 36 ページの「RGM の有効な名前」
- 38 ページの「RGM 値」

RGM の有効な名前

RGM 名は、次のカテゴリに分類されます。

- リソースグループ名
- リソースタイプ名
- リソース名
- プロパティー名
- 列挙定数名

リソースタイプ名を除く名前の規則

リソースタイプ名を除き、すべての名前が次の規則に従う必要があります。

- 名前は ASCII で指定する必要があります。
- 名前は文字で始まる必要があります。
- 名前には、大文字と小文字、数字、ダッシュ (-)、および下線 (_) を含めることができます。
- 名前に使用できる文字の最大数は 255 です。

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形式は、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` 指令が含まれている場合、形式は次のとおりです。

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

- リソースタイプの RTR ファイルに `#$upgrade` 指令が含まれていない場合、形式は次のとおりです。

  vendor-id.*base-rt-name*

ピリオドによって、*vendor-id* と *base-rt-name* が区切られます。コロンによって、*base-rt-name* と *rt-version* が区切られます。

この形式での変数の要素は次のとおりです。

vendor-id

RTR ファイル内の `Vendor_id` リソースタイププロパティーの値であるベンダー ID 接頭辞を指定します。リソースタイプを開発している場合は、ベンダーを一意に識別するベンダー ID 接頭辞 (企業のティッカーシンボルなど) を選択します。

base-rt-name

RTR ファイル内の `Resource_type` リソースタイププロパティーの値である基本リソースタイプ名を指定します。

rt-version

RTR ファイル内の `RT_version` リソースタイププロパティーの値であるバージョン接尾辞を指定します。RTR ファイルに `#$upgrade` 指令が含まれている場合、バージョン接尾辞は、完全なリソースタイプ名の一部にすぎません。

注記 - 基本リソースタイプ名の 1 つのバージョンしか登録されていない場合、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、バージョン番号接尾辞、またはその両方を省略できます。

詳細は、35 ページの「リソースタイププロパティー」を参照してください。

例 2 `#$upgrade` 指令を使用したリソースタイプの完全な名前

この例では、RTR ファイル内の各プロパティーが次のように設定されている場合のリソースタイプの完全な名前を示します。

- `Vendor_id=ORCL`

- `Resource_type=sample`
- `RT_version=2.0`

この RTR ファイルで定義されるリソースタイプの完全な名前は次のとおりです。

```
ORCL.sample:2.0
```

例 3 `#$upgrade` 指令を使用しないリソースタイプの完全な名前

この例では、RTR ファイル内の各プロパティーが次のように設定されている場合のリソースタイプの完全な名前を示します。

- `Vendor_id=ORCL`
- `Resource_type=abc`

この RTR ファイルで定義されるリソースタイプの完全な名前は次のとおりです。

```
ORCL.abc
```

RGM 値

RGM 値は、プロパティー値と記述値の 2 つのカテゴリに分類されます。この両方のカテゴリが同じ規則を共有します。

- 値は ASCII で指定する必要があります。
- 値の最大長は 4M バイトから 1 を引いた数、つまり 4,194,303 バイトです。
- 値に次の文字を含めることはできません。
  - NULL
  - 改行
  - コンマ (,)
  - セミコロン (;)

♦ ♦ ♦ 第 2 章

2 データサービスリソースの管理

この章では、Oracle Solaris Cluster の保守コマンドを使用して、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方法について説明します。ほかのツールを使用して手順を完了できるかどうかを確認するには、32 ページの「データサービスリソースの管理のためのツール」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「Oracle Solaris Cluster データサービスの計画」および『Concepts for Oracle Solaris Cluster 4.4』を参照してください。

この章には次のセクションがあります。

- 93 ページの「リソースタイプ、リソースグループ、およびリソース構成情報の表示」
- 94 ページの「リソースタイプ、リソースグループ、およびリソースプロパティーの変更」
- 102 ページの「リソースに関する `STOP_FAILED` エラーフラグのクリア」
- 104 ページの「`Start_failed` リソース状態のクリア」
- 110 ページの「事前登録されたリソースタイプのアップグレード」
- 112 ページの「不注意による削除のあとの事前登録されたリソースタイプの再登録」
- 112 ページの「リソースグループへのノードの追加またはリソースグループからのノードの削除」
- 122 ページの「Oracle Solaris SMF サービスを Oracle Solaris Cluster とともに実行できるようにする」
- 132 ページの「Oracle Solaris Cluster データサービスの障害モニターの調整」

データサービスリソースを管理するためのタスクの概要

次の表に、Oracle Solaris 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タスクの要約を示します。この表にはまた、タスクを実行するための詳細な手順への相互参照も示されています。

表 3 データサービスリソースを管理するためのタスク

タスク	参照先
リソースタイプを登録する	43 ページの「リソースタイプを登録する方法」
リソースタイプをアップグレードする	45 ページの「リソースタイプのアップグレードをインストールおよび登録する方法」
リソースタイプをダウングレードする	52 ページの「リソースをそのリソースタイプの古いバージョンにダウングレードする方法」
フェイルオーバーまたはスケーラブルリソースグループを作成する	53 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方法」 55 ページの「スケーラブルリソースグループを作成する方法」
リソースグループに論理ホスト名または共有アドレスリソースやデータサービスリソースを追加する	61 ページの「リソースグループに論理ホスト名リソースを追加する方法 (`clsetup`)」 64 ページの「リソースグループに論理ホスト名リソースを追加する方法 (CLI)」 66 ページの「リソースグループに共有アドレスリソースを追加する方法 (`clsetup`)」 68 ページの「リソースグループに共有アドレスリソースを追加する方法 (CLI)」

タスク	参照先
	70 ページの「リソースグループにフェイルオーバーアプリケーションリソースを追加する方法」 72 ページの「リソースグループにスケーラブルアプリケーションリソースを追加する方法」
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、さらにリソースグループとそれに関連付けられているリソースをオンラインにする	78 ページの「リソースを有効にする方法」 76 ページの「リソースグループをオンラインにする方法」 77 ページの「リソースグループを優先プライマリに切り替える方法」
リソースグループを静止する	80 ページの「リソースグループを静止する方法」 80 ページの「リソースグループをただちに静止する方法」
リソースグループの自動復旧アクションを中断および再開する	82 ページの「リソースグループの自動復旧アクションを中断する方法」 82 ページの「リソースグループの自動復旧アクションをただちに中断する方法」 83 ページの「リソースグループの自動復旧アクションを再開する方法」
リソースには関係なく、リソースモニターを無効および有効にする	84 ページの「リソースの障害モニターを無効にする方法」 84 ページの「リソースの障害モニターを有効にする方法」
クラスタからリソースタイプを削除する	85 ページの「リソースタイプを削除する方法」
クラスタからリソースグループを削除する	87 ページの「リソースグループを削除する方法」
リソースグループからリソースを削除する	88 ページの「リソースを削除する方法」
リソースグループのプライマリを切り替える	89 ページの「リソースグループの現在のプライマリを切り替える方法」
リソースを無効にして、そのリソースグループを UNMANAGED 状態に移行する	91 ページの「リソースを無効にし、そのリソースグループを UNMANAGED 状態に移行する方法」
リソースタイプ、リソースグループ、およびリソース構成情報を表示する	93 ページの「リソースタイプ、リソースグループ、および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、およびリソースプロパティーを変更する	94 ページの「リソースタイププロパティーを変更する方法」 96 ページの「リソースグループプロパティーを変更する方法」 97 ページの「リソースプロパティーを変更する方法」
失敗した Resource Group Manager (RGM) プロセスのエラーフラグをクリアする	103 ページの「リソースに関する STOP_FAILED エラーフラグをクリアする方法」

タスク	参照先
`Start_failed` リソース状態をクリアする	105 ページの「リソースグループのスイッチオーバーによって `Start_failed` リソース状態をクリアする方法」 107 ページの「リソースグループの再起動によって `Start_failed` リソース状態をクリアする方法」 109 ページの「リソースの無効化および有効化によって `Start_failed` リソース状態をクリアする方法」
組み込みリソースタイプ `LogicalHostname` および `SharedAddress` を再登録する	112 ページの「不注意による削除のあとに事前登録されたリソースタイプを再登録する方法」
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 リストを更新し、リソースグループのノードリストを更新する	113 ページの「リソースグループへのノードの追加」
リソースグループからノードを削除する	116 ページの「リソースグループからのノードの削除」
組み込みリソースタイプ `LogicalHostname` および `SharedAddress` をアップグレードする	44 ページの「リソースタイプのアップグレード」
Solaris SMF サービスを Oracle Solaris Cluster とともに実行できるようにする	122 ページの「Oracle Solaris SMF サービスを Oracle Solaris Cluster とともに実行できるようにする」
Oracle Solaris Cluster データサービスの障害モニターを調整する	132 ページの「Oracle Solaris Cluster データサービスの障害モニターの調整」

注記 - この章の手順では、Oracle Solaris Cluster の保守コマンドを使用してこれらのタスクを完了する方法について説明します。また、その他のツールを使用してリソースを管理することもできます。これらのオプションの詳細は、32 ページの「データサービスリソースの管理のためのツール」を参照してください。

Oracle Solaris Cluster データサービスの構成および管理

Oracle Solaris Cluster データサービスを構成するには、次のタスクを実行します。

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

初期構成のあとにデータサービスの構成を更新するには、この章の手順を使用します。たとえば、リソースタイプ、リソースグループ、およびリソースプロパティーを

変更するには、94 ページの「リソースタイプ、リソースグループ、およびリソースプロパティーの変更」に進みます。

リソースタイプの登録

リソースタイプは、特定のタイプのすべてのリソースに適用される共通のプロパティーとコールバックメソッドを指定します。そのタイプのリソースを作成する前に、リソースタイプを登録する必要があります。リソースタイプの詳細は、第1章「Oracle Solaris Cluster データサービスの計画」を参照してください。

管理者は、ゾーンクラスタの内部に存在するリソースタイプ登録 (RTR) ファイルを指定することによって、ゾーンクラスタのリソースタイプを登録できます。つまり、そのファイルはゾーンのルートパスの下に存在する必要があります。ゾーンクラスタの内部にある RTR ファイル内の `Global_zone` プロパティーを `TRUE` に設定することはできません。ゾーンクラスタの内部にある RTR ファイルのタイプを `RTR_LOGICAL_HOSTNAME` や `RTR_SHARED_ADDRESS` にすることはできません。

注記 - Oracle Solaris の Trusted Extensions 機能を使用するゾーンクラスタでリソースタイプを登録し、かつ `Global_zone` リソースタイププロパティーを `TRUE` に設定する場合は、RTR ファイルをグローバルクラスタの `/usr/cluster/lib/rgm/rtreg` ディレクトリ内に配置する必要があります。

管理者はまた、`/usr/cluster/lib/rgm/rtreg` の場所からゾーンクラスタのリソースタイプを登録することもできます。ゾーンクラスタにいる管理者は、このディレクトリ内のどの RTR ファイルも変更できません。これにより、ゾーンクラスタから直接には設定できないいずれかのプロパティーが RTR ファイルに含まれている場合でも、ゾーンクラスタのシステムリソースタイプを登録できます。このプロセスにより、システムリソースタイプの配信のためのセキュアな方法が提供されます。

▼ リソースタイプを登録する方法

注記 - この手順は、いずれかのクラスタノードから実行します。

始める前に 登録する予定のリソースタイプの名前を用意していることを確認してください。リソースタイプ名は、データサービス名の省略名です。

1. **クラスタメンバーで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 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。

3. **リソースタイプが登録されたことを確認します。**

```
# clresourcetype show
```

例 4 リソースタイプの登録

次の例では、Oracle Solaris Cluster 構成内の HA Oracle Server アプリケーションを表す `SUNW.oracle_server:8` リソースタイプを登録します。

```
# clresourcetype register SUNW.oracle_server:8
# clresourcetype show SUNW.oracle_server:8

Resource Type:                                  SUNW.oracle_server:8
RT_description:                                 Resource type for Oracle Server
RT_version:                                     8
API_version:                                    2
RT_basedir:                                     /opt/SUNWscor/oracle_server
Single_instance:                                False
Proxy:                                          False
Init_nodes:                                     All potential masters
Installed_nodes:                                <All>
Failover:                                       True
Pkglist:                                        <NULL>
RT_system:                                      False
Global_zone:                                    False
```

次の手順 リソースタイプを登録したら、リソースグループを作成し、そのリソースグループにリソースを追加することができます。詳細は、53 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:

- `clresourcetype`(8CL)
- `clresourcegroup`(8CL)
- `clresource`(8CL)

リソースタイプのアップグレード

リソースタイプをアップグレードすると、リソースタイプの新しいバージョンで導入された新機能を使用できるようになります。リソースタイプの新しいバージョンは、以前のバージョンとは次の点で異なる可能性があります。

- リソースタイププロパティーのデフォルト設定が変更されている可能性があります。

- リソースタイプの新しい拡張プロパティーが導入されている可能性があります。
- リソースタイプの既存の拡張プロパティーが削除されている可能性があります。
- リソースタイプで宣言される一連の標準プロパティーが変更されている可能性があります。
- `min`、`max`、`arraymin`、`arraymax`、`default`、`tunability` などのリソースプロパティーの属性が変更されている可能性があります。
- 一連の宣言されるメソッドが異なる可能性があります。
- メソッドの実装や障害モニターが変更されている可能性があります。

リソースタイプをアップグレードするには、次のセクションで説明されているタスクを実行します。

1. 45 ページの「リソースタイプのアップグレードをインストールおよび登録する方法」
2. 46 ページの「既存のリソースをリソースタイプの新しいバージョンに移行する方法」

▼ リソースタイプのアップグレードをインストールおよび登録する方法

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何を実行する必要があるかを確認するには、リソースタイプに関するドキュメントを参照してください。次のリストのうちの 1 つのアクションが必要になります。

- ノードを非クラスタモードでリブートする必要があります。
- ノードをクラスタモードで実行されたままにすることができますが、リソースタイプのすべてのインスタンスのモニタリングを無効にする必要があります。
- ノードをクラスタモードで実行されたままにし、かつリソースタイプのすべてのインスタンスのモニタリングをオンにしたままにすることができます。

ノードを非クラスタモードでリブートする必要がある場合は、ローリングアップグレードを実行することによって、サービスが失われることのないようにします。ローリングアップグレードでは、各ノードに個別にパッケージをインストールし、その間は残りのノードをクラスタモードで実行されたままにします。

1. **クラスタメンバーで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**
2. **リソースタイプのアップグレードのパッケージを、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にインストールします。**
3. **リソースタイプの新しいバージョンを登録します。**

正しいバージョンのリソースタイプが確実に登録されるようにするには、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*

この形式については、37 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` プロパティーを、リソースタイプのアップグレード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。**

この手順は、リソースタイプのアップグレードのパッケージがすべてのクラスタノードにインストールされるわけではない場合に実行する必要があります。

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

```
# 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。

▼ 既存のリソースをリソースタイプの新しいバージョンに移行する方法

次の手順では、`clresource` コマンドを使用してこのタスクを実行する方法について説明します。ただし、このタスクでは `clresource` コマンドの使用に限定されるわけではありません。`clresource` コマンドの代わりに、`clsetup` コマンドの Oracle Solaris Cluster Manager または Resource Group オプションを使用してこのタスクを実行できます。詳細は、`clsetup(8CL)` を参照してください。

始める前に リソースをリソースタイプの新しいバージョンにいつ移行できるかを確認するには、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。

- いつでも

- リソースがモニターされていない場合のみ
- リソースがオフラインである場合のみ
- リソースが無効になっている場合のみ
- リソースグループが非管理状態にある場合のみ

手順によっては、リソースの既存のバージョンをアップグレードできないと記載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、次の代替方法を検討してください。

- リソースを削除し、それを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
- リソースをリソースタイプの古いバージョンのままにする

1. **クラスタメンバーで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **移行されるリソースタイプのリソースごとに、リソースまたはそのリソースグループの状態を適切な状態に変更します。**

   - **リソースをいつでも移行できる場合、アクションは必要ありません。**

   - **リソースがモニターされていない場合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
     ```

注記 - 移行するリソースにほかのリソースが依存している場合、この手順は失敗します。この状況では、出力されるエラーメッセージを参照して、依存するリソースの名前を確認します。次に、移行するリソースとすべての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。

   - **リソースが無効になっている場合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
     ```

注記 - 移行するリソースにほかのリソースが依存している場合、この手順は失敗します。この状況では、出力されるエラーメッセージを参照して、依存するリソースの名前を確認します。次に、移行するリソースとすべての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非管理状態にある場合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 -g resource-group +
# clresourcegroup offline resource-group
# clresourcegroup unmanage resource-group
```

これらのコマンドの各項目は次のとおりです。

resource-group

非管理状態にするリソースグループを指定します。

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

必要に応じて、同じリソースのほかのプロパティーを同じコマンドで適切な値に設定します。これらのプロパティーを設定するには、このコマンドの -p オプションを指定します。

ほかのプロパティーを設定する必要があるかどうかを確認するには、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。ほかのプロパティーの設定は、次の理由で必要になることがあります。

- リソースタイプの新しいバージョンで拡張プロパティーが導入されている。
- リソースタイプの新しいバージョンで既存のプロパティーのデフォルト値が変更されている。

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

注記 - リソースタイプの既存のバージョン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. **ステップ 2 で入力したコマンドを逆にすることによって、リソースまたはリソースグループの以前の状態を復元します。**

- **リソースをいつでも移行できる場合、アクションは必要ありません。**

注記 - いつでも移行できるリソースを移行したあと、リソース検証に正しいリソースタイプバージョンが表示されないことがあります。この状況では、リソース検証に正しいリソースタイプバージョンが確実に表示されるように、リソースの障害モニターを無効にしてふたたび有効にします。

- **リソースがモニターされていない場合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
  ```

注記 - ステップ 2 で、移行するリソースに依存するほかのリソースを無効にした場合は、それらの依存するリソースも有効にします。

- **リソースが無効になっている場合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
  ```

注記 - ステップ 2 で、移行するリソースに依存するほかのリソースを無効にした場合は、それらの依存するリソースも有効にします。

- **リソースグループが非管理状態にある場合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 5 オフラインである場合にのみ移行できるリソースの移行

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できるリソースの移行を示します。新しいリソースタイプのパッケージには、新しいパスに配置されるメソッドが含まれます。メソッドはインストール中に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がインストールされるまで、リソースを無効にする必要はありません。

この例でのリソースの特性は次のとおりです。

- 新しいリソースタイプバージョンは 2.0 です。
- リソース名は `myresource` です。
- リソースタイプ名は `myrt` です。
- 新しい RTR ファイルは `/opt/XYZmyrt/etc/XYZ.myrt` に存在します。
- 移行されるリソースに関する依存関係は存在しません。
- 移行されるリソースを、それが属するリソースグループをオンラインにしたままオフラインにすることができます。

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

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

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

例 6 モニターされていない場合のみ移行できるリソースの移行

この例では、リソースがモニターされていない場合のみ移行できるリソースの移行を示します。新しいリソースタイプのパッケージには、モニターと RTR ファイルのみが含まれます。モニターはインストール中に上書きされるため、アップグレードパッケージがインストールされる前に、リソースのモニタリングを無効にする必要があります。

この例でのリソースの特性は次のとおりです。

- 新しいリソースタイプバージョンは 2.0 です。
- リソース名は `myresource` です。
- リソースタイプ名は `myrt` です。
- 新しい RTR ファイルは `/opt/XYZmyrt/etc/XYZ.myrt` に存在します。

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がインストールされる前に、リソースのモニタリングを無効にするために次のコマンドが実行されます。

   ```
   # clresource unmonitor myresource
   ```
2. アップグレードパッケージが、サプライヤの指示に従ってすべてのクラスタノードにインストールされます。
3. リソースタイプの新しいバージョンを登録するために、次のコマンドが実行されます。

   ```
   # clresourcetype register -f /opt/XYZmyrt/etc/XYZ.myrt myrt
   ```
4. `Type_version` プロパティーを新しいバージョンに変更するために、次のコマンドが実行されます。

   ```
   # clresource set -p Type_version=2.0 myresource
   ```
5. リソースのモニタリングをその移行後に有効にするために、次のコマンドが実行されます。

   ```
   # clresource monitor myresource
   ```

▼ リソースタイプの古い未使用のバージョンを登録解除する方法

既存のリソースをすべてリソースタイプの最新バージョンに移行しており、そのリソースタイプの古いバージョンがもう必要なくなった場合は、古いバージョンを登録解除します。

1. **クラスタメンバーで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **登録されているリソースタイプバージョンのリストを取得し、もう使用しないバージョンを登録解除します。**

```
# clresourcetype list | grep myrt
XYZ.myrt:1.0
XYZ.myrt:2.0
# clresourcetype unregister XYZ.myrt:1.0
```

3. **リソースタイプの `RT_system` プロパティーが `TRUE` に設定されている場合は、リソースタイプを削除する前に、まずそれを `FALSE` に設定する必要があります。**

   ステップ 2 の `clresourcetype unregister` を実行したあとに次のテキストが表示された場合は、下の出力の次に表示されているコマンドを入力することにより、このプロパティーを `TRUE` に設定する必要があります。

```
clrt: (C944871) Operation not allowed on system resource type <XYZ.myrt:1.0>
# clresourcetype set -p RT_system=FALSE XYZ.myrt:1.0
# clresourcetype unregister XYZ.myrt:1.0
```

   デフォルトでは、事前にインストールされたリソースタイプ `LogicalHostname` および `SharedAddress` では `RT_system` プロパティーが `TRUE` に設定されています。これらの事前にインストールされたいずれかのリソースタイプの新しいバージョンにアップグレードしている場合は、古いバージョンを登録解除する前に `RT_system` を `FALSE` に設定する必要があります。この例では、既存の `LogicalHostname` リソースはすべて、`LogicalHostname` のバージョン 4.0 に移行されています。

```
# clresourcetype list
…
SUNW.LogicalHostname.3
SUNW.LogicalHostname:4
…
# clresourcetype set -p RT_system=FALSE SUNW.LogicalHostname:3
# clresourcetype unregister SUNW.LogicalHostname:3
```

リソースタイプのダウングレード

リソースをそのリソースタイプの古いバージョンにダウングレードできます。リソースをリソースタイプの古いバージョンにダウングレードするための条件は、リソースタイプの新しいバージョンにアップグレードするための条件より制限的です。そのリソースを含むリソースグループが非管理状態にある必要があります。

▼ リソースをそのリソースタイプの古いバージョンにダウングレードする方法

次の手順では、clresource コマンドを使用してこのタスクを実行する方法について説明します。ただし、このタスクでは clresource コマンドの使用に限定されるわけではありません。clresource コマンドの代わりに、clsetup コマンドの Oracle Solaris Cluster Manager または Resource Group オプションを使用してこのタスクを実行できます。詳細は、clsetup(8CL) を参照してください。

1. **クラスタメンバーで、RBAC の承認 `solaris.cluster.modify` および `solaris.cluster.admin` を提供する `root` 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内のすべてのリソースを無効にします。**

   `clresource disable -g` *resource-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リソースタイプの古いバージョンを再登録します。**

   この手順は、ダウングレード先のバージョンがすでに登録されていない場合にのみ実行します。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、この手順を省略してください。

   `clresourcetype register` *resource-type-name*

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

   必要に応じて、同じリソースのほかのプロパティーを同じコマンドで適切な値に編集します。

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

7. **ステップ 3 で無効にしたすべてのリソースを有効にします。**

   `#` **`clresource enable -g`** *resource-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ノード群で共にオンラインまたはオフラインになります。空のリソースグループを作成してから、そのリソースグループにリソースを配置する必要があります。

リソースグループのタイプには、**フェイルオーバー**と**スケーラブル**の 2 つがあります。フェイルオーバーリソースグループは常に 1 つのノードでのみオンラインにできますが、スケーラブルリソースグループは同時に複数のノードでオンラインにできます。

次の手順では、`clresourcegroup` コマンドを使用してリソースグループを作成する方法について説明します。詳細は、`clresourcegroup(8CL)` のマニュアルページを参照してください。

リソースグループの概念については、第1章「Oracle Solaris Cluster データサービスの計画」および『Concepts for Oracle Solaris Cluster 4.4』を参照してください。

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用して、フェイルオーバーまたはスケーラブルリソースグループを作成することもできます。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の「Oracle Solaris Cluster Manager にアクセスする方法」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方法

フェイルオーバーリソースグループには、次のタイプのリソースが含まれます。

- ネットワークアドレスリソース。これは、組み込みリソースタイプ `LogicalHostname` および `SharedAddress` のインスタンスです。
- フェイルオーバーリソース。これは、フェイルオーバーデータサービスのためのデータサービスアプリケーションリソースです。

ネットワークアドレスリソースとそれが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するか、またはスイッチオーバーされるとクラスタノード間で移動します。

この手順は、いずれか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、RBAC の承認 `solaris.cluster.modify` を提供する `root` 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 nodelist] resource-group
```

-n *nodelist*

このリソースグループをマスターできるノードの、コンマで区切られた順序付きリストを指定します。

このリストはオプションです。このリストを省略した場合は、クラスタ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。

resource-group

追加するフェイルオーバーリソースグループの選択した名前を指定します。この名前は ASCII 文字で始まる必要があります。

3. **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
```

例 7 フェイルオーバーリソースグループの作成

この例では、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。クラスタノード `phys-schost-1` および `phys-schost-2` 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ら、こ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、59 ページの「リソースグループへのリソースの追加」を参照してください。

参照 `clresourcegroup`(8CL) 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方法

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で使用されます。共有アドレス機能は、スケーラブルサービスの複数のインスタンスを 1 つのサービスとして見せることができる Oracle Solaris Cluster ネットワーキング機能です。最初に、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する必要があります。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。スケーラブルサービスの各インスタンスは、異なるクラスタノード上で実行する必要があります。

注記 - この手順は、いずれか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、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **この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 -S [-p Maximum_primaries=m] [-p Desired_primaries=n] \
[-n nodelist] resource-group
```

-S
: このリソースグループを複数マスターにすることを指定します。-p `Maximum_primaries` および -p `Desired_primaries` オプションが省略された場合は、両方のプロパティーがリソースグループのノードリスト内のノードの数に設定されます。

-p `Maximum_primaries=`*m*
: このリソースグループのアクティブなプライマリの最大数を指定します。

-p `Desired_primaries=`*n*
: このリソースグループが起動を試みるべきアクティブなプライマリの数を指定します。

-n *nodelist*

このリソースグループを使用可能にするノードの、コンマで区切られた順序付きリストを指定します。

このリストはオプションです。このリストを省略した場合は、クラスタ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。

resource-group

追加するスケーラブルリソースグループの選択した名前を指定します。この名前は ASCII 文字で始まる必要があります。

4. スケーラブル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
```

例 8 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` のグローバルクラスタでホストされます。この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 -S \
-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ら、そのリソースグループにスケーラブルアプリケーションリソースを追加できます。詳細は、72 ページの「リソースグループにスケーラブルアプリケーションリソースを追加する方法」を参照してください。

参照 `clresourcegroup(8CL)` のマニュアルページ。

共有ファイルシステム上でのフェイルオーバーおよびスケーラブルデータサービスの構成

NAS デバイスがインストールおよび構成されたあと、`ScalMountPoint` リソースを使用するとフェイルオーバーおよびスケーラブルアプリケーションを構成できます。

`ScalMountPoint` リソースタイプのインスタンスは、次のいずれかのタイプのファイルシステムのマウントポイントを表します。

- QFS 共有ファイルシステム
- ネットワーク接続ストレージ (NAS) デバイス上のファイルシステム

この NAS デバイスとファイルシステムがすでに、Oracle Solaris Cluster ソフトウェアで使用できるように構成されている必要があります。

`ScalMountPoint` リソースタイプは、スケーラブルリソースタイプです。このリソースタイプのインスタンスは、リソースグループが現在オンラインになっているどのノード上でもオンラインです。

▼ `ScalMountPoint` リソースを使用してフェイルオーバーアプリケーションを構成する方法

始める前に この手順では、長形式の Oracle Solaris Cluster コマンドを使用して説明します。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、コマンドは同一です。

この手順を実行するには、RBAC (役割に基づくアクセス制御) の承認 `solaris.cluster.read` および `solaris.cluster.modify` を提供する `root` 役割になります。

1. **NAS NFS ファイルシステムの `ScalMountPoint` リソースを含むスケーラブルリソースグループを作成します。**

```
# clresourcegroup create -p RG_mode=Scalable \
-p Desired_primaries=num_active_primary \
-p Maximum_primaries=max_num_active_primary scalmp-rg
# clresourcetype register SUNW.ScalMountPoint
# clresource create -g scalmp-rg -t SUNW.ScalMountPoint \
```

```
-p TargetFileSystem=nas_device:path \
-p FileSystemType=nas \
-p MountPointDir=fs_mountpoint scalmp-rs
# clresourcegroup online -eM scalmp-rg
```

2. **(オプション) アプリケーションバイナリが NAS NFS ファイルシステムを使用しており、ストレージ障害が検出されたらリソースが自動的にフェイルオーバーするようにする場合は、`RebootOnFailure` プロパティーを `True` に設定します。**

   このプロパティーを設定すると、ストレージ接続で障害が発生した場合にリソースが `STOP_FAILED` 状態にならなくなります。代わりに、`ScalMountPoint` リソースが存在するノードがリブートされ、リソースは別のクラスタノード上で再起動します。

   注記 - この障害ケースが、ほかのサービスの可用性に悪影響を与える可能性があります。このプロパティーを設定する前に、クラスタノード上で実行されているすべてのサービスへのその影響を考慮したことを確認してください。代わりに、このサービスをゾーンクラスタで構成することによって、`RebootOnFailure` 設定のほかのサービスへの影響を制限できます。その場合、リブートはそのゾーンクラスタ内のサービスにのみ影響を与えます。

   ```
   # clresource set -p RebootOnFailure=True scalable-mount-point-resource
   ```

3. **フェイルオーバーアプリケーションリソースを含むフェイルオーバーリソースグループを作成します。**

   ```
   # clresourcegroup create -p rg_affinities=++scalmp-rg app-fo-rg
   ```

   このフェイルオーバーアプリケーションリソースグループは、ステップ 1 で作成されたリソースグループに対する強い肯定的なアフィニティーを持っている必要があります。

   ```
   # clresourcetype register app_resource_type

   # clresource create -g app-fo-rg -t app_resource_type \
   -p Resource_dependencies_offline_restart=scalmp-rs \
   …
   app-fo-rs
   ```

   このフェイルオーバーアプリケーションリソースは、ステップ 1 で作成された `ScalMountPoint` リソースに対するオフライン再起動依存関係を持っている必要があります。

   ```
   # clresourcegroup online -eM app-fo-rg
   ```

▼ `ScalMountPoint` リソースを使用してスケーラブルアプリケーションを構成する方法

始める前に この手順では、長形式の Oracle Solaris Cluster コマンドを使用して説明します。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、コマンドは同一です。

この手順を実行するには、RBAC の承認 `solaris.cluster.read` および `solaris.cluster.modify` を提供する `root` 役割になります。

1. **NAS NFS ファイルシステムの `ScalMountPoint` リソースを含むスケーラブルリソースグループを作成します。**

```
# clresourcegroup create -p RG_mode=Scalable \
-p Desired_primaries=num_active_primary \
-p Maximum_primaries=max_num_active_primary scalmp-rg

# clresourcetype register SUNW.ScalMountPoint

# clresource create -g scalmp-rg -t SUNW.ScalMountPoint \
-p TargetFileSystem=nas_device:path \
-p FileSystemType=nas \
-p MountPointDir=fs_mountpoint scalmp-rs

# clresourcegroup online -eM scalmp-rg
```

2. **アプリケーションリソースを含むスケーラブルリソースグループを作成します。**

```
# clresourcegroup create -p RG_mode=Scalable \
-p Maximum_primaries=max_num_active_primary \
-p Desired_primaries=num_active_primary \
-p rg_affinities=++scalmp-rg app-rg
```

このアプリケーションリソースグループは、ステップ 1 で作成されたリソースグループに対する強い肯定的なアフィニティーを持っている必要があります。

```
# clresourcetype register app_resource_type
# clresource create -g app-rg -t app_resource_type \
…
-p Scalable=True \
-p resource_dependencies_offline_restart=scalmp-rs app-rs
# clresourcegroup online -eM app-rg
```

このアプリケーションリソースは、ステップ 1 で作成された `ScalMountPoint` リソースに対するオフライン再起動依存関係を持っている必要があります。

リソースグループへのリソースの追加

リソースは、リソースタイプのインスタンス化です。RGM でリソースを管理するには、リソースグループにリソースを追加しておく必要があります。このセクションでは、次の 3 つ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

Oracle Solaris Cluster には、リソースグループにリソースを追加するための次のツールが用意されています。

- Oracle Solaris Cluster Manager ブラウザインタフェース。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。
- `clsetup` ユーティリティー。詳細は、`clsetup`(8CL) のマニュアルページを参照してください。
- Oracle Solaris Cluster の保守コマンド。

`clsetup` ユーティリティーのウィザードを使用するか、または Oracle Solaris Cluster の保守コマンドを使用すると、リソースグループに論理ホスト名リソースを追加できます。

Oracle Solaris Cluster Manager を使用して、論理ホスト名リソースとそれを含むリソースグループを 1 回の操作で作成することもできます。

Oracle Solaris Cluster Manager および `clsetup` ユーティリティーでは、対話形式でリソースグループにリソースを追加できます。これらのリソースを対話形式で構成すると、コマンドの構文エラーや省略から生じる構成エラーの可能性が低減されます。`clsetup` ユーティリティーおよび Oracle Solaris Cluster Manager では、必要なすべてのリソースが作成され、リソース間で必要なすべての依存関係が設定されることが保証されます。

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内に構成します。フェイルオーバーデータサービスのためのデータサービスリソースをフェイルオーバーリソースグループ内に構成します。フェイルオーバーリソースグループには、論理ホスト名リソースと、データサービス用のアプリケーションリソースの両方が含まれています。スケーラブルリソースグループには、スケーラブルサービス用のアプリケーションリソースのみが含まれています。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、個別のフェイルオーバーリソースグループ内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間で有効にするには、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の間の依存関係を指定する必要があります。

注記 - `DEPRECATED` フラグは、論理ホスト名または共有アドレスリソースを非推奨のアドレスとしてマークします。これらのアドレスは、フェイルオーバーやスイッチオーバーによって異なるクラスタノードに移行できるため、アウトバウンドリクエストには適していません。

注記 - パブリックネットワーク管理 (PNM) オブジェクトには、IPMP、トランクおよびデータリンクマルチパス (DLMP) リンクアグリゲーション、およびリンクアグリゲーションに直接基づく VNIC が含まれます。

リソースの詳細は、『Concepts for Oracle Solaris Cluster 4.4』および第1章「Oracle Solaris Cluster データサービスの計画」を参照してください。

▼ リソースグループに論理ホスト名リソースを追加する方法 (clsetup)

次の手順では、clsetup ユーティリティーを使用してリソースグループに論理ホスト名リソースを追加する方法について説明します。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、Oracle Solaris Cluster の長い形式の保守コマンドを使用します。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、コマンドは同一です。

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用して、論理ホスト名リソースまたはノードごとの論理ホスト名リソースと、それを含む新しいリソースグループを 1 回の操作で作成することもできます。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。ログインしたあと、「タスク」をクリックし、「論理ホスト名」または「ノードごとの論理ホスト名」をクリックしてウィザードを起動します。

このウィザードでは、すべてのクラスタノードが同じ root パスワードを持つ必要があります。

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。

- リソースによって使用可能になる論理ホスト名ごとに 1 つ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る。
- PNM オブジェクトを使用している場合は、論理ホスト名リソースをオンラインにできるノードでオブジェクトが構成されている。パブリックネットワーク管理 (PNM) オブジェクトには、インターネットプロトコルネットワークマルチパス (IPMP) グループ、トランクおよびデータリンクマルチパス (DLMP) リンクアグリゲーション、およびリンクアグリゲーションに直接基づく VNIC が含まれます。
- すべての論理ホスト名の IP アドレスのサブネットとネットマスクのエントリが /etc/netmasks ファイルにあることを確認してください。必要に応じて、/etc/netmasks ファイルを編集して、不足しているエントリがある場合は追加します。

1. **いずれかのクラスタノード上で root 役割になります。**

2. **clsetup ユーティリティーを起動します。**

```
# 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番号を入力します。**

「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番号を入力します。**

   clsetup ユーティリティーは、このタスクを実行するための前提条件のリストを表示します。

5. **前提条件を満たしていることを確認します。**

   clsetup ユーティリティーは、論理ホスト名リソースをオンラインにできるクラスタノード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できるノードを選択します。**

   - **任意の順序で一覧表示されたすべてのノードのデフォルトの選択を受け入れるには、a と入力します。**
   - **一覧表示されたノード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またはスペースで区切られたリストを入力します。**
   - **すべてのノードを特定の順序で選択するには、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ます。**

     各ノードが、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一覧表示されていることを確認してください。リスト内の最初のノードは、このリソースグループのプライマリノードです。

7. **ノードの選択を確定するには、d と入力します。**

   clsetup ユーティリティーは、このリソースが使用可能にする論理ホスト名を指定できる画面を表示します。

8. **このリソースが利用可能にする論理ホスト名を入力します。**

   - **指定した論理ホスト名で複数の PNM オブジェクトが構成されている場合は、clsetup ユーティリティーによって、使用する PNM オブジェクトを指定できる画面が表示されます。**

     ステップ 9 に進みます。
   - **指定した論理ホスト名で 1 つの PNM オブジェクトのみが構成されている場合は、clsetup ユーティリティーによって、このユーティリティーが作成する Oracle Solaris Cluster オブジェクトの名前が一覧表示されます。**

     ステップ 10 にスキップします。

9. **使用可能な PNM オブジェクトのリストから、クラスタノードごとに 1 つのオブジェクトを選択します。**

   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 Oracle Solaris Cluster オブジェクトの名前を一覧表示します。

10. **いずれかの Oracle Solaris Cluster オブジェクトに別の名前が必要な場合は、次のように名前を変更します。**

    a. **変更する名前のオプション番号を入力します。**

       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 Oracle Solaris Cluster オブジェクトの名前のリストに戻ります。

11. **選択した Oracle Solaris Cluster オブジェクト名を確定するには、d と入力します。**

    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 Oracle Solaris Cluster 構成に関する情報を表示します。

12. **構成を作成するには、c と入力します。**

    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。構成が完了すると、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実行したコマンドを一覧表示します。

13. **(オプション) clsetup ユーティリティーが終了するまで繰り返し q と入力し、Return キーを押します。**

    必要に応じて、ほかの必要なタスクを実行している間、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できます。clsetup の終了を選択した場合は、ユーザーがこのユーティリティーを再起動すると、既存の論理ホスト名リソースグループがユーティリティーによって認識されます。

14. **論理ホスト名リソースが作成されたことを確認します。**

    この目的には、clresource(8CL) 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 *node_name*-rg という名前を割り当てます。

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

▼ リソースグループに論理ホスト名リソースを追加する方法 (CLI)

注記 - リソースグループに論理ホスト名リソースを追加すると、そのリソースの拡張プロパティーが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、リソースグループにリソースを追加したあとに、そのリソースを変更する必要があります。詳細は、101 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方法」を参照してください。

この手順は、いずれか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いることを確認してください。
  - リソースの追加先とな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予定のホスト名
- すべての論理ホスト名の IP アドレスのサブネットとネットマスクのエントリが `/etc/netmasks` ファイルにあることを確認してください。必要に応じて、`/etc/netmasks` ファイルを編集して、不足しているエントリがある場合は追加します。

1. **クラスタメンバーで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **リソースグループに論理ホスト名リソースを追加します。**

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

-g *resource-group*

このリソースが存在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内のサービスと通信するために使用する UNIX ホスト名 (論理ホスト名) のコンマ区切りリストを指定します。

完全修飾ホスト名が必要な場合は、`-h` オプションを使用して完全修飾名を指定する必要があります。

-N *netiflist*

各ノード上に存在する PNM オブジェクトを識別する、オプションのコンマ区切りリストを指定します。*netiflist* 内の各要素は、`netif@node` の形式である必要があります。`netif` は、`sc_ipmp0` などの PNM オブジェクト名として指定できます。ノードは、`sc_ipmp0@1` や `sc_ipmp@phys-schost-1` などのノード名またはノード ID で識別できます。

注記 - Oracle Solaris Cluster は、`netif` に対するアダプタ名の使用をサポートしていません。

resource

選択したオプションのリソース名を指定します。リソース名で完全修飾名を使用することはできません。

3. 論理ホスト名リソースが追加されたことを確認します。

```
# clresource show resource
```

例 9 リソースグループへの論理ホスト名リソースの追加

この例では、リソースグループ (`resource-group-1`) への論理ホスト名リソース (`resource-1`) の追加を示します。

```
# 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{phys-schost-1}:                         True
Enabled{phys-schost-2}:                         True
Monitored{phys-schost-1}:                       True
Monitored{phys-schost-2}:                       True
```

例 10 IPMP グループを識別する論理ホスト名リソースの追加

この例では、リソースグループ `nfs-fo-rg` への次の論理ホスト名リソースの追加を示します。

- ノード 1 およびノード 2 上の IPMP グループ `sc_ipmp0` を識別する、`cs23-rs` という名前のリソース
- ノード 1 およびノード 2 上の IPMP グループ `sc_ipmp1` を識別する、`cs24-rs` という名前のリソース

```
# 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ら、そのリソースをオンラインにするために 76 ページの「リソースグループをオンラインにする方法」を参照してください。

注意事項 リソースを追加すると、そのリソースが Oracle Solaris Cluster ソフトウェアによって検証されます。検証が失敗した場合、`clreslogicalhostname` コマンドは、エラー

メッセージを出力して終了します。検証が失敗した理由を確認するには、各ノード上の syslog にエラーメッセージがないかどうかチェックしてください。このメッセージは、必ずしも clreslogicalhostname コマンドを実行したノードではなく、検証を実行したノード上で表示されます。

参照 clreslogicalhostname(8CL) のマニュアルページ。

▼ リソースグループに共有アドレスリソースを追加する方法 (clsetup)

次の手順では、clsetup ユーティリティーを使用してリソースグループに共有アドレスリソースを追加する方法について説明します。この手順は、いずれかのクラスタノードから実行します。

この手順では、Oracle Solaris Cluster の長い形式の保守コマンドを使用します。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、コマンドは同一です。

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。

- リソースによって使用可能に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る。
- PNM オブジェクトを使用している場合は、共有アドレスリソースをオンラインにできるノードでオブジェクトが構成されている。
- すべての論理ホスト名の IP アドレスのサブネットとネットマスクのエントリが /etc/netmasks ファイルにあることを確認してください。必要に応じて、/etc/netmasks ファイルを編集して、不足しているエントリがある場合は追加します。

1. **いずれかのクラスタノード上で root 役割になります。**

2. **clsetup ユーティリティーを起動します。**

   ```
   # 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番号を入力します。**
   「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番号を入力します。**
   clsetup ユーティリティーは、このタスクを実行するための前提条件のリストを表示します。

5. **前提条件を満たしていることを確認します。**

clsetup ユーティリティーは、共有アドレスリソースをオンラインにできるクラスタノードを一覧表示します。

6. **共有アドレスリソースをオンラインにできるノードを選択します。**

   - **任意の順序で一覧表示されたすべてのノードのデフォルトの選択を受け入れるには、a と入力します。**
   - **一覧表示されたノード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またはスペースで区切られたリストを入力します。**
   - **すべてのノードを特定の順序で選択するには、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ます。**

7. **ノードの選択を確定するには、d と入力します。**

   clsetup ユーティリティーは、このリソースが使用可能にする共有アドレスを指定できる画面を表示します。

8. **このリソースが利用可能にする共有アドレスを入力します。**

   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 Oracle Solaris Cluster オブジェクトの名前を一覧表示します。

9. **いずれかの Oracle Solaris Cluster オブジェクトに別の名前が必要な場合は、次のように名前を変更します。**

   a. **変更する名前のオプション番号を入力します。**

      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 Oracle Solaris Cluster オブジェクトの名前のリストに戻ります。

10. **選択した Oracle Solaris Cluster オブジェクト名を確定するには、d と入力します。**

    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 Oracle Solaris Cluster 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 と入力します。**

    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。構成が完了すると、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実行したコマンドを一覧表示します。

12. **(オプション) `clsetup` ユーティリティーが終了するまで繰り返し `q` と入力し、Return キーを押します。**

    必要に応じて、ほかの必要なタスクを実行している間、`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できます。`clsetup` の終了を選択した場合は、ユーザーがこのユーティリティーを再起動すると、既存の共有アドレスリソースグループがユーティリティーによって認識されます。

13. **共有アドレスリソースが作成されたことを確認します。**

    この目的には、`clresource`(8CL) 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 *node_name*`-rg` という名前を割り当てます。

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

▼ リソースグループに共有アドレスリソースを追加する方法 (CLI)

注記 - リソースグループに共有アドレスリソースを追加すると、そのリソースの拡張プロパティーが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、リソースグループにリソースを追加したあとに、そのリソースを変更する必要があります。詳細は、101 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方法」を参照してください。

この手順は、いずれか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。
  - リソースの追加先となるリソースグループの名前。このグループは、以前に作成したフェイルオーバーリソースグループである必要があります。
  - リソースグループに追加する予定のホスト名。
- すべての論理ホスト名の IP アドレスのサブネットとネットマスクのエントリが `/etc/netmasks` ファイルにあることを確認してください。必要に応じて、`/etc/netmasks` ファイルを編集して、不足しているエントリがある場合は追加します。

1. **クラスタメンバーで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **リソースグループに共有アドレスリソースを追加します。**

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

-g *resource-group*

リソースグループ名を指定します。

-h *hostnamelist,…*

共有アドレスホスト名のコンマ区切りリストを指定します。

-X *auxnodelist*

共有アドレスをホストできるが、フェイルオーバーが発生してもプライマリとして機能することのないクラスタノードを識別するノード名または ID のコンマ区切り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ノードリストで潜在的な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、このリストはデフォルトで、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。

-N *netiflist*

各ノード上に存在する PNM オブジェクトを識別する、オプションのコンマ区切りリストを指定します。*netiflist* 内の各要素は、`netif@node` の形式である必要があります。`netif` は、`sc_ipmp0` などの PNM オブジェクト名として指定できます。ノードは、`sc_ipmp0@1` や `sc_ipmp@phys-schost-1` などのノード名またはノード ID で識別できます。

注記 - Oracle Solaris Cluster は、`netif` に対するアダプタ名の使用をサポートしていません。

---

resource

選択したオプションのリソース名を指定します。

3. 共有アドレスリソースが追加および検証されたことを確認します。

```
# clresource show resource
```

例 11 リソースグループへの共有アドレスリソースの追加

この例では、リソースグループ (`resource-group-1`) への共有アドレスリソース (`resource-1`) の追加を示します。

```
# 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{phys-schost-1}:                         False
Enabled{phys-schost-2}:                         False
Monitored{phys-schost-1}:                       True
Monitored{phys-schost-2}:                       True
```

次の手順 共有アドレスリソースを追加したら、76 ページの「リソースグループをオンラインにする方法」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、そのリソースが Oracle Solaris Cluster ソフトウェアによって検証されます。検証が失敗した場合、`clressharedaddress` コマンドは、エラーメッセージを出力して終了します。検証が失敗した理由を確認するには、各ノード上の `syslog` にエラーメッセージがないかどうかチェックしてください。このメッセージは、必ずしも `clressharedaddress` コマンドを実行したノードではなく、検証を実行したノード上で表示されます。

参照 `clressharedaddress(8CL)` のマニュアルページ。

▼ リソースグループにフェイルオーバーアプリケーションリソースを追加する方法

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内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。

注記 - この手順は、いずれか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いることを確認してください。

- リソースの追加先となるフェイルオーバーリソースグループの名前
- このリソースのリソースタイプの名前
- この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 (以前に同じリソースグループに含めた論理ホスト名)

注記 - この手順は、プロキシリソースにも適用されます。

---

1. **クラスタメンバーで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **リソースグループにフェイルオーバーアプリケーションリソースを追加します。**

```
# 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*

このリソースのリソースタイプの名前を指定します。

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

このリソースに対して設定する拡張プロパティーのコンマ区切りリストを指定します。設定できる拡張プロパティーは、リソースタイプによって異なります。どの拡張プロパティーを設定するかを確認するには、リソースタイプに関するドキュメントを参照してください。

node-specifier は、-p 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、リソースが作成されたときに (1 つまたは複数の) 拡張プロパティーを指定した (1 つまたは複数の) ノードでのみ設定することを示します。クラスタ内のその他のノード上の指定された拡張プロパティーは設定されません。*node-specifier* を含めない場合は、クラスタ内のすべてのノード上の指定された拡張プロパティーが設定されます。*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ーをノード `phys-schost-1` でのみ設定することを示します。ほとんどのシェルでは、二重引用符 (“) が必要です。

-p*standard-property=value*,…

このリソースに対して設定する標準プロパティーのコンマ区切りリストを指定します。設定できる標準プロパティーは、リソースタイプによって異なります。どの標準プロパティーを設定するかを確認するには、`rt_properties(7)`、`cluster(8CL)`、`rg_properties(7)`、`r_properties(7)`、`property_attribu`の各マニュアルページを参照してください。

resource

追加するリソースの選択した名前を指定します。

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および検証されたことを確認します。

```
# clresource show resource
```

例 12 リソースグループへのフェイルオーバーアプリケーションリソースの追加

この例では、リソースグループ (resource-group-1) へのリソース (resource-1) の追加を示します。このリソースは、以前に定義した同じフェイルオーバーリソースグループ内に存在する必要のある論理ホスト名リソース (schost-1、schost-2) に依存します。

```
# clresource create -g resource-group-1 -t resource-type-1 \
-p Resource_dependencies=schost-1,schost2resource-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{phys-schost-1}:                         False
Enabled{phys-schost-2}:                         False
Monitored{phys-schost-1}:                       True
Monitored{phys-schost-2}:                       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ら、76 ページの「リソースグループをオンラインにする方法」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、そのリソースが Oracle Solaris Cluster ソフトウェアによって検証されます。検証が失敗した場合、clresource コマンドは、エラーメッセージを出力して終了します。検証が失敗した理由を確認するには、各ノード上の syslog にエラーメッセージがないかどうかチェックしてください。このメッセージは、必ずしも clresource コマンドを実行したノードではなく、検証を実行したノード上で表示されます。

参照 clresource(8CL) のマニュアルページ。

▼ リソースグループにスケーラブルアプリケーションリソースを追加する方法

スケーラブルアプリケーションリソースは、Oracle Solaris 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アプリケーションリソースです。スケーラブルアプリケーションリソースは複数マスターリソースグループ内に存在し、1 つまたは複数の共有アドレスリソースに対する依存関係を持っています。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存在します。

注記 - この手順は、いずれか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いることを確認してください。

- リソースの追加先となるスケーラブルリソースグループの名前
- このリソースのリソースタイプの名前
- この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 (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)

注記 - この手順は、プロキシリソースにも適用されます。

---

1. **クラスタメンバーで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **リソースグループにスケーラブルアプリケーションリソースを追加します。**

```
# clresource create -S -g resource-group -t resource-type \
-p Resource_dependencies=network-resource[,network-resource...] \
-p Scalable=True
[-p "extension-property[{node-specifier}]"=value,…] [-p standard-property=value,…] resource
```

-S

このリソースグループを複数マスターにすることを指定します。-p `Maximum_primaries` および -p `Desired_primaries` オプションが省略された場合は、両方のプロパティーがリソースグループのノードリスト内のノードの数に設定されます。

-g *resource-group*

以前に作成したスケーラブルサービスリソースグループの名前を指定します。

-t *resource-type*

このリソースのリソースタイプの名前を指定します。

-p `Resource_dependencies=`*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 Oracle Solaris 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ことを指定します。

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

このリソースに対して設定する拡張プロパティーのコンマ区切りリストを指定します。設定できる拡張プロパティーは、リソースタイプによって異なります。どの拡張プロパティーを設定するかを確認するには、リソースタイプに関するドキュメントを参照してください。

node-specifier は、-p 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、リソースが作成されたときに (1 つまたは複数の) 拡張プロパティーを指定した (1 つまたは複数の) ノードでのみ設定することを示します。クラスタ内のその他のノード上の指定された拡張プロパティーは設定されません。*node-specifier* を含めない場合は、クラスタ内のすべてのノード上の指定された拡張プロパティーが設定されます。*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ーをノード `phys-schost-1` でのみ設定することを示します。ほとんどのシェルでは、二重引用符 (“) が必要です。

-p *standard-property=value,…*

このリソースに対して設定する標準プロパティーのコンマ区切りリストを指定します。設定できる標準プロパティーは、リソースタイプによって異なります。スケーラブルサービスの場合、通常は `Port_list`、`Load_balancing_weights`、および `Load_balancing_policy` プロパティーを設定します。どの標準プロパティーを設定するかを確認するには、`cluster(8CL)`、`rt_properties(7)`、`rg_properties(7)`、`r_properties(7)`、`property_attri` の各マニュアルページを参照してください。

resource

追加するリソースの選択した名前を指定します。

リソースは有効状態で作成されます。

3. **スケーラブルアプリケーションリソースが追加および検証されたことを確認します。**

```
# clresource show resource
```

例 13 リソースグループへのスケーラブルアプリケーションリソースの追加

この例では、リソースグループ (`resource-group-1`) へのリソース (`resource-1`) の追加を示します。`resource-group-1` は、使用中のネットワークアドレス (次の例の `schost-1` および `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。このリソースは、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内に存在する必要のある共有アドレスリソース (`schost-1`、`schost-2`) に依存します。

```
# clresource create -S -g resource-group-1 -t resource-type-1 \
-p Resource_dependencies=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{phys-schost-1}:                         False
Enabled{phys-schost-2}:                         False
Monitored{phys-schost-1}:                       True
Monitored{phys-schost-2}:                       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ら、76 ページの「リソースグループをオンラインにする方法」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、そのリソースが Oracle Solaris Cluster ソフトウェアによって検証されます。検証が失敗した場合、`clresource` コマンドは、エラーメッセージを出力して終了します。検証が失敗した理由を確認するには、各ノード上の `syslog` にエラーメッセージがないかどうかチェックしてください。このメッセージは、必ずしも `clresource` コマンドを実行したノードではなく、検証を実行したノード上で表示されます。

参照 `clresource(8CL)` のマニュアルページ。

リソースグループのオンライン化

リソースが HA サービスの提供を開始できるようにするには、次の操作を実行する必要があります。

- リソースグループをオンラインにする
- リソースをそのリソースグループ内で有効にする
- リソースモニターを有効にする
- リソースグループを管理状態にする

これらのタスクは個別に実行することも、1 つのコマンドを使用して実行することもできます。

リソースグループをオンラインにすると、そのリソースグループは構成され、使用できるようになります。リソースまたはノードに障害が発生した場合、RGM はリソースグループを代替ノード上でオンラインに切り替えることによって、そのリソースグループの可用性を維持します。

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用して、現在のプライマリノードのリストでリソースグループをオンラインにすることもできます。また、そのリソースをクラスタ全体で管理状態にすることもできます。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。

▼ リソースグループをオンラインにする方法

このタスクは、いずれか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、RBAC の承認 `solaris.cluster.admin` を提供する `root` 役割になります。**

2. **リソースグループをオンラインにするためのコマンドを入力します。**

   - **無効のままにする必要のあるリソースや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のコンマ区切りリストを指定します。これらのリソースグループは存在している必要があります。このリストには、1 つのリソースグループ名または複数のリソースグループ名を含めることができます。

   - **リソースグループがオンラインになったときにリソースとその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 -em rg-list
     ```

     rg-list

     オンラインにするリソースグループの名前のコンマ区切りリストを指定します。これらのリソースグループは存在している必要があります。このリストには、1 つのリソースグループ名または複数のリソースグループ名を含めることができます。

注記 - オンラインにするいずれかのリソースグループで、ほかのリソースグループに対する強いアフィニティーが宣言されている場合、この操作は失敗することがあります。詳細は、192 ページの「オンラインリソースグループのクラスタノード間での分散」を参照してください。

---

3. **ステップ 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。**

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

   ```
   # clresourcegroup status
   ```

例 14 リソースグループのオンライン化

この例では、リソースグループ `resource-group-1` をオンラインにして、そのステータスを検証する方法を示します。また、このリソースグループ内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 -eM resource-group-1
# clresourcegroup status
```

次の手順 リソースグループをオンラインにしたときに、そのリソースと障害モニターをまだ有効にしていない場合は、有効にする必要のあるすべてのリソースの障害モニターを有効にします。詳細は、84 ページの「リソースの障害モニターを有効にする方法」を参照してください。

参照 `clresourcegroup(8CL)` のマニュアルページ。

リソースグループの優先プライマリへの切り替え

`clresourcegroup remaster` コマンドは、リソースグループを優先ノード上でオンラインに切り替えるために、そのリソースグループを現在のプライマリからオフラインに切り替えることができます。`clresourcegroup online` コマンドと同様に、RGM は、そのリソースグループの `Nodelist` プロパティーに基づいて一連の最優先ノードを計算しますが、次の要素も考慮に入れます。

- `RG_affinities` プロパティーの設定
- `Load_factors` プロパティーの設定 (ノードの負荷制限や現在の負荷と比較します)
- 各リソースグループの障害履歴

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用して、リソースグループのマスターを再作成することもできます。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。

▼ リソースグループを優先プライマリに切り替える方法

このタスクは、いずれか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、RBAC の承認 `solaris.cluster.admin` を提供する `root` 役割になります。**
2. **リソースグループをその優先プライマリに切り替えるためのコマンドを入力します。**

- **無効のままにする必要のあるリソースや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 remaster rg-list
  ```

  rg-list

  優先プライマリに切り替えるリソースグループの名前のコンマ区切りリストを指定します。これらのリソースグループは存在している必要があります。このリストには、1 つのリソースグループ名または複数のリソースグループ名を含めることができます。

- **リソースグループが優先プライマリに切り替えられたときにリソースとその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 remaster -emM rg-list
  ```

3. **ステップ 2 で指定した各リソースグループが優先プライマリに切り替えられことを確認します。**

   このコマンドの出力は、切り替えられたリソースグループの新しいノードを示します。

   ```
   # clresourcegroup status
   ```

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。

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用して、リソースを有効にすることもできます。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。

▼ リソースを有効にする方法

この手順は、いずれかのクラスタノードから実行します。

始める前に 有効にする予定のリソースを作成しており、その名前を用意していることを確認してください。

1. **クラスタメンバーで、RBAC の承認 `solaris.cluster.admin` を提供する `root` 役割になります。**

2. **リソースを有効にします。**

```
# clresource enable [-n nodelist] resource
```

-n *nodelist*

リソースを有効にするノードの、コンマで区切られた順序付きリストを指定します。

このリストはオプションです。このリストを省略した場合は、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。

注記 - -n オプションで複数のノードを指定した場合、指定できるリソースは 1 つだけです。

---

resource

有効にするリソースの名前を指定します。

3. **リソースが有効になったことを確認します。**

```
# clresource status
```

このコマンドの出力は、有効にしたリソースの状態を示します。

参照 `clresource(8CL)` のマニュアルページ。

リソースグループの静止

START または STOP メソッドが失敗した場合に、リソースグループがあるノードから別のノードに連続して切り替えるのを阻止するには、そのリソースグループを静止状態にします。リソースグループを静止状態にするには、`clresourcegroup quiesce` コマンドを発行します。

リソースグループを静止した場合、実行中のリソースメソッドは、完了するまで実行することを許可されます。重大な問題が発生した場合は、リソースグループをただちに静止しなければならないことがあります。それには、-k コマンドオプションを指定します。これにより、次のメソッドが強制終了されます。

- `Prenet_start`
- `Start`
- `Monitor_start`
- `Monitor_stop`

- Stop
- Postnet_stop

注記 - このコマンドオプションを指定しても、Init、Fini、Boot、および Update メソッドは強制終了されません。

ただし、メソッドを強制終了してリソースグループをただちに静止した場合は、そのリソースのいずれかが Start_failed や Stop_failed などのエラー状態のままになることがあります。これらのエラー状態は、ユーザー自身でクリアする必要があります。

▼ リソースグループを静止する方法

1. **RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**
2. **リソースグループを静止します。**

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

▼ リソースグループをただちに静止する方法

1. **RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**
2. **リソースグループをただちに静止します。**

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

そのリソースグループに関連付けられている Prenet_start、Start、Monitor_start、Monitor_stop、Stop、および Postnet_stop メソッドがただちに強制終了されます。そのリソースグループが静止状態になります。

clresourcegroup quiesce -k コマンドは、指定されたリソースグループが静止状態に達するまでブロックされます。

リソースグループの自動復旧アクションの中断および再開

リソースグループの自動復旧アクションを一時的に中断できます。リソースグループの自動復旧は、クラスタ内にある問題を調査して修正するために、中断する必要がある場合があります。または、リソースグループサービス上で保守を行う必要がある場合もあります。

リソースグループの自動復旧アクションを中断するには、clresourcegroup suspend コマンドを発行します。自動復旧アクションを再開するには、clresourcegroup resume コマンドを発行します。

リソースグループの自動復旧アクションを中断する場合は、そのリソースグループの静止状態への移行も行います。

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。オンラインかオフラインかにかかわらず、中断されたデータサービスは現在の状態のままです。指定したノード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。

次のいずれかの条件を満たすリソースグループの自動復旧アクションを中断すると、依存関係またはアフィニティーが中断され、適用されなくなります。

- 別のリソースに対する再起動依存関係を持つリソースが含まれている。
- 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が宣言されている。

これらのいずれかのカテゴリのリソースグループを中断すると、Oracle Solaris Cluster により、依存関係またはアフィニティーも中断されることを示す警告が表示されます。

注記 - 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復旧アクションを中断または再開する機能には影響を与えません。ただし、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中断した場合は、警告メッセージが生成されます。RG_system プロパティーは、リソースグループにクリティカルなシステムサービスが含まれていることを指定します。RG_system プロパティーが TRUE に設定されている場合は、ユーザーがリソースグループやそのリソースを不注意で停止、削除、または変更することが回避されます。

メソッドを強制終了して自動復旧をただちに中断する

リソースグループの自動復旧アクションを中断した場合、実行中のリソースメソッドは、完了するまで実行することを許可されます。重大な問題が発生した場合は、リソースグループの自動復旧アクションをただちに中断しなければならないことがあります。それには、-k コマンドオプションを指定します。これにより、次のメソッドが強制終了されます。

- Prenet_start
- Start
- Monitor_start
- Monitor_stop

- `Stop`
- `Postnet_stop`

注記 - このコマンドオプションを含めても、`Init`、`Fini`、`Boot`、および `Update` メソッドは強制終了されません。

ただし、メソッドを強制終了して自動復旧アクションをただちに中断した場合は、そのリソースのいずれかが `Start_failed` や `Stop_failed` などのエラー状態のままになることがあります。これらのエラー状態は、ユーザー自身でクリアする必要があります。

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用して、リソースのモニタリングを停止または開始したり、リソースグループを中断したりすることもできます。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。

▼ リソースグループの自動復旧アクションを中断する方法

1. **RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**
2. **リソースグループの自動復旧アクションを中断します。**

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

自動復旧アクションを再開するまで、指定したリソースグループが自動的に起動、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。83 ページの「リソースグループの自動復旧アクションを再開する方法」を参照してください。

▼ リソースグループの自動復旧アクションをただちに中断する方法

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用して、リソースグループの即時中断を実行することもできます。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。

---

1. **RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **リソースグループの自動復旧アクションをただちに中断します。**

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

そのリソースグループに関連付けられている `Prenet_start`、`Start`、`Monitor_start`、`Monitor_stop`、`Stop`、および `Postnet_stop` メソッドがただちに強制終了されます。そのリソースグループの自動復旧アクションが中断されます。自動復旧アクションを再開するまで、このリソースグループが自動的に起動、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。83 ページの「リソースグループの自動復旧アクションを再開する方法」を参照してください。

`clresourcegroup suspend -k` コマンドは、指定されたリソースグループが静止状態に達するまでブロックされます。

▼ リソースグループの自動復旧アクションを再開する方法

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用して、中断されたリソースグループの自動復旧を開始することもできます。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。

---

1. **RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **リソースグループの自動復旧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および有効化

このセクションの手順では、リソース自体ではなく、リソースの障害モニターを無効または有効にする方法について説明します。リソースは、その障害モニターが無効になっている間も引き続き正常に動作できます。ただし、障害モニターが無効になっていて、データサービスの障害が発生した場合、自動障害復旧は開始されません。詳細は、`clresource(8CL)` のマニュアルページを参照してください。

これらの手順は、いずれかのクラスタノードから実行します。

▼ リソースの障害モニターを無効にする方法

1. **いずれかのクラスタメンバーで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **リソースの障害モニターを無効にします。**

```
# clresource unmonitor [-n nodelist] resource
```

-n *nodelist*

リソースのモニタリングを無効にするノードの、コンマで区切られた順序付きリストを指定します。

このリストはオプションです。このリストを省略した場合は、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モニタリングが無効になります。

注記 - -n オプションで複数のノードを指定した場合、指定できるリソースは 1 つだけです。

---

resource

1 つまたは複数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、モニターされるフィールド (`RS Monitored`) をチェックすることによって、リソースの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 **15** リソースの障害モニターの無効化

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

▼ リソースの障害モニターを有効にする方法

1. **いずれかのクラスタメンバーで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **リソースの障害モニターを有効にします。**

```
# clresource monitor [-n nodelist] resource
```

-n *nodelist*

リソースをモニターするノードの、コンマで区切られた順序付きリストを指定します。

このリストはオプションです。このリストを省略した場合は、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モニターされます。

注記 - -n オプションで複数のノードを指定した場合、指定できるリソースは 1 つだけです。

resource

1 つまたは複数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、モニターされるフィールド (`RS Monitored`) をチェックすることによって、リソースの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 16 リソースの障害モニターの有効化

```
# 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。ただし、リソースタイプを削除する場合は、次の手順に従います。

注記 - この手順は、いずれか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方法

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

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識別するには、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、RBAC の承認 `solaris.cluster.modify` を提供する `root` 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. **リソースタイプを登録解除します。**

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

   resource-type
   登録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たことを確認します。**

   ```
   # clresourcetype show
   ```

例 17 リソースタイプの削除

この例では、リソースタイプ (`resource-type-1`) のすべてのリソースを無効にして削除してから、そのリソースタイプを登録解除する方法を示します。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:

- `clresource`(8CL)
- `clresourcetype`(8CL)

リソースグループの削除

CLI でリソースグループを削除するには、まず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。

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用して、リソースグループとそのすべてのリソースを削除したり、リソースグループをオフラインにしたりすることもできます。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。

この手順は、いずれか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方法

始める前に 削除するリソースグループ内のすべてのリソースを識別するには、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、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **次のコマンドを実行して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. **削除するリソースグループ内のすべてのリソースを無効にします。**

   ```
   # 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
   削除され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
   削除され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たことを確認します。**

```
# clresourcegroup show
```

例 18 リソースグループの削除

この例では、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、そのリソースグループを削除する方法を示します。

```
# 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:

- clresource(8CL)
- clresourcegroup(8CL)

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。

注記 - この手順は、いずれかのクラスタノードから実行します。

▼ リソースを削除する方法

1. **クラスタメンバーで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 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たことを確認します。**

```
# clresource show
```

例 19 リソースの削除

この例では、リソース (resource-1) を無効にして削除する方法を示します。

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

参照 clresource(8CL) のマニュアルページ。

リソースグループの現在のプライマリの切り替え

リソースグループをその現在のプライマリから、新しいプライマリになる別のノードにスイッチオーバーするには、次の手順を使用します。

▼ リソースグループの現在のプライマリを切り替える方法

注記 - この手順は、いずれか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。

- 次の情報を用意している。
  - スイッチオーバーするリソースグループの名前。
  - リソースグループをオンラインにするか、またはオンラインのままにするノードの名前。
- リソースグループをオンラインにするか、またはオンラインのままにするノードがクラスタ内に存在する。
- これらのノードが、切り替えるリソースグループの潜在的なマスターとして設定されている。

リソースグループの潜在的なプライマリのリストを表示するには、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、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **リソースグループを新しい一連のプライマリに切り替えます。**

```
# clresourcegroup switch [-n nodelist] resource-group
```

-n *nodelist*

このリソースグループをマスターできるグローバルクラスタノードの、コンマで区切られた順序付きリストを指定します。このリソースグループは、ほかのすべてのノード上でオフラインに切り替えられます。

このリストはオプションです。このリストを省略した場合は、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。

注記 - 切り替えるいずれかのリソースグループで、ほかのリソースグループに対する強いアフィニティーが宣言されている場合は、切り替えの試みが失敗するか、または委託されることがあります。詳細は、192 ページの「オンラインリソースグループのクラスタノード間での分散」を参照してください。

---

3. **リソースグループが新しいプライマリに切り替えられたことを確認します。**

このコマンドの出力は、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ます。

```
# clresourcegroup status
```

例 20 リソースグループの新しいプライマリへの切り替え

この例では、リソースグループ `resource-group-1` をその現在のプライマリ `phys-schost-1` から潜在的なプライマリ `phys-schost-2` に切り替える方法を示します。

1. このリソースグループが `phys-schost-1` 上でオンラインであることを確認するために、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
```

2. 切り替えを実行するために、次のコマンドを実行します。

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

3. このグループが phys-schost-2 上でオンラインに切り替えられたことを確認するために、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(8CL) のページ。

リソースの無効化およびそのリソースグループの UNMANAGED 状態への移行

場合によっては、リソースグループに対する管理手順を実行する前に、そのリソースグループを UNMANAGED 状態に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、そのリソースグループに含まれているすべてのリソースを無効にしたあと、その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(8CL) のマニュアルページを参照してください。

注記 - この手順は、いずれかのクラスタノードから実行します。

▼ リソースを無効にし、そのリソースグループを UNMANAGED 状態に移行する方法

Oracle Solaris Cluster Manager ブラウザインタフェースを使用して、リソースを無効にし、リソースグループを非管理状態に移動することもできます。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。

注記 - 共有アドレスリソースが無効になっても、そのリソースは、一部のホストからの ping コマンドに引き続き応答できる可能性があります。無効になった共有アドレスリソースが ping コマンドに応答できなくなるようにするには、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。詳細は、ping(8) のマニュアルページを参照してください。

始める前に 次の情報を用意していることを確認してください。

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

この手順に必要なリソースおよびリソースグループ名を確認するには、次のように入力します。

```
# clresourcegroup show -v
```

1. **いずれかのクラスタメンバーで、RBAC の承認 `solaris.cluster.admin` を提供する `root` 役割になります。**

2. **リソースグループ内のすべてのリソースを無効にします。**

```
# clresource disable [-n nodelist] -g resource-group +
```

-n *nodelist*

リソースを無効にするノードの、コンマで区切られた順序付きリストを指定します。

このリストはオプションです。このリストを省略した場合は、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。

注記 - -n オプションで複数のノードを指定した場合、指定できるリソースは 1 つだけです。

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` 状態に移行します。**

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

resource-group

UNMANAGED 状態に移行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、そのリソースグループが `UNMANAGED` 状態にあることを確認します。**

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

例 21 リソースの無効化およびそのリソースグループの UNMANAGED 状態への移行

この例では、リソース (resource-1) を無効にしたあと、そのリソースグループ (resource-group-1) を UNMANAGED 状態に移行する方法を示します。

```
# 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 詳細は、次のマニュアルページを参照してください。

- clresource(8CL)
- clresourcegroup(8CL)

リソースタイプ、リソースグループ、およびリソース構成情報の表示

リソース、リソースグループ、またはリソースタイプに対する管理手順を実行する前に、これらのオブジェクトの現在の構成設定を表示します。

注記 - リソース、リソースグループ、およびリソースタイプの構成設定は、どのクラスタノードからでも表示できます。

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 のみに関する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(8CL)
- clresourcegroup(8CL)
- clresource(8CL)

リソースタイプ、リソースグループ、およびリソースプロパティーの変更

Oracle Solaris Cluster では、リソースタイプ、リソースグループ、およびリソースを構成するための標準プロパティーが定義されています。これらの標準プロパティーは、次のセクションで説明されています。

- 35 ページの「リソースタイププロパティー」
- 35 ページの「リソースプロパティー」
- 35 ページの「リソースグループプロパティー」

リソースにはまた、そのリソースを表すデータサービスに対して事前に定義された拡張プロパティーもあります。データサービスの拡張プロパティーについては、そのデータサービスに関するドキュメントを参照してください。

プロパティーを変更できるかどうかを確認するには、プロパティーの説明にあるプロパティーの調整可能なエントリに関するセクションを参照してください。

次の手順では、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にプロパティーを変更する方法について説明します。

▼ リソースタイププロパティーを変更する方法

注記 - この手順は、いずれか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いることを確認してください。

- 変更するリソースタイプの名前。
- 変更するリソースタイププロパティーの名前。リソースタイプでは、特定のプロパティーしか変更できません。プロパティーを変更できるかどうかを確認するに

は、rt_properties(7) のマニュアルページにあるプロパティーの調整可能なエントリに関するセクションを参照してください。

注記 - Installed_nodes プロパティーを明示的に変更することはできません。このプロパティーを変更するには、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、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **`clresourcetype` コマンドを実行して、この手順に必要なリソースタイプの名前を確認します。**

```
# clresourcetype show -v
```

3. **リソースタイププロパティーを変更します。**

リソースタイプでは、特定のプロパティーしか変更できません。プロパティーを変更できるかどうかを確認するには、rt_properties(7) のマニュアルページにあるプロパティーの調整可能なエントリに関するセクション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ーの名前と、そのプロパティーの新しい値を指定します。

Installed_nodes プロパティーを明示的に変更することはできません。このプロパティーを変更するには、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ーが変更されたことを確認します。**

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

例 22 リソースタイププロパティーの変更

この例では、SUNW.apache プロパティーを変更して、このリソースタイプがクラスタノード phys-schost-1 および phys-schost-2 にインストールされることを定義する方法を示します。

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

```
Resource Type:                                  SUNW.apache:4.2
RT_description:                                 Apache Web Server on Oracle Solaris
 Cluster
RT_version:                                     4.2
API_version:                                    2
RT_basedir:                                     /opt/SUNWscapc/bin
Single_instance:                                False
Proxy:                                          False
Init_nodes:                                     All potential masters
Installed_nodes:                                All
Failover:                                       False
Pkglist:                                        <NULL>
RT_system:                                      False
Global_zone:                                    False
```

▼ リソースグループプロパティーを変更する方法

この手順では、リソースグループプロパティーを変更する方法について説明します。リソースグループプロパティーについては、`rg_properties(7)` のマニュアルページを参照してください。

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用してリソースグループプロパティーを編集することもできます。「リソースグループ」をクリックし、リソースグループをクリックしてそのページに移動し、「プロパティー」タブをクリックします。「編集」をクリックして編集可能なプロパティーのリストにアクセスします。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。

この手順は、いずれか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いることを確認してください。

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

1. **クラスタメンバーで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **リソースグループプロパティーを変更します。**

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

-p *property* 変更するプロパティー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ーが変更されたことを確認します。**

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

例 23 リソースグループプロパティーの変更

この例では、リソースグループ (resource-group-1) の Failback プロパティーを変更する方法を示します。

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

▼ リソースプロパティーを変更する方法

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

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用してリソースプロパティーを編集することもできます。「リソースグループ」をクリックして、リソースが属するリソースグループをクリックします。リソースをクリックしてそのページに移動し、「プロパティー」タブをクリックし、「編集」をクリックして編集可能なプロパティーのリストにアクセスします。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。

- 標準リソースプロパティーについては、r_properties(7) のマニュアルページを参照してください。
- リソースの拡張プロパティーについては、そのリソースのリソースタイプに関するドキュメントを参照してください。

注記 - この手順は、いずれか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いることを確認してください。

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

1. **クラスタメンバーで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **現在のリソースプロパティーの設定を表示します。**

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

3. **リソースプロパティーを変更します。**

```
# 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ー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ーの名前を指定します。

node-specifier は、-p 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、リソースが作成されたときに (1 つまたは複数の) 拡張プロパティーを指定した (1 つまたは複数の) ノードでのみ設定することを示します。クラスタ内のその他のノード上の指定された拡張プロパティーは設定されません。*node-specifier* を含めない場合は、クラスタ内のすべてのノード上の指定された拡張プロパティーが設定されます。*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ーをノード `phys-schost-1` でのみ設定することを示します。ほとんどのシェルでは、二重引用符 (“) が必要です。

注記 - *node-specifier* で指定した拡張プロパティーは、RTR ファイル内でノード単位のプロパティーとして宣言されている必要があります。

---

resource

リソースの名前を指定します。

4. **リソースプロパティーが変更されたことを確認します。**

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

例 24 標準リソースプロパティーの変更

この例では、リソース (`resource-1`) の system-defined `Start_timeout` プロパティーを変更する方法を示します。

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

例 25 拡張リソースプロパティーの変更

この例では、リソース (`resource-1`) の拡張プロパティー (`Log_level`) を変更する方法を示します。

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

▼ リソース依存関係プロパティーを変更する方法

この手順では、リソース依存関係プロパティーを設定する方法について説明します。RGM は、あるリソースの別のリソースに対する依存関係をサポートしています。リソースの依存関係をノード単位で指定できますが、これはリソースのノードごとのインスタンスで異なる可能性があります。ノードごとのインスタンスとは、さまざまなノードで同時に (1 つの複数マスターリソースグループ内で) オンラインであるか、またはばらばらに (1 つのフェイルオーバーリソースグループ内で) オンラインであるリソースインスタンスです。リソースプロパティーについては、`r_properties(7)` のマニュアルページを参照してください。

`clsetup` ユーティリティーまたは CLI を使用して、リソース依存関係を設定できます。次の手順は、`clsetup` ユーティリティーでの手順を示しています。

1. **いずれかのクラスタノード上で root 役割になります。**

2. **`clsetup` ユーティリティーを起動します。**

```
# clsetup
```

`clsetup` のメインメニューが表示されます。

注記 - CLI を使用して、クラスタノードのサブセットに対するノード単位の依存関係を設定するには、ノード単位の各依存関係を *resourcename*@*nodename* という形式で指定します。

---

3. **リソースグループのオプション番号を入力します。**
「リソースグループ」メニューが表示されます。

4. **リソースのプロパティーを変更するためのオプション番号を入力します。**
「リソースのプロパティーの変更」画面に、このタスクの説明が表示されます。

5. **yes と入力します。**
このタスクのためのオプションのメニューが表示されます。

6. **標準リソースプロパティーを変更するためのオプション番号を入力します。**
このタスクのためのオプションのメニューが表示されます。

7. **プロパティーを変更するリソースに対応するオプション番号を入力します。**
リソースがこの状態にある間に変更できる標準プロパティーだけが表示されます。特定のプロパティーを変更するためにリソースを無効にすることが必要になる場合があります。標準リソースプロパティーの設定の詳細は、`r_properties(7)` のマニュアルページを参照してください。

resource_dependencies、resource_dependencies_weak、resource_dependencies_restart、または resource_dependencies_offline_restart プロパティーを変更することを選択できます。

8. **リソース状態が表示されているときに Return キーを押します。**

9. **変更する依存関係プロパティーのオプション番号を入力します。**
   現在のプロパティー名、タイプ、説明、および値が表示されます。

10. **依存関係リストの新しい値を入力します。**
    このリソースが依存する各リソースを、*resource-name*、*resource-name*{*qualifier*}、または *resource-name@node* の形式を使用して指定します。詳細は、画面上のテキストを参照してください。

    たとえば、resource_dependencies プロパティーの値を rs1 から rs1@mynode1,rs2@mynode2,rs3 に変更できます。

11. **yes と入力します。**

12. **「コマンドが正常に完了しました」というメッセージが表示されたあとに Return キーを押して、依存関係が設定されたことを確認します。**
    入力した新しい値が、そのプロパティーの「現在の設定」列に表示されます。

例 26 リソース依存関係プロパティーの変更

次の例は、clresource コマンドを使用して、2 つの異なる論理ホスト名リソースに依存するノード単位のリソース依存関係を設定する方法を示しています。この例では、gds-rs と呼ばれるスケーラブルリソースを使用し、ptrancos1 上の trancos-3-rs および ptrancos2 上の trancos-4-rs に対する gds-rs の依存関係を設定します。

```
ptrancos1# clresource set -p resource_dependencies=trancos-3-rs@ptrancos1, \
trancos-4-rs@ptrancos2 gds-rs
ptrancos1# clresource show -p resource_dependencies gds-rs

=== Resources ===

Resource:                                       gds-rs
Resource_dependencies: trancos-3-rs@ptrancos1 trancos-4-rs@ptrancos2

--- Standard and extension properties ---
```

例 27 リソースの依存関係のプロパティーの表示

次の例は、scha_resource_get command コマンドを使用して、2 つの異なる論理ホストリソースに依存するノード単位のリソースの依存関係を取得する方法を示しています。ノード単位のリソースの依存関係を設定するには、clresource set コマンドを使用する必要があります。この例では、gds-rs と呼ばれるスケーラブルリソースを使

用し、ptrancos1 上の trancos-3-rs および ptrancos2 上の trancos-4-rs に対する gds-rs の依存関係を設定します。

ptrancos1 ノードから:

```
ptrancos1(/root)$ scha_resource_get -O RESOURCE_DEPENDENCIES -R gds-rs
trancos-3-rs
ptrancos1(/root)$ scha_resource_get -O RESOURCE_DEPENDENCIES_NODE -R gds-rs ptrancos1
trancos-3-rs
ptrancos1(/root)$ scha_resource_get -O RESOURCE_DEPENDENCIES_NODE -R gds-rs ptrancos2
trancos-4-rs
ptrancos1(/root)$ scha_resource_get -Q -O RESOURCE_DEPENDENCIES -R gds-rs
trancos-3-rs@ptrancos1
trancos-4-rs@ptrancos2
ptrancos1(/root)$ scha_resource_get -O NETWORK_RESOURCES_USED -R gds-rs
trancos-3-rs
```

ptrancos2 ノードから:

```
ptrancos2(/root)$ scha_resource_get -O RESOURCE_DEPENDENCIES -R gds-rs
trancos-4-rs
ptrancos2(/root)$ scha_resource_get -O RESOURCE_DEPENDENCIES_NODE -R gds-rs ptrancos1
trancos-3-rs
ptrancos2(/root)$ scha_resource_get -O RESOURCE_DEPENDENCIES_NODE -R gds-rs ptrancos2
trancos-4-rs
ptrancos2(/root)$ scha_resource_get -Q -O RESOURCE_DEPENDENCIES -R gds-rs
trancos-3-rs@ptrancos1
trancos-4-rs@ptrancos2
ptrancos2(/root)$ scha_resource_get -O NETWORK_RESOURCES_USED -R gds-rs
trancos-4-rs
```

▼ 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方法

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。同じクラスタ上で実行され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できます。論理ホスト名リソースまたは共有アドレスリソースのフェイルオーバー中に、クラスタ上で実行されているネームサービスもフェイルオーバーしている可能性があります。論理ホスト名リソースまたは共有アドレスリソースがそのフェイルオーバーしているネームサービスを使用していると、そのリソースはフェイルオーバーに失敗します。

注記 - 同じクラスタ上で実行され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、そのクラスタ上のその他のサービスの可用性が低下する場合があります。

このようなフェイルオーバーの失敗を回避するには、論理ホスト名リソースまたは共有アドレスリソースを、ネームサービスを省略するように変更します。ネームサービスを省略するようにリソースを変更するには、そのリソースの CheckNameService 拡

張プロパティーを false に設定します。CheckNameService プロパティーはいつでも変更できます。

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用して、論理ホスト名または共有アドレスリソースを編集することもできます。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。

始める前に リソースタイプのバージョンが 2 未満である場合は、リソースを変更しようとする前に、リソースタイプをアップグレードする必要があります。詳細は、110 ページの「事前登録された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、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **リソースプロパティーを変更します。**

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

`-p CheckNameService=false`

リソースの CheckNameService 拡張プロパティーを 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。

リソースに関する STOP_FAILED エラーフラグのクリア

Failover_mode リソースプロパティーが NONE または SOFT に設定されている場合は、そのリソースの STOP メソッドの失敗によって次の影響が発生します。

- 個々のリソースが STOP_FAILED 状態に移行します。
- そのリソースを含むリソースグループが ERROR_STOP_FAILED 状態に移行します。

この状況では、次の操作を実行できません。

- すべてのノード上のリソースグループのオンライン化
- リソースグループへのリソースの追加
- リソースグループからのリソースの削除
- リソースグループのプロパティーの変更
- リソースグループ内のリソースのプロパティーの変更

▼ リソースに関する STOP_FAILED エラーフラグをクリアする方法

この手順は、いずれかのクラスタノードから実行します。

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用して、リソースの STOP_FAILED 状態をクリアすることもできます。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。

始める前に 次の情報を用意していることを確認してください。

- リソースが STOP_FAILED 状態にあるノードの名前
- STOP_FAILED 状態にあるリソースおよびリソースグループの名前

1. **クラスタメンバーで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **どのリソースが、どのノード上で STOP_FAILED 状態に移行したかを識別します。**

   ```
   # clresource status
   ```

3. **リソースが STOP_FAILED 状態にあるノード上で、それらのリソースとそのモニターを手動で停止します。**

   この手順では、プロセスを強制終了するか、またはリソースタイプに固有のコマンドやその他のコマンドを実行することが必要になる場合があります。

4. **リソースに関する STOP_FAILED エラーフラグをクリアします。**

   ```
   # 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 状態にあるノードの名前のコンマ区切りリストを指定します。このリストには、1 つのノード名または複数のノード名を含めることができます。

   resource
   : リソースの名前を指定します。

5. **ステップ 4 で STOP_FAILED フラグをクリアしたノード上で、リソースグループの状態をチェックします。**

```
# clresourcegroup status
```

リソースグループの状態は、OFFLINE または ONLINE になっているはずです。

次の状況の組み合わせでは、リソースグループは ERROR_STOP_FAILED 状態のままになります。

- STOP メソッドの失敗が発生したときに、リソースグループがオフラインに切り替えられていた。
- 停止に失敗したリソースが、リソースグループ内のほかのリソースに対する依存関係を持っていた。

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

   a. **適切なノード上で、リソースグループをオフラインに切り替えます。**

   ```
   # 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 ONLINE 状態に切り替えます。**

参照 次のマニュアルページ:

- clresource(8CL)
- clresourcegroup(8CL)

Start_failed リソース状態のクリア

Start_failed リソース状態は、Start または Prenet_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が結果的にオンラインになったことを示します。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 プロパティーに 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、ユーザーまたは Oracle Solaris Cluster ソフトウェアがそのリソースグループに対してアクションを実行することを妨げません。その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。

この状態をクリアするには、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って Start_failed リソース状態をクリアする方法

この手順は、いずれかのクラスタノードから実行します。

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用して、リソースグループを現在のプライマリノードから別のプライマリノードに切り替えることもできます。ログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。

始める前に 次の条件が満たされているか確認します。

- 次の情報を用意している。
  - スイッチオーバーするリソースグループの名前
  - リソースグループをスイッチオーバーするノードの名前
- リソースグループをオンラインにするか、またはオンラインのままにするノードがクラスタ内に存在する。

1. **クラスタメンバーで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **リソースグループを新しいノードに切り替えます。**

```
# clresourcegroup switch [-n nodelist] resource-group
```

-n *nodelist*

このリソースグループをマスターできるノードの、コンマで区切られた順序付きリストを指定します。このリソースグループは、ほかのすべてのノード上でオフラインに切り替えられます。

このリストはオプションです。このリストを省略した場合は、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。

注記 - 切り替えるいずれかのリソースグループで、ほかのリソースグループに対する強いアフィニティーが宣言されている場合は、切り替えの試みが失敗するか、または委託されることがあります。詳細は、192 ページの「オンラインリソースグループのクラスタノード間での分散」を参照してください。

---

3. **リソースグループが新しいノードに切り替えられ、Start_failed リソース状態がクリアされたことを確認します。**

```
# clresourcegroup status
```

このコマンドの出力は、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ます。

例 28 リソースグループのスイッチオーバーによる Start_failed リソース状態のクリア

この例では、resource-group-1 リソースグループ内の rscon リソース上で発生した Start_failed リソース状態をクリア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クラスタノード phys-schost-2 に切り替えることによって、この状態をクリアします。

1. リソースが phys-schost-1 上で Start_failed リソース状態にあることを確認するために、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. 切り替えを実行するために、次のコマンドを実行します。

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

3. この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 リソースステータスがクリアされたことを確認するために、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(8CL) のマニュアルページ。

▼ リソースグループの再起動によって Start_failed リソース状態をクリアする方法

この手順は、いずれかのクラスタノードから実行します。

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用して、リソースグループを再起動することもできます。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。

始める前に 次の条件が満たされているか確認します。

- 次の情報を用意している。
  - 再起動するリソースグループの名前
  - リソースグループを再起動するノードの名前
- リソースグループをオンラインにするか、またはオンラインのままにするノードがクラスタノードである。

1. **クラスタメンバーで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 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*

リソースグループを再起動するノードの名前を指定します。このリソースグループは、ほかのすべてのノード上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` リソース状態がクリアされたことを確認します。**

```
# clresourcegroup status
```

このコマンドの出力は、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ます。

例 29 リソースグループの再起動による Start_failed リソース状態のクリア

この例では、resource-group-1 リソースグループ内の rscon リソース上で発生した Start_failed リソース状態をクリアする方法を示します。このコマンドは、クラスタノード phys-schost-1 上のリソースグループを再起動することによって、この状態をクリアします。

1. リソースが phys-schost-1 上で Start_failed リソース状態にあることを確認します。

```
# 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. リソースを再起動します。

```
# 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 リソースステータスがクリアされたことを確認します。

```
# 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(8CL) 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って Start_failed リソース状態をクリアする方法

この手順は、いずれかのクラスタノードから実行します。

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用して、リソースを無効および有効にすることもできます。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。

始める前に 無効化および有効化を行うリソースの名前を用意していることを確認してください。

1. **クラスタメンバーで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 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` リソース状態がクリアされたことを確認します。**

```
# clresource status
```

このコマンドの出力は、無効になったあと、ふたたび有効になったリソースの状態を示します。

例 30 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` のクリア

この例では、リソースの無効化および有効化によって、`rscon` リソース上で発生した `Start_failed` リソース状態をクリアする方法を示します。

1. リソースが `Start_failed` リソース状態にあることを確認します。

```
# 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. リソースを無効にして、ふたたび有効にします。

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

3. リソースがふたたび有効になり、`Start_failed` リソースステータスがクリアされたことを確認します。

```
# 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(8CL)` のマニュアルページ。

事前登録されたリソースタイプのアップグレード

以前のバージョンのクラスタソフトウェアでは、次の事前登録されたリソースタイプが拡張されました。

- 論理ホスト名を表す `SUNW.LogicalHostname`
- 共有アドレスを表す `SUNW.SharedAddress`

これらの拡張の目的は、論理ホスト名リソースと共有アドレスリソースを、名前解決のためのネームサービスを省略するように変更できるようにすることでした。

次のリストにあるすべての条件に当てはまる場合は、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 Oracle Solaris Cluster からアップグレードしようとしている
- そのリソースタイプの新機能を使用する必要がある

リソースタイプをアップグレードする方法について説明する一般的な手順については、44 ページの「リソースタイプのアップグレード」を参照してください。事前

登録されたリソースタイプのアップグレードを完了するために必要な情報は、続くサブセクションに記載されています。

リソースタイプの新しいバージョンを登録するための情報

登録されているリソースタイプのバージョンを判定するには、次のリストから 1 つのコマンドを使用します。

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

例 31 `SUNW.LogicalHostname` リソースタイプの新しいバージョンの登録

この例では、アップグレード中に `SUNW.LogicalHostname` リソースタイプのバージョン 4 を登録するためのコマンドを示します。

```
# clresourcetype register SUNW.LogicalHostname:4
```

リソースタイプの既存のインスタンスを移行するための情報

事前登録されたリソースタイプのインスタンスを移行するために必要な情報は次のとおりです。

- 移行はいつでも実行できます。
- 事前登録されたリソースタイプの新機能を使用する必要がある場合、`Type_version` プロパティーの必要な値は `2` です。
- ネームサービスを省略するようにリソースを変更する場合は、そのリソースの `CheckNameService` 拡張プロパティーを `false` に設定します。

例 32 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、名前解決のためのネームサービスを省略するように変更されます。

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

不注意による削除のあとの事前登録されたリソースタイプの再登録

リソースタイプ SUNW.LogicalHostname および SUNW.SharedAddress は事前登録されています。すべての論理ホスト名および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。これらの 2 つのリソースタイプを登録する必要はありませんが、これらが不注意で削除されることがあります。リソースタイプを不注意で削除した場合は、次の手順を使用して再登録します。

注記 - 事前登録されたリソースタイプをアップグレードする場合は、110 ページの「事前登録されたリソースタイプのアップグレード」の手順に従って新しいリソースタイプバージョンを登録します。

この手順は、いずれかのクラスタノードから実行します。

▼ 不注意による削除のあとに事前登録されたリソースタイプを再登録する方法

- **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。このリソースタイプは、SUNW.LogicalHostname または SUNW.SharedAddress のどちらかです。

例 33 不注意による削除のあとの事前登録されたリソースタイプの再登録

この例では、SUNW.LogicalHostname リソースタイプを再登録する方法を示します。

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

参照 clresourcetype(8CL) 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またはリソースグループからのノードの削除

このセクションの手順を使用すると、次のタスクを実行できます。

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

これらの手順は、フェイルオーバーまたはスケーラブルリソースグループにノードを追加するか、またはこれらのリソースグループからノードを削除するかに応じて、若干異なります。

フェイルオーバーリソースグループには、フェイルオーバーサービスとスケーラブルサービスの両方で使用されるネットワークリソースが含まれています。クラスタに接続されている各 IP サブネットワークには、フェイルオーバーリソースグループで指定され、そこに含まれている独自のネットワークリソースがあります。このネットワークリソースは、論理ホスト名リソースまたは共有アドレスリソースのどちらかです。各ネットワークリソースには、そこで使用される PNM オブジェクトのリストが含まれています。フェイルオーバーリソースグループの場合は、そのリソースグループに含まれているネットワークリソースごとの PNM オブジェクトの完全なリスト (`netiflist` リソースプロパティー) を更新する必要があります。

スケーラブルリソースグループに対する手順は次のとおりです。

1. そのスケーラブルリソースが使用す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に対する手順を繰り返します。
2. そのスケーラブルグループを、新しい一連のホスト上でマスターされるように変更します。

詳細は、`clresourcegroup(8CL)` のマニュアルページを参照してください。

注記 - どちらの手順も、いずれかのクラスタノードから実行します。

リソースグループへのノードの追加

リソースグループにノードを追加するための次の手順は、そのリソースグループがスケーラブルリソースグループまたはフェイルオーバーリソースグループのどちらであるかによって異なります。詳細な手順については、次のセクションを参照してください。

- 114 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方法」
- 114 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方法」

この手順を完了するには、次の情報を指定する必要があります。

- すべてのクラスタノードの名前とノード ID
- ノードの追加先となるリソースグループの名前

- すべてのノード上のリソースグループによって使用されるネットワークリソースをホストする PNM オブジェクトの名前

また、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方法

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を新しいノード上で実行されるようにします。**

   詳細は、次の手順のステップ 1 からステップ 5 を参照してください。

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

   この手順によって `nodelist` の以前の値が上書きされるため、そ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ここに含める必要があります。

   ```
   # clresourcegroup set [-n nodelist] resource-group
   ```

   -n *nodelist*

   このリソースグループをマスターできるノードの、コンマで区切られた順序付きリストを指定します。このリソースグループは、ほかのすべてのノード上でオフラインに切り替えられます。

   このリストはオプションです。このリストを省略した場合は、クラスタ内のすべてのノードに `Nodelist` プロパティーが設定されます。

   resource-group

   このノードの追加先となるリソースグループの名前を指定します。

3. **(オプション) スケーラブルリソースの `Load_balancing_weights` プロパティーを更新して、リソースグループに追加するノードにウェイトを割り当てます。**

   そうしない場合、このウェイトはデフォルトで `1` です。詳細は、`clresourcegroup(8CL)` 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方法

1. **現在のノードリストと、リソースグループ内のリソースごとに構成されている PNM オブジェクト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
```

注記 - nodelist および netiflist のコマンド行の出力では、ノードがノード名で識別されます。ノード ID を識別するには、clnode show -v | grep -i *node-id* コマンドを実行してください。

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

この手順によって netiflist の以前の値が上書きされるため、すべての PNM オブジェクトをここに含める必要があります。

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

-p netiflist=*netiflist*

各ノード上に存在する PNM オブジェクトを識別するコンマ区切りリストを指定します。*netiflist* 内の各要素は、netif@node の形式である必要があります。netif は、sc_ipmp0 などの PNM オブジェクト名として指定できます。ノードは、sc_ipmp0@1 や sc_ipmp@phys-schost-1 などのノード名またはノード ID で識別できます。

network-resource

netiflist のエントリ上でホストされるネットワークリソース (論理ホスト名または共有アドレス) の名前を指定します。

3. HAStoragePlus AffinityOn 拡張プロパティーが True に等しい場合は、適切な Solaris Volume Manager ディスクセットにノードを追加します。

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

-s *disk-set*

metaset コマンドの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されるノードを指定します。

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

この手順によって nodelist の以前の値が上書きされるため、そ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ここに含める必要があります。

```
# clresourcegroup set [-n nodelist] resource-group
```

-n *nodelist*

このリソースグループをマスターできるノードの、コンマで区切られた順序付きリストを指定します。このリソースグループは、ほかのすべてのノード上でオフラインに切り替えられます。

このリストはオプションです。このリストを省略した場合は、クラスタ内のすべてのノードに `Nodelist` プロパティーが設定されます。

resource-group

このノードの追加先とな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 -v resource-group | grep -i nodelist
# clresourcegroup show -v resource-group | grep -i netiflist
```

例 34 リソースグループへのノードの追加

この例では、論理ホスト名リソース `schost-2` を含むリソースグループ `resource-group-1` にクラスタノード `phys-schost-2` を追加する方法を示します。

```
# 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 IPMP
interfaces on each node
Res property type: stringarray
Res property value: sc_ipmp0@1 sc_ipmp0@3
```

ノード 1 と 3 だけが IPMP グループを割り当てられています。ノード 2 の IPMP グループを追加する必要があります。

```
# 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 -nphys-schost-1,phys-schost-2,phys-schost-3 resource-group-1
# clresourcegroup show -v resource-group-1 | grep -i nodelist
Nodelist: phys-schost-1phys-schost-2phys-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のノードの削除

リソースグループからノードを削除するための手順はすべて、そのリソースグループがスケーラブルリソースグループまたはフェイルオーバーリソースグループのどちらであるかによって異なります。詳細な手順については、次のセクションを参照してください。

- 118 ページの「スケーラブルリソースグループからノードを削除する方法」
- 119 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方法」
- 120 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方法」

注記 - 削除するノードがノード単位のリソース依存関係に表示される場合は、リソースグループから削除する前に、まずノード単位の依存関係からそのノードを削除する必要があります。詳細は、99 ページの「リソース依存関係プロパティーを変更する方法」を参照してください。

この手順を完了するには、次の情報を指定する必要があります。

- すべてのクラスタノードのノード名とノード ID

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

- ノードの削除元となる (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 PNM オブジェクトの名前

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

さらに、削除するノード上でこのリソースグループが**マスターされていない**ことを必ず確認してください。削除するノード上でこのリソースグループが**マスターされている**場合は、`clresourcegroup` コマンドを実行して、このリソースグループをそのノードからオフラインに切り替えてください。次の `clresourcegroup` コマンドは、そのノードが *new-masters* に含まれていない場合、このリソースグループを指定されたノードからオフラインにします。

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

-n *new-masters*

リソースグループをマスターするようにするノードを指定します。

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、削除するノード上でマスターされています。

詳細は、`clresourcegroup`(8CL) のマニュアルページを参照してください。

注意 - すべてのリソースグループからノードを削除するときに、スケーラブルサービスの構成を使用している場合は、まずスケーラブルリソースグループからノードを削除します。次に、フェイルオーバーグループからノードを削除します。

▼ スケーラブルリソースグループからノードを削除する方法

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

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

さらに、スケーラブルリソースグループの `RG_dependencies` プロパティーは、フェイルオーバーリソースグループに対する依存関係を持つスケーラブルグループを構成するように設定されます。このプロパティーについては、`rg_properties(7)` のマニュアルページ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、『Concepts for Oracle Solaris Cluster 4.4』を参照してください。

スケーラブルリソースグループからノードを削除すると、そのノード上ではスケーラブルサービスがオンラインにならなくなります。スケーラブルリソースグループからノードを削除するには、次の手順を実行します。

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

   # **`clresourcegroup set [-n`** *nodelist*`]` *scalable-resource-group*

   -n *nodelist*

   このリソースグループをマスターできるノードの、コンマで区切られた順序付きリストを指定します。このリソースグループは、ほかのすべてのノード上でオフラインに切り替えられます。

   このリストはオプションです。このリストを省略した場合は、クラスタ内のすべてのノードに `Nodelist` プロパティーが設定されます。

   scalable-resource-group

   ノードの削除元となるリソースグループの名前を指定します。

2. **(オプション) 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します。**

   詳細は、120 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方法」を参照してください。

3. **(オプション) スケーラブルリソースの `Load_balancing_weights` プロパティーを更新して、リソースグループから削除するノードのウェイトを削除します。**

参照 clresourcegroup(8CL) 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方法

フェイルオーバーリソースグループからノードを削除するには、次の手順を実行します。

注意 - すべてのリソースグループからノードを削除するときに、スケーラブルサービスの構成を使用している場合は、まずスケーラブルリソースグループからノードを削除します。次に、この手順を使用して、フェイルオーバーグループからノードを削除します。

そのフェイルオーバーリソースグループに、スケーラブルサービスが使用する共有アドレスリソースが含まれている場合は、120 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方法」を参照してください。

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

   この手順によってノードが削除され、ノードリストの以前の値が上書きされます。そ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ここに必ず含めてください。

   ```
   # clresourcegroup set [-n nodelist] failover-resource-group
   ```

   -n *nodelist*

   このリソースグループをマスターできるノードの、コンマで区切られた順序付きリストを指定します。このリソースグループは、ほかのすべてのノード上でオフラインに切り替えられます。

   このリストはオプションです。このリストを省略した場合は、クラスタ内のすべてのノードに `Nodelist` プロパティーが設定されます。

   failover-resource-group

   ノードの削除元とな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リソースごとに構成されている PNM オブジェクト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。**

   この手順によって `netiflist` の以前の値が上書きされます。すべての PNM オブジェクトをここに含めてください。

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

注記 - 前のコマンド行の出力では、ノードがノード名で識別されます。ノード ID を見つけるには、コマンド行 `clnode show -v | grep -i "Node ID"` を実行してください。

-p `netiflist`=*netiflist*

各ノード上に存在する PNM オブジェクトを識別するコンマ区切りリストを指定します。*netiflist* 内の各要素は、`netif@node` の形式である必要があります。`netif` は、`sc_ipmp0` などの PNM オブジェクト名として指定できます。ノードは、`sc_ipmp0@1` や `sc_ipmp@phys-schost-1` などのノード名またはノード ID で識別できます。

network-resource

`netiflist` のエントリ上でホストされるネットワークリソースの名前を指定します。

注記 - Oracle Solaris Cluster は、`netif` に対するアダプタ名の使用をサポートしていません。

4. **更新された情報を確認します。**

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

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

スケーラブルサービスが使用する共有アドレスリソースを含むフェイルオーバーリソースグループでは、ノードは次の場所に表示される可能性があります。

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

フェイルオーバーリソースグループのノードリストからノードを削除するには、119 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方法」の手順に従います。

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再作成する必要があります。

フェイルオーバーグループのノードリストからノードを削除した場合は、引き続きそのノード上で共有アドレスリソースを使用してスケーラブルサービスを提供できま

す。引き続き共有アドレスリソースを使用するには、そのノードを共有アドレスリソースの auxnodelist に追加する必要があります。auxnodelist にノードを追加するには、次の手順を実行します。

注記 - 次の手順はまた、共有アドレスリソースの auxnodelist からノードを**削除する**ためにも使用できます。auxnodelist からノードを削除するには、共有アドレスリソースを削除して再作成する必要があります。

始める前に すべての論理ホスト名の IP アドレスのサブネットとネットマスクのエントリが /etc/netmasks ファイルにあることを確認してください。必要に応じて、/etc/netmasks ファイルを編集して、不足しているエントリがある場合は追加します。

1. **スケーラブルサービスリソースをオフラインに切り替えます。**
2. **フェイルオーバーリソースグループから共有アドレスリソースを削除します。**
3. **共有アドレスリソースを作成します。**
   フェイルオーバーリソースグループから削除したノードのノード ID またはノード名を auxnodelist に追加します。

```
# 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

目的のノードが追加または削除された、変更された新しい auxnodelist。

shared-address

共有アドレスの名前。

例 35 リソースグループからのノードの削除

この例では、論理ホスト名リソース (schost-1) を含むリソースグループ (resource-group-1) からノード (phys-schost-3) を削除する方法を示します。

```
# clresourcegroup show -v resource-group-1 | grep -i nodelist
Nodelist: phys-schost-1phys-schost-2phys-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 IPMP
interfaces on each node
( Res property type: stringarray
Res property value: sc_ipmp0@1sc_ipmp0@2sc_ipmp0@3
```

(*sc_ipmp0@3 is the IPMP group to be removed.)*

```
# clresource set-pnetiflist=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
```

Oracle Solaris SMF サービスを Oracle Solaris Cluster とともに実行できるようにする

サービス管理機能 (SMF) では、ノードのブート中やサービスの障害時に、SMF サービスを自動的に起動および再起動できます。SMF により、単一ホスト上の SMF サービスに対してある程度の高可用性が促進されます。この機能は、クラスタアプリケーションの高可用性とスケーラビリティーを促進する Oracle Solaris Cluster Resource Group Manager (RGM) と類似しています。SMF サービスと RGM は、互いに補完的な機能です。

Oracle Solaris Cluster には、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で SMF サービスを Oracle Solaris Cluster とともに実行できるようにするために使用可能な 3 つの SMF プロキシリソースタイプが含まれています。これらのプロキシリソースタイプを次に示します。

- `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、互いに関連する一連の SMF サービスを 1 つのリソースである *SMF* プロキシリソースにカプセル化して Oracle Solaris Cluster によって管理されるようにすることができます。この機能では、SMF は、単一のノード上の SMF サービスの可用性を管理します。Oracle Solaris Cluster は、SMF サービスのクラスタ全体にわたる高可用性とスケーラビリティーを提供します。

SMF プロキシリソースタイプを使用すると、SMF で制御された独自のサービスを Oracle Solaris Cluster に統合することにより、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、これらのサービスがクラスタ全体にわたるサービスの可用性を備えるようにすることができます。SMF サービスを SMF プロキシリソースに統合すると、SMF サービスは、デフォルトのリスタータでは管理されなくなります。Oracle Solaris Cluster によって委託されたリスタータが 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あり、その使用にも制限はありません。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できます。SMF プロキシリソースは、ほかのリソースと同じ方法で作成したり、管理したりできます。SMF プロキシリソースがほかのリソースとは異

なる点が 1 つあります。どの SMF プロキシリソースタイプのリソースを作成する場合も、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る必要があります。この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のパスです。このファイル内の各行は 1 つの SMF サービス専用であり、`svc fmri、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 `restarter_svc_test_1:default` と `restarter_svc_test_2:default` の 2 つのサービスを管理する必要がある場合、このファイルには次の 2 行を含めるようにしてください。

```
<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>
```

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、グローバルクラスタ内に存在できます。同じプロキシリソースの下にあるサービスはすべて、同じゾーン内に存在する必要があります。

注意 - プロキシリソース内にカプセル化された SMF サービスを無効または有効にするために SMF `svcadm` を使用しないでください。プロキシリソース内にカプセル化された (SMF リポジトリ内の) SMF サービスのプロパティーを変更しないでください。

- 123 ページの「フェイルオーバープロキシリソース構成への SMF サービスのカプセル化」
- 126 ページの「マルチマスタープロキシリソース構成への SMF サービスのカプセル化」
- 128 ページの「スケーラブルプロキシリソース構成への SMF サービスのカプセル化」

▼ フェイルオーバープロキシリソース構成への SMF サービスのカプセル化

フェイルオーバー構成については、53 ページの「リソースグループの作成」を参照してください。

注記 - この手順は、いずれか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、RBAC の承認 `solaris.cluster.modify` を提供する `root` 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. **プロキシリソースタイプが登録されたことを確認します。**

```
# 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 *nodelist*

このリソースグループをマスターできるノードの、コンマで区切られた順序付きリストを指定します。

このリストはオプションです。このリストを省略した場合は、すべてのクラスタノード上でリソースグループが構成されます。

resource-group

追加するリソースグループの選択した名前を指定します。この名前は ASCII 文字で始まる必要があります。

5. **SMF リソースグループが作成されたことを確認します。**

```
# clresourcegroup status resource-group
```

6. **リソースグループに SMF フェイルオーバーアプリケーションリソースを追加します。**

```
# clresource create -g resource-group -t SUNW.Proxy_SMF_failover \
-p Port_list=portnumber/protocol \
-x Proxied_service_instances=/tmp/dns_svcs.txt
```

-g *resource-group*

以前に作成した SMF 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。

-p `Port_list`=*portnumber/protocol*

インスタンスがアクティビティーを待機するために使用するポート番号を指定します。プロトコルは、`tcp` または `udp` のどちらかです。

-p `Proxied_service_instances`

SMF サービスと、それに対応するプロキシされる SMF サービスのマニフェストのマッピングを指定する、作成済みのファイルのパスを指定します。上の例では、`/tmp/dns_svcs.txt` がテキストファイルのパスです。

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および検証されたことを確認します。**

```
# 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 resource-group
```

注記 - clresource status コマンドを使用して SMF プロキシリソースタイプの状態を表示すると、ステータスは「オンラインですがモニターされていません」として表示されます。これはエラーメッセージではありません。SMF プロキシリソースが有効になって実行されると、SMF プロキシリソースタイプのリソースに対してモニタリングサポートが提供されないため、このステータスメッセージが表示されます。

例 36 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:                 2.0
API_version:                7
RT_basedir:                 /opt/SUNWscsmf/bin
Single_instance:            False
Proxy:                      False
Init_nodes:                 All potential masters
Installed_nodes:            <All>
Failover:                   True
Pkglist:                    <NULL>
RT_system:                  False
Global_zone:                False
```

例 37 リソースグループへの 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追加

この例では、リソースグループ resource-group-1 へのプロキシリソースタイプ SUN.Proxy_SMF_failover 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:                                   2.0
Group:                                          resource-group-1
R_description:
Resource_project_name:                          default
Enabled{phys-schost-1}:                         True
Monitored{phys-schost-1}:                       True
```

▼ マルチマスタープロキシリソース構成への SMF サービスのカプセル化

1. **クラスタメンバーで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 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 -S [-p Maximum_primaries=m] [-p Desired_primaries=n] \
[-n node-zone-list] resource-group
```

-S

このリソースグループを複数マスターにすることを指定します。-p `Maximum_primaries` および -p `Desired_primaries` オプションが省略された場合は、両方のプロパティーがリソースグループのノードリスト内のノードの数に設定されます。

-p `Maximum_primaries=`*m*

このリソースグループのアクティブなプライマリの最大数を指定します。

-p `Desired_primaries=`*n*

このリソースグループが起動を試みるべきアクティブなプライマリの数を指定します。

-n *nodelist*

このリソースグループを使用可能にするノードの、コンマで区切られた順序付きリストを指定します。

このリストはオプションです。このリストを省略した場合は、すべてのクラスタノード上でリソースグループが構成されます。

resource-group

追加するスケーラブルリソースグループの選択した名前を指定します。この名前は ASCII 文字で始まる必要があります。

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. **リソースグループに SMF プロキシマルチマスターリソースを追加します。**

```
# clresource create -g resource-group -t SUNW.Proxy_SMF_multimaster \
```

```
-p Port_list=portnumber/protocol \
-x Proxied_service_instances=/tmp/dns_svcs.txt
```

-g *resource-group*

以前に作成した SMF マルチマスターリソースグループの名前を指定します。

-p `Port_list`=*portnumber/protocol*

インスタンスがアクティビティーを待機するために使用するポート番号を指定します。プロトコルは、`tcp` または `udp` のどちらかです。

-p `Proxied_service_instances`

SMF サービスと、それに対応するプロキシされる SMF サービスのマニフェストのマッピングを指定する、作成済みのファイルのパスを指定します。上の例では、`/tmp/dns_svcs.txt` がテキストファイルのパスです。

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および検証されたことを確認します。

```
# 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 resource-group
```

注記 - `clresource status` コマンドを使用して SMF プロキシリソースタイプの状態を表示すると、ステータスは「**オンラインですがモニターされていません**」として表示されます。これはエラーメッセージではありません。SMF プロキシリソースが有効になって実行されると、SMF プロキシリソースタイプのリソースに対してモニタリングサポートが提供されないため、このステータスメッセージが表示されます。

例 38 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:               2.0
API_version:              7
RT_basedir:               /opt/SUNWscsmf/bin
Single_instance:          False
Proxy:                    False
Init_nodes:               All potential masters
Installed_nodes:          <All>
Failover:                 True
Pkglist:                  <NULL<
RT_system:                False
Global_zone:              False
```

例 39 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およびリソースグループへの追加

この例では、マルチマスタープロキシリソースタイプ SUN.Proxy_SMF_multimaster の作成およびリソースグループ resource-group-1 への追加を示します。

```
# clresourcegroup create -S \
-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:                           2.0
Group:                                  resource-group-1
R_description:
Resource_project_name:                  default
Enabled{phys-schost-1}:                 True
Monitored{phys-schost-1}:               True
```

▼ スケーラブルプロキシリソース構成への SMF サービスのカプセル化

スケーラブル構成については、55 ページの「スケーラブルリソースグループを作成する方法」を参照してください。

注記 - この手順は、いずれか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、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **SMF プロキシスケー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. **このスケー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を作成するには、53 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方法」を参照してください。**

4. **フェイルオーバーリソースグループに共有アドレスリソースを追加します。**
   68 ページの「リソースグループに共有アドレスリソースを追加する方法 (CLI)」を参照してください。

5. **SMF プロキシスケーラブルリソースグループを作成します。**

```
# clresourcegroup create -S [-p Maximum_primaries=m] [-p Desired_primaries=n] \
[-n node-zone-list] resource-group
```

   -S

   このリソースグループを複数マスターにすることを指定します。-p `Maximum_primaries` および -p `Desired_primaries` オプションが省略された場合は、両方のプロパティーがリソースグループのノードリスト内のノードの数に設定されます。

   -p `Maximum_primaries`=*m*

   このリソースグループのアクティブなプライマリの最大数を指定します。

   -p `Desired_primaries`=*n*

   このリソースグループが起動を試みるべきアクティブなプライマリの数を指定します。

   -n *nodelist*

   このリソースグループを使用可能にするノードの、コンマで区切られた順序付きリストを指定します。

   このリストはオプションです。このリストを省略した場合は、クラスタ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。

   resource-group

   追加するスケーラブルリソースグループの選択した名前を指定します。この名前は ASCII 文字で始まる必要があります。

6. **スケーラブル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
```

7. **ステップ 5 で作成したスケーラブルリソースグループに SMF プロキシスケーラブルリソースを追加します。**

```
# clresource create-g resource-group -t SUNW.Proxy_SMF_scalable \
-p Resource_dependencies=network-resource[,network-resource...] \
-p Scalable=True \
-p Port_list=portnumber/protocol \
-x Proxied_service_instances=/tmp/dns_svcs.txt
```

-p Resource_dependencies=*network-resource*[,*network-resource*…]

このリソースが依存する、ステップ 3 で作成したスケーラブルネットワークリソースの名前を指定します。

-g *resource-group*

以前に作成した SMF プロキシスケーラブルリソースグループの名前を指定します。

-p Scalable=True

このリソースが Oracle Solaris 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ことを指定します。詳細は、72 ページの「リソースグループにスケーラブルアプリケーションリソースを追加する方法」を参照してください。

リソースは有効状態で作成されます。

8. **SMF プロキシスケーラブルアプリケーションリソースが追加および検証されたことを確認します。**

```
# clresource show resource
```

9. **SMF プロキシスケーラブルリソースグループをオンラインにします。**

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

注記 - `clresource status` コマンドを使用して SMF プロキシリソースタイプの状態を表示すると、ステータスは「オンラインですがモニターされていません」として表示されます。これはエラーメッセージではありません。SMF プロキシリソースが有効になって実行されると、SMF プロキシリソースタイプのリソースに対してモニタリングサポートが提供されないため、このステータスメッセージが表示されます。

例 40 SMF プロキシスケー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:             2.0
API_version:            7
RT_basedir:             /opt/SUNWscsmf/bin
Single_instance:        False
Proxy:                  False
Init_nodes:             All potential masters
Installed_nodes:        <All>
Failover:               True
Pkglist:                <NULL>
RT_system:              False
Global _zone:           False
```

例 41 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および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_scalalble の作成およびリソースグループ resource-group-1 への追加を示します。

```
# clresourcegroup create -S \
-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 \
-p resource_dependencies=net-res -p port_list=1080/tcp \
-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:                       2.0
Group:                              resource-group-1
R_description:
Resource_project_name:              default
Enabled{phys-schost-1}:             True
Monitored{phys-schost-1}:           True
```

resource_dependencies および使用するポート番号を選択できます。

Oracle Solaris Cluster データサービスの障害モニターの調整

Oracle Solaris Cluster 製品に付属する各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しない終了を検出する
- データサービスの健全性をチェックする

障害モニターは、データサービスが記述された対象のアプリケーションを表すリソースに含まれています。このリソースは、データサービスを登録および構成するときに作成します。詳細は、データサービスに関するドキュメントを参照してください。

このリソースの標準プロパティーと拡張プロパティーによって、障害モニターの動作が制御されます。これらのプロパティーのデフォルト値によって、障害モニターの事前設定された動作が決定されます。事前設定された動作は、ほとんどの Oracle Solaris Cluster のインストールに適しているはずです。したがって、障害モニターの調整は、この事前設定された動作を変更する必要がある場合のみにしてください。

障害モニターを調整するには、次のタスクを実行します。

- 132 ページの「障害モニターの検証間隔を設定する」
- 133 ページの「障害モニターの検証タイムアウトを設定する」
- 134 ページの「継続的な障害とみなす基準を定義する」
- 135 ページの「リソースのフェイルオーバー動作を指定する」

これらのタスクは、データサービスを登録および構成するときに実行します。詳細は、データサービスに関するドキュメントを参照してください。

注記 - 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を設定する

リソースが正しく動作しているかどうかを確認するために、障害モニターは、この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、リソースの可用性やシステムのパフォーマンスに次のような影響を与えます。

- 障害モニターの検証間隔は、障害を検出したり、障害に対応したりするために必要な時間の長さに影響を与えます。そのため、障害モニターの検証間隔を短くすると、障害を検出したり、障害に対応したりするために必要な時間も短くなります。この短縮によって、リソースの可用性が向上します。

- 障害モニターの各検証によって、プロセッササイクルやメモリーなどのシステムリソースが消費されます。そのため、障害モニターの検証間隔を短くすると、システムのパフォーマンスが低下します。

障害モニターの最適な検証間隔はまた、リソース内の障害に対応するために必要な時間によっても異なります。この時間は、リソースの複雑さが、そのリソースの再起動などの操作に必要な時間にどのような影響を与えるかによって異なります。

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` 標準プロパティーを必要な間隔 (秒単位) に設定します。

障害モニターの検証タイムアウトを設定する

障害モニターの検証タイムアウトは、その障害モニターが、検証に対するリソースからの応答を待機する時間の長さを指定します。障害モニターは、このタイムアウト内に応答を受信しなかった場合、そのリソースを障害があるとして処理します。リソースが障害モニターの検証に応答するために必要な時間は、そ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、そのデータサービスに関するドキュメントを参照してください。

リソースが応答するために必要な時間はまた、次に示すような、障害モニターやアプリケーションには関連のない要素によっても異なります。

- 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には、リソースの `Probe_timeout` 拡張プロパティーを必要なタイムアウト (秒単位) に設定します。

ほとんどのリソースタイプの障害モニターの検証では、検証の実行時間がタイムアウト制限に近づいたときに通知を送信するように `Timeout_threshold` プロパティーを構成することもできます。そのような通知は、偽のフェイルオーバーを引き起こす可能性がある、設定が低すぎる検証タイムアウトを特定するために役立ちます。`Timeout_threshold` プロパティーの詳細は、`r_properties(7)` のマニュアルページを参照してください。

継続的な障害とみなす基準を定義する

リソース内の一時的な障害によって発生する中断を最小限に抑えるために、障害モニターは、このような障害に対応してそのリソースを再起動します。継続的な障害に対しては、リソースの再起動より根本的なアクションが必要になります。

- フェイルオーバーリソースの場合、障害モニターは、そ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、障害モニターは、そのリソースをオフラインにします。

リソースの完全な障害の数が、`Retry_count` 標準プロパティーで指定された再試行回数を超えると、障害モニターは障害を継続的として処理します。継続的な障害とみなす基準の定義により、クラスタのパフォーマンス特性や可用性の要件に対応するための再試行回数と再試行間隔を設定できるようになります。

このセクションの内容は次のとおりです。

- 134 ページの「リソースの完全な障害と部分的な障害」
- 135 ページの「再試行回数と再試行間隔のその他のプロパティーに対する依存関係」
- 135 ページの「再試行回数と再試行間隔を設定するための標準プロパティー」

リソースの完全な障害と部分的な障害

障害モニターは、一部の障害をリソースの完全な障害として処理します。完全な障害によって、通常、サービスは完全に失われます。完全な障害の例を次に示します。

- データサービスサーバーのプロセスの予期しない終了
- 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

完全な障害を検出すると、障害モニターは、その再試行間隔内の完全な障害のカウントを 1 増やします。

障害モニターは、その他の障害をリソースの部分的な障害として処理します。部分的な障害は完全な障害に比べて重大性が低く、通常はサービスが低下しますが、サービスが完全に失われることはありません。部分的な障害の例として、障害モニターの検証がタイムアウトする前のデータサービスサーバーからの不完全な応答があります。

部分的な障害を検出すると、障害モニターは、その再試行間隔内の完全な障害のカウントを分数の量だけ増やします。部分的な障害は、その再試行間隔にわたって引き続き累積されます。

部分的な障害の次の特性は、データサービスによって異なります。

- 障害モニターが部分的な障害として処理する障害のタイプ
- 部分的な各障害によって完全な障害のカウントに追加される分数の量

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、そのデータサービスに関するドキュメントを参照してください。

再試行回数と再試行間隔のその他のプロパティーに対する依存関係

障害のあるリソースの 1 回の再起動に必要な最大時間は、次のプロパティーの値の合計値です。

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

再試行間隔内に再試行回数に達するための十分な時間が確保されるようにするには、次の式を使用して再試行間隔の値と再試行回数値を計算します。

retry_interval >= 2 x *retry_count* × (*thorough_probe_interval* + *probe_timeout*)

2 の係数は、リソースのフェイルオーバーやオフライン化をただちに引き起こすわけではない部分的な検証障害を考慮しています。

再試行回数と再試行間隔を設定するための標準プロパティー

再試行回数と再試行間隔を設定するには、リソースの次の標準プロパティーを設定します。

- 再試行回数を設定するには、`Retry_count` 標準プロパティーを完全な障害の許可される最大数に設定します。
- 再試行間隔を設定するには、`Retry_interval` 標準プロパティーを必要な間隔 (秒単位) に設定します。

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によって、次の障害への RGM の対応方法が決定されます。

- 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、そのリソースの `Failover_mode` 標準プロパティーを設定します。このプロパティーの指定可能な値については、`r_properties(7)` のマニュアルページにある `Failover_mode` 標準プロパティーの説明を参照してください。

♦ ♦ ♦ 第 3 章

3 高可用性ローカルファイルシステムリソースの管理

この章では、Oracle Solaris Cluster の保守コマンドを使用して、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方法について説明します。ほかのツールを使用して手順を完了できるかどうかを確認するには、32 ページの「データサービスリソースの管理のためのツール」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「Oracle Solaris Cluster データサービスの計画」および『Concepts for Oracle Solaris Cluster 4.4』を参照してください。

この章には次のセクションがあります。

- 132 ページの「Oracle Solaris Cluster データサービスの障害モニターの調整」

高可用性ローカルファイルシステムを管理するためのタスクの概要

次の表に、Oracle Solaris 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タスクの要約を示します。この表にはまた、タスクを実行するための詳細な手順への相互参照も示されています。

表 4 高可用性ローカルファイルシステムを管理するためのタスク

タスク	参照先
リソースグループの HAStoragePlus を、これらのリソースグループとデバイスグループの間で起動の同期をとるように設定する	142 ページの「新しいリソースの HAStoragePlus リソースタイプを設定する方法」 145 ページの「既存のリソースの HAStoragePlus リソースタイプを設定する方法」 147 ページの「UFS ファイルシステムを使用してクラスタファイルシステムの HAStoragePlus リソースを設定する方法」 149 ページの「グローバル ZFS ストレージプールの HAStoragePlus リソースを設定する方法 」 159 ページの「clsetup ユーティリティーを使用して HAStoragePlus リソースタイプを設定する方法」
ローカル ZFS ファイルシステムを高可用性にするように HAStoragePlus を設定する	164 ページの「ローカル ZFS ファイルシステムを高可用性にするように HAStoragePlus リソースタイプを設定する方法」
高可用性ローカルファイルシステムのリソースをオンラインで変更する	172 ページの「高可用性ローカルファイルシステムのリソースのオンラインでの変更」
グローバル ZFS ストレージプールを管理するリソースをオンラインで変更する	154 ページの「グローバル ZFS ストレージプールを管理する HAStoragePlus リソースのオンラインでの変更」
HAStoragePlus リソースのクラスタ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	184 ページの「HAStoragePlus リソースのクラスタファイルシステムのローカルファイルシステムへの変更」
HAStoragePlus リソースタイプをアップグレードする	44 ページの「リソースタイプのアップグレード」

注記 - この章の手順では、Oracle Solaris Cluster の保守コマンドを使用してこれらのタスクを完了する方法について説明します。また、その他のツールを使用してリソースを管理することもできます。これらのオプションの詳細は、32 ページの「データサービスリソースの管理のためのツール」を参照してください。

HAStoragePlus を使用したリソースグループとデバイスグループの間の起動の同期

クラスタがブートしたり、サービスが別のノードにフェイルオーバーしたりしたあと、グローバルデバイスやローカルおよびクラスタファイルシステムが使用可能になるまでに時間がかかることがあります。ただし、データサービスは、グローバルデバイスや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、その 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グローバル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していると、その START メソッドはタイムアウトします。この状況では、そのデータサービスが使用するリソースグループの状態をリセットし、データサービスを手動で再起動する必要があります。

これらの追加の管理タスクを回避するには、HAStoragePlus リソースタイプを使用します。データサービスリソースがグローバル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すべてのリソースグループに HAStoragePlus のインスタンスを追加します。これらのリソースタイプのインスタンスは、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、グローバルデバイスやローカルおよびクラスタファイルシステムが使用可能になるまで強制的に待たせるなどの操作を実行できます。

アプリケーションリソースが HAStoragePlus リソース上に構成されている場合、アプリケーションリソースはその下にある HAStoragePlus リソースとのオフライン再起動の依存関係を定義しなければなりません。これにより、そのアプリケーションリソースは依存する HAStoragePlus リソースがオンラインになったあとにオンラインになり、HAStoragePlus リソースがオフラインになる前にオフラインになることが保証されます。

次のコマンドは、アプリケーションリソースから HAStoragePlus リソースへのオフライン再起動依存関係を作成します。

```
# clrs set -p Resource_dependencies_offline_restart=hasp_rs  applicaton_rs
```

HAStoragePlus リソースを作成するには、142 ページの「新しいリソースの HAStoragePlus リソースタイプを設定する方法」を参照してください。

HAStoragePlus による管理対象エンティティーのモニタリング

HAStoragePlus リソースタイプによって管理されているエンティティーはすべてモニターされます。SUNW.HAStoragePlus リソースタイプは、グローバルデバイス、ファイルシステム、ZFS ストレージプールなどの HAStoragePlus リソースによって管理さ

れるエンティティーの健全性をモニターするための障害モニターを提供します。障害モニターは定期的に障害プローブを実行します。いずれかのエンティティーが使用できない状態になると、リソースが再起動されるか、別のノードへのフェイルオーバーが実行されます。複数のエンティティーがモニターされている場合、障害モニターはすべてのエンティティーを同時にプローブします。モニタリングを有効にする前に、管理対象エンティティーへのすべての構成変更が完了していることを確認してください。

注記 - HAStoragePlus リソースの障害モニターは、ファイルシステムの読み取りと書き込みを行なうことで、管理対象のデバイスおよびファイルシステムを検証します。読み取り操作が I/O スタックのソフトウェアによってブロックされており、`HAStoragePlus` リソースがオンラインである必要がある場合、ユーザーは障害モニターを無効にしなければなりません。

管理対象エンティティーのモニタリングを有効にするプロパティーの詳細は、`SUNW.HAStoragePlus(7)` のマニュアルページを参照してください。

管理対象エンティティーのモニタリングを有効および無効にする手順については、84 ページの「リソースの障害モニターを有効にする方法」を参照してください。

管理対象エンティティーのタイプに応じて、障害モニターは、読み取りまたは書き込みを行うことによりターゲットを検証します。複数のエンティティーがモニターされている場合、障害モニターはすべてのエンティティーを同時にプローブします。

表 5 障害モニターの検証内容

モニター対象エンティティー	障害モニターの検証内容
グローバルデバイス	■ デバイスグループはオンラインか、または縮退状態か。 ■ デバイスは読み取り可能か。
raw デバイスグループ	■ デバイスグループはオンラインか、または縮退状態か。 ■ デバイスグループの各デバイスについて、そのパス (`/dev/global/rdsk/`*device*) は使用可能か。 ■ すべてのデバイスのパーティションが読み取り可能か。
Solaris Volume Manager デバイスグループ	■ デバイスグループはオンラインか、または縮退状態か。 ■ メタセットのパス (`/dev/md/`*metaset*) は有効か。 ■ Solaris Volume Manager によって、デバイスグループのプライマリからのステータスが報告されたか。 　■ ミラー化されていないメタデバイスが「保守が必要」、「最後にエラー」、「使用不可」のいずれかのエラー状態にないか。 　■ ミラーの少なくとも 1 つのサブミラーがエラー状態にないか。すべてのサブミラーではなく、一部のサブミラーのエラーが部分的なエラーとして処理されているか。 ■ ミラー化されていないメタデバイスはプライマリから読み取り可能か。 ■ ミラーの一部のサブミラーが読み取り可能か。すべてのサブミラーではなく、一部のサブミラーのエラーが部分的なエラーとして処理されているか。

モニター対象エンティティー	障害モニターの検証内容
ファイルシステム (UFS、QFS、および PxFS を含む)	■ ファイルシステムはマウントされているか。 ■ ファイルシステム配下のすべてのデバイスが読み取り可能か。 ■ `IOOption` プロパティーが `ReadOnly` に設定されている場合、ファイルシステムは読み取り可能か。 ■ `IOOption` プロパティーが `ReadWrite` に設定されている場合、ファイルシステムは書き込み可能か。 ■ ファイルシステムが読み取り専用でマウントされているが、`IOOption` プロパティーが `ReadWrite` に設定されている場合、障害モニターは警告を発行したあと (書き込みではなく) 読み取りを試行します。 ■ ファイルシステムが割り当て制限に達したときに `HAStoragePlus` リソースがオフラインにならないようにするには、`IOOption` を `ReadOnly` に設定します。`ReadOnly` オプションにより、障害モニターはファイルシステムへの書き込みを試行しなくなります。
ZFS ストレージプール	■ プールステータスは「OK」か、または「縮退」か。 ■ レガシー以外の各ファイルシステムはマウントされているか。 ■ `IOOption` プロパティーが `ReadOnly` に設定されている場合、レガシー以外の各ファイルシステムは読み取り可能か。 ■ `IOOption` プロパティーが `ReadWrite` に設定されている場合、レガシー以外の各ファイルシステムは書き込み可能か。 ■ レガシー以外のファイルシステムが読み取り専用でマウントされているが、`IOOption` プロパティーが `ReadWrite` に設定されている場合、障害モニターは警告を発行したあと (書き込みではなく) 読み取りを試行します。 ■ ファイルシステムが割り当て制限に達したときに `HAStoragePlus` リソースがオフラインにならないようにするには、`IOOption` を `ReadOnly` に設定します。`ReadOnly` オプションにより、障害モニターはファイルシステムへの書き込みを試行しなくなります。 **注記 -** トップレベルの ZFS ストレージデバイスへのすべての接続が失われた場合、ZFS ストレージプールまたは関連するファイルシステムに関する問い合わせはハングアップします。障害モニターがハングアップしないようにするには、ZFS ストレージプールの `fail_mode` プロパティーを `panic` に設定する必要があります。

リソースの障害モニターを有効にする手順については、84 ページの「リソースの障害モニターを有効にする方法」を参照してください。

管理対象エンティティーのモニタリングのトラブルシューティング

管理対象エンティティーに対するモニタリングが有効になっていない場合は、次のトラブルシューティング手順を実行してください。

1. `hastorageplus_probe` プロセスが実行されていることを確認します。
2. コンソールにエラーメッセージが表示されていないか調べます。
3. `syslog` ファイルへのデバッグメッセージを有効にします。

```
# mkdir -p /var/cluster/rgm/rt/SUNW.HAStoragePlus:11

# echo 11 > /var/cluster/rgm/rt/SUNW.HAStoragePlus:11/loglevel
```

また、/etc/syslog.conf ファイルもチェックして、機能レベルが daemon.debug であるメッセージが /var/adm/messages ファイルに記録されていることを確認してください。まだ存在しない場合は、/var/adm/messages アクションに daemon.debug エントリを追加します。

ゾーンクラスタの HAStoragePlus リソースを構成するための追加の管理タスク

ゾーンクラスタの HAStoragePlus リソースを構成する場合は、グローバルクラスタの手順を実行する前に、次の追加タスクを実行する必要があります。

- ファイルシステムマウントポイントで UFSまたはスタンドアロン QFSなどのファイルシステムを構成している間に、それらのファイルシステムをゾーンクラスタに対して構成する必要があります。ファイルシステムをゾーンクラスタに対して構成する方法の詳細は、『安装和配置 Oracle Solaris Cluster 4.4 环境』 の 「ローカルファイルシステムを特定のゾーンクラスタノードに追加する方法 (CLI)」を参照してください。
- グローバルデバイスパスでグローバルデバイスを構成している間に、それらのデバイスをゾーンクラスタに対して構成する必要があります。グローバルデバイスをゾーンクラスタに対して構成する方法の詳細は、『安装和配置 Oracle Solaris Cluster 4.4 环境』 の 「ゾーンクラスタにストレージデバイスを追加する」を参照してください。
- ZFS ストレージプールを使用して ZFS ファイルシステムを構成している間に、その ZFS プールをゾーンクラスタに対して構成する必要があります。ZFS ファイルシステムをゾーンクラスタに対して構成する方法の詳細は、『安装和配置 Oracle Solaris Cluster 4.4 环境』 の 「ZFS ストレージプールをゾーンクラスタに追加する方法 (clsetup)」を参照してください。

▼ 新しいリソースの HAStoragePlus リソースタイプを設定する方法

次の例では、リソースグループ resource-group-1 に次のデータサービスが含まれています。

- HA for Oracle iPlanet Web Server。これは、/global/resource-group-1 に依存します。

- HA for Oracle。これは /dev/global/dsk/d5s2 に依存します。
- HA for NFS。これは dsk/d6 に依存します。

注記 - 高可用性ローカルファイルシステムとして ZFS ストレージプールで HAStoragePlus リソースを作成するには、164 ページの「ローカル ZFS ファイルシステムを高可用性にするように HAStoragePlus リソースタイプを設定する方法」セクションを参照してください。

---

resource-group-1 内の新しいリソースの HAStoragePlus リソース hastorageplus-1 を作成するには、139 ページの「HAStoragePlus を使用したリソースグループとデバイスグループの間の起動の同期」を参照してから、次の手順を実行します。

HAStoragePlus リソースを作成するには、156 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、RBAC の承認 solaris.cluster.modify および solaris.cluster.admin を提供する root 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. **このリソースタイプが登録されているかどうかを確認します。**

   次のコマンドは、登録されているリソースタイプのリストを出力します。

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

4. **必要な場合は、このリソースタイプを登録します。**

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

5. **HAStoragePlus リソース hastorageplus-1 を作成し、ファイルシステムマウントポイントとグローバルデバイスパスを定義します。**

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

   GlobalDevicePaths には、次の値を含めることができます。

   - グローバルデバイスグループ名 (nfs-dg、dsk/d5 など)
   - グローバルデバイスへのパス (/dev/global/dsk/d1s2、/dev/md/nfsdg/dsk/d10 など)

   FilesystemMountPoints には、次の値を含めることができます。

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

注記 - HAStoragePlus には、ZFS ファイルシステムストレージプールを構成するために使用される Zpools 拡張プロパティーと、ZFS ファイルシステムストレージプールのデバイスを検索する場所を指定するために使用される ZpoolsSearchDir 拡張プロパティーがあります。ZpoolsSearchDir 拡張プロパティーのデフォルト値は /dev/dsk です。ZpoolsSearchDir 拡張プロパティーは、zpool(8) コマンドの -d オプションと類似しています。

リソースは有効状態で作成されます。

6. **resource-group-1 にリソース (Oracle iPlanet Web Server、Oracle、および NFS) を追加し、それらの依存関係を hastorageplus-1 に設定します。**

   たとえば、Oracle iPlanet Web Server の場合は、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 Resource_dependencies=schost-1 -p Port_list=80/tcp \
-p Resource_dependencies_offline_restart=hastorageplus-1 resource
```

   リソースは有効状態で作成されます。

7. **リソース依存関係が正しく構成されたことを確認します。**

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

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

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

アフィニティースイッチオーバー

HAStoragePlus リソースタイプには、HAStoragePlus が GLobalDevicePaths および FileSystemMountPoints 拡張プロパティーで定義されているグローバルデバイスに対してアフィニティースイッチオーバーを実行する必要があるかどうかを指定するブール値である、別の拡張プロパティー AffinityOn が含まれています。詳細は、SUNW.HAStoragePlus(7) のマニュアルページを参照してください。

注記 - スケーラブルサービスの場合、AffinityOn フラグの設定は無視されます。スケーラブルリソースグループでアフィニティースイッチオーバーを実行できません。

▼ 既存のリソースの HAStoragePlus リソースタイプを設定する方法

始める前に 139 ページの「HAStoragePlus を使用したリソースグループとデバイスグループの間の起動の同期」を参照してください。

1. **このリソースタイプが登録されているかどうかを確認します。**

   次のコマンドは、登録されているリソースタイプのリストを出力します。

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

2. **必要な場合は、このリソースタイプを登録します。**

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

3. **HAStoragePlus リソース hastorageplus-1 を作成します。**

   ```
   # clresource create -g resource-group \
   -t SUNW.HAStoragePlus -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、既存の各リソースの依存関係を設定します。**

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

5. **リソース依存関係が正しく構成されたことを確認します。**

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

クラスタファイルシステムの HAStoragePlus リソースの構成

クラスタファイルシステムで HAStoragePlus リソースが構成され、オンラインになると、これらのファイルシステムは確実に使用可能になります。クラスタファイルシステムは、UFS および ZFS 上でサポートされています。データサービスが I/O 集中型である場合は、ローカルファイルシステムの HAStoragePlus を使用してください。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、185 ページの「HAStoragePlus リソースのクラスタ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方法」を参照してください。

クラスタファイルシステムは、ループバックマウントメカニズムを使用して、HAStoragePlus リソースのゾーンクラスタに対して構成できます。SUNW.HAStoragePlus リソースタイプは、クラスタファイルシステムをグローバルクラスタ内にマウントすることによって、そのファイルシステムをゾーンクラスタから使用

可能にします。このリソースタイプは、次に、リソースグループがオンラインであるゾーンクラスタノード上でループバックマウントを実行します。

注記 - フェイルオーバーリソースグループを使用している場合、リソースグループは 1 つのノード上でしかオンラインになりません。スケーラブルリソースグループを使用している場合は、`Desired_primaries` プロパティーによって、リソースグループがオンラインになるノードの数が定義されます。

ゾーンクラスタの `HAStoragePlus` リソースタイプで構成されているクラスタファイルシステムは、`clzonecluster` コマンドを使用して、ゾーンクラスタでの使用を承認するようにしてください。詳細は、`clzonecluster`(8CL) のマニュアルページおよび『安装和配置 Oracle Solaris Cluster 4.4 环境』 の 「クラスタファイルシステムをゾーンクラスタに追加する方法 (clsetup)」を参照してください。

このセクションには、次の情報が含まれます。

- 146 ページの「UFS クラスタファイルシステムの `/etc/vfstab` 内のサンプルエントリ」
- 147 ページの「UFS ファイルシステムを使用してクラスタファイルシステムの `HAStoragePlus` リソースを設定する方法」
- 149 ページの「グローバル ZFS ストレージプールの `HAStoragePlus` リソースを設定する方法 」
- 154 ページの「クラスタファイルシステムの `HAStoragePlus` リソースを削除する方法」

UFS クラスタファイルシステムの `/etc/vfstab` 内のサンプルエントリ

次の例は、UFS クラスタファイルシステムに使用されるグローバルデバイスの `/etc/vfstab` ファイル内のエントリを示しています。

注記 - UFS クラスタファイルシステムの `/etc/vfstab` ファイル内のエントリは、マウントオプションに `global` キーワードを含めるようにしてください。

例 42 Solaris Volume Manager を使用したグローバルデバイスの `/etc/vfstab` 内のエントリ

この例では、Solaris Volume Manager を使用するグローバルデバイスの `/etc/vfstab` ファイル内のエントリを示します。

```
/dev/md/kappa-1/dsk/d0     /dev/md/kappa-1/rdsk/d0
/global/local-fs/nfs ufs    5  yes     logging,global
```

▼ UFS ファイルシステムを使用してクラスタファイルシステムの HAStoragePlus リソースを設定する方法

クラスタファイルシステムに UFS ファイルシステムを使用する HAStoragePlus リソースを構成するには、このタスクを実行します。

ZFS ストレージプールを使用する HAStoragePlus でクラスタファイルシステムを作成する場合は、代わりに 149 ページの「グローバル ZFS ストレージプールの HAStoragePlus リソースを設定する方法 」を参照してください。

1. **クラスタ内のいずれかのノードで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **必要に応じて、フェイルオーバーまたはスケーラブルリソースグループを作成します。**

   - **フェイルオーバーグループを作成するには、次の手順を実行します。**

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

   - **スケーラブルグループを作成するには、次の手順を実行します。**

     ```
     # clresourcegroup create -S [-p Maximum_primaries=m] [-p Desired_primaries=n] \
     [-n node-list] resource-group
     ```

3. **`SUNW.HAStoragePlus` リソースタイプを登録します。**

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

4. **`HAStoragePlus` リソースを作成し、ファイルシステムマウントポイントを定義します。**

   ```
   # clresource create -g resource-group -t SUNW.HAStoragePlus \
   -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* にデータサービスリソースを追加し、それらの依存関係を *hasp-resource* に設定します。**

   ```
   # clresource set -p  Resource_dependencies_offline_restart= \
   hasp-resource  application-resource
   ```

6. **`HAStoragePlus` リソースを含むリソースグループをオンラインおよび管理状態にします。**

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

例 43 グローバルクラスタ内のクラスタファイルシステムを使用した HAStoragePlus リソースの設定

この例では、フェイルオーバーリソースグループ用に、グローバルクラスタ内のクラスタファイルシステム /global/ufs を使用して HAStoragePlus リソースを構成する方法を示します。

```
phys-schost-1# vi /etc/vfstab
#device         device          mount           FS      fsck    mount   mount
#to mount       to fsck         point           type    pass    at boot options
#
/dev/md/apachedg/dsk/d0 /dev/md/apachedg/rdsk/d0 /global/ufs ufs 2 yes global, logging

# clresourcegroup create hasp-rg
# clresourcetype register SUNW.HAStoragePlus
# clresource create -g hasp-rg -t SUNW.HAStoragePlus -p \
FileSystemMountPoints=/global/ufs hasp-rs
# clresourcegroup online -M hasp-rg
```

例 44 ゾーンクラスタ内のクラスタファイルシステムを使用した HAStoragePlus リソースの設定

この例では、スケーラブルリソースグループ用に、ゾーンクラスタ内のクラスタファイルシステム /global/ufs を使用して HAStoragePlus リソースを構成する方法を示します。このクラスタファイルシステムは、マウントポイント /zone/ufs 上でゾーンクラスタノードに対して使用できます。この構成例は、グローバルファイルシステム /global/ufs がグローバルクラスタ内にマウントされ、あとでリソースグループがオンラインである 2 つのゾーンクラスタノードにループバックマウントされるようにします。

```
phys-schost-1# vi /etc/vfstab
#device         device          mount           FS      fsck    mount   mount
#to mount       to fsck         point           type    pass    at boot options
#
/dev/md/apachedg/dsk/d0 /dev/md/apachedg/rdsk/d0 /global/ufs ufs 2 yes global, logging

# clzonecluster configure sczone
clzc:sczone> add fs
clzc:sczone:fs> set dir=/zone/ufs
clzc:sczone:fs> set special=/global/ufs
clzc:sczone:fs> set type=lofs
clzc:sczone:fs> end
clzc:sczone:fs> exit

# clresourcegroup create -Z sczone -p desired_primaries=2 -p maximum_primaries=2 hasp-rg
# clresourcetype register -Z sczone SUNW.HAStoragePlus
# clresource create -Z sczone -g hasp-rg -t SUNW.HAStoragePlus -p
FileSystemMountPoints=/zone/ufs hasp-rs
# clresourcegroup online -Z sczone -M hasp-rg
```

▼ グローバル ZFS ストレージプールの HAStoragePlus リソースを設定する方法

クラスタファイルシステムに ZFS ストレージプールを使用する HAStoragePlus リソースを作成するには、このタスクを実行します。

UFS ファイルシステムを使用する HAStoragePlus でクラスタファイルシステムを作成する場合は、代わりに 147 ページの「UFS ファイルシステムを使用してクラスタファイルシステムの HAStoragePlus リソースを設定する方法」を参照してください。

1. ZFS ストレージプールを作成します。

```
# zpool create mypool /dev/dsk/c0t0d0
```

注意 - 構成済みの定足数デバイスを 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がストレージプールに追加されると、ディスクのラベルが EFI ディスクに変更され、定足数構成情報が失われ、さらにディスクはクラスタに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。あるいは、ディスクの構成を解除し、それをストレージプールに追加したあと、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。

Oracle Solaris Cluster 構成内に ZFS ストレージプールを作成する場合は、次の要件に従ってください。

- ZFS ストレージプールの作成元とな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なことを確認してください。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されている必要があります。
- zpool(8) コマンドに対して指定した Oracle Solaris デバイス識別子 (たとえば、/dev/dsk/c0t0d0) が、cldevice list -v コマンドに認識されることを確認してください。

注記 - 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。

- 最適なパフォーマンスのために、ディスク全体を使用して ZFS ストレージプールを作成します。Oracle Solaris 論理デバイスを ZFS ファイルシステムとして指定すると、ディスクの書き込みキャッシュを有効にすることによってパフォーマンスが向上します。ディスク全体が指定された場合、ZFS ファイルシステムはそのディスクに EFI でラベル付けします。
- DID デバイス上に zpool を作成している場合は、スライスを指定する必要があります。`/dev/did/dsk/d`*N* は、ディスクラベルを破損させる可能性があるため使用しないでください。

ZFS ストレージプールを作成する方法については、『Managing ZFS File Systems in Oracle Solaris 11.4』 の 「Creating ZFS Storage Pools」を参照してください。

2. **新しく作成された ZFS zpool をエクスポートします。**

```
# zpool export mypool
```

3. **クラスタ内のいずれかのノードで、`root` 役割になるか、承認 `solaris.cluster.modify` を提供する役割になります。**

4. **必要に応じて、フェイルオーバーまたはスケーラブルリソースグループを作成します。**

   - フェイルオーバーグループを作成するには、次の手順を実行します。

```
# clresourcegroup create mypool-rg
```

   - スケーラブルグループを作成するには、次の手順を実行します。

```
# clresourcegroup create -S [-p Maximum_primaries=m] [-p Desired_primaries=n] \
[-n node-zone-list] mypool-rg
```

5. **`SUNW.HAStoragePlus` リソースタイプを登録します。**

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

6. **ZFS ファイルシステムの `HAStoragePlus` リソースを作成します。**

   - **ZFS ストレージプールのデバイスを検索するデフォルトの場所は `/dev/dsk` です。`HAStoragePlus` は、必要な ZFS プールのデバイスグループを自動的に作成し、その `poolaccess` プロパティーを `global` に設定します。**

```
# clresource create -g resource-group -t SUNW.HAStoragePlus \
-p GlobalZpools=mypool
 \
mypool hasp-resource
```

- **ZFS ストレージプールのデバイスを検索するデフォルトの場所が `/dev/dsk` 以外の場合、まず ZFS プールのデバイスグループを手動で作成してから、各 ZFS プールの `HAStoragePlus` リソースを作成する必要があります。次に例を示します。**

```
# cldevicegroup create [-p poolaccess=global] \
[-p searchpaths=/dev/did/dsk] [-p readonly=false]\
[-n node-list] -t zpool mypool

# clresource create -g resource-group -t SUNW.HAStoragePlus \
-p GlobalZpools=mypool hasp-resource
```

リソースは有効状態で作成されます。

7. ***hasp-resource* に対するデータサービスリソースの依存関係を設定します。**

```
# clresource set -p  Resource_dependencies_offline_restart=\
hasp-resource  application-resource
```

8. **`HAStoragePlus` リソースを含むリソースグループをオンラインおよび管理状態にします。**

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

この時点で、ストレージプール内に追加の ZFS ファイルシステムを作成することもできます。これらのファイルシステムはすべて、クラスタファイルシステムとしてグローバルにマウントされます。ストレージプール内に追加の ZFS ファイルシステムを作成する場合は、ステップ 9 およびステップ 10 に従ってください。

ZFS プール内に ZFS ファイルシステムを作成する場合は、次の要件に従ってください。

- 同じ ZFS ストレージプール内に複数の ZFS ファイルシステムを作成できます。
- `HAStoragePlus` は、ZFS ファイルシステム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 ファイルシステムを `FilesystemMountPoints` 拡張プロパティーに含めないでください。
- 必要に応じて、ZFS の `failmode` プロパティーの設定を、`continue` または `panic` のどちらか、要件に当てはまる方に変更します。

注記 - ZFS プールの failmode プロパティーは、デフォルトでは wait に設定されています。この設定により、HAStoragePlus リソースがブロックされる場合があり、それによってリソースグループのフェイルオーバーが妨げられることがあります。zpool の推奨される設定は failmode=continue です。この zpool を管理している HAStoragePlus リソースで、RebootOnFailure プロパティーを TRUE に設定します。あるいは、zpool failmode=panic を設定して、ストレージの損失時のパニック、クラッシュダンプ、およびフェイルオーバーを保証することもできます。failmode=panic の設定は、RebootOnFailure プロパティーの設定に関係なく機能します。ただし、RebootOnFailure=TRUE を設定すると、そのモニターでストレージの停止をすばやく検出できるため、応答性が向上する可能性があります。

- ZFS ファイルシステムを作成するときに、その暗号化を選択できます。HAStoragePlus リソースは、リソースのオンライン化中にプール内のすべてのファイルシステムを自動的にマウントします。マウント中に鍵またはパスフレーズを対話式に入力する必要がある暗号化されたファイルシステムでは、リソースをオンラインにする際に問題が発生します。

  問題を回避するために、HAStoragePlus リソースを使用してクラスタによって管理されている ZFS ストレージプールの暗号化されたファイルシステムには keysource=raw | hex | passphrase,prompt|pkcs11: を使用しないでください。HAStoragePlus リソースがオンラインになるクラスタノードから鍵またはパスフレーズの場所にアクセスできる場合は、keysource=raw | hex | passphrase, file://|https:// を使用できます。

9. **(オプション) zfs コマンドを実行して追加のファイルシステムを作成するために、プライマリノードを見つけます。**

   ```
   # clresourcegroup status mypool-rg
   ```

   グループがオンラインになっているノードを特定します。

10. **(オプション) プライマリノード上で zfs create コマンドを実行して、追加のファイルシステムを作成します。**

    ```
    # zfs create mypool/filesystem1
    ```

    各クラスタノード上で次のコマンドを実行すると、ファイルシステムがグローバルにマウントされていることを確認できます。

    ```
    # ls -l /mypool/filesystem1
    ```

例 45 グローバルクラスタ内のクラスタファイルシステムを含む ZFS プールを使用した HAStoragePlus リソースの設定

この例では、グローバルクラスタ内の ZFS ストレージプール mypool を管理するために HAStoragePlus リソースを構成する方法を示します。mypool のファイルシステム

データセットは、/mypool の下にマウントされ、すべてのグローバルクラスタノードで使用可能になります。

次のコマンドは大域ゾーンで実行されます。

```
phys-schost-1# zpool create mypool /dev/dsk/c1d5
# clresourcegroup create hasp-rg
# clresourcetype register SUNW.HAStoragePlus
# clresource create -g hasp-rg -t SUNW.HAStoragePlus \
-p GlobalZpools=mypool hasp-rs
# clresourcegroup online -M hasp-rg
```

仮想デバイスの詳細は、zpool(8) のマニュアルページを参照してください。

例 46 ゾーンクラスタ内の ZFS クラスタファイルシステムを使用した HAStoragePlus リソースの設定

この例では、ゾーンクラスタ sczone 内のクラスタファイルシステム /mypool/fs1 を使用して HAStoragePlus リソースを構成する方法を示します。この例では、ゾーンクラスタ内のスケーラブルリソースグループを使用して、ファイルシステムをすべてのゾーンクラスタノードで使用可能にします。このクラスタファイルシステムは、マウントポイント /global/fs1 上でゾーンクラスタノードに対して使用できます。

この構成例では、グローバルクラスタ内にマウントされたクラスタファイルシステム /mypool/fs1 が使用され、あとでリソースグループがオンラインであるゾーンクラスタノードにループバックマウントされます。

次のコマンドは大域ゾーンで実行されます。

```
phys-schost-1# zpool create mypool /dev/dsk/c1d5
# clresourcegroup create hasp-rg
# clresourcetype register SUNW.HAStoragePlus
# clresource create -g hasp-rg -t SUNW.HAStoragePlus \
-p GlobalZpools=mypool hasp-rs
# clresourcegroup online -M hasp-rg

# clzonecluster configure sczone
clzc:sczone> add fs
clzc:sczone:fs> set dir=/global/fs1
clzc:sczone:fs> set special=/mypool/fs1
clzc:sczone:fs> set type=lofs
clzc:sczone:fs> end
clzc:sczone:fs> exit

# clresourcegroup create -Z sczone \
-p RG_affinities=++global:hasp-rg \
zc-hasp-rg
# clresourcetype register -Z sczone SUNW.HAStoragePlus
# clresource create -Z sczone -g zc-hasp-rg -t SUNW.HAStoragePlus \
-p FileSystemMountPoints=/global/fs1 zc-hasp-rs

# clresourcegroup online -Z sczone -M zc-hasp-rg
```

仮想デバイスの詳細は、zpool(8) のマニュアルページを参照してください。

▼ クラスタファイルシステムの HAStoragePlus リソースを削除する方法

- **クラスタファイルシステムで構成されている HAStoragePlus リソースを無効にして削除します。**

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

グローバル ZFS ストレージプールを管理する HAStoragePlus リソースのオンラインでの変更

▼ オンライン HAStoragePlus リソースにグローバル ZFS ストレージプールを追加する方法

オンライン HAStoragePlus リソースにグローバル ZFS ストレージプールを追加すると、HAStoragePlus リソースは次の操作を実行します。

- グローバル ZFS ストレージプールをインポートします。
- グローバル 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します。

1. **クラスタ内のいずれかのノードで、root 役割になるか、承認 solaris.cluster.modify 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グローバル ZFS ストレージプールを確認します。**

```
# clresource show -p GlobalZpools hasp-resource
```

hasp-resource

グローバル ZFS ストレージプールの追加先となる 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、新しいグローバル ZFS ストレージプールを追加します。**

```
# clresource set -p GlobalZpools+=mypool hasp-resource
```

-p GlobalZpools+=*mypool*

追加する新しいグローバル ZFS ストレージプールの名前を指定します。

hasp-resource

ZFS ストレージプールの追加先となる HAStoragePlus リソースを指定します。

4. **HAStoragePlus リソースが管理している ZFS ストレージプールの新しいリストを ステップ 2 で生成したリストと比較します。**

```
# clresource show -p GlobalZpools hasp-resource
```

hasp-resource

ZFS ストレージプールを追加した先の HAStoragePlus リソースを指定します。

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

HAStoragePlus リソースがオンラインであるが、障害が発生した場合、リソースの検証には成功しました。ただし、HAStoragePlus リソースによる ZFS ファイルシステムのインポートとマウントの試みが失敗しました。この場合は、システムコンソール上または /var/adm/messages ディレクトリ内のエラーメッセージを確認します。エラーを修正したあとで、前の一連のステップを繰り返します。

```
# clresource status -t SUNW.HAStoragePlus +
```

▼ オンライン HAStoragePlus リソースからグローバル ZFS ストレージプールを削除する方法

オンライン HAStoragePlus リソースからグローバル ZFS ストレージプールを削除すると、HAStoragePlus リソースは次の操作を実行します。

- グローバル 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します。
- プライマリノードからグローバル ZFS ストレージプールをエクスポートします。

1. **クラスタ内のいずれかのノードで、root 役割になるか、承認 solaris.cluster.modify 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グローバル ZFS ストレージプールを確認します。**

```
# clresource show -p GlobalZpools hasp-resource
```

hasp-resource

ZFS ストレージプールを削除する HAStoragePlus リソースを指定します。

3. **HAStoragePlus 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、グローバル ZFS ストレージプールを削除します。**

```
# clresource set -p GlobalZpools-=mypool  hasp-resource
```

`-p` GlobalZpools-=*mypool*

削除するグローバル ZFS ストレージプールの名前を指定します。

hasp-resource

グローバル ZFS ストレージプールを削除する `HAStoragePlus` リソースを指定します。

4. **`HAStoragePlus` リソースが現在管理しているグローバル ZFS ストレージプールの新しいリストをステップ 2 で生成したリストと比較します。**

```
# clresource show -p GlobalZpools hasp-resource
```

hasp-resource

グローバル ZFS ストレージプールを削除した元の `HAStoragePlus` リソースを指定します。

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

`HAStoragePlus` リソースがオンラインであるが、障害が発生した場合、リソースの検証には成功しました。ただし、`HAStoragePlus` リソースによるグローバル ZFS ファイルシステムのアンマウントとエクスポートの試みが失敗しました。この場合は、システムコンソール上または `/var/adm/messages` ディレクトリ内のエラーメッセージを確認します。エラーを修正したあとで、前の一連のステップを繰り返します。

```
# clresource status -t SUNW.HAStoragePlus +
```

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、I/O 集中型のデータサービスのパフォーマンスが向上します。Oracle Solaris Cluster 環境でローカルファイルシステムを高可用性にするには、`HAStoragePlus` リソースタイプを使用します。

クラスタファイルシステムまたはローカルファイルシステムを指定できます。クラスタファイルシステムには、クラスタ内のすべてのノード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からアクセス可能です。`SUNW.HAStoragePlus` 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、ベースとなるデバイスは Oracle Solaris Cluster グローバルデバイスである必要があります。

これらのファイルシステムマウントポイントは、`paths[,…]` という形式で定義されます。このプロパティーのデフォルト設定は空のリストです。

SUNW.HAStoragePlus リソースタイプを使用すると、ファイルシステムをゾーンクラスタノードから使用可能にすることができます。ゾーンクラスタの SUNW.HAStoragePlus リソースタイプで構成されているファイルシステムは、clzonecluster コマンドを使用して、ゾーンクラスタでの使用を承認するようにしてください。詳細は、clzonecluster(8CL) のマニュアルページおよび『安装和配置 Oracle Solaris Cluster 4.4 环境』 の 「ゾーンクラスタにファイルシステムを追加する」を参照してください。

注記 - ローカルファイルシステムには UFS、QFS、および ZFS が含まれます。

I/O 集中型である Oracle Solaris Cluster データサービスごとの手順では、各データサービスを HAStoragePlus リソースタイプで動作するように構成する方法について説明します。詳細は、個々の Oracle Solaris Cluster データサービスガイドを参照してください。

注記 - ルートファイルシステムを高可用性にするために HAStoragePlus リソースタイプを使用しないでください。

Oracle Solaris 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用意されています。

- **Oracle Solaris Cluster Manager** ブラウザインタフェース。ログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。
- **clsetup ユーティリティー**。
- **Oracle Solaris Cluster の保守コマンド**。

Oracle Solaris Cluster Manager および clsetup ユーティリティーでは、対話形式でリソースグループにリソースを追加できます。これらのリソースを対話形式で構成すると、コマンドの構文エラーや省略から生じる構成エラーの可能性が低減されます。Oracle Solaris Cluster Manager および clsetup ユーティリティーでは、必要なすべてのリソースが作成され、リソース間で必要なすべての依存関係が設定されることが保証されます。

高可用性ローカルファイルシステムのための構成要件

マルチホストディスク上のすべてのファイルシステムが、これらのマルチホストディスクに直接接続されているどのホストからでも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、高可用性ローカルファイルシステムを次のように構成します。

- 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グローバルデバイス上にあることを確認します。

- これらのグローバルデバイスを指定する HAStoragePlus リソースの AffinityOn 拡張プロパティーを True に設定します。

  HAStoragePlus リソースの Zpools 拡張プロパティーは、AffinityOn 拡張プロパティーを無視します。
- フェイルオーバーリソースグループ内に 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でのフェイルバックの設定が同じであることを確認します。

注記 - 高可用性ローカルファイルシステムのグローバルデバイスでのボリュームマネージャーの使用はオプションです。

ボリュームマネージャーを使用しないデバイスでのデバイス名の形式

ボリュームマネージャーを使用していない場合は、ベースとなるストレージデバイスの名前に適切な形式を使用してください。使用する形式は、ストレージデバイスのタイプによって次のように異なります。

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

これらのデバイス名の置き換え可能な要素は次のとおりです。

- *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 内のサンプルエントリ

次の例は、高可用性ローカルファイルシステムに使用されるグローバルデバイスの /etc/vfstab ファイル内のエントリを示しています。

注記 - ZFS では /etc/vfstab ファイルを使用しません。

例 47 ボリュームマネージャーを使用しないグローバルデバイスの /etc/vfstab 内のエントリ

この例では、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グローバルデバイスの /etc/vfstab ファイル内の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 48 Solaris Volume Manager を使用したグローバルデバイスの /etc/vfstab 内のエントリ

この例では、Solaris Volume Manager を使用するグローバルデバイスの /etc/vfstab ファイル内のエントリを示します。

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

注記 - SUNW.HAStoragePlus リソースタイプを使用して、ファイルシステムをゾーンクラスタに対して構成する場合は、同じファイルシステムエントリをゾーンクラスタ構成に追加する必要があります。

▼ clsetup ユーティリティーを使用して HAStoragePlus リソースタイプを設定する方法

次の手順では、clsetup ユーティリティーを使用して SUNW.HAStoragePlus リソースタイプを設定する方法について説明します。この手順は、いずれかのクラスタノードから実行します。

この手順では、Oracle Solaris Cluster の長い形式の保守コマンドを使用します。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、コマンドは同一です。

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用して、HAStoragePlus リソースとそれを含む新しいリソースグループを 1 回の操作で作成することもできます。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。ログインしたあと、「タスク」をクリックし、「高可用性ストレージ」をクリックしてウィザードを起動します。

このウィザードでは、すべてのクラスタノードが同じ root パスワードを持つ必要があります。

始める前に

- グローバルクラスタに UFS 以外のファイルシステムの必要なボリューム、zpool、ディスクグループ、ファイルシステム、およびマウントポイントが作成されていることを確認します。
- ゾーンクラスタの場合、それが使用するすべてのファイルシステムが、clzonecluster コマンドを使用してゾーンクラスタに追加されていることを確認します。詳細は、『安装和配置 Oracle Solaris Cluster 4.4 环境』 の 「ゾーンクラスタにファイルシステムを追加する」を参照してください。

1. **いずれかのクラスタノード上で root 役割になります。**

2. **clsetup ユーティリティーを起動します。**

   ```
   # 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番号を入力します。**
   「データサービス」メニューが表示されます。

4. **高可用性ストレージを構成するためのオプション番号を入力します。**
   clsetup ユーティリティーは、このタスクを実行するための前提条件のリストを表示します。

5. **前提条件を満たしていることを確認します。**
   clsetup ユーティリティーは、高可用性 HAStoragePlus リソースをマスターできるクラスタノードのリストを表示します。

6. **高可用性 HAStoragePlus リソースをマスターできるノードを選択します。**

   - **任意の順序で一覧表示されたすべてのノードのデフォルトの選択を受け入れるには、a と入力します。**

   - **一覧表示されたノード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またはスペースで区切られたリストを入力します。**
     各ノードが、HAStoragePlus 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一覧表示されていることを確認してください。リスト内の最初のノードは、このリソースグループのプライマリノードです。

   - **すべてのノードを特定の順序で選択するには、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ます。**

7. **ノードの選択を確定するには、d と入力します。**
   clsetup ユーティリティーは、データが格納される共有ストレージのタイプ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タイプのオプション番号を入力します。**
   clsetup ユーティリティーは、クラスタで構成されているファイルシステムマウントポイントのリストを表示します。グローバルクラスタによって使用される UFS ファイルシステムの既存のマウントポイントが存在しない場合は、clsetup ユーティリティーを使用すると新しいマウントポイントを定義できます。

9. **デフォルトマウントディレクトリ、raw デバイスパス、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マウントポイントのプロパティーを返します。

10. マウントポイントを作成するには、d と入力します。

clsetup ユーティリティーは、使用可能なファイルシステムマウントポイントを表示します。

注記 - c オプションを使用すると、別の新しいマウントポイントを定義できます。

---

11. ファイルシステムマウントポイントを選択します。

- **任意の順序で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の選択を受け入れるには、a と入力します。**
- **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区切りのリストを入力します。**

12. ノードの選択を確定するには、d と入力します。

clsetup ユーティリティーは、クラスタで構成されているグローバルディスクセットとデバイスグループのリストを表示します。

13. グローバルデバイスグループを選択します。

- **任意の順序で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の選択を受け入れるには、a と入力します。**
- **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、デバイスグループのオプション番号の、コンマまたはスペース区切りのリストを入力します。**

14. ノードの選択を確定するには、d と入力します。

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 Oracle Solaris Cluster オブジェクトの名前を表示します。

15. いずれかの Oracle Solaris Cluster オブジェクトに別の名前が必要な場合は、次のように名前を変更します。

a. 変更する名前の番号を入力します。

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 Oracle Solaris Cluster オブジェクトの名前のリストに戻ります。

16. **選択した Oracle Solaris Cluster オブジェクト名を確定するには、d と入力します。**

    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 Oracle Solaris Cluster 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 と入力します。**

    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。構成が完了すると、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実行したコマンドを一覧表示します。

18. **(オプション) clsetup ユーティリティーが終了するまで繰り返し q と入力し、Return キーを押します。**

    必要に応じて、ほかの必要なタスクを実行している間、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できます。clsetup の終了を選択した場合は、ユーザーがこのユーティリティーを再起動すると、既存のリソースグループがユーティリティーによって認識されます。

19. **HAStoragePlus リソースが作成されたことを確認します。**

    この目的には、clresource(8CL) ユーティリティーを使用します。

```
# clresource show resource-group
```

▼ ZFS 以外のファイルシステムを高可用性にするように HAStoragePlus リソースタイプを設定する方法

次の手順では、ZFS 以外のファイルシステムを高可用性にするように HAStoragePlus リソースタイプを設定する方法について説明します。

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用して、HAStoragePlus リソースとそれを含む新しいリソースグループを 1 回の操作で作成することもできます。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。ログインしたあと、「タスク」をクリックし、「高可用性ストレージ」をクリックしてウィザードを起動します。

このウィザードでは、すべてのクラスタノードが同じ root パスワードを持つ必要があります。

---

1. **グローバルクラスタ内のいずれかのノードで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 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
```

3. **`HAStoragePlus` リソースタイプを登録します。**

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

4. **`HAStoragePlus` リソースを作成し、ファイルシステムマウントポイントを定義します。**

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

5. **`HAStoragePlus` リソースを含むリソースグループをオンラインおよび管理状態にします。**

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

例 **49** UFS ファイルシステムをグローバルクラスタで高可用性にするための `HAStoragePlus` リソースタイプの設定

この例では、ファイルシステム */web-1* が、そのファイルシステムをグローバルクラスタで高可用性にするように `HAStoragePlus` リソースに対して構成されると仮定します。

```
phys-schost-1# vi /etc/vfstab
#device         device          mount           FS      fsck    mount   mount
#to mount       to fsck         point           type    pass    at boot options
#
# /dev/md/apachedg/dsk/d0 /dev/md/apachedg/rdsk/d0 /web-1 ufs 2 no logging

# clresourcegroup create hasp-rg
# clresourcetype register SUNW.HAStoragePlus
# clresource create -g hasp-rg -t SUNW.HAStoragePlus -p FileSystemMountPoints=/global/ufs-1 hasp-rs
# clresourcegroup online -M hasp-rg
```

例 **50** UFS ファイルシステムをゾーンクラスタで高可用性にするための `HAStoragePlus` リソースタイプの設定

この例では、ファイルシステム */web-1* が、そのファイルシステムを *sczone* という名前のゾーンクラスタで高可用性にするように `HAStoragePlus` リソースに対して構成されると仮定します。ローカルファイルシステムが、`SUNW.HAStoragePlus` リソースタイプを使用してゾーンクラスタに対して高可用性ローカルファイルシステムとして構成されている場合、HAStoragePlus リソースは、そのゾーンクラスタ構成内のファイルシステム情報を読み取ります。

```
# clzonecluster configure sczone
clzc:sczone> add fs
```

```
clzc:sczone:fs> set dir=/web-1
clzc:sczone:fs> set special=/dev/md/apachedg/dsk/d0
clzc:sczone:fs> set raw=/dev/md/apachedg/rdsk/d0
clzc:sczone:fs> set type=ufs
clzc:sczone:fs> add options [logging]
clzc:sczone:fs> end
clzc:sczone:fs> exit

# clresourcegroup create -Z sczone hasp-rg
# clresourcetype register -Z sczone SUNW.HAStoragePlus
# clresource create -Z sczone -g hasp-rg \
-t SUNW.HAStoragePlus -p FileSystemMountPoints=/web-1 hasp-rs
# clresourcegroup online -Z sczone -M hasp-rg
```

▼ ローカル ZFS ファイルシステムを高可用性にするように HAStoragePlus リソースタイプを設定する方法

ローカル ZFS ファイルシステムを高可用性にするには、次の主要なタスクを実行します。

- ZFS ストレージプールを作成します。
- その ZFS ストレージプール内に ZFS ファイルシステムを作成します。
- ZFS ストレージプールを管理する HAStoragePlus リソースを設定します。

このセクションでは、これらのタスクを完了する方法について説明します。

注意 - クラスタによってすでに管理されている ZFS プールを手動でインポートすることを予定している場合は、そのプールが複数のノードにインポートされないようにしてください。複数のノードにプールをインポートすると、問題が発生する場合があります。詳細は、180 ページの「HAStoragePlus リソースによって管理されている ZFS プール構成の変更」を参照してください。

注記 - Oracle Solaris Cluster Manager ブラウザインタフェースを使用して、ZFS ファイルシステム用の HAStoragePlus リソースとそれを含む新しいリソースグループを 1 回の操作で作成することもできます。Oracle Solaris Cluster Manager のログイン手順については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「Oracle Solaris Cluster Manager にアクセスする方法」を参照してください。ログインしたあと、「タスク」をクリックし、「高可用性ストレージ」をクリックしてウィザードを起動します。

このウィザードでは、すべてのクラスタノードが同じ root パスワードを持つ必要があります。

1. **ZFS ストレージプールを作成します。**

注意 - 構成済みの定足数デバイスを 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がストレージプールに追加されると、ディスクのラベルが EFI ディスクに変更され、定足数構成情報が失われ、さらにディスクはクラスタに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。あるいは、ディスクの構成を解除し、それをストレージプールに追加したあと、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。

Oracle Solaris Cluster 構成内に ZFS ストレージプールを作成する場合は、次の要件に従ってください。

- ZFS ストレージプールの作成元とな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なことを確認してください。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されている必要があります。
- `zpool(8)` コマンドに対して指定した Oracle Solaris デバイス識別子 (たとえば、`/dev/dsk/c0t0d0`) が、`cldevice list -v` コマンドに認識されることを確認してください。

注記 - 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。

- 最適なパフォーマンスのために、ディスク全体を使用して ZFS ストレージプールを作成します。Oracle Solaris 論理デバイスを ZFS ファイルシステムとして指定すると、ディスクの書き込みキャッシュを有効にすることによってパフォーマンスが向上します。ディスク全体が指定された場合、ZFS ファイルシステムはそのディスクに EFI でラベル付けします。
- DID デバイス上に zpool を作成している場合は、スライスを指定する必要があります。`/dev/did/dsk/d`*N* は、ディスクラベルを破損させる可能性があるため使用しないでください。

ZFS ストレージプールを作成する方法については、『Managing ZFS File Systems in Oracle Solaris 11.4』 の 「Creating ZFS Storage Pools」を参照してください。

2. **作成した ZFS ストレージプール内に ZFS ファイルシステムを作成します。**

   ZFS プール内に ZFS ファイルシステムを作成する場合は、次の要件に従ってください。

   - 同じ ZFS ストレージプール内に複数の ZFS ファイルシステムを作成できます。
   - `HAStoragePlus` は、ZFS ファイルシステム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 ファイルシステムを FilesystemMountPoints 拡張プロパティーに含めないでください。
- 必要に応じて、ZFS の failmode プロパティーの設定を、continue または panic のどちらか、要件に当てはまる方に変更します。

注記 - ZFS プールの failmode プロパティーは、デフォルトでは wait に設定されています。この設定により、HAStoragePlus リソースがブロックされる場合があり、それによってリソースグループのフェイルオーバーが妨げられることがあります。zpool の推奨される設定は failmode=continue です。この zpool を管理している HAStoragePlus リソースで、RebootOnFailure プロパティーを TRUE に設定します。あるいは、zpool failmode=panic を設定して、ストレージの損失時のパニック、クラッシュダンプ、およびフェイルオーバーを保証することもできます。failmode=panic の設定は、RebootOnFailure プロパティーの設定に関係なく機能します。ただし、RebootOnFailure=TRUE を設定すると、そのモニターでストレージの停止をすばやく検出できるため、応答性が向上する可能性があります。

- ZFS ファイルシステムを作成するときに、その暗号化を選択できます。HAStoragePlus リソースは、リソースのオンライン化中にプール内のすべてのファイルシステムを自動的にマウントします。マウント中に鍵またはパスフレーズを対話式に入力する必要がある暗号化されたファイルシステムでは、リソースをオンラインにする際に問題が発生します。問題を回避するために、HAStoragePlus リソースを使用してクラスタによって管理されている ZFS ストレージプールの暗号化されたファイルシステムには keysource=raw | hex | passphrase, prompt|pkcs11: を使用しないでください。HAStoragePlus リソースがオンラインになるクラスタノードから鍵またはパスフレーズの場所にアクセスできる場合は、keysource=raw | hex | passphrase,file://|https:// を使用できます。

ZFS ストレージプール内に ZFS ファイルシステムを作成する方法については、『Oracle Solaris 11.4 での ZFS ファイルシステムの管理』 の 「ZFS ストレージプールを作成する」を参照してください。

3. **クラスタ内のいずれかのノードで、RBAC の承認 solaris.cluster.modify を提供する root 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 ファイルシステムの 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を検索するデフォルトの場所は /dev/dsk です。これは、ZpoolsSearchDir 拡張プロパティーを使用してオーバーライドできます。

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および管理状態にします。

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

例 51 ローカル ZFS ファイルシステムをグローバルクラスタで高可用性にするための HAStoragePlus リソースタイプの設定

次の例は、ローカル ZFS ファイルシステムを高可用性にするためのコマンドを示しています。

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

DID Device          Full Device Path
----------          ----------------
d1                  phys-schost-1:/dev/rdsk/c0t0d0
d2                  phys-schost-1:/dev/rdsk/c0t1d0
d3                  phys-schost-1:/dev/rdsk/c1t8d0
d3                  phys-schost-2:/dev/rdsk/c1t8d0
d4                  phys-schost-1:/dev/rdsk/c1t9d0
d4                  phys-schost-2:/dev/rdsk/c1t9d0
d5                  phys-schost-1:/dev/rdsk/c1t10d0
d5                  phys-schost-2:/dev/rdsk/c1t10d0
d6                  phys-schost-1:/dev/rdsk/c1t11d0
d6                  phys-schost-2:/dev/rdsk/c1t11d0
d7                  phys-schost-2:/dev/rdsk/c0t0d0
d8                  phys-schost-2:/dev/rdsk/c0t1d0
     Solaris デバイス識別子を指定することによって、ディスクスライスを使用して ZFS ストレージプールを作成できます。
# zpool create HAzpool c1t8d0s2
     または、論理デバイス識別子を指定することによって、ディスクスライスを使用して ZFS ストレージプールを作成できます
# zpool create HAzpool /dev/did/dsk/d3s2
# zfs create HAzpool/export
# zfs create HAzpool/export/home
# clresourcegroup create hasp-rg
# clresourcetype register SUNW.HAStoragePlus
# clresource create -g hasp-rg -t SUNW.HAStoragePlus -p Zpools=HAzpool hasp-rs
# clresourcegroup online -M hasp-rg
```

例 52 ローカル ZFS ファイルシステムをゾーンクラスタで高可用性にするための HAStoragePlus リソースタイプの設定

次の例は、ゾーンクラスタ *sczone* でローカル ZFS ファイルシステムを高可用性にするための手順を示しています。

```
phys-schost-1# cldevice list -v
```

```
# zpool create HAzpool c1t8d0
# zfs create HAzpool/export
# zfs create HAzpool/export/home
# clzonecluster configure sczone
clzc:sczone> add dataset
clzc:sczone:fs> set name=HAzpool
clzc:sczone:fs> end
clzc:sczone:fs> exit
# clresourcegroup create -Z sczone hasp-rg
# clresourcetype register -Z sczone SUNW.HAStoragePlus
# clresource create -Z sczone -g hasp-rg -t SUNW.HAStoragePlus \
-p Zpools=HAzpool hasp-rs
# clresourcegroup online -Z -sczone -M hasp-rg
```

▼ ローカル ZFS ファイルシステムを高可用性にする HAStoragePlus リソースを削除する方法

- **ローカル ZFS ファイルシステムを高可用性にする HAStoragePlus リソースを無効にし、削除します。**

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

ゾーンクラスタ間での高可用性ローカルファイルシステムの共有

SUNW.HAStoragePlus リソースタイプを使用すると、グローバルクラスタリソースによって管理されている高可用性ローカルファイルシステムディレクトリをゾーンクラスタに対して共有できます。この方法によりストレージが統合され、異なるゾーンクラスタ上で実行されている異なるアプリケーションで高可用性ローカルファイルシステムが共有されます。ゾーンクラスタへのファイルシステムの追加については、『安装和配置 Oracle Solaris Cluster 4.4 环境』 の 「ゾーンクラスタにファイルシステムを追加する」を参照してください。

このセクションでは、ゾーンクラスタ間で高可用性ローカルファイルシステムディレクトリを共有するための要件および手順について説明します。

ゾーンクラスタに対して高可用性ローカルファイルシステムディレクトリを共有するための構成要件

グローバルクラスタリソースによって管理されている高可用性ローカルファイルシステムのディレクトリは、ゾーンクラスタに対して共有できます。高可用性ローカル

ファイルシステムディレクトリを共有するには、構成が次の要件を満たしている必要があります。

- 共有されるディレクトリが属するファイルシステムを含むグローバルクラスタのフェイルオーバーリソースグループ内に `HAStoragePlus` リソースを作成します。
- 共有する高可用性ローカルファイルシステムのディレクトリは、`lofs` ファイルシステムとしてゾーンクラスタに対して構成されている必要があります。
- `lofs` ファイルシステムを含むゾーンクラスタのフェイルオーバーリソースグループ内に `HAStoragePlus` リソースを作成します。
- ゾーンクラスタリソースは、グローバルクラスタリソースに対するオフライン再起動依存関係を持っている必要があります。
- ゾーンクラスタリソースのリソースグループは、グローバルクラスタリソースのリソースグループに対する強い肯定的なアフィニティーまた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を持っている必要があります。

注記 - 高可用性ローカルファイルシステムを共有しているアプリケーションは、アプリケーションの共用関係による可用性への影響を受けます。ノード上のアプリケーション障害やそのフェイルオーバーの試みがほかのアプリケーションに連鎖的に影響する可能性があり、アプリケーションが強制的に別のノードにフェイルオーバーされます。ファイルシステムを共有しているアプリケーションの数を減らすことによって、この問題を軽減してください。共有されているファイルシステムが UFS である場合は、ゾーンクラスタに対してクラスタファイルシステムを構成することを選択できます。147 ページの「UFS ファイルシステムを使用してクラスタファイルシステムの `HAStoragePlus` リソースを設定する方法」を参照してください。

▼ ゾーンクラスタに対して高可用性ローカルファイルシステムディレクトリを共有するように HAStoragePlus リソースタイプを設定する方法

次の手順では、*zone-cluster-name* という名前のゾーンクラスタに対して高可用性ローカルファイルシステム (UFS、QFS など) または ZFS プールディレクトリを共有するように `HAStoragePlus` リソースタイプを設定する方法について説明します。

1. **グローバルクラスタ内のいずれかのノードで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**
   ゾーンクラスタからグローバルクラスタへの依存関係やアフィニティーは、承認されたクラスタノード管理者しか設定できないため、この手順はグローバルクラスタ内のノードから実行します。

2. **グローバルクラスタ内にフェイルオーバーリソースグループを作成します。**

   ```
   # clresourcegroup create gc-hasp-resource-group
   ```

3. グローバルクラスタで **HAStoragePlus** リソースタイプを登録します。

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

4. ゾーンクラスタに対して共有するディレクトリが含まれている高可用性ローカルファイルシステムを含むグローバルクラスタのフェイルオーバーリソースグループ内に **HAStoragePlus** リソースを作成します。

```
# clresource create -g gc-hasp-resource-group -t HAStoragePlus \
-p FilesystemMountPoints=mount-point \
-p Zpools=pool gc-hasp-resource
```

5. グローバルクラスタ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 gc-hasp-resource-group
```

6. **lofs** ファイルシステムとしてゾーンクラスタに対して共有される高可用性ローカルファイルシステムのディレクトリを構成します。

```
# clzonecluster configure zoneclustername
clzc:zoneclustername> add fs
clzc:zoneclustername:fs> set dir = shared-dir-mount-point-in-zc
clzc:zoneclustername:fs> set special = shared-directory
clzc:zoneclustername:fs> set type = lofs
clzc:zoneclustername:fs> end
clzc:zoneclustername> exit
#
```

7. グローバルクラスタのフェイルオーバーリソースグループに対する強い肯定的なアフィニティーまた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を持つゾーンクラスタ内にフェイルオーバーリソースグループを作成します。

```
# clresourcegroup create -Z zoneclustername \
-p RG_affinities=++global:gc-hasp-resource-group \
zc-hasp-resource-group
OR
# clresourcegroup create -Z zoneclustername \
-p RG_affinities=+++global:gc-hasp-resource-group zc-hasp-resource-group
```

8. ゾーンクラスタで **HAStoragePlus** リソースタイプを登録します。

```
# clresourcetype register -Z zoneclustername SUNW.HAStoragePlus
```

9. ゾーンクラスタのフェイルオーバーリソースグループ内に **HAStoragePlus** リソースを作成します。ゾーンクラスタに対して共有するグローバルクラスタリソースに対する依存関係を持つ共有ディレクトリの **lofs** ファイルシステムを含むゾーンクラスタを構成します。

```
 # clresource create -Z zoneclustername -t SUNW.HAStoragePlus -g zc-hasp-resource-group \
-p FilesystemMountPoints=shared-dir-mount-point-in-zc \
-p Resource_dependencies_offline_restart=global:gc-hasp-resource zc-hasp-resource
```

10. ゾーンクラスタの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 -Z zoneclustername -M zc-hasp-resource-group
```

例 53 ゾーンクラスタに対して UFS 高可用性ローカルファイルシステムディレクトリを共有するための HAStoragePlus リソースタイプの設定

次の例は、*sczone* と呼ばれるゾーンクラスタに対して UFS 高可用性ローカルファイルシステム (/local/fs) の /local/fs/home ディレクトリを共有する方法を示しています。

```
# clresourcegroup create gc-hasp-rg
# clresourcetype register -Z sczone SUNW.HAStoragePlus
# vi /etc/vfstab /dev/md/dg1/dsk/d0 /dev/md/dg1/rdsk/d0 /local/fs ufs 2 no logging
# clresource create -g gc-hasp-rg -t SUNW.HAStoragePlus \
-p FilesystemMountPoints=/local/fs gc-hasp-rs
# clresourcegroup online -M gc-hasp-rg
```

上の手順により、グローバルクラスタで実行されている *gc-hasp-rs* リソースが、高可用性ローカルファイルシステム /local/fs を確実に管理するようになります。

```
# clzonecluster configure sczone
clzc:sczone> add fs
clzc:sczone:fs> set dir = /share/local/fs/home
clzc:sczone:fs> set special = /local/fs/home
clzc:sczone:fs> set type = lofs
clzc:sczone:fs> end
clzc:sczone> exit
```

上の構成により、高可用性ローカルファイルシステムのディレクトリ */local/fs/home* がゾーンクラスタ *sczone* のマウントポイント */share/local/fs/home* で使用可能になります。

```
# clresourcegroup create -Z sczone \
-p RG_affinities=++global:gc-hasp-rg zc-hasp-rg
# clresourcetype register -Z sczone SUNW.HAStoragePlus
# clresource create -Z sczone -t HAStoragePlus -g zc-hasp-rg \
-p FilesystemMountPoints=/share/local/fs/home \
-p Resource_dependencies_offline_restart=global:gc-hasp-rs zc-hasp-rs
# clresourcegroup online -Z sczone -M zc-hasp-rg
```

上の手順は、共有ディレクトリを lofs ファイルシステムとして管理するゾーンクラスタリソースを作成します。この例のステップは、QFS ファイルシステムに適用できます。

例 54 ゾーンクラスタに対して ZFS プールディレクトリを共有するための HAStoragePlus リソースタイプの設定

次の例は、*sczone* と呼ばれるゾーンクラスタに対して ZFS プールの「tank」ディレクトリ */tank/home* を共有する方法を示しています。

```
# clresourcegroup create gc-hasp-rg
# clresourcetype register SUNW.HAStoragePlus
# clresource create -g gc-hasp-rg -t SUNW.HAStoragePlus \
-p Zpools=tank gc-hasp-rs
# clresourcegroup online -M gc-hasp-rg
```

上の手順により、ZFS 高可用性ローカルファイルシステムが、グローバルクラスタで実行されている *gc-hasp-rs* によって確実に管理されるようになります。

```
# clzonecluster configure sczone
```

```
clzc:sczone> add fs
clzc:sczone:fs> set dir = /share/tank/home
clzc:sczone:fs> set special = /tank/home
clzc:sczone:fs> set type = lofs
clzc:sczone:fs>end
clzc:sczone> exit
#
```

上の構成により、ZFS プールの「tank」ディレクトリ */tank/home* がゾーンクラスタ *sczone* のマウントポイント */share/tank/home* で使用可能になります。

```
 # clresourcegroup create -Z sczone \
-p RG_affinities=++global:gc-hasp-rg zc-hasp-rg
# clresourcetype register -Z sczone SUNW.HAStoragePlus
# clresource create -Z sczone -t HAStoragePlus -g zc-hasp-rg \
-p FilesystemMountPoints=/share/tank/home \
-p Resource_dependencies_offline_restart=global:gc-hasp-rs zc-hasp-rs
# clresourcegroup online -Z sczone -M zc-hasp-rg
```

上の手順は、共有ディレクトリを `lofs` ファイルシステムとして管理するゾーンクラスタリソースを作成します。

高可用性ローカルファイルシステムのリソースのオンラインでの変更

高可用性ローカルファイルシステムを表すリソースを変更している間、そのファイルシステムを使用可能な状態に維持することが必要になる場合があります。たとえば、ストレージが動的にプロビジョニングされているために、ファイルシステムを使用可能な状態に維持することが必要になる場合があります。この状況では、高可用性ローカルファイルシステムを表すリソースを、そのリソースがオンラインである間に変更します。

Oracle Solaris Cluster 環境では、高可用性ローカルファイルシステムは `HAStoragePlus` リソースによって表されます。Oracle Solaris Cluster では、オンライン `HAStoragePlus` リソースを次のように変更できます。

- `HAStoragePlus` リソースへのファイルシステムの追加
- `HAStoragePlus` リソースからのファイルシステムの削除

Oracle Solaris Cluster ソフトウェアでは、ファイルシステムがオンラインである間に、そのファイルシステムの名前を変更することはできません。

注記 - ゾーンクラスタの `HAStoragePlus` リソースで構成されているファイルシステムを削除する場合は、そのゾーンクラスタからもファイルシステム構成を削除する必要があります。ゾーンクラスタからファイルシステムを削除する方法については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「ゾーンクラスタからファイルシステムを削除する」を参照してください。

このセクションでは次の情報を提供します。

▼ オンライン HAStoragePlus リソースに ZFS 以外のファイルシステムを追加する方法

HAStoragePlus リソースにローカルファイルシステムまたはクラスタファイルシステムを追加すると、HAStoragePlus リソースは、そのファイルシステムを自動的にマウントします。

1. **クラスタのいずれかのノードで、RBAC の承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **クラスタの各ノード上の `/etc/vfstab` ファイル内に、追加する各ファイルシステムのマウントポイントに対する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、「mount options」フィールドを global オプションを含むように設定します。

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*

ファイルシステムの追加先となる HAStoragePlus リソースを指定します。

-G *hasp-rg*

HAStoragePlus リソースを含むリソースグループを指定します。

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

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

HAStoragePlus リソースがすでに管理しているファイルシステムのマウントポイントと、追加するファイルシステムのマウントポイントのコンマ区切りリストを指定します。このリスト内の各エントリの形式は *LocalZonePath*:*GlobalZonePath* です。この形式で、グローバルパスはオプションです。グローバルパスが指定されていない場合、グローバルパスはローカルパスと同じです。

hasp-resource

ファイルシステムの追加先となる HAStoragePlus リソースを指定します。

5. **HAStoragePlus リソースのマウントポイントリストと、ステップ 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*

ファイルシステムの追加先となる HAStoragePlus リソースを指定します。

-G *hasp-rg*

HAStoragePlus リソースを含むリソースグループを指定します。

6. **HAStoragePlus リソースがオンラインで、かつ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、かつ障害が発生した場合、リソースの検証には成功しましたが、HAStoragePlus によるファイルシステムのマウントの試みが失敗しました。

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

例 55 オンライン HAStoragePlus リソースへのファイルシステムの追加

この例では、オンライン HAStoragePlus リソースにファイルシステムを追加する方法を示します。

- HAStoragePlus リソースは rshasp という名前であり、リソースグループ rghasp に含まれています。
- rshasp という名前の HAStoragePlus リソースは、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すでに管理しています。
- 追加され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、追加されるファイルシステムに対するエントリが各クラスタノード上の /etc/vfstab ファイルにすでに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 HAStoragePlus リソースから ZFS 以外のファイルシステムを削除する方法

HAStoragePlus リソースからファイルシステムを削除する場合、HAStoragePlus リソースは、ローカルファイルシステムをクラスタファイルシステムまたはグローバルファイルシステムとは異なる方法で処理します。

- HAStoragePlus リソースは、オフラインになるときに、ローカルファイルシステムを自動的にアンマウントします。

- HAStoragePlus リソースは、オフラインになるときに、クラスタファイルシステムまたはグローバル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 HAStoragePlus リソースからファイルシステムを削除する前に、そのファイルシステムがどのアプリケーションからも使用されていないことを確認してください。オンライン HAStoragePlus リソースからファイルシステムを削除すると、そのファイルシステムが強制的にアンマウントされることがあります。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た場合、そのアプリケーションは失敗するか、またはハングアップします。

1. **クラスタのいずれかのノードで、承認 `solaris.cluster.modify` を提供する `root` 役割になります。**

2. **HAStoragePlus リソースがすでに管理しているファイルシステムのマウントポイントのリストを取得します。**

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

`-R` *hasp-resource*

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

`-G` *hasp-rg*

HAStoragePlus リソースを含むリソースグループを指定します。

3. **HAStoragePlus リソースの `FileSystemMountPoints` 拡張プロパティーを、HAStoragePlus リソース内に残るファイルシステムのマウントポイントのみを含むように変更します。**

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

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

HAStoragePlus リソース内に残るファイルシステムのマウントポイントのコンマ区切りリストを指定します。このリストに、削除するファイルシステムのマウントポイントが含まれていてはいけません。

hasp-resource

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

4. **HAStoragePlus リソースのマウントポイントリストと、ステップ 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 HAStoragePlus リソースを指定します。

`-G` *hasp-rg*

HAStoragePlus リソースを含むリソースグループを指定します。

5. HAStoragePlus リソースがオンラインで、かつ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、かつ障害が発生した場合、リソースの検証には成功しましたが、HAStoragePlus によるファイルシステムのアンマウントの試みが失敗しました。

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

6. (オプション) クラスタの各ノード上の /etc/vfstab ファイルから、削除する各ファイルシステムのマウントポイントに対するエントリを削除します。

例 56 オンライン HAStoragePlus リソースからのファイルシステムの削除

この例では、オンライン HAStoragePlus リソースからファイルシステムを削除する方法を示します。

- HAStoragePlus リソースは rshasp という名前であり、リソースグループ rghasp に含まれています。
- rshasp という名前の HAStoragePlus リソースは、次のマウントポイントを持つファイルシステムをすでに管理しています。
  - /global/global-fs/fs
  - /global/local-fs/fs
- 削除され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
```

▼ オンライン HAStoragePlus リソースに ZFS ストレージプールを追加する方法

オンライン HAStoragePlus リソースに ZFS ストレージプールを追加すると、HAStoragePlus リソースは次の操作を実行します。

- 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。

注意 - クラスタによってすでに管理されているプールを手動でインポートすることを予定している場合は、そのプールが複数のノードにインポートされないようにしてください。複数のノードにプールをインポートすると、問題が発生する場合があります。

HAStoragePlus リソースを含むクラスタによって管理されている ZFS プールの構成を変更する場合は、180 ページの「HAStoragePlus リソースによって管理されている ZFS プール構成の変更」を参照してください。

1. **クラスタ内のいずれかのノードで、承認 solaris.cluster.modify を提供する root 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確認します。**

```
# 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となる 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と、追加する新しい ZFS ストレージプール名のコンマ区切りリストを指定します。

hasp-resource

ZFS ストレージプールの追加先となる HAStoragePlus リソースを指定します。

4. **HAStoragePlus リソースが管理している ZFS ストレージプールの新しいリストを ステップ 2 で生成したリストと比較します。**

```
# 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 ストレージプールを追加した先の HAStoragePlus リソースを指定します。

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

HAStoragePlus リソースがオンラインであるが、障害が発生した場合、リソースの検証には成功しました。ただし、HAStoragePlus リソースによる ZFS ファイルシステムのインポートとマウントの試みが失敗しました。この場合は、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 HAStoragePlus リソースから ZFS ストレージプールを削除する方法

オンライン HAStoragePlus リソースから ZFS ストレージプールを削除すると、HAStoragePlus リソースは次の操作を実行します。

- 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します。
- ノードから ZFS ストレージプールをエクスポートします。

1. **クラスタ内のいずれかのノードで、承認 solaris.cluster.modify を提供する root 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確認します。**

```
# 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削除する HAStoragePlus リソースを指定します。

3. **HAStoragePlus 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*"

HAStoragePlus リソースが現在管理している ZFS ストレージプール名から、削除する ZFS ストレージプール名を除いた内容のコンマ区切りリストを指定します。

hasp-resource

ZFS ストレージプールを削除する HAStoragePlus リソースを指定します。

4. **HAStoragePlus リソースが現在管理している ZFS ストレージプールの新しいリストを ステップ 2 で生成したリストと比較します。**

```
# 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 ストレージプールを削除した元の HAStoragePlus リソースを指定します。

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

HAStoragePlus リソースがオンラインであるが、障害が発生した場合、リソースの検証には成功しました。ただし、HAStoragePlus リソースによる ZFS ファイルシステムのアンマウントとエクスポートの試みが失敗しました。この場合は、前の一連の手順を繰り返す必要があります。

```
# clresource status -t SUNW.HAStoragePlus +
```

HAStoragePlus リソースによって管理されている ZFS プール構成の変更

HAStoragePlus リソースによって管理されている ZFS プール構成を変更するには、そのプールが複数のノードにインポートされることのないようにする必要があります。複数のノードへのインポートを実行すると、重大な結果を招く場合があり、また ZFS プールが破損する可能性もあります。

次の手順は、プール構成の変更を実行するときに複数のインポートを回避するのに役立ちます。

- 181 ページの「オフラインの HAStoragePlus リソースによって管理される ZFS プール構成を変更する方法」

- 182 ページの「オンラインの HAStoragePlus リソースによって管理される ZFS プール構成を変更する方法」

▼ オフラインの HAStoragePlus リソースによって管理される ZFS プール構成を変更する方法

1. **構成の変更が必要な ZFS プールが、どのノードにもインポートされていないことを確認します。**

   ```
   # zpool list zfs-pool-name
   ```

   このコマンドを、この ZFS プールに物理的に接続されているすべてのクラスタノード上で実行します。

2. **この ZFS プールに物理的に接続されているクラスタノード上で、`force` オプションを使用せずに代替ルートにプールをインポートします。**

   ```
   # zpool import -R zfs-pool-name
   ```

   インポートが成功した場合は、ステップ 3 に進みます。インポートが失敗した場合は、以前にこのプールにアクセスしたクラスタノードが、プールをエクスポートせずにシャットダウンしている可能性があります。下のサブステップに従って、クラスタノードがこの ZFS プールを使用していないことを確認してから、このプールを強制的にインポートします。

   a. **次のようなエラーメッセージのためにインポートが失敗したかどうかを確認します。その場合は、ステップ 2b およびステップ 2c に進みます。**

      ```
      Cannot import 'zfs-pool-name': pool may be in use from other system, it was last
      accessed by hostname (hostid: hostid) on accessed-date.
      ```

   b. **このプールが、ここに最後にアクセスしたマシン上で使用されていないことを確認します。**

      ```
      hostname# zpool list zfs-pool-name
      ```

   c. **この ZFS プールがそのノード上で使用されていない場合は、プールを強制的にインポートします。**

      ```
      # zpool import -f zfs-pool-name
      ```

3. **ZFS プール構成の変更を実行します。**

4. **この ZFS プールをエクスポートし、プールが使用されていないことを確認します。**

   ```
   # zpool export zfs-pool-name
   # zpool list zfs-pool-name
   ```

▼ オンラインの HAStoragePlus リソースによって管理される ZFS プール構成を変更する方法

1. **この ZFS プールがインポートされているクラスタノードを見つけます。**
   これは HAStoragePlus リソースがオンラインであるノードです。

   ```
   # clresource show hasp-rs-managing-pool

   === Cluster Resources ===
   Resource Name           Node Name       Status        Message
   -------------           ----------      -------       --------
   hasp-rs-managing-pool   phys-node-1     Offline       Offline
                           phys-node-2     Online        Online

   phys-node-2# zpool list zfs-pool-name
   ```

2. **ZFS プール構成の変更を実行します。**

▼ HAStoragePlus リソースの FileSystemMountPoints プロパティー変更後の障害から復旧する方法

FileSystemMountPoints 拡張プロパティーの変更中に障害が発生した場合、HAStoragePlus リソースのステータスはオンラインおよび障害発生です。障害が修正されたあと、HAStoragePlus リソースのステータスはオンラインです。

1. **変更の試みが失敗する原因となった障害を特定します。**

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

   障害のある HAStoragePlus リソースのステータスメッセージは障害を示します。考えられる障害は次のとおりです。

   - ファイルシステムが格納されているはずのデバイスが存在しない。
   - fsck コマンドによるファイルシステムの修復の試みが失敗した。
   - 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ない。
   - 追加しようとしたファイルシステムをマウントできない。
   - 削除しようとしたファイルシステムをアンマウントできない。

2. **変更の試みが失敗する原因となった障害を修正します。**

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

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

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

高可用性ローカルファイルシステムを変更しようとして失敗したときに指定したマウントポイントのコンマ区切りリストを指定します。

hasp-resource

変更する HAStoragePlus リソースを指定します。

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

```
# clresource status
```

例 57 障害のある HAStoragePlus リソースのステータス

この例では、障害のある HAStoragePlus リソースのステータスを示します。このリソースに障害があるのは、fsck コマンドによるファイルシステムの修復の試みが失敗したためです。

```
# 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 プロパティー変更後の障害から復旧する方法

Zpools 拡張プロパティーの変更中に障害が発生した場合、HAStoragePlus リソースのステータスはオンラインおよび障害発生です。障害が修正されたあと、HAStoragePlus リソースのステータスはオンラインです。

1. 変更の試みが失敗する原因となった障害を特定します。

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

障害のある HAStoragePlus リソースのステータスメッセージは障害を示します。考えられる障害は次のとおりです。

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

注記 - 破損した ZFS プールをインポートする場合の最適なオプションは、`Continue` を選択してエラーメッセージを表示することです。その他の選択には、`Wait` (成功またはノードパニックの発生までハングアップする) または `Panic` (ノードでパニックが発生する) があります。

2. **変更の試みが失敗する原因となった障害を修正します。**

3. **`HAStoragePlus` リソースの `Zpools` 拡張プロパティー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から、削除する ZFS ストレージプール名を除いた内容のコンマ区切りリストを指定します。

hasp-resource

変更する `HAStoragePlus` リソースを指定します。

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

```
# clresource status
```

例 58 障害のある `HAStoragePlus` リソースのステータス

この例では、障害のある `HAStoragePlus` リソースのステータスを示します。このリソースに障害があるのは、ZFS プール *zpool* がインポートに失敗したためです。

```
# 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 リソースのクラスタファイルシステムの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クラスタ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 リソースのクラスタ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方法

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. **リソースごと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. **クラスタ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。**

   この手順を、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 -eM** *resource-group*

HAStoragePlus リソース内の ZFS ストレージプールのデータセットのアクセシビリティーをフェイルオーバーとグローバルの間で変更する

ZFS ストレージプールが HAStoragePlus リソースによって管理されている場合は、プールのデータセットのアクセシビリティーをフェイルオーバーとグローバルの間で変更できます。

▼ HAStoragePlus リソース内のプールのデータセットのアクセシビリティーをフェイルオーバーからグローバルに変更する方法

1. **zpool に依存しているリソースを無効にします。**

   この手順によって zpool がエクスポートされます。

   ```
   # clrs disable <resource-list>
   ```

2. **フェイルオーバー ZFS プールを HAStoragePlus リソースから削除し、グローバル zpool デバイスグループが存在する場合はそれを削除します。**

   この操作によって zpool がエクスポートされます。

   ```
   # clresource set -p GlobalZpools-=zpool  resource
   # cldg list zpool
   ```

   グローバル zpool デバイスグループが存在する場合は、それを削除します。

   ```
   # cldg delete zpool
   ```

3. **HAStoragePlus リソースを変更して、グローバルアクセス用としてプールを追加します。**

   この操作によって、ZPOOL タイプのクラスタデバイスグループが zpool の名前で作成されます。

   ```
   # clresource set -p GlobalZpools+=zpool  resource
   ```

4. **この手順のステップ 1 で無効にしたリソースを有効にします。**

   ```
   # clrs enable <resource-list>
   ```

▼ HAStoragePlus リソース内のプールのデータセットのアクセシビリティーをグローバルからフェイルオーバーに変更する方法

1. **グローバルからフェイルオーバーに変更しようとしている zpool の HAStoragePlus リソースが含まれているリソースグループの rg_mode を確認します。**

   Oracle Solaris Cluster では、スケーラブルリソースグループの HAStoragePlus リソースにフェイルオーバー zpool を追加することは許可されません。

   ```
   # clrg show rg_mode resource-group
   ```

   - **rg_mode が Failover の場合は、ステップ 2 に進みます。**

- **rg_mode が Scalable の場合は、次のステップを実行します。**

  a. **zpool をフェイルオーバープールにしたあとで、zpool をホストするために既存のフェイルオーバーリソースグループを見つけるか、フェイルオーバーリソースグループを作成します。**

  ```
  # clrg create resource-group
  ```

2. **zpool に依存しているリソースを無効にします。**
   この手順によって zpool がエクスポートされます。

   ```
   # clrs disable resource-list
   ```

3. **フェイルオーバー ZFS プールを HAStoragePlus リソースから削除します。**
   この操作によって zpool がエクスポートされます。

   ```
   # clresource set -p GlobalZpools-=zpool resource
   ```

4. **ステップ 1 で rg_mode が Scalable であった場合、ステップ 1a で作成または特定したリソースグループに移動し、そのリソースグループ内の HAStoragePlus リソースに zpool を追加します。または、そのリソースグループ内に新しい HAStoragePlus リソースを作成したあと、この zpool に依存しているリソースからこの HAStoragePlus リソースへの依存関係を設定します。**
   これらのステップは、新しい HAStoragePlus リソースの作成、および resource_dependencies_offline_restart 依存関係の設定方法を示しています。使用する構成に必要なステップを実行してください。

   ```
   # clrs create -t HAStoragePlus -p zpools=zpool -g resource-group resource
   # clrg online -eM resource-group
   # clrs set -p Resource_dependencies_offline_restart+=resource resource-list
   ```

5. **ステップ 1a で新しい HAStoragePlus リソースを作成しなかった場合は、ステップ 3 で変更した HAStoragePlus リソースを変更します。**

   ```
   # clresource set -p Zpools+=zpool resource
   ```

6. **この手順のステップ 1 で無効にしたリソースを有効にします。**

   ```
   # clrs enable resource-list
   ```

HAStoragePlus リソースタイプのアップグレード

HAStoragePlus リソースタイプでは、高可用性ローカルファイルシステムをオンラインで変更できます。次のリストにあるすべての条件に当てはまる場合は、HAStoragePlus 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 Oracle Solaris Cluster からアップグレードしようとしている。
- HAStoragePlus リソースタイプの新機能を使用する必要がある。

リソースタイプをアップグレードする方法について説明する一般的な手順については、44 ページの「リソースタイプのアップグレード」を参照してください。HAStoragePlus リソースタイプのアップグレードを完了するために必要な情報は、以降のサブセクションに記載されています。

リソースタイプの新しいバージョンを登録するための情報

登録されているリソースタイプのバージョンを確認するには、次のリストにあるコマンドを使用します。

- cluster show コマンドは、クラスタのリソースタイプの名前とバージョンを一覧表示します。
- clresourcetype list -v コマンドは、各リソースタイプのノードリストを一覧表示します。

このリソースタイプに対する 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のインスタンスを移行するための情報

Oracle Solaris Cluster 4.4 には、SUNW.HAStoragePlus リソースタイプのバージョン 11 が含まれています。既存のクラスタを以前の Oracle Solaris Cluster リリースから Oracle Solaris Cluster 4.4 に更新する場合は、187 ページの「HAStoragePlus リソースタイプのアップグレード」の手順に従って、リソースをそのリソースタイプの最新バージョンに移行してください。

HAStoragePlus リソースタイプのインスタンスを移行するには、次の情報を使用します。

- HAStoragePlus の現在のバージョンが 10 以下の場合、バージョン 11 への移行は、リソースの障害モニターが無効になっているときに実行できます。
- clresource unmonitor コマンドを使用して障害モニターを無効にし、移行が完了したら clresource monitor コマンドを使用して再度有効にします。

- HAStoragePlus の現在のバージョンを確認するには、clresourcetype list HAStoragePlus コマンドを実行し、バージョン接尾辞を探します (たとえば、SUNW.HAStoragePlus:11 はバージョン 11)。

HAStoragePlus リソースに依存しているすべてのアプリケーションリソースについて、アプリケーションリソースの resource_dependencies_offline_restart プロパティーが、依存する HAStoragePlus リソースに設定されていることを確認してください。HAStoragePlus リソースタイプのバージョン 11 の場合、これは必須です。リソースタイプの以前のバージョンでは、resource_dependencies プロパティーまたは resource_dependencies_offline_restart プロパティーを使用できました。

たとえば、既存のアプリケーションリソース app-rs の resource_dependencies が HAStoragePlus リソース hasp-rs に設定されている場合、次のコマンドがこの依存関係を offline-restart 依存関係に変更します。

```
# clresource set -p resource_dependencies-=hasp-rs resource_dependencies_offline_restart+=hasp-rs
 app-rs
```

♦ ♦ ♦ 第 4 章

ロードバランシングの管理

この章では、Oracle Solaris Cluster の保守コマンドを使用して、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方法について説明します。ほかのツールを使用して手順を完了できるかどうかを確認するには、32 ページの「データサービスリソースの管理のためのツール」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「Oracle Solaris Cluster データサービスの計画」および『Concepts for Oracle Solaris Cluster 4.4』を参照してください。

この章には次のセクションがあります。

リソースグループの負荷を分散するためのタスクの概要

次の表に、Oracle Solaris 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タスクの要約を示します。この表にはまた、タスクを実行するための詳細な手順への相互参照も示されています。

表 6 データサービスリソースを管理するためのタスク

タスク	参照先
オンラインリソースグループをクラスタノード間で分散させる	192 ページの「オンラインリソースグループのクラスタノード間での分散」
リソースグループの負荷をクラスタノード間で分散させる	203 ページの「ノード間でのリソースグループの負荷分散の構成」

注記 - この章の手順では、Oracle Solaris Cluster の保守コマンドを使用してこれらのタスクを完了する方法について説明します。また、その他のツールを使用してリソースを管理することもできます。これらのオプションの詳細は、32 ページの「データサービスリソースの管理のためのツール」を参照してください。

オンラインリソースグループのクラスタノード間での分散

可用性の最大化またはパフォーマンスの最適化のために、サービスの一部の組み合わせでは、オンラインリソースグループのクラスタノード間での特定の分散が必要になります。オンラインリソースグループを分散させるには、次の目的のためにリソースグループ間のアフィニティーを作成します。

- リソースグループが最初にオンラインになったときに必要な分散を適用する
- 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試みたあとに必要な分散を保持する

このセクションでは、リソースグループのアフィニティーを使用してクラスタノード間でオンラインリソースグループを分散させる方法を示す、次の例について説明します。

- リソースグループの別のリソースグループとの共用関係の適用
- リソースグループの別のリソースグループとの優先共用関係の指定
- リソースグループのセットの負荷を分散させる
- クリティカルサービスが優先されることを指定する
- リソースグループのフェイルオーバーまたはスイッチオーバーを委託する
- リソースグループ間のアフィニティーを組み合わせてより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によって、各リソースグループをどのノード上で同時にオンラインにできるかが制限されます。各アフィニティーでは、ソースリソースグループによって、1 つまたは複数のターゲットリソースグループに対するアフィニティーが宣言されます。リソースグループ間の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 1 つまたは複数のターゲットリソースグループ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。このリストには 1 つのアフィニティーまたは複数のアフィニティーを指定できます。

リスト内の各アフィニティーを次のように指定します。

operator target-rg

注記 - *operator* と *target-rg* の間にはスペースを含めないでください。

operator

作成するアフィニティーのタイプを指定します。詳細は、表7を参照してください。

target-rg

作成す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 7 リソースグループ間のアフィニティーのタイプ

演算子	アフィニティーのタイプ	効果
+	弱い肯定的	可能な場合、ソースは、ターゲットがオンラインまたは起動中である 1 つまたは複数のノード上でオンラインになります。ただし、ソースとターゲットは、異なるノード上でオンラインになることを許可されます。
++	強い肯定的	ソースは、ターゲットがオンラインまたは起動中である 1 つまたは複数のノード上でのみオンラインになります。ソースとターゲットは、異なるノード上でオンラインになることを許可されません。
-	弱い否定的	可能な場合、ソースは、ターゲットがオンラインまたは起動中ではない 1 つまたは複数のノード上でオンラインになります。ただし、ソースとターゲットは、同じノード上でオンラインになることを許可されます。
--	強い否定的	ソースは、ターゲットがオンラインではない 1 つまたは複数のノード上でのみオンラインになります。ソースとターゲットは、同じノード上でオンラインになることを許可されません。
+++	フェイルオーバー委託付きの強い肯定的	ソースによるフェイルオーバーの試みがターゲットに委託される点を除き、強い肯定的と同じです。詳細は、199 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の委託」を参照してください。

弱いアフィニティーは、`Nodelist` の優先順序よりも優先されます。

ほかのリソースグループの現在の状態によって、どのノード上でも強いアフィニティーを満たすことができないことがあります。この状況では、その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になります。ほかのリソースグループの状態が変化して強いアフィニティーを満たすことができるようになった場合、その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。

注記 - ソースリソースグループ上で、複数のターゲットリソースグループに対して強いアフィニティーを宣言する場合は注意してください。宣言されているすべての強いアフィニティーを満たすことができない場合、そのソースリソースグループはオフラインのままになります。

リソースグループの別のリソースグループとの共用関係の適用

あるリソースグループによって表されるサービスが 2 つ目のリソースグループ内のサービスに強く依存しすぎているために、両方のサービスを同じノード上で実行しなければならないことがあります。たとえば、互いに依存する複数のサービスデーモンで構成されているアプリケーションでは、すべてデーモンを同じノード上で実行することが必要な場合があります。

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がもう一方のサービスのリソースグループと共用関係を持つように強制します。リソースグループの別のリソースグループとの共用関係を適用するには、そのリソースグループ上で、ほかのリソースグループに対する強い肯定的なアフィニティーを宣言します。

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

source-rg

強い肯定的なアフィニティーのソースであるリソースグループを指定します。このリソースグループは、別のリソースグループに対する強い肯定的なアフィニティーを宣言している場所の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ーのターゲットであるリソースグループを指定します。このリソースグループは、強い肯定的なアフィニティーを宣言している相手のリソースグループです。

リソースグループは、強い肯定的なアフィニティーを持っている相手のリソースグループに追従します。ターゲットリソースグループが別のノードに再配置された場合、ソースリソースグループは、ターゲットと同じノードに自動的に切り替えられます。ただし、強い肯定的なアフィニティーを宣言しているリソースグループが、そのアフィニティーのターゲットがまだ実行されていないノードにフェイルオーバーすることは妨げられます。

注記 - 妨げられるのは、リソースモニターによって開始されたフェイルオーバーだけです。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が実行されているノードに障害が発生した場合は、両方のリソースグループが同じ動作中のノード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` が、リソースグループ `rg2` に対する強い肯定的なアフィニティーを宣言しているとします。`rg2` が別のノードにフェイルオーバーした場合は、`rg1` もそのノードにフェイルオーバーします。このフェイルオーバーは、`rg1` 内のすべてのリソースが動作中であっても実行されます。ただし、`rg1` 内のリソースが `rg1` を `rg2` が実行されていないノードにフェイルオーバーしようとした場合、この試みはブロックされます。

強い肯定的なアフィニティーのターゲットをオンラインにしたとき、その強い肯定的なアフィニティーのソースは、すべてのノード上でオフラインである可能性があります。この状況では、その強い肯定的なアフィニティーのソースは、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` が、リソースグループ `rg2` に対する強い肯定的なアフィニティーを宣言しているとします。両方のリソースグループは最初、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` をオンラインにした場合、`rg1` は同じノード上で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のために、リソースグループが自動的にオンラインになることを回避できます。

強い肯定的なアフィニティーを宣言しているリソースグループをフェイルオーバーできるようにする必要がある場合は、フェイルオーバーを委託する必要があります。詳細は、199 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の委託」を参照してください。

例 59 リソースグループの別のリソースグループとの共用関係の適用

この例では、リソースグループ `rg1` を、リソースグループ `rg2` に対する強い肯定的なアフィニティーを宣言するように変更するためのコマンドを示します。このアフィニティー関係の結果として、`rg1` は、`rg2` が実行されているノード上でのみ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すると仮定します。

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

リソースグループの別のリソースグループとの優先共用関係の指定

あるリソースグループによって表されるサービスが、2 つ目のリソースグループ内のサービスを使用することがあります。その結果、これらのサービスは、同じノード上で実行されている場合にもっとも効率的に実行されます。たとえば、データベースを使用するアプリケーションは、そのアプリケーションとデータベースが同じノード上で実行されている場合にもっとも効率的に実行されます。ただし、効率の低下の方がリソースグループのフェイルオーバーの増加より害は少ないため、各サービスを別のノード上で実行できます。

この状況では、可能であれば両方のリソースグループが共用されるように指定します。リソースグループの別のリソースグループとの優先共用関係を指定するには、そのリソースグループ上で、ほかのリソースグループに対する弱い肯定的なアフィニティーを宣言します。

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

source-rg

弱い肯定的なアフィニティーのソースであるリソースグループを指定します。このリソースグループは、別のリソースグループに対する弱い肯定的なアフィニティーを宣言している場所の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ーのターゲットであるリソースグループを指定します。このリソースグループは、弱い肯定的なアフィニティーを宣言している相手のリソースグループです。

あるリソースグループ上で別のリソースグループに対する弱い肯定的なアフィニティーを宣言することによって、両方のリソースグループが同じノード上で実行される確率が高くなります。弱い肯定的なアフィニティーのソースは、その弱い肯定的なアフィニティーのターゲットがすでに実行されているノード上で最初にオンラインになります。ただし、リソースモニターのために弱い肯定的なアフィニティーのターゲットがフェイルオーバーした場合、そのアフィニティーのソースはフェイルオーバーしません。同様に、弱い肯定的なアフィニティーのターゲットがスイッチオーバーされた場合、そのアフィニティーのソースはフェイルオーバーしません。どちらの状況でも、このソースは、すでに実行されているノード上でオンラインのままになります。

注記 - 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が実行されているノードに障害が発生した場合は、両方のリソースグループが同じ動作中のノード上で再起動されます。

例 60 リソースグループの別のリソースグループとの優先共用関係の指定

この例では、リソースグループ `rg1` を、リソースグループ `rg2` に対する弱い肯定的なアフィニティーを宣言するように変更するためのコマンドを示します。このアフィニティー関係の結果として、`rg1` と `rg2` は、同じノード上で最初にオンラインになります。ただし、`rg2` 内のリソースのために `rg2` がフェイルオーバーした場合、`rg1` は、これらのリソースグループが最初にオンラインになったノード上でオンラインのままになります。この例では、両方のリソースグループが存在すると仮定します。

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

リソースグループのセットのクラスタノード間での均等な分散

リソースグループのセット内の各リソースグループにより、クラスタに同じ負荷が課されることがあります。この状況では、リソースグループをクラスタノード間で均等に分散させることによって、クラスタ上の負荷を分散させることができます。

リソースグループのセットをクラスタノード間で均等に分散させるには、各リソースグループ上で、セット内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、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ている場所の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のコンマ区切りリストを指定します。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している相手のリソースグループです。

あるリソースグループ上で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することによって、リソースグループが常に、クラスタ内のもっとも負荷の軽いノード上でオンラインになることが保証されます。そのノード上では、もっとも少ない数のほかのリソースグループが実行されています。そのため、違反する弱い否定的なアフィニティーの数は最小です。

例 61　リソースグループのセットのクラスタノード間での均等な分散

この例では、リソースグループ `rg1`、`rg2`、`rg3`、および `rg4` を、これらのリソースグループが確実にクラスタ内の使用可能なノード間で均等に分散されるように変更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` が存在すると仮定します。

```
# 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
```

クリティカルサービスが優先されることの指定

クラスタが、ミッションクリティカルサービスと非クリティカルサービスの組み合わせを実行するように構成されていることがあります。たとえば、クリティカルな顧客サービスをサポートするデータベースが、非クリティカルな調査タスクと同じクラスタ内で実行されることがあります。

非クリティカルサービスがクリティカルサービスのパフォーマンスに影響を与えることのないようにするには、そのクリティカルサービスが優先されることを指定しま

す。クリティカルサービスが優先されることを指定することにより、非クリティカルサービスがクリティカルサービスと同じノード上で実行されることが回避されます。

すべてのノードが動作中の場合、クリティカルサービスは、非クリティカルサービスとは別のノード上で実行されます。ただし、クリティカルサービスの障害のために、非クリティカルサービスが実行されているノードにそのサービスがフェイルオーバーすることがあります。この状況では、そのノードのコンピューティングリソースがミッションクリティカルサービスに完全に専用になるようにするために、非クリティカルサービスをただちにオフラインにします。

クリティカルサービスが優先されることを指定するには、それぞれの非クリティカルサービスのリソースグループ上で、そのクリティカルサービスを含むリソースグループに対する強い否定的なアフィニティーを宣言します。

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

noncritical-rg

非クリティカルサービスを含むリソースグループを指定します。このリソースグループは、別のリソースグループに対する強い否定的なアフィニティーを宣言している場所のリソースグループです。

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

クリティカルサービスを含むリソースグループを指定します。このリソースグループは、強い否定的なアフィニティーを宣言している相手のリソースグループです。

リソースグループは、強い否定的なアフィニティーを持っている相手のリソースグループから遠ざかります。

強い否定的なアフィニティーのターゲットをオフラインにしたとき、その強い否定的なアフィニティーのソースは、すべてのノード上でオフラインである可能性があります。この状況では、その強い否定的なアフィニティーのソースは自動的にオンラインになります。リソースグループは一般に、ノードリスト内のノードの順序および宣言されているアフィニティーに基づいて、最優先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` が、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ーを宣言しているとします。リソースグループ `rg1` が最初、すべてのノード上でオフラインであるのに対して、リソースグループ `rg2` はあるノード上でオンラインです。管理者が `rg2` をオフラインにした場合、`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のために、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ことを回避できます。

例 62 クリティカルサービスが優先されることの指定

この例では、非クリティカルリソースグループ ncrg1 および ncrg2 を、これらのリソースグループよりクリティカルリソースグループ mcdbrg が確実に優先されるように変更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 mcdbrg、ncrg1、および ncrg2 が存在すると仮定します。

```
# clresourcegroup set -p RG_affinities=--mcdbrg ncrg1 ncrg2
```

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委託

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実行されていないノードに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できるようにする必要がある場合は、そのフェイルオーバーを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。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、アフィニティーのソースはそのターゲットとともに強制的にフェイルオーバーされます。

注記 - ++ 演算子で指定されてい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をスイッチオーバーしなければならないことがあります。この状況では、アフィニティーのターゲットとアフィニティーの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上で、ほか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 -p RG_affinities=+++target-rg  source-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側の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している場所の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先の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を宣言している相手のリソースグループです。

リソースグループがフェイルオーバー委託付きの強い肯定的なアフィニティーを宣言できる相手は、最大 1 つのリソースグループです。ただし、指定されたリ

ソースグループは、任意の数のほかのリソースグループによって宣言されたフェイルオーバー委託付きの強い肯定的なアフィニティーのターゲットになることができます。

フェイルオーバー委託付きの強い肯定的なアフィニティーは、完全に対称的であるわけではありません。ターゲットは、ソースがオフラインのままである間にオンラインになることができます。ただし、ターゲットがオフラインである場合、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 3 つ目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ている場合、フェイルオーバーまたはスイッチオーバーは、さらにその 3 つ目のリソースグループに委託されます。その 3 つ目のリソースグループはフェイルオーバーまたはスイッチオーバーを実行して、ほか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します。

例 63 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委託

この例では、リソースグループ `rg1` を、リソースグループ `rg2` に対するフェイルオーバー委託付きの強い肯定的なアフィニティーを宣言するように変更するためのコマンドを示します。このアフィニティー関係の結果として、`rg1` は、`rg2` にフェイルオーバーまたはスイッチオーバーを委託します。この例では、両方のリソースグループが存在すると仮定します。

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

リソースグループ間のアフィニティーの組み合わせ

複数のアフィニティーを組み合わせることによって、より複雑な動作を作成できます。たとえば、アプリケーションの状態を、関連するレプリカサーバーで記録することもできます。この例でのノード選択の要件は次のとおりです。

- レプリカサーバーは、アプリケーションとは別のノード上で実行する必要があります。
- アプリケーションがその現在のノードからフェイルオーバーする場合、そのアプリケーションは、レプリカサーバーが実行されているノードにフェイルオーバーします。
- レプリカサーバーが実行されているノードにアプリケーションがフェイルオーバーした場合、そのレプリカサーバーは別のノードにフェイルオーバーする必要があります。ほかに使用可能なノードがない場合、レプリカサーバーはオフラインになる必要があります。

これらの要件を満たすには、アプリケーションとレプリカサーバーのため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、レプリカサーバーを含むリソースグループに対する弱い肯定的なアフィニティーを宣言します。
- レプリカサーバーを含むリソースグループは、アプリケーションを含むリソースグループに対する強い否定的なアフィニティーを宣言します。

例 64 リソースグループ間のアフィニティーの組み合わせ

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

- リソースグループ `app-rg` は、状態がレプリカサーバーによって追跡されるアプリケーションを表します。
- リソースグループ `rep-rg` は、レプリカサーバーを表します。

この例では、各リソースグループはアフィニティーを次のように宣言します。

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

この例では、両方のリソースグループが存在すると仮定します。

```
# clresourcegroup set -p RG_affinities=+rep-rg app-rg
# clresourcegroup set -p RG_affinities=--app-rg rep-rg
```

ゾーンクラスタリソースグループのアフィニティー

クラスタ管理者は、ゾーンクラスタ内のリソースグループと、ゾーンクラスタ内の別のリソースグループまたはグローバルクラスタ上のリソースグループの間のアフィニティーを指定できます。

ゾーンクラスタ内のリソースグループ間のアフィニティーを指定するには、次のコマンドを使用できます。

```
# clresourcegroup set -p RG_affinities=affinity-typetarget-zc:target-rg  source-zc:source-rg
```

ゾーンクラスタ内のリソースグループのアフィニティーのタイプには、次のいずれかを指定できます。

- + (弱い肯定的)
- ++ (強い肯定的)
- +++ (フェイルオーバー委託付きの強い肯定的)
- - (弱い否定的)
- -- (強い否定的)

例 65 ゾーンクラスタ内のリソースグループ間の強い肯定的なアフィニティーの指定

この例では、ゾーンクラスタ内のリソースグループ間の強い肯定的なアフィニティーを指定するためのコマンドを示します。

ゾーンクラスタ ZC1 内のリソースグループ RG1 は、ゾーンクラスタ ZC2 内のリソースグループ RG2 に対する強い肯定的なアフィニティーを宣言します。

ゾーンクラスタ ZC1 内のリソースグループ RG1 と、別のゾーンクラスタ ZC2 内のリソースグループ RG2 の間の強い肯定的なアフィニティーを指定する必要がある場合は、次のコマンドを使用します。

```
# clresourcegroup set -p RG_affinities=++ZC2:RG2 ZC1:RG1
```

例 66 ゾーンクラスタ内のリソースグループとグローバルクラスタ内のリソースグループの間の強い否定的なアフィニティーの指定

この例では、ゾーンクラスタ内のリソースグループ間の強い否定的なアフィニティーを指定するためのコマンドを示します。ゾーンクラスタ ZC1 内のリソースグループ RG1 と、グローバルクラスタ内のリソースグループ RG2 の間の強い否定的なアフィニティーを指定する必要がある場合は、次のコマンドを使用します。

```
# clresourcegroup set -p RG_affinities=--global:RG2 ZC1:RG1
```

▼ ゾーンクラスタ内のリソースグループとグローバルクラスタ内のリソースグループの間のフェイルオーバー委託を構成する方法

この手順では、ゾーンクラスタリソースがグローバルクラスタリソースに依存する構成を設定する方法について説明します。この構成では、ゾーンクラスタリソースがオンラインになっているゾーンクラスタノードが停止すると、確実にフェイルオーバーが発生します。

1. **グローバルクラスタの 1 つのノードから、リソースグループを作成します。**

   ```
   # clrg create global_zone-rg
   ```

2. **グローバルクラスタの 1 つのノードから、ゾーンクラスタリソースが依存するリソースを作成します。**

   ```
   # clrs create -t <rt-type> -p <property>=<value> -g global_zone-rg
   global_zone-rs
   ```

3. **ゾーンクラスタの 1 つのゾーンノードから、リソースグループを作成します。**

   ```
   # clrg create zone_cluster-rg
   ```

4. **グローバルクラスタの 1 つのノードから、`rg_affinities` をゾーンクラスタ内のリソースグループに設定します。**

```
# clrg set -p RG_affinities=+++<zone-cluster-name>:zone_cluster-rg
global_zone-rg
```

5. **ゾーンクラスタの 1 つのゾーンノードから、グローバルクラスタリソースに依存するリソースを作成します。**

ここで設定する依存関係には、必要に応じて `Resource_dependencies_restart`、`Resource_dependencies_offline_restart`、`Resource_dependen` または `Resource_dependencies_weak` のいずれかを指定できます。

```
# clrs create -g zone_cluster-rg -t <rt-type> -p <property>=<value> -p
resource_dependencies_offline_restart=global:global_zone-rs zone_cluster-rs
```

この手順で示されているように、リソースの依存関係はゾーンクラスタリソースからグローバルクラスタリソースへ向かっていますが、リソースグループのアフィニティーはグローバルクラスタリソースグループからゾーンクラスタリソースグループへ向かっています。

リソースグループの 1 つが障害回復保護グループに含まれている場合、保護グループを有効にするためには、保護グループのプロパティー `External_Dependency_Allowed` を true に設定しておく必要があります。

ノード間でのリソースグループの負荷分散の構成

負荷制限を設定することにより、ノード間でのリソースグループの自動負荷分散を有効にすることができます。リソースグループに負荷係数を割り当てると、その負荷係数はノードの定義済み負荷制限に対応します。

デフォルトの動作では、リソースグループの負荷は、すべての使用可能なノードに均等に分散されます。各リソースグループはそのノードリストのノード上で起動されます。Resource Group Manager (RGM) は、構成済みの負荷分散ポリシーをもっとも満たしているノードを選択します。RGM によってリソースグループがノードに割り当てられると、各ノード上のリソースグループの負荷係数が合計され、合計負荷が算出されます。次に、合計負荷がそのノードの負荷制限と比較されます。

負荷制限は、グローバルクラスタまたはゾーンクラスタで構成できます。

各ノードでの負荷分散を制御するために設定する係数には、負荷制限、リソースグループ優先度、およびプリエンプションモードがあります。グローバルクラスタでは、`Concentrate_load` プロパティーを設定することで、負荷制限を超過しない範囲で可能な最小限のノードにリソースグループの負荷を集中させるか、使用可能なすべてのノードにできるだけ均等に負荷を分散させるか、優先する負荷分散ポリシーを選択できます。デフォルトの動作は、リソースグループの負荷を分散させます。各リ

ソースグループはまだ、負荷係数および負荷制限の設定とは関係なく、そのノードリスト内のノード上でのみ実行するように制限されています。

注記 - コマンド行、Oracle Solaris Cluster Manager インタフェース、または `clsetup` ユーティリティーを使用して、リソースグループの負荷分散を構成できます。次の手順は、`clsetup` ユーティリティーを使用して、リソースグループの負荷分散を構成する方法を示したものです。コマンド行を使用してこれらの手順を実行する方法については、『管理 Oracle Solaris Cluster 4.4 配置』 の 「負荷制限の構成」を参照してください。

ここでは、次の手順について説明します。

- 204 ページの「ノードの負荷制限を構成する方法」
- 205 ページの「リソースグループの優先度を設定する方法」
- 206 ページの「リソースグループの負荷係数を設定する方法」
- 207 ページの「リソースグループのプリエンプションモードを設定する方法」
- 208 ページの「クラスタ内の少数のノードに負荷を集中させる方法」

▼ ノードの負荷制限を構成する方法

各クラスタノードには、独自の一連の負荷制限を設定できます。リソースグループに負荷係数を割り当てると、その負荷係数はノードの定義済み負荷制限に対応します。弱い負荷制限 (超過できる) を設定することも、強い負荷制限 (超過できない) を設定することもできます。

1. **クラスタの 1 つのアクティブノード上で `root` 役割になります。**

2. **`clsetup` ユーティリティーを起動します。**

   ```
   phys-schost# clsetup
   ```

   `clsetup` メニューが表示されます。

3. **「その他のクラスタタスク」メニュー項目を選択します。**

   「ほかのクラスタタスクメニュー」が表示されます。

4. **「リソースグループの負荷分散の管理」メニュー項目を選択します。**

   「リソースグループ負荷分散管理メニュー」が表示されます。

5. **「負荷制限の管理」メニュー項目を選択します。**

   「負荷制限の管理」メニューが表示されます。

6. **yes と入力し、Return キーを押して続行します。**

7. **実行する操作に対応するオプション番号を入力します。**
   負荷制限を作成、負荷制限を変更、または負荷制限を削除できます。

8. **負荷制限の作成を選択した場合は、負荷制限を設定するノードに対応するオプション番号を選択します。**
   2 番目のノードに負荷制限を設定する場合は、2 番目のノードのオプション番号を選択します。負荷制限を構成するすべてのノードを選択したあとで、q と入力します。

9. **yes と入力して、ステップ 8 で選択したノードを確定します。**

10. **負荷制限の名前を入力します。**
    たとえば、負荷制限の名前として `mem_load` と入力します。

11. **弱い制限値の指定について yes または no を入力します。**
    yes と入力した場合は、弱い制限値を入力して Enter キーを押します。

12. **強い制限値の指定について yes または no を入力します。**
    yes と入力した場合は、強い制限値を入力して Enter キーを押します。

13. **yes と入力して、負荷制限の作成を続行します。**

14. **yes と入力して、更新を続行します。**
    「コマンドが正常に完了しました」というメッセージが、選択したノードの弱い制限値および強い制限値とともに表示されます。Return キーを押して続行します。

15. **`clsetup` ユーティリティーのプロンプトに従って、負荷制限を変更または削除できます。**
    q と入力し Return キーを押して、前のメニューに戻ります。

▼ リソースグループの優先度を設定する方法

高い優先度を持つようにリソースグループを構成すると、特定のノードから移動させられる可能性が低くなります。負荷制限を超過した場合、優先度の低いリソースグループを強制的にオフラインにすることができます。

1. **クラスタの 1 つのアクティブノード上で `root` 役割になります。**

2. **`clsetup` ユーティリティーを起動します。**

```
phys-schost# clsetup
```

clsetup メニューが表示されます。

3. **「その他のクラスタタスク」メニュー項目を選択します。**
   「ほかのクラスタタスクメニュー」が表示されます。
4. **「リソースグループの負荷分散の管理」メニュー項目を選択します。**
   「リソースグループ負荷分散管理メニュー」が表示されます。
5. **「リソースグループごとの優先順位の設定」メニュー項目を選択します。**
   「リソースグループの優先順位の設定」メニューが表示されます。
6. **yes と入力し、Return キーを押します。**
7. **リソースグループのオプションを入力します。**
   既存の優先度の値が表示されます。デフォルトの優先度の値は 500 です。
8. **新しい優先度値を入力します。**
9. **yes と入力して入力を確認します。**
10. **Return キーを押して前のメニューに戻ります。**
    「リソースグループ負荷分散管理メニュー」が表示されます。

▼ リソースグループの負荷係数を設定する方法

負荷係数とは、負荷制限に関して負荷に割り当てる値です。負荷係数はリソースグループに割り当てられ、これらの負荷係数はノードの定義済みの負荷制限に対応します。

1. **クラスタの 1 つのアクティブノード上で root 役割になります。**
2. **clsetup ユーティリティーを起動します。**

```
phys-schost# clsetup
```

clsetup メニューが表示されます。

3. **「その他のクラスタタスク」メニュー項目を選択します。**
   「ほかのクラスタタスクメニュー」が表示されます。

4. **「リソースグループの負荷分散の管理」メニュー項目を選択します。**
   「リソースグループ負荷分散管理メニュー」が表示されます。

5. **「リソースグループごとの負荷係数の設定」メニュー項目を選択します。**
   「リソースグループの負荷係数の設定」メニューが表示されます。

6. **yes と入力します。**

7. **リソースグループのオプション番号を入力します。**

8. **適切な負荷係数を入力します。**
   たとえば、`mem_load@50` と入力することにより、選択したリソースグループに対して `mem_load` という負荷係数を設定できます。終了したら、`Ctrl-D` を押します。

9. **Return キーを押して更新を開始します。**

10. **Return キーを押して前のメニューに戻ります。**
    「リソースグループ負荷分散管理メニュー」が表示されます。

▼ リソースグループのプリエンプションモードを設定する方法

`preemption_mode` プロパティーは、リソースグループが、ノードの過負荷のために、優先度の高いリソースグループによってノードから横取りされるかどうかを指定します。このプロパティーはノード間でリソースグループを移動するコストを示します。

1. **クラスタの 1 つのアクティブノード上で `root` 役割になります。**

2. **`clsetup` ユーティリティーを起動します。**

   ```
   phys-schost# clsetup
   ```

   `clsetup` メニューが表示されます。

3. **「その他のクラスタタスク」メニュー項目を選択します。**
   「ほかのクラスタタスクメニュー」が表示されます。

4. **「リソースグループの負荷分散の管理」メニュー項目を選択します。**
   「リソースグループ負荷分散管理メニュー」が表示されます。

5. **「リソースグループごとのプリエンプションモードの設定」メニュー項目を選択します。**

「リソースグループのプリエンプションモードの設定」メニューが表示されます。

6. **yes と入力します。**

7. **リソースグループのオプション番号を入力します。**
   リソースグループにプリエンプションモードが設定されている場合は、次のように表示されます。

```
The preemption mode property of "rg11" is currently set to the following: preemption mode:
 Has_Cost
```

8. **必要なプリエンプションモードのオプション番号を入力します。**
   選択肢は Has_cost、No_cost、Never の 3 つです。

9. **yes と入力して、更新を続行します。**

10. **Return キーを押して前のメニューに戻ります。**
    「リソースグループ負荷分散管理メニュー」が表示されます。

▼ クラスタ内の少数のノードに負荷を集中させる方法

Concentrate_load プロパティーを FALSE に設定すると、クラスタは、リソースグループの負荷をそのリソースグループのノードリスト内の使用可能なすべてのノード間で均等に分散します。デフォルトでは、Concentrate_load プロパティーは FALSE に設定されています。

このプロパティーを TRUE に設定すると、クラスタは構成済みのどの強い負荷制限値または弱い負荷制限値も超えることなく、リソースグループの負荷を可能なかぎり少ない数のノードに集中させようとします。

注記 - リソースグループ RG2 がリソースグループ RG1 に対する ++ または +++ アフィニティーを宣言している場合に Concentrate_load=TRUE を指定する際は、RG2 にゼロ以外の負荷係数を設定しないようにしてください。代わりに、RG1 と同じノードでオンラインになる RG2 による追加の負荷を考慮して、より大きな負荷係数を RG1 に設定します。これにより、Concentrate_load 機能が目的どおりに動作できるようになります。あるいは、RG2 に対して負荷係数を設定するが、これらの負荷係数には強い負荷制限値を設定せずに、弱い制限値のみを設定することもできます。これにより、弱い負荷制限を超えた場合であっても、RG2 をオンラインにできます。

Concentrate_load プロパティーを設定できるのはグローバルクラスタ内だけであり、ゾーンクラスタでこのプロパティーを設定することはできません。ゾーンクラスタでは、デフォルト設定は常に FALSE です。

1. **クラスタの 1 つのアクティブノード上で `root` 役割になります。**

2. **`clsetup` ユーティリティーを起動します。**

   ```
   phys-schost# clsetup
   ```

   `clsetup` メニューが表示されます。

3. **「その他のクラスタタスク」メニュー項目を選択します。**
   「ほかのクラスタタスクメニュー」が表示されます。

4. **「クラスタの concentrate_load プロパティーを設定します」メニュー項目を選択します。**
   「クラスタの負荷集中プロパティーの設定」メニューが表示されます。

5. **yes と入力します。**
   TRUE または FALSE の現在の値が表示されます。

6. **yes と入力して値を変更します。**

7. **yes と入力して、更新を続行します。**

8. **Return キーを押して前のメニューに戻ります。**
   「ほかのクラスタタスクメニュー」が表示されます。

索引

さ

た

な

は

ま

や

ら

P

R

S

T

U

V

Z