Veritas™ Cluster Server Agents for Veritas™ Volume Replicator 設定ガイド

Solaris

5.0 Maintenance Pack 3

symantec™

Veritas™ Cluster Server Agents for Veritas™ Volume Replicator 設定ガイド

このマニュアルで説明するソフトウェアは、使用許諾契約に基づいて提供され、その内容に同意する場合にのみ使用することができます。

Product Version:5.0 MP3

Documentation version:5.0 MP3.0

法定通知

Symantec Corporation
20330 Stevens Creek Blvd.
Cupertino, CA 95014

http://www.symantec.com

テクニカルサポート

ご購入先にお問い合わせください。

目次

1

VCS Agents for VVR の概要

この章では以下の項目について説明しています。

VCS Agents for VVR について

エージェントは、あらかじめ定義されたリソースタイプを管理するプロセスです。エージェントが起動されると、Veritas Cluster Server（VCS）から設定情報を入手します。その後に、周期的にリソースを監視し、VCS のリソースの状態を更新します。

一般的に、エージェントは次の事柄を実行します。

- リソースをオンライン化する
- リソースをオフライン化する
- リソースを監視し、何らかの状態が変化した場合に VCS に報告する

VCS Agents for VVR は、RVG（Replicated Volume Group）を監視し管理します。各エージェントには、リソースタイプとして定義される VCS タイプの宣言文とエージェント実行ファイルが含まれます。VCS Agents for VVR には次のエージェントがあります。

フェールオーバーアプリケーションのエージェント

- 「RVG エージェント」
- 「RVGPrimary エージェント」
- 「RVGSnapshot エージェント」

p.9 の「フェールオーバーアプリケーションのエージェントの機能」を参照してください。

並列アプリケーションのエージェント

- 「RVGShared エージェント」
- 「RVGSharedPri エージェント」
- 「RVGLogowner エージェント」

p.17 の「並列アプリケーションのエージェントの機能」を参照してください。

VCS クラスタの概念

リソース、属性およびサービスグループは、クラスタの機能に不可欠なコンポーネントです。

詳細については、『Veritas Cluster Server ユーザーズガイド』を参照してください。

リソース	リソースとは、ディスク、ボリューム、ファイルシステムマウントポイント、ネットワークインターフェースカード（NIC）、IP アドレス、アプリケーション、データベースなどのハードウェアやソフトウェアの構成要素です。各リソースが連携して、クライアント/サーバー環境で、クライアントにサービスを提供します。types.cf ファイルには付属エージェントリソースタイプの属性が定義されています。VCS 設定ファイルの main.cf には、リソースの属性値が含まれています。main.cf ファイルは、types.cf に記述されているリソースを、include 文により取り込んでいます。また、main.cf ファイルは、VVRTypes.cf ファイルに定義されている VVR リソースタイプも include 文により取り込んでいます。
属性	属性には、クラスタ、ノード、サービスグループ、リソース、リソースタイプおよびエージェントに関する情報が含まれます。ある属性にある値を設定すると、リソースは特定の方法で機能するようになります。リソースの属性値を変更することにより、VCS エージェントによるリソースの管理方法を変更します。各属性には、定義と値があります。属性を定義するには、そのデータ形式および値の種類を指定します。また、属性には、値が指定されていない場合に割り当てられるデフォルト値もあります。
サービスグループ	サービスグループは関連するリソースによって構成されています。あるサービスグループがオンライン化されると、そのグループ内のすべてのリソースがオンライン化されます。

VCS エージェントとそのリソースは、コマンドラインまたは VCS Java コンソールおよび Web コンソールから動的に設定または修正できます。main.cf ファイルは直接編集できますが、main.cf ファイルを編集する前に VCS を停止する必要があります。VCS Agents for VVR 用の main.cf ファイルの例は、/etc/VRTSvcs/conf/sample_vvr ディレクトリにあります。

詳細については、『Veritas Cluster Server ユーザーズガイド』の VCS の管理についての章を参照してください。

フェールオーバーアプリケーションのエージェントの機能

ここでは各エージェントがどのように機能するかを説明し、各エージェントに対するエントリポイント、状態の定義および属性を概説し、各エージェントの依存関係グラフについて説明します。

VCS Agents for VVR には次のエージェントがあります。

- 「RVG エージェント」
- 「RVGPrimary エージェント」
- 「RVGSnapshot エージェント」

RVG エージェント

RVG エージェントを使うと、クラスタのプライマリ VVR ノードと別のクラスタのセカンダリ VVR ノードを管理することで、クラスタ間でのレプリケーションが可能になります。そして、この VVR のサービスをそれぞれのクラスタ内で、フェールオーバーさせることができます。これにより、レプリケーションの高可用性を実現します。

メモ: RVG は、プライマリ VVR ノードからセカンダリ VVR ノードへのフェールオーバー機能を提供する RVGPrimary エージェントをサポートします。プライマリ VVR ノードで災害が発生し、プライマリクラスタのすべてのノードが使えない場合は、RVG エージェントは、プライマリ VVR ノードからセカンダリ VVR ノードにフェールオーバーしません。VCS グローバルクラスタを使用すると、プライマリの役割をプライマリ VVR ノードからセカンダリ VVR ノードにフェールオーバーできます。

RVG エージェントの主な機能を次に示します。

- VVR のプライマリおよびセカンダリノードをクラスタ化することで、単一点障害によるサービスの停止を防ぎます。
- VVR を使用する VCS 管理アプリケーションの起動プロセスを、VCS のサービスグループをオンラインにするのと同じくらい容易にします。

- クラスタ内のノードに障害が発生したとしても、クラスタ内の他のノードがレプリケーションを継続するため、更新分のデータを失うことはありません。
- RVG のリソースタイプを加えることで、VVR を任意の VCS クラスタに追加できるようにします。

ユーザー環境を作成する際に参考となるエージェント用の設定ファイルのサンプルは、次のディレクトリにあります。

/etc/VRTSvcs/conf/sample_vvr/RVG

メモ: このリリースでは、RVG エージェントの Primary、SRL、RLinks 属性をサポートしません。以前のリリースの設定がある場合は、アップグレード時にこれらの属性を削除する必要があります。削除しないと設定できません。

RVG エージェントの機能、そのエントリポイント、状態の定義は次のとおりです。

説明	RVG のオンライン化、RVG への読み込みおよび書き込みの監視、および RVG のオフライン化を実行します。これはフェールオーバーリソースです。
エントリポイント	■ online: DiskGroup エージェントが RVG をリカバリしたかどうかを確認します。リカバリしていない場合、データボリュームと SRL（Storage Replicator Log）のリカバリと開始を行い、RVG と RVG に設定されているすべての RLINK のリカバリを行い、その後に、RVG を開始します。 ■ offline: RVG を停止します。 ■ clean: RVG を停止します。 ■ info: RDS（Replicated Data Set）のレプリケーションの状態に関する情報を取得します。 ■ monitor: vxprint コマンドを使用して、RVG の状態を監視します。 **メモ:** RVG リソースは、ローカルの RVG へのアクセスのみ監視します。レプリケーション自体を監視するわけではありません。
障害の検出	RVG の状態が ENABLED/ACTIVE ではない場合、RVG リソースには障害が発生した、と判断します。
状態の定義	ONLINE - RVG が ENABLED/ACTIVE 状態にあることを示します。 OFFLINE - RVG が DISABLED/CLEAN 状態にあることを示します。

RVG エージェントの属性は次のとおりです。

表 1-1 RVG エージェントの属性

属性	データ形式と値の種類	定義
RVG	文字列 - スカラー	バンカー RVG の名前です。
DiskGroup	文字列 - スカラー	RVG に関連付けられるディスクグループです。
StorageDG	文字列 - スカラー	バンカー RVG の名前です。
StorageRVG	文字列 - スカラー	バンカー RVG の名前です。
StorageHostIds	文字列 - キーリスト	バンカークラスタ内の各ノードのホスト ID をスペースで区切ったリストです。

RVG エージェントのタイプの定義

```
type RVG (
  static str ArgList[] = { RVG, DiskGroup, StorageDG, StorageRVG, StorageHostIds}
  str RVG
  str DiskGroup
  str StorageDG
  str StorageRVG
  str StorageHostIds[]
  static int NumThreads = 1
 )
```

info エントリポイントの使用

info エントリポイントは、RDS のレプリケーションの状態に関する情報を表示します。デフォルトでは、情報間隔は 0 に設定されています。デフォルトの情報間隔を変更するには、次のコマンドを使用します。

```
# hatype -modify resourcetype_name InfoInterval interval
```

たとえば、RVG リソースタイプの情報間隔を 60 秒に設定するには、次のコマンドを入力します。

```
# hatype -modify RVG InfoInterval 60
```

情報間隔とは、VCS が info エントリポイントを実行して、レプリケーションの状態を更新する間隔を表します。上の例では、情報間隔は 60 に設定され、VCS は 60 秒ごとにレプリケーションの状態を更新します。info エントリポイントの出力を表示するには、次のコマンドを使用します。

```
# hares -value resource_name ResourceInfo
```

info エントリポイントの出力は、/var/VRTSvcs/log/engine_A.log ファイルにも記録されます。

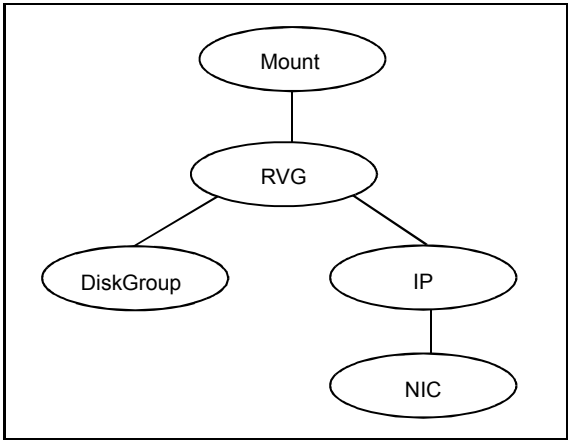
RVG エージェントの依存関係グラフ

RVG リソースでは、RDS 内の RVG（Replicated Volume Group）を定義します。RVG リソースは、DiskGroup リソースに依存します。また、RVG リソースは、レプリケーションに使用する IP リソースにも依存します。

通常のサービスグループでは、Mount リソースなどの上位アプリケーションリソースは Volume リソースに依存しますが、VVR 環境では、Mount リソースは RVG リソースに依存している必要があります。

依存関係の詳細については、『Veritas Cluster Server ユーザーズガイド』を参照してください。

図 1-1 RVG エージェントの依存関係グラフ



RVGPrimary エージェント

RVGPrimary エージェントを使うことにより、VCS 環境で VVR RDS のマイグレーションとテイクオーバーが可能になります。タイプ RVGPrimary のリソースをオンラインにすると、ローカルホストの RVG がプライマリになります。このエージェントは、プライマリとセカンダリ両ホストがクラスタ化されている場合、特に VCS RDC や、VCS グローバルクラスタを使っている場合は、VCS が管理しているアプリケーションが、書き込み可能なレプリケーションディスクを利用できるよう完全自動化するのに便利です。

RVGPrimary エージェントの主な機能を次に示します。

- 広い範囲でアプリケーションをフェールオーバーする場合、手動での VVR プライマリとセカンダリとの役割を移行する作業を省きます。
- ハードテイクオーバーを試す前に移行を試すことによって、レプリケーションボリュームを再同期する必要性を最小限にします。
- 役割を移行する前に、プライマリとセカンダリ両方の RDS が完全に同期されるのを待ちます。
- テイクオーバー後に復帰する場合は、停止したプライマリの自動高速フェールバック再同期をサポートします。

ユーザー環境を作成する際に参考となるエージェント用の設定ファイルのサンプルは、/etc/VRTSvcs/conf/sample_vvr/RVGPrimary にあります。

RVGPrimary エージェントの機能、そのエントリポイントは次のとおりです。

説明	アプリケーションフェールオーバーが発生したときにセカンダリをプライマリに移行またはテイクオーバーします。
エントリポイント	online - RVG の現在の役割を決定します。RVG がセカンダリの場合、もとのプライマリからの未送信の書き込みを待って、移行を試みます。もとのプライマリがダウンしている場合はテイクオーバーを試みます。RVG がプライマリの場合、何も実行せずにオンラインになります。 offline - 何も実行しません。 clean - 何も実行しません。 monitor - 何も実行しません。RVG の実際の監視は RVG エージェントによって行われます。
障害の検出	RVG の実際の監視は RVG エージェントによって行われます。何らかの原因により、VCS の外部で VVR プライマリの移行が実行されると、Mount など他のリソースにすぐに障害が発生するため、このエージェントによる監視は必要ありません。

RVGPrimary エージェントの属性は次のとおりです。

表 1-2 RVGPrimary エージェントの属性

属性	データ形式と値の種類	定義
RvgResourceName	文字列 - スカラー	バンカー RVG の名前です。
AutoTakeover	整数 - スカラー	テイクオーバー後、もとのプライマリが復帰したときに、エージェントがもとのプライマリの高速フェールバック再同期を自動的に実行するかどうかを指定するフラグです。

属性	データ形式と値の種類	定義
AutoResync	整数 - スカラー	テイクオーバー後、もとのプライマリが復帰したときに、エージェントがもとのプライマリの高速フェールバック再同期を自動的に実行するかどうかを指定するフラグです。

RVGPrimary エージェントのタイプの定義

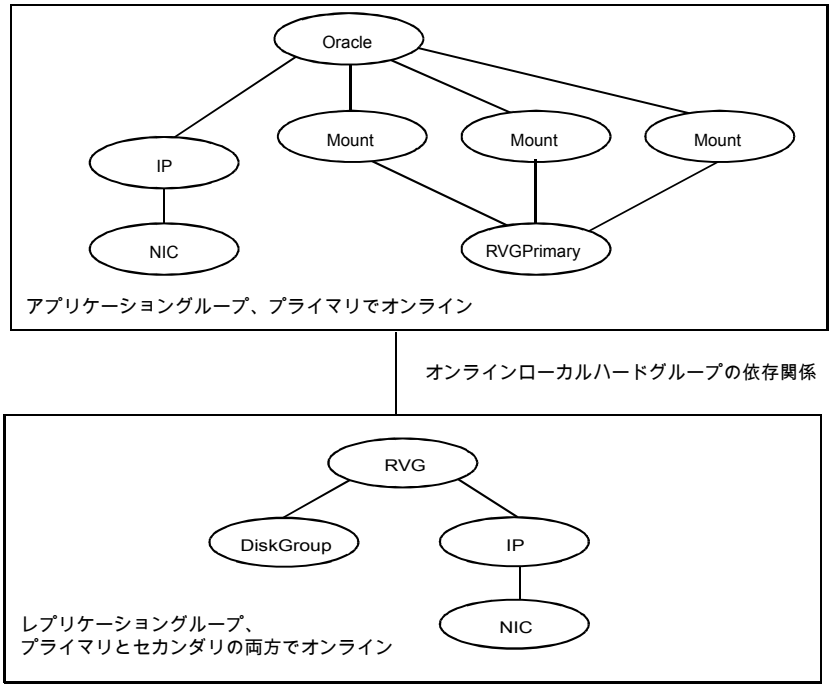
```
type RVGPrimary (
 static keylist SupportedActions = { fbsync }
 static int InfoTimeout = 0
 static int NumThreads = 1
 static int OnlineRetryLimit = 1
 static str ArgList[] = { RvgResourceName, AutoTakeover, AutoResync }
 str RvgResourceName
 int AutoTakeover = 1
 int AutoResync = 0
)
```

RVGPrimary エージェントの依存関係グラフ

RVGPrimary エージェントは、通常、online local firm グループの依存関係にある 2 つのグループで、RVG エージェントと連携して使用されます。親グループには、実際のアプリケーション、ファイルシステムを管理するリソースおよび RVGPrimary リソースが含まれ、子グループには、RVG や DiskGroup タイプリソースなどのストレージインフラストラクチャを管理するリソースが含まれます。

RVGPrimary エージェントを使用した VVR 環境の詳しい設定に関する情報は、『Veritas Cluster Server ユーザーズガイド』を参照してください。

図 1-2 RVGPrimary エージェントの依存関係グラフ



RVGSnapshot エージェント

RVGSnapshot エージェントは、セカンダリ RVG で、領域最適化スナップショットを自動的に作成します。レプリケーションデータに影響を与えずに、スナップショットボリュームのマウントと書き込みを実行できるため、領域最適化スナップショットはファイアドリルに組み込むのに有効なツールであり、広域なフェールオーバーを確実に可能にします。このエージェントを、レプリケーションされているアプリケーションを管理する VCS Mount エージェントや VCSの他のエージェントと組み合わせることにより、特別なファイアドリルサービスグループを生成でき、それを定期的にスケジュールされた間隔でオンラインとオフラインにすることで、ディザスタリカバリ環境での堅牢性をより高めます。

エージェント自体に加えて、ファイアドリル用の VVR と VCS インフラストラクチャとファイアドリルを実行するスクリプト /opt/VRTSvcs/bin/fdsched、それらの結果を統合するテキストベースのウィザード /opt/VRTSvcs/bin/fdsetup がこのパッケージに含まれています。

詳しい説明は、『Veritas Cluster Server ユーザーズガイド』にあります。

RVGSnapshot エージェントの主な機能を次に示します。

- アプリケーションに影響を与えることなく、広域フェールオーバーをシミュレートするマウントが可能となるような、VVR セカンダリへの領域最適化スナップショットの生成処理を自動化します。
- VCS によって完全に管理されているファイアドリルを効果的に設定し、スケジュールするウィザードを利用できます。

次の表は、fdsetup ウィザードがファイアドリルグループ用に適切なリソースを設定する場合の、RVGSnapshot エージェントの機能、エントリポイント、状態定義をまとめたものです。

説明	VVR セカンダリ RDS にあるすべてのボリュームについて、トランザクション的に整合性がとれており、スペースが最適化されたスナップショットを生成、破棄します。
エントリポイント	online - RDS 内のすべてのボリュームについて、トランザクション的に整合性がとれているスナップショットを生成します。 offline - スナップショットを破棄します。 clean - 失敗したスナップショットの生成と破棄をクリーンにします。 monitor - 何の処理も行いません。スナップショットの失敗は、マウントされているファイルシステムの Mount リソースの失敗によって示されます。
障害の検出	スナップショットの生成がオンライン中に成功しなかった場合、タイムアウトのため RVGSnapshot リソースに障害が発生した、と判断します。
状態の定義	ONLINE - スナップショットが作成されたことを表します。 OFFLINE - スナップショットが破棄されたことを表します。

RVGSnapshot エージェントの属性は次のとおりです。

表 1-3 RVGSnapshot エージェントの属性

必須属性	データ形式と値の種類	定義
RvgResourceName	文字列 - スカラー	バンカー RVG の名前です。
CacheObj	文字列 - スカラー	スペースが最適化されたスナップショットに必要なキャッシュオブジェクトの名前です。キャッシュオブジェクトが存在しない場合は、fdsetup ウィザードがそれを生成します。
Prefix	文字列 - スカラー	スナップショットされたボリュームを生成する場合に、実際のボリュームの名前の先頭に付加されるトークンです。

必須属性	データ形式と値の種類	定義
オプション属性	データ形式と値の種類	定義
DestroyOnOffline	整数 - スカラー	テイクオーバー後、もとのプライマリが復帰したときに、エージェントがもとのプライマリの高速フェールバック再同期を自動的に実行するかどうかを指定するフラグです。ファイアドリル用のスナップショットは、処理効率への影響を減らすため長期間保存されているスナップショットは削除してください。ただし、データを保持する必要がある場合、この値は 0 に設定します。デフォルトは 1 (true) です。
FDFile	一時 文字列 - スカラー	ファイアドリルスケジュールは、システム名と、ファイルへのパスを使ってこの属性を更新します。このファイルには、RVGSnapshot リソースを含むグループに対する最後に行ったファイアドリルの結果が含まれます。

RVGSnapshot エージェントのタイプの定義

```
type RVGSnapshot (
 static keylist RegList = { Prefix }
 static int InfoTimeout = 0
 static int NumThreads = 1
 static str ArgList[] = { RvgResourceName, CacheObj, Prefix, DestroyOnOffline }
 str RvgResourceName
 str CacheObj
 str Prefix
 boolean DestroyOnOffline = 1
 temp str FDFile
)
```

並列アプリケーションのエージェントの機能

並列アプリケーションのエージェントには次のものが含まれます。

- 「RVGShared エージェント」
- 「RVGLogowner エージェント」
- 「RVGSharedPri エージェント」

RVGShared エージェント

RVGShared エージェントを使うと、並列アプリケーションを設定して、クラスタ内で RVG を使用することができます。RVGShared エージェントによって、共有ディスクグループ環境の RVG が監視されます。RVGShared エージェントは、VCS 内のパラレルグループとして設定する必要があります。通常 RVGShared リソースは、VCS クラスタのすべてのノードで同時にオンラインまたはオフラインとなります。ユーザー環境を作成する際に参考となるエージェント用の設定ファイルのサンプル
は、/etc/VRTSvcs/conf/sample_vvr/RVGLogowner にあります。

RVGShared エージェントの機能、そのエントリポイント、状態の定義は次のとおりです。

説明	共有環境の RVG を監視します。これはパラレルリソースです。
エントリポイント	online - RVG が起動しているかどうかを確認します。RVG が起動していない場合は、RVG を復旧して起動します。 offline - 何も実行しません。 clean - 何も実行しません。 info: RDS（Replicated Data Set）のレプリケーションの状態に関する情報を取得します。 p.11 の「info エントリポイントの使用」を参照してください。 monitor - RVG が起動している場合は、状態が ONLINE と表示されます。RVG が起動していない場合は、状態が OFFLINE と表示されます。
状態の定義	ONLINE - RVG が ENABLED/ACTIVE 状態にあることを示します。 OFFLINE - RVG が ENABLED/ACTIVE 状態にないか、管理者がオフラインのエントリポイントを呼び出したことを示します。

RVGShared エージェントの属性は次のとおりです。

表 1-4 RVGShared エージェントの属性

属性	データ形式と値の種類	定義
RVG	文字列 - スカラー	監視される RVG の名前です。
DiskGroup	文字列 - スカラー	RVGに関連付けられる共有ディスクグループです。

RVGShared エージェントのタイプの定義

```
type RVGShared (
 static str ArgList[] = { RVG, DiskGroup }
```

```
  str RVG
  str DiskGroup
  static int NumThreads = 1
)
```

RVGShared エージェントの依存関係グラフ

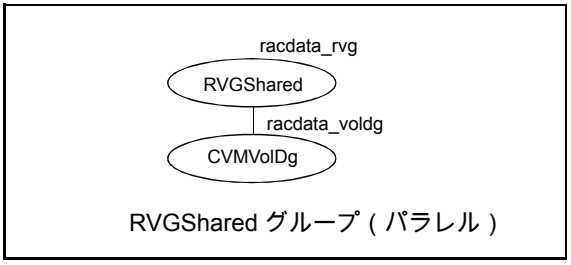
RVGShared リソースでは、RDS 内の RVG を定義します。RVGShared リソースは、CVMVolDg リソースに依存します。

RVGShared リソースは、パラレルグループに設定する必要があります。

p.35 の「並列アプリケーションの設定例」を参照してください。

依存関係の詳細については、『Veritas Cluster Server ユーザーズガイド』を参照してください。

図 1-3 RVGShared エージェントの依存関係グラフ



メモ: CVMVolDg リソースの CVMVolume 属性に RVG の一部であるボリュームを追加しないでください。RVG 内のボリュームは、RVGShared リソースによって管理されます。

RVGLogowner エージェント

RVGLogowner エージェントでは、クラスタ内のノードをログ所有者として割り当てたり、割り当ての解除を行います。VVR を使ってデータのレプリケーションを行うには、プライマリとセカンダリ間のネットワーク接続が必要です。共有ディスクグループ環境では、セカンダリに対するデータのレプリケーションを行えるのはログ所有者である 1 つのノードのみです。

レプリケーションの高可用性を実現するには、ログ所有者のノードが高可用性を実現している必要があります。ログ所有者の高可用性を実現するには、RVGLogownerリソースをフェールオーバーグループのリソースとして設定する必要があります。また、クラスタ内のあるノードから別のノードへログ所有者のレプリケーションおよびフェールオーバーを可能にするために、ログ所有者に仮想 IP を設定する必要があります。仮想 IP は IP リソースとして設定します。

p.21 の「RVGLogowner エージェントの依存関係グラフ」を参照してください。

ログ所有者の詳細については、『Veritas Volume Replicator 管理者ガイド』を参照してください。ユーザー環境を作成する際に参考となるエージェント用の設定ファイルのサンプルは、/etc/VRTSvcs/conf/sample_vvr/RVGLogowner にあります。

RVGLogowner エージェントの機能、そのエントリポイント、状態の定義は次のとおりです。

説明

CVM クラスタ内のノードをログ所有者として割り当てたり、割り当ての解除を行います。これはフェールオーバーリソースです。

オプション

online - ログ所有者をノードに割り当てます。

offline - ログ所有者のノードの割り当てを解除します。

monitor - ノードがログ所有者で RVG が ENABLED/ACTIVE 状態の場合は、ONLINE を返します。ノードがログ所有者で ENABLED/ACTIVE 状態ではない場合、または状態にかかわらずノードがログ所有者ではない場合は、OFFLINE を返します。

メモ: ログ所有者が監視される RVG のリソースタイプは RVGShared として設定する必要があります。

clean - ログ所有者のノードの割り当てを解除します。

状態の定義

ONLINE - クラスタ内でノードが RVG のログ所有者であることを示します。

OFFLINE - クラスタ内でノードが RVG のログ所有者ではないことを示します。

RVGLogowner エージェントの属性は次のとおりです。

表 1-5 RVGLogowner エージェントの属性

属性	データ形式と値の種類	定義
RVG	文字列 - スカラー	監視されるRVGの名前です。
DiskGroup	文字列 - スカラー	RVG に関連付けられるディスクグループです。
バンカーの属性	データ形式と値の種類	定義
StorageDG	文字列 - スカラー	バンカーディスクグループの名前です。
StorageRVG	文字列 - スカラー	バンカー RVG の名前です。

属性	データ形式と値の種類	定義
StorageHostIds	文字列 - キーリスト	バンカークラスタ内の各ノードのホストIDをスペースで区切ったリストです。

RVGLogowner エージェントのタイプの定義

```
type RVGLogowner (
  static str ArgList[] = { RVG, DiskGroup, StorageDG, StorageRVG,
StorageHostIds}
  str RVG
  str DiskGroup
  str StorageDG
  str StorageRVG
  str StorageHostIds
  static int NumThreads = 1
)
```

RVGLogowner エージェントの依存関係グラフ

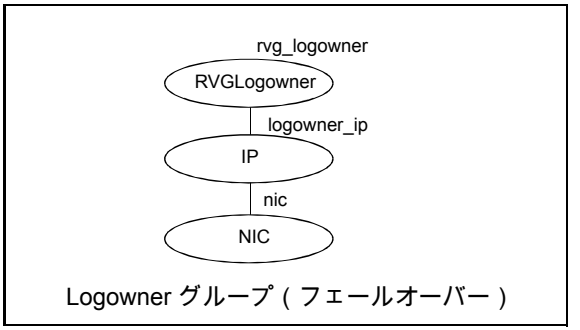
RVGLogowner リソースでは、クラスタ内の RVG の ログ所有者を定義します。
RVGLogowner リソースは、レプリケーションに使用する IP リソースに依存します。

RVGLogowner リソースは、フェールオーバーグループに設定する必要があります。
RVGLogowner グループは、別のグループに構成されている RVGSharedPri エージェントおよび RVGShared エージェントと適切なサービスグループ依存関係を持ち、これらと連携して使用されます。

p.35 の「並列アプリケーションの設定例」を参照してください。

依存関係の詳細については、『Veritas Cluster Server ユーザーズガイド』を参照してください。

図 1-4 RVGLogowner エージェントの依存関係グラフ



RVGSharedPri エージェント

RVGSharedPri エージェントを使うことにより、VCS 環境でパラレルグループの VVR RDS の移行とテイクオーバーが可能になります。RVGSharedPri タイプのリソースをオンラインにすると、ローカルホストの RVG がプライマリになります。このエージェントは、VCS グローバルクラスタを使用してプライマリとセカンダリの両ホストをクラスタ化して、VCS が管理しているアプリケーションが、書き込み可能なレプリケーションディスクを利用できるよう完全自動化するときに便利です。

RVGSharedPri エージェントの主な機能を次に示します。

- 広い範囲でアプリケーションをフェールオーバーする場合、手動での VVR プライマリとセカンダリとの役割を移行する作業を省きます。
- ハードテイクオーバーを試す前に移行を試すことによって、レプリケーションボリュームを再同期する必要性を最小限にします。
- 役割を移行する前に、プライマリとセカンダリ両方の RDS が完全に同期されるのを待ちます。
- テイクオーバー後に復帰する場合は、停止したプライマリの自動高速フェールバック再同期をサポートします。

サンプル設定ファイルは、/etc/VRTSvcs/conf/sample_rac/ ディレクトリにあり、ファイル名の一部に CVR を含みます。これらのサンプルファイルは、VRTSdbac パッケージの一部としてインストールされ、ユーザー環境を作成する際に参考になります。

RVGSharedPri エージェントの機能、そのエントリポイントは次のとおりです。

説明	パラレルサービスグループにフェールオーバーが発生したときにセカンダリをプライマリに移行またはテイクオーバーします。

エントリポイント

online - RVG の現在の役割を決定します。RVG がセカンダリの場合、もとのプライマリからの未送信の書き込みを待って、移行を試みます。もとのプライマリがダウンしている場合はテイクオーバーを試みます。RVG がプライマリの場合、何も実行せずにオンラインになります。

offline - 何も実行しません。

clean - 何も実行しません。

monitor - 何も実行しません。RVG の実際の監視は RVGShared エージェントによって行われます。

障害の検出

RVG の実際の監視は RVGShared エージェントによって行われます。何らかの原因により、VCS の外部で VVR プライマリの移行が実行されると、Mount など他のリソースに即座に障害が発生するため、このエージェントによる監視は必要ありません。

表 1-6 RVGSharedPri エージェントの属性

属性	データ形式と値の種類	定義
RvgResourceName	文字列 - スカラー	このエージェントが監視するリソースタイプの名前です。つまり、RVGShared エージェントを使用するように設定された RVG リソースタイプの名前です。必要な VVR オブジェクト名です。たとえば、RVG の名前、ディスクグループ名、RLINK 名、SRL 名は、このエージェントが直接 VCS にクエリーを送信して検出します。
AutoTakeover	整数 - スカラー	テイクオーバー後、もとのプライマリが復帰したときに、エージェントがもとのプライマリの高速フェールバック再同期を自動的に実行するかどうかを指定するフラグです。
AutoResync	整数 - スカラー	テイクオーバー後、もとのプライマリが復帰したときに、エージェントがもとのプライマリの高速フェールバック再同期を自動的に実行するかどうかを指定するフラグです。
VCSResLock	文字列 - スカラー	この属性は、VCS の内部使用のために予約されます。

RVGSharedPri エージェントのタイプの定義

```
type RVGSharedPri (
 static keylist SupportedActions = { fbsync, resync }
 static int NumThreads = 1
```

```
 static int OnlineRetryLimit = 1
 static str ArgList[] = { RvgResourceName, "RvgResourceName:RVG",
 "RvgResourceName:DiskGroup", AutoTakeover, AutoResync }
 str RvgResourceName
 int AutoTakeover = 1
 int AutoResync = 0
 temp str VCSResLock
)
```

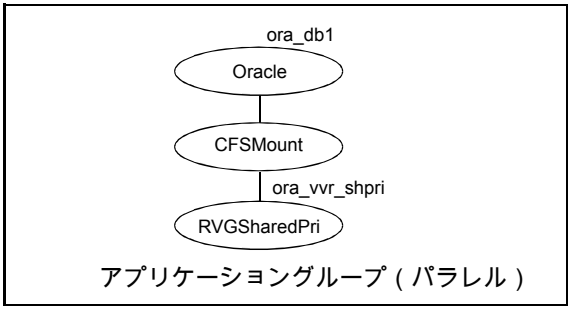
RVGSharedPri エージェントの依存関係グラフ

RVGSharedPri エージェントは、別のグループに構成されている RVGShared エージェントおよび RVGLogowner エージェントと適切なサービスグループ依存関係を持ち、これらと連携して使用されます。

p.35 の「並列アプリケーションの設定例」を参照してください。

RVGSharedPri エージェントは、パラレルサービスグループに設定する必要があります。このアプリケーションサービスグループには、実際のアプリケーションおよびファイルシステムを管理するリソースと、RVGSharedPri エージェントが含まれます。

図 1-5 RVGSharedPri エージェントの依存関係グラフ



ハイブリッドアプリケーションのエージェントの機能

ハイブリッドアプリケーションのエージェントには次のものが含まれます。

- 「RVG エージェント」
- 「RVGPrimary エージェント」

ハイブリッド設定とは、RDC 用の、フェールオーバーグループとパラレルサービスグループを組み合わせた設定です。ハイブリッドサービスグループは、システムゾーン内のフェールオーバーグループやシステムゾーン間のパラレルグループとして動作します。このサービスグループは、システムゾーン間でフェールオーバーすることはできません。ただし、同

じシステムゾーン内のシステム間でのみ、ハイブリッドサービスグループの切り替え操作が可能です。

RVG エージェントおよび RVGPrimary エージェントの詳細については、「RVG エージェント」および「RVGPrimary エージェント」をそれぞれ参照してください。これらの項には、RVG エージェントおよび RVGPrimary エージェントのエントリポイント、状態の定義および属性に関する情報が記載されています。さらに、RDC を設定する際は、RVG エージェントおよび RVGPrimary エージェントの次の属性を設定する必要があります。

表 1-7 RDC の属性

オプション属性	データ形式と値の種類	定義
SystemZones	整数 - 関連付け	フェールオーバーゾーンを示します。

RDC は、共有ストレージとは対照的な VVR を使用して、セカンダリのデータへのアクセスを提供します。RDC は、1 つの VCS クラスタ内に存在します。フェールオーバーグループとして設定されたアプリケーショングループは、プライマリホスト上でのみオンラインになります。プライマリサイトに障害が発生した場合は、セカンダリがプライマリに昇格し、新プライマリホスト上でアプリケーションがオンライン化されます。

共有ストレージが不足している場合や、プライマリサイトとセカンダリサイトを SAN によって相互接続する場合には、RDC 設定が最適です。ただし、プライマリサイトとセカンダリサイト間の通信にデュアル構成専用の LLT リンクが使われている場合に限ります。

RDC の詳細については、『Veritas Cluster Server ユーザーズガイド』を参照してください。

VCS 環境における VVR の設定方法の概要

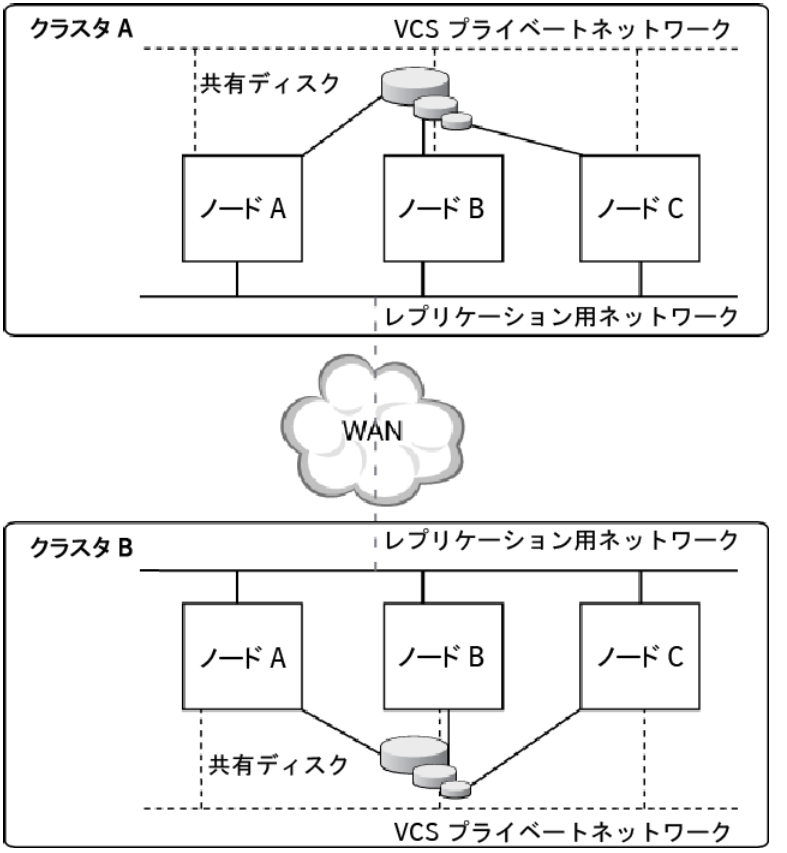
この項では、レプリケーションに関連するアプリケーションの高可用性を実現するために、VCS 環境で VVR を設定する方法の概要を説明します。

VCS 環境に VVR を設定する場合は、次のタスクを順番どおりに実行します。

- RDS（RDS）の作成を含めて、VVR 構成を設定します。
- VVR エージェントのサービスグループを作成し、リソースとサービスグループの適切な依存関係を追加します。

VCS 環境における一般的な VVR 設定

2つのクラスタ環境による VCS 環境での VVR のレプリケーションの設定は、次のとおりです。

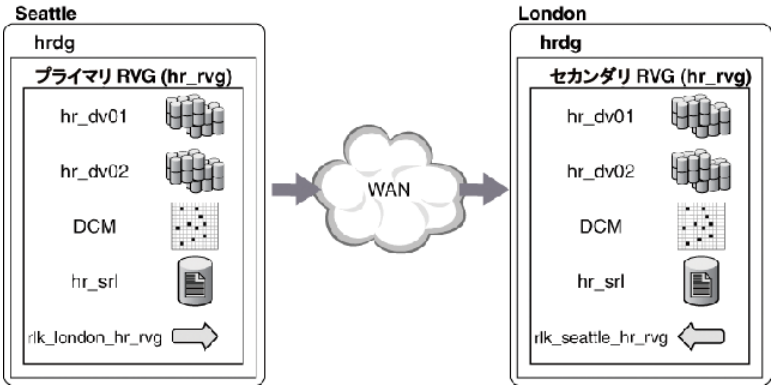


VCS 環境における VVR 設定例

次の例では、2 つのクラスタが、それぞれ別のサイトに置かれています。VVR は WAN を使用してサイト間でデータをレプリケーションします。

1 つ目のクラスタはシアトルにあり、Seattle と名前が付けられています。クラスタ Seattle は、seattle1 と seattle2 の 2 つのノードで構成されます。2 つ目のクラスタはロンドンにあり、名前は London です。このクラスタ London も、london1 と london2 の 2 つのノードで構成されます。クラスタ Seattle の 2 つのノードにはプライマリ RVG が含まれています。クラスタ London の 2 つのノードにはセカンダリ RVG が含まれています。次の図は、RVG エージェントが使用する VVR コンポーネントの名前を示しています。

図 1-6 VCS 環境における VVR 設定例



2

高可用性を目的としたエージェントの設定

この章では以下の項目について説明しています。

VCS 環境で VVR を設定するための必要条件

VCS 環境で VVR を設定するための必要条件は、次のとおりです。

- VVR でのレプリケーションの推奨設定に従います。
  レプリケーションの設定の詳細については、『Veritas Volume Replicator 管理者ガイド』を参照してください。
- レプリケーションサービスグループに含まれるノードは、同じポート番号をレプリケーションで使用します。そのため、ノードによっては VVR の設定前に、ポート番号を変更する必要があります。
- ノードが複数の NIC を備えており、レプリケーションに使用しているネットワークに複数の NIC で接続されている場合は、各 NIC に異なる MAC アドレスが必要です。これは、プライマリおよびセカンダリサイトのすべてのノードに当てはまります。

- 次の必要条件は RVG エージェントにのみ適用されます。VCS では、ディスクグループの noautoimport 属性の設定が必要です。
  noautoimport 属性の設定の詳細については、『Veritas Cluster Server 付属エージェントリファレンスガイド』を参照してください。

エージェントの推奨設定

次のリストは、エージェントの推奨設定を示します。

- 1 つのサービスグループに DiskGroup と RVG リソースが 1 つずつ存在している必要があります。
- ディスクグループを DiskGroup リソースとして設定した場合は、このディスクグループ内のすべての RVG を RVG リソースとして設定する必要があります。
  ディスクグループを CVMVolDG リソースとして設定した場合は、すべての RVG を RVGShared リソースとして設定する必要があります。
- フェールオーバーアプリケーションを設定するときは、RVG エージェント、RVGPrimary エージェントおよび RVGSnapshot エージェントを使用します。
- 並列アプリケーションを設定するときは、RVGShared エージェント、RVGSharedPri エージェントおよび RVGLogowner エージェントを使用します。設定に複数の RVGLogowner リソースがある場合、RVGLogowner リソースを含むサービスグループの AutoStartList 属性で、ホストの順番を入れ替えることをお勧めします。このように設定すると、VCS はクラスタ内の別々のノードで RVGLogowner リソースをオンラインにします。これにより負荷分散が円滑化します。たとえば、RVGLogowner リソースを含む最初のサービスグループは次のようになります。

  ```
  AutoStartList = { seattle1, seattle2 }
  ```

  これに対し、次のサービスグループは次のようになります。

  ```
  AutoStartList = { seattle2, seattle1 }
  ```

  など。
- CVM グループで RVGShared リソースを設定しないでください。RVGShared リソースおよび CVMVolDg リソースを含む異なるグループで RVGShared リソースを設定してください。
- ボリュームセットが RVG に完全に関連付けられている場合、つまりそのコンポーネントボリュームすべてが RVG に関連付けられている場合は、ボリュームの追加と同様にして、このボリュームセットをエージェント設定に追加できます。コンポーネントボリューム名ではなく Mount リソース内のボリュームセットを指定します。
  p.36 の「例 - VCS 環境における VVR の設定」を参照してください。

メモ: エージェントは、RVG に部分的に関連付けられた、つまり 1 つ以上のコンポーネントボリュームが RVG に関連付けられていないボリュームセットのマウントはサポートしません。

RVG でのボリュームセットの使用方法の詳細については、『Veritas Volume Replicator 管理者ガイド』を参照してください。

VCS 設定への VVR エージェントの追加

次の場合に、VCS 設定に VVR エージェントを追加できます。

- VCS が実行中の場合
- VCS が停止している場合

システムでアプリケーションを停止せずにエージェントを追加するには、次の手順を実行します。

VCS が実行中の場合にエージェントを追加する方法

1. クラスタのいずれかのノードに root ユーザーとしてログインします。
2. クラスタ内の任意のシステムで次のコマンドを入力して、VCS 設定モードを読み書き両用に設定します。

   ```
   # haconf -makerw
   ```

3. 次のスクリプトを実行して、VCS 設定を更新します。

   ```
   # /etc/VRTSvcs/conf/sample_vvr/addVVRTypes.sh
   ```

4. バージョン 1.0 または 1.1 のエージェントがインストールされている場合は、次の手順を実行します。
   - 各依存関係に対して次のコマンドを実行して、Rvolume リソースに対する依存関係のリンクを解除します。

     ```
     # hares -unlink parent_resource child_resource
     ```

   - 各リソースに対して次のコマンドを実行して、main.cf ファイルから RVolume リソースを削除します。

     ```
     # hares -delete resource
     ```

   - RVG リソース用の依存関係を生成します。
     p.12 の「RVG エージェントの依存関係グラフ」を参照してください。

次のコマンドを入力します。

```
# hares -link parent_resource child_resource
```

5 次のコマンドを使って、既存の設定に対する変更を確実に保存し、以後の変更が行われないようにします。

```
# haconf -dump -makero
```

エージェントを新規にインストールしている場合は、これで設定は終了です。エージェントをアップグレードしている場合は、続いて手順 6 と 7 を実行します。

6 新しいエージェントをインストールする前にエージェントを停止した場合は、次のコマンドを入力してシステム上のエージェントを起動します。

```
# haagent -start agent_name -sys system_name
```

[ログファイルの中からメッセージを検索してください(Please look for messages in the log file)]というメッセージが表示された場合は、/var/VRTSvcs/log/engine_A.log というファイルを調べ、各エージェントが起動したことを示すメッセージが存在するかどうかを確認してください。

ps コマンドでエージェントが動作しているかどうかを確認することも可能です。

7 インストールをする前に RVG サービスグループをオフラインにした場合は、次のコマンドを使用してオンラインにします。

```
# hagrp -online service_group -sys system_name
```

main.cf ファイルを編集して、エージェントを追加できます。main.cf ファイルを編集する前に VCS を停止する必要があります。次の手順を実行します。

VCS が停止している場合にエージェントを追加する方法

1 クラスタのいずれかのノードに root ユーザーとしてログインします。

2 次のコマンドを使って、既存の設定に対する変更を確実に保存します。そして、/etc/VRTSvcs/conf/config ディレクトリの main.cf ファイルを変更している間、その他の変更が行われないようにします。

現在、VCS クラスタが書き込み可能な場合は、次のコマンドを実行します。

```
# haconf -dump -makero
```

VCS クラスタがすでに読み込み専用の場合は、次のコマンドを実行します。

```
# haconf -dump
```

3 VCS を実行したまま、設定ファイルを編集しないでください。次のコマンドを入力すると、すべてのシステムで had デーモンが停止し、リソースを VCS 以外からコントロールすることが可能になります。

```
# hastop -all -force
```

4 /etc/VRTSvcs/conf から /etc/VRTSvcs/conf/config ディレクトリへ VVRTypes.cf ファイルをコピーします。

5 /etc/VRTSvcs/conf/config ディレクトリにある main.cf ファイルに VVRTypes を追加します。

バージョン 3.5 以下のエージェントがインストールされている場合は、ファイル内の行を次のように変更します。

main.cf ファイルに含まれる変更前の行

```
include "SRVMTypes.cf"
```

この行を次のように変更します。

```
include "VVRTypes.cf"
```

エージェントを新規にインストールする場合は、次の行を main.cf ファイルに追加します。

```
include "VVRTypes.cf"
```

6 バージョン 1.0 または 1.1 のエージェントがインストールされている場合は、次の手順を実行します。

- RVolume リソースおよび RVolume リソースタイプとの依存関係を解除します。
- RVG リソースの依存関係を新規に作成します。
  p.12 の「RVG エージェントの依存関係グラフ」を参照してください。

7 このバージョンのエージェントは、RVG リソースの Primary、SRL、RLINK 属性をサポートしません。既存の RVG リソースでこれらの属性を使用していたり定義していたりする場合は、これらの属性を削除する必要があります。

8 /etc/VRTSvcs/conf/config/main.cf ファイルの構文を検証します。

```
# hacf -verify /etc/VRTSvcs/conf/config
```

9 両方のクラスタ内のすべてのシステムで VCS エンジンを起動します。

p.34 の「プライマリとセカンダリの両方のクラスタ内のすべてのシステムで VCS を起動する方法」を参照してください。

すべてのシステムでの VCS の起動

プライマリとセカンダリの両方のクラスタ内のすべてのシステムで VCS を起動する方法

1 プライマリクラスタ内の main.cf を変更したシステムで、VCS エンジンを起動します。

```
# hastart
```

2 hastatus コマンドを入力します。

```
# hastatus
```

3 "LOCAL_BUILD" または "RUNNING"?がメッセージ欄に表示されたら、もう一方のシステムで VCS を起動します。

```
# hastart
```

4 すべてのサービスグループのリソースがオンラインであることを確認します。すべてのシステムで次のコマンドを入力します。

```
# hagrp -display
```

5 セカンダリクラスタ内の main.cf を変更したシステムから VCS エンジンを起動します。

```
# hastart
```

6 hastatus コマンドを入力します。

```
# hastatus
```

7 "LOCAL_BUILD" または "RUNNING"?がメッセージ欄に表示されたら、もう一方のシステムで VCS を起動します。

```
# hastart
```

8 サービスグループとそのリソースがオンラインであることを確認します。すべてのシステムで次のコマンドを入力します。

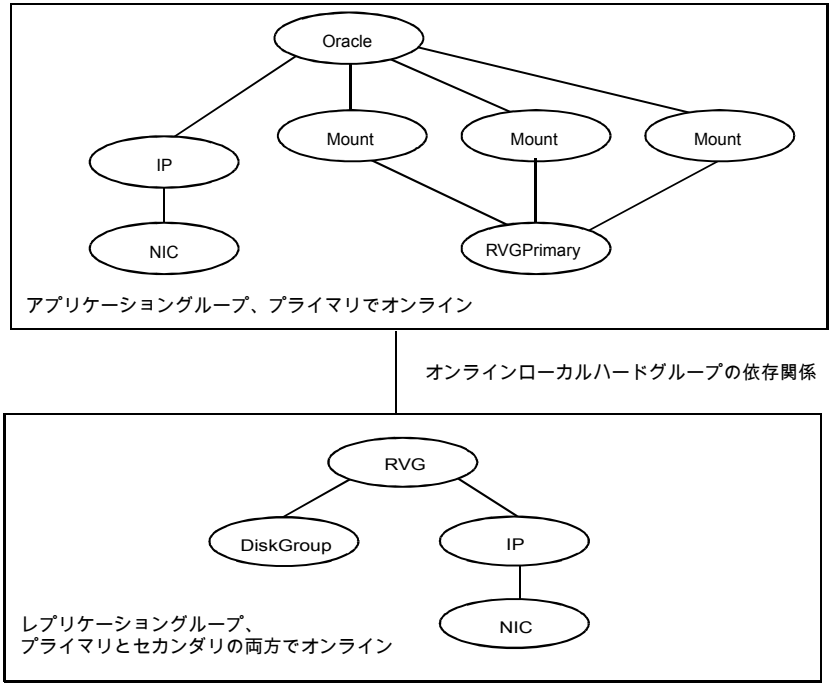
```
# hagrp -display
```

フェールオーバーアプリケーションの設定例

次の例では、RVG を使用することで、2 つのクラスタ間のフェールオーバーアプリケーションの高可用性を実現しています。アプリケーションサービスグループのリソースには、アプ

リケーション、Mount、NIC、IP および RVGPrimary が含まれます。レプリケーショングループのリソースには、RVG、IP、NIC および DiskGroup が含まれます。アプリケーショングループは、レプリケーショングループに対して、online local hard グループの依存関係を持ちます。

図 2-1 RVG と RVGPrimary エージェント - サービスグループとリソースの依存関係

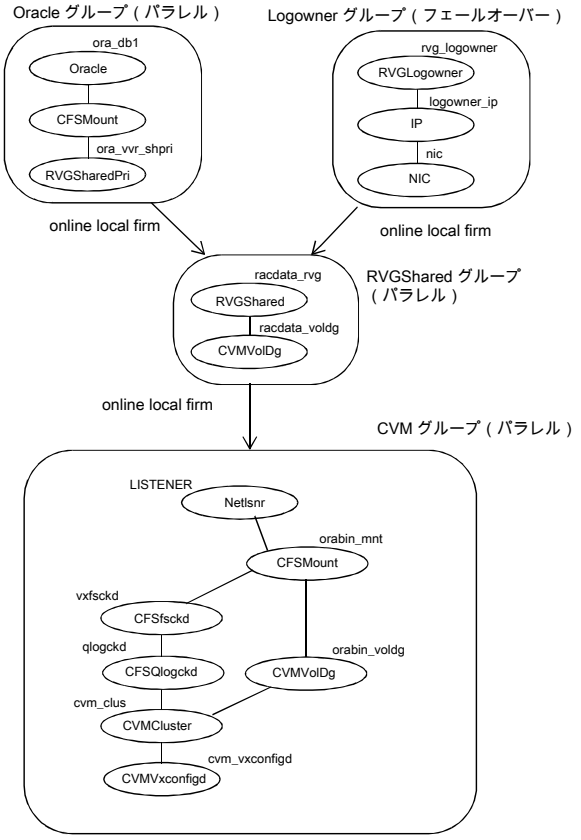


並列アプリケーションの設定例

次の例では、RVG を使用することで、2 つのクラスタ間の並列アプリケーションの高可用性を実現しています。ここでは、Oracle サービスグループがアプリケーショングループで、CFSMount リソースがそのグループに含まれています。Logowner サービスグループがフェールオーバーグループで、ログ所有者を管理します。サービスグループ RVGShared が、アプリケーションで使用される RVG を管理します。Oracle および CVM グループが、パラレルグループとして設定されます。

サービスグループ Logowner および Oracle は、サービスグループ RVGShared に依存します。RVGShared は、共有環境の RVG を管理するため、CVM サービスグループに依存します。

図 2-2 RVGShared、RVGLogowner および RVGSharedPri エージェント - サービスグループとリソースの依存関係



例 - VCS 環境における VVR の設定

VCS で VVR を設定するには、次に示すタスクを順番どおりに実行する必要があります。

- 「VVR の設定」
- 「VVR レプリケーションの状態の確認」
- 「フェールオーバーアプリケーションのエージェントの設定」
- 「並列アプリケーションのエージェントの設定」

VVR の設定を行う前に、VVR がインストールされているクラスタ内のすべてのノードが、レプリケーション用に同じポート番号を使用するかどうかを確認します。ポート番号を確認

し変更するにはvrportコマンドを使用します。ポート番号がすべてのノードで同じである場合は、VCS 設定に VVR エージェントを追加します。

vrportコマンドの使用方法については、『Veritas Volume Replicator 管理者ガイド』を参照してください。

VVR の設定

この項では、サンプルの VVR 設定を行う手順について説明します。この例で設定する VVR 設定は、RVG エージェントに適用されます。RVG エージェントのサンプル設定ファイルで使用されている名前を使用します。VVR を設定する手順は、すべての VVR エージェントで共通です。他のエージェントを設定するときは、/etc/VRTSvcs/conf/sample_vvr ディレクトリにある各サンプル設定ファイルを使用してください。

VVR の設定の詳細については、『Veritas Volume Replicator 管理者ガイド』を参照してください。

例では次の表の名前を使用しています。

クラスタ名: Seattle

ディスクグループ	hrdg
プライマリ RVG	hr_rvg
london1 へのプライマリ RLINK	rlk_london_hr_rvg
プライマリデータボリューム 1	hr_dv01
プライマリデータボリューム 2	hr_dv02
プライマリボリュームセット(データボリューム hr_dv03、hr_dv04 を含む)	hr_vset01
hr_rvg のプライマリ SRL	hr_srl
クラスタ IP	10.216.144.160

クラスタ名: London

ディスクグループ	hrdg
セカンダリ RVG	hr_rvg
seattle へのセカンダリ RLINK	rlk_seattle_hr_rvg
セカンダリデータボリューム 1	hr_dv01
セカンダリデータボリューム 2	hr_dv02

セカンダリボリュームセット(データボリューム hr_dv03、hr_dv04 を含む)	hr_vset01
hr_rvg のセカンダリ SRL	hr_srl
クラスタ IP	10.216.144.162

ここに示す例は、ホスト seattle1 と london1 に hrdg という名前のディスクグループがあり、例に示す VVR オブジェクトを作成するのに十分な空き領域があることを前提としています。seattle1 と london1 に VVR を設定し、/etc/VRTSvcs/conf/sample_vvr/RVG ディレクトリ内のサンプル設定ファイル main.cf.seattle と main.cf.london に使用するオブジェクトを追加します。

p.26 の「VCS 環境における VVR 設定例」を参照してください。

VVR 設定を行う方法

1 london1:

- セカンダリデータボリュームを作成します。

```
# vxassist -g hrdg make hr_dv01 100M ¥
    layout=mirror logtype=dcm mirror=2
# vxassist -g hrdg make hr_dv02 100M ¥
    layout=mirror logtype=dcm mirror=2
```

- セカンダリのボリュームセット用のデータボリュームを作成し、ボリュームセットを作成します。

```
# vxassist -g hrdg make hr_dv03 100M ¥
    layout=mirror logtype=dcm mirror=2
# vxassist -g hrdg make hr_dv04 100M ¥
    layout=mirror logtype=dcm mirror=2
# vxmake -g hrdg vset hr_vset01 ¥
    appvols=hr_dv03,hr_dv04
```

- セカンダリ SRL を作成します。

```
# vxassist -g hrdg make hr_srl 200M mirror=2
```

2 seattle1:

- プライマリデータボリュームを作成します。

```
# vxassist -g hrdg make hr_dv01 100M ¥
    layout=mirror logtype=dcm mirror=2
# vxassist -g hrdg make hr_dv02 100M ¥
    layout=mirror logtype=dcm mirror=2
```

- プライマリのボリュームセット用のデータボリュームを作成し、ボリュームセットを作成します。

```
# vxassist -g hrdg make hr_dv03 100M ¥
    layout=mirror logtype=dcm mirror=2
# vxassist -g hrdg make hr_dv04 100M ¥
    layout=mirror logtype=dcm mirror=2
# vxmake -g hrdg vset hr_vset01 ¥
    appvols=hr_dv03,hr_dv04
```

- プライマリ SRL を作成します。

```
# vxassist -g hrdg make hr_srl 200M mirror=2
```

- プライマリ RVG を作成します。

```
# vradmin -g hrdg createpri hr_rvg ¥
    hr_dv01,hr_dv02,hr_vset01 hr_srl
```

- レプリケーションに使用される仮想 IP アドレスを決定し、この IP が設定されているデバイスのインターフェースを確認します。この IP 用のデバイスインターフェースが設定されていない場合は、デバイスを設定し、各 OS に応じたコマンドを使用して、IP を立ち上げます。レプリケーション用に使用されるこの IP アドレスは、この RVG サービスグループ用の IP リソースとして設定する必要があります。
- セカンダリ RVG を作成します。

```
# vradmin -g hrdg addsec hr_rvg 10.216.144.160 ¥
    10.216.144.162 prlink=rlk_london_hr_rvg ¥
    srlink=rlk_seattle_hr_rvg
```

メモ: RLINK は、フェールオーバーが正常に実行されるように、仮想 IP アドレスを参照する必要があります。仮想 IP アドレス 10.216.144.160 から仮想 IP アドレス 10.216.144.162 に、またはその逆方向に ping できるようにしてください。

- レプリケーションを開始します。

```
# vradmin -g hrdg -f startrep hr_rvg
```

3 seattle1 と seattle2 に次のディレクトリを作成します。これらのディレクトリは、seattle サイトでボリュームセット hr_vset01 のボリューム hr_dv01 と hr_dv02 のマウントポイントとして使用されます。

```
# mkdir /hr_mount01
# mkdir /hr_mount02
# mkdir /hr_mount03
```

4 seattle1 で、ボリューム hr_dv01 と hr_dv02、ボリュームセット hr_vset01 にファイルシステムを作成します。

VVR レプリケーションの状態の確認

seattle1 と london1 間のレプリケーションの状態をテストし、VVR が正しく設定されていることを確認します。

レプリケーション状態を確認する方法

1 各ノードで次のコマンドを入力します。

```
# vxprint -g hrdg hr_rvg
```

2 出力で次の点を確認します。

- RVG の状態が ENABLED/ACTIVE になっている。
- RLINK の状態が、CONNECT/ACTIVE になっている。

フェールオーバーアプリケーションのエージェントの設定

この項では、フェールオーバーアプリケーションの VVR エージェントの設定方法について説明します。

p.47 の「並列アプリケーションのエージェントの設定」を参照してください。

VCS の停止中または実行中に RVG エージェントと RVGPrimary エージェントを設定できます。サンプル設定ファイル、main.cf.seattle と main.cf.london はそれぞれ /etc/VRTSvcs/conf/sample_vvr/RVG ディレクトリと /etc/VRTSvcs/conf/sample_vvr/RVGPrimary ディレクトリにあり、参照用に使用できます。

次の手順のいずれかを使用して、既存の VCS 設定に RVG リソースを追加できます。

- 「VCS 実行中のエージェントの設定」
- 「VCS 停止時のエージェントの設定」

VCS 実行中のエージェントの設定

この項の例では、VCS を実行中に RVG および RVGPrimary エージェントを設定する方法について説明します。

p.34 の「フェールオーバーアプリケーションの設定例」を参照してください。

メモ: この例は、リソースや属性の生成や変更時に、参照用として使用します。

VCS の実行中に既存の VCS 設定にエージェントリソースを追加するには、次の手順を実行します。

- レプリケーションサービスグループを作成します。
- アプリケーションサービスグループを作成します。

プライマリクラスタ Seattle 内のシステム seattle1 で次の手順を実行してから、セカンダリクラスタ London 内のシステム london1 で同じ手順を繰り返します（説明に従って若干変更します）。

レプリケーションサービスグループを作成する方法

1. root でログインします。
2. 次のコマンドを発行して、VCS 設定モードを読み書き両用にします。

```
# haconf -makerw
```

3. レプリケーションサービスグループ `VVRGrp` をクラスタに追加します。このグループにはすべてのストレージとレプリケーションのリソースが含まれます。サービスグループの属性 `SystemList` および `AutoStartList` を修正して、`SystemList` および `AutoStartList` を設定します。

```
# hagrp -add VVRGrp
# hagrp -modify VVRGrp SystemList seattle1 0 seattle2 1
# hagrp -modify VVRGrp AutoStartList seattle1 seattle2
```

セカンダリクラスタで、seattle1 と seattle2 を london1 と london2 に置き換えます。

4. DiskGroup リソース `Hr_Dg` をサービスグループ `VVRGrp` に追加して、リソースの属性を修正します。

```
# hares -add Hr_Dg DiskGroup VVRGrp
# hares -modify Hr_Dg DiskGroup hrdg
```

5 NIC リソース vvrnic をサービスグループ VVRGrp に追加して、リソースの属性を修正します。

```
# hares -add vvrnic NIC VVRGrp
# hares -modify vvrnic Device qfe3
```

6 IP リソース vvrip をサービスグループ VVRGrp に追加して、リソースの属性を修正します。

```
# hares -add vvrip IP VVRGrp
# hares -modify vvrip Device qfe3
# hares -modify vvrip Address 192.168.40.20
# hares -modify vvrip NetMask "255.255.248.0"
```

セカンダリクラスタで、アドレスに適切な IP を使用します。たとえば次のように指定します。

```
# hares -modify vvrip Address 192.168.40.21
```

7 これまでの手順で追加したリソースの依存関係を指定します。

```
# hares -link Hr_Rvg vvrip
# hares -link Hr_Rvg Hr_Dg
# hares -link vvrip vvrnic
```

8 VVRGrp 内のすべてのリソースを有効化します。

```
# hagrp -enableresources VVRGrp
```

9 VCS の設定を保存して閉じます。

```
# haconf -dump -makero
```

プライマリクラスタ Seattle 内のシステム seattle1 で次の手順を実行してから、セカンダリクラスタ London 内のシステム london1 で同じ手順を繰り返します（説明に従って若干変更します）。

アプリケーションサービスグループを作成する方法

1 root でログインします。

2 次のコマンドを発行して、VCS 設定モードを読み書き両用にします。

```
# haconf -makerw
```

3 サービスグループ ORAGrp をクラスタ Seattle に追加します。このグループには、アプリケーション固有のすべてのリソースが含まれます。サービスグループの属性 SystemList、AutoStartList、ClusterList を設定します。

```
# hagrp -add ORAGrp
# hagrp -modify ORAGrp SystemList seattle1 0 seattle2 1
# hagrp -modify ORAGrp AutoStartList seattle1 seattle2
# hagrp -modify ORAGrp ClusterList  Seattle 0 London 1
```

セカンダリで、次のように seattle1 と seattle2 を london1 と london2 に置き換えます。

```
# hagrp -add ORAGrp
# hagrp -modify ORAGrp SystemList london1 0 london2 1
# hagrp -modify ORAGrp AutoStartList london1 london2
# hagrp -modify ORAGrp ClusterList  Seattle 0 London 1
```

4 NIC リソース oranic をサービスグループ ORAGrp に追加して、リソースの属性を修正します。

```
# hares -add oranic NIC ORAGrp
# hares -modify oranic Device hme0
```

5 IP リソース oraip をサービスグループ ORAGrp に追加して、リソースの属性を修正します。

```
# hares -add oraip IP ORAGrp
# hares -modify oraip Device hme0
# hares -modify oraip Address 192.168.40.1
# hares -modify oraip NetMask "255.255.248.0"
```

セカンダリで、IP リソース用のアドレス属性を適切に修正します。

6 RVG リソース Hr_Rvg 内の Volume リソース hr_dv01 をマウントするために Mount リソース Hr_Mount01 を追加します。

```
# hares -add Hr_Mount01 Mount ORAGrp
# hares -modify Hr_Mount01 MountPoint /hr_mount01
# hares -modify Hr_Mount01 BlockDevice /dev/vx/dsk/Hr_Dg/hr_dv01
# hares -modify Hr_Mount01 FSType vxfs
# hares -modify Hr_Mount01 FsckOpt %-n
# hares -modify Hr_Mount01 MountOpt rw
```

7 RVG リソース Hr_Rvg 内の Volume リソース hr_dv02 をマウントするために Mount リソース Hr_Mount02 を追加します。

```
# hares -add Hr_Mount02 Mount ORAGrp
# hares -modify Hr_Mount02 MountPoint /hr_mount02
# hares -modify Hr_Mount02 BlockDevice /dev/vx/dsk/Hr_Dg/hr_dv02
# hares -modify Hr_Mount02 FSType vxfs
# hares -modify Hr_Mount02 FsckOpt %-n
# hares -modify Hr_Mount02 MountOpt rw
```

8 RVG リソース Hr_Rvg 内のボリュームセット hr_vset01 をマウントするために Mount リソース Hr_Mount03 を追加します。

```
# hares -add Hr_Mount03 Mount ORAGrp
# hares -modify Hr_Mount03 MountPoint /hr_mount03
# hares -modify Hr_Mount03 BlockDevice /dev/vx/dsk/ Hr_Dg/hr_vset01
# hares -modify Hr_Mount03 FSType vxfs
# hares -modify Hr_Mount03 FsckOpt %-n
# hares -modify Hr_Mount03 MountOpt rw
```

9 Oracle リソースの Hr_Oracle を追加します。

```
# hares -add Hr_Oracle Oracle ORAGrp
# hares -modify Hr_Oracle Sid hr1
# hares -modify Hr_Oracle Owner oracle
# hares -modify Hr_Oracle Home "/hr_mount01/OraHome1"
# hares -modify Hr_Oracle Pfile "inithr1.ora"
# hares -modify Hr_Oracle User dbtest
# hares -modify Hr_Oracle Pword dbtest
# hares -modify Hr_Oracle Table oratest
# hares -modify Hr_Oracle MonScript "./bin/Oracle/SqlTest.pl"
# hares -modify Hr_Oracle StartUpOpt STARTUP
# hares -modify Hr_Oracle ShutDownOpt IMMEDIATE
# hares -modify Hr_Oracle AutoEndBkup 1
```

10 Oracle リスナーリソースの LISTENER を追加します。

```
# hares -add LISTENER Netlsnr ORAGrp
# hares -modify LISTENER Owner oracle
# hares -modify LISTENER Home "/hr_mount01/OraHome1"
# hares -modify LISTENER Listener LISTENER
# hares -modify LISTENER EnvFile "/oracle/.profile"
# hares -modify LISTENER MonScript "./bin/Netlsnr/LsnrTest.pl"
```

11 RVGPrimary リソースの Hr_RvgPri を追加します。

```
# hares -add Hr_RvgPri RVGPrimary ORAGrp
# hares -modify Hr_RvgPri RvgResourceName Hr_Rvg
```

12 これまでの手順で追加したリソースの依存関係を指定します。

```
# hares -link LISTENER Hr_Oracle
# hares -link LISTENER oraip
# hares -link Hr_Oracle Hr_Mount01
# hares -link Hr_Oracle Hr_Mount02
# hares -link Hr_Mount01 rvg-pri
# hares -link Hr_Mount02 rvg-pri
# hares -link Hr_Mount03 rvg-pri
# hares -link oraip oranic
```

13 この手順を行う前に、アプリケーションサービスグループとレプリケーションサービスグループの両方が存在している必要があります。まだレプリケーションサービスグループを作成していない場合は、ここで作成してください。

p.41 の「レプリケーションサービスグループを作成する方法」を参照してください。

アプリケーションサービスグループとレプリケーションサービスグループを作成した後、ORAGrp と VVRGrp 間で online local hard グループの依存関係を指定します。

```
# hagrp -link ORAGrp VVRGrp online local hard
```

14 ORAGrp 内のすべてのリソースを有効化します。

```
# hagrp -enableresources ORAGrp
```

15 VCS の設定を保存して閉じます。

```
# haconf -dump -makero
```

16 サービスグループがオンラインでない場合は、オンラインにします。

```
# hagrp -online VVRGrp -sys seattle1
# hagrp -online ORAGrp -sys seattle1
```

17 次のコマンドを実行して、seattle1 でサービスグループ ORAGrp がオンラインであることを確認します。

```
# hagrp -state ORAGrp
```

VCS 停止時のエージェントの設定

プライマリクラスタとセカンダリクラスタの最初のノードで、次の手順を実行し、サンプルの設定ファイルを使用しながらRVGエージェントを設定します。この例では、seattle1が最初のプライマリノードで、london1 が最初のセカンダリノードです。

VCS が停止している場合にエージェントを設定する方法

1. root でログインします。
2. 次のコマンドを使って、既存の設定に対する変更を確実に保存します。そして、main.cf を変更している間、各システムの状態、アプリケーションやシステムの設定などを変更しないようにします。

   現在、VCS クラスタが書き込み可能な場合は、次のコマンドを実行します。

   ```
   # haconf -dump -makero
   ```

   VCS クラスタがすでに読み込み専用の場合は、次のコマンドを実行します。

   ```
   # haconf -dump
   ```

3. VCS を起動している間は、設定ファイルを編集しないでください。次のコマンドを使うことにより、すべてのシステムで had デーモンが停止し、リソースを VCS 以外からコントロールすることが可能になります。

   ```
   # hastop -all -force
   ```

4. main.cf ファイルのバックアップコピーを作成します。

   ```
   # cd /etc/VRTSvcs/conf/config
   # cp main.cf main.cf.orig
   ```

5. プライマリとセカンダリクラスタ用の main.cf ファイルを編集します。/etc/VRTSvcs/conf/sample_vvr/RVGPrimary ディレクトリにあるファイル main.cf.seattle と main.cf.london は、それぞれプライマリクラスタとセカンダリクラスタの参考として使用できます。
6. ファイルを保存して閉じます。
7. /etc/VRTSvcs/conf/config/main.cf ファイルの構文を検証します。

   ```
   # hacf -verify /etc/VRTSvcs/conf/config
   ```

8 両方のクラスタ内のすべてのシステムで VCS を起動します。

p.34 の「プライマリとセカンダリの両方のクラスタ内のすべてのシステムで VCS を起動する方法」を参照してください。

9 サービスグループを管理します。

p.53 の「サービスグループの管理」を参照してください。

並列アプリケーションのエージェントの設定

共有環境の並列アプリケーションで使用される RVG を監視し管理するには、RVGShared エージェント、RVGSharedPri エージェントおよび RVGLogowner エージェントを使用します。

メモ: 各ノードで適当な期間にわたり vxstat コマンドを実行して、最も多くの書き込みを行うノードを判別し、その後、レプリケーションを設定してからそのノードをログ所有者として指定します。

エージェントを設定するための前提条件は次のとおりです。

- プライマリサイトとセカンダリサイト間でレプリケーションを設定する必要があります。
  共有環境でのレプリケーションの詳細については、『Veritas Volume Replicator 管理者ガイド』を参照してください。
- プライマリサイトとセカンダリサイトはグローバルクラスタ内に設定する必要があります。また、アプリケーションサービスはグローバルサービスグループとして設定する必要があります。
  グローバルクラスタの設定の詳細については、『Veritas Cluster Server ユーザーズガイド』を参照してください。

サンプル設定ファイルは、/etc/VRTSvcs/conf/sample_rac/ ディレクトリにあり、ファイル名の一部に CVR を含みます。これらのサンプルファイルは、VRTSdbac パッケージの一部としてインストールされ、ユーザー環境を作成する際に参考になります。エージェントは、コマンドラインまたは VCS Java コンソールおよび Web コンソールから設定できます。

詳細については、『Veritas Cluster Server ユーザーズガイド』を参照してください。

プライマリクラスタの VCS 設定を変更する方法

1 2 つのサービスグループ、RVGLogowner リソースを含むログ所有者グループと RVGShared リソースレプリケーションオブジェクトを含む RVG グループを新規に定義します。

2 ログ所有者グループで、RVG の RLINK で使用される IP および NIC リソースと、RVG とその関連ディスクグループが属性として定義される RVGLogowner リソースを定義します。

3 RVG サービスグループで、RVG リソースを監視する RVGShared エージェントを設定します。RVG サービスグループは共有グループなので、RVG は、共有ディスクグループおよびその起動モードを定義する CVMVolDg リソースに依存するように設定される必要があります。

パラレルサービスグループ内の RVGShared リソースと CVMVolDg リソースを定義して、サービスグループがすべてのクラスタノードで同時にオンラインになるようにします。

4 RVGSharedPri リソースを既存のアプリケーションサービスグループに追加して、サービスグループをグローバルグループに定義します。

グローバルグループの作成方法については、『Veritas Cluster Server ユーザーズガイド』を参照してください。

5 CVMVolDg リソースを既存のアプリケーションサービスグループから新規作成された RVGShared サービスグループに移動します。

6 サービスグループの依存関係を次のように設定します。

- RVG ログ所有者サービスグループは、RVG を含むサービスグループに対して、online local firm グループの依存関係を持ちます。
- RVG サービスグループは、CVM サービスグループに対して、online local firm グループの依存関係を持ちます。
- アプリケーションサービスグループは、RVG サービスグループに対して、online local firm グループの依存関係を持ちます。

セカンダリクラスタの VCS 設定を変更する方法

1 セカンダリクラスタのノードに root ユーザーとしてログインします。

2 次のコマンドを使って、既存の設定に対する変更を確実に保存します。そして、main.cf を変更している間、各システムの状態、アプリケーションやシステムの設定などを変更しないようにします。

現在、VCS クラスタが書き込み可能な場合は、次のコマンドを実行します。

```
# haconf -dump -makero
```

VCS クラスタがすでに読み込み専用の場合は、次のコマンドを実行します。

```
# haconf -dump
```

3 main.cf ファイルの編集中に VCS が実行されていないようにします。このためには、hastop コマンドを使用して、すべてのシステム上の VCS エンジンを停止してリソースを利用可能にします。

```
# hastop -all -force
```

4 main.cf ファイルのバックアップコピーを作成します。

```
# cd /etc/VRTSvcs/conf/config
# cp main.cf main.orig
```

5 vi またはその他のテキストエディタを使用して、main.cf ファイルで次の変更を行います。

- セカンダリクラスタの CVM グループを編集します。CVM グループの編集には、プライマリクラスタの CVM グループを参照してください。
- ログ所有者グループと RVG サービスグループを追加します。
- アプリケーションサービスグループを追加します。セカンダリクラスタのアプリケーションサービスグループのモデルとして、プライマリクラスタのアプリケーションサービスグループを参照してください。
- サービスグループはグローバルグループなので、プライマリクラスタのグループと同じ名前を割り当てます。
- ClusterList および ClusterFailOverPolicy クラスタ属性を定義します。
- RVGSharedPri リソースをサービスグループに含めます。

6 main.cf ファイルを保存して閉じます。

7 /etc/VRTSvcs/conf/config/main.cf ファイルの構文を検証します。

```
# hacf -verify /etc/VRTSvcs/conf/config
```

8 両方のクラスタ内のすべてのシステムで VCS を起動します。

p.34 の「プライマリとセカンダリの両方のクラスタ内のすべてのシステムで VCS を起動する方法」を参照してください。

このとき、プライマリクラスタ内の両方のシステムでアプリケーショングループがオンラインである必要があります。

このとき、セカンダリクラスタ内のアプリケーションサービスグループはオフラインであり、CVM、RVG ログ所有者および RVG グループはオンラインである必要があります。

バンカーレプリケーション設定用のエージェントの設定

この項では、バンカーレプリケーション設定、つまりバンカーサイトを含む RDS 用に VCS エージェントを設定する方法について説明します。バンカーは、STORAGE プロトコルまたは IP を使用して設定できます。

VCS エージェントの設定については、次のいずれかの項を参照してください。

- 「STORAGE プロトコルを使用したバンカー用の VCS 設定」
- 「IP を使用したバンカー用の VCS 設定」

STORAGE プロトコルを使用したバンカー用の VCS 設定

STORAGEプロトコルを使用してバンカーを設定すると、バンカーRVGを含むディスクグループがプライマリノードにインポートされます。プライマリRVGがVCSクラスタにある場合、プライマリRVGがオンラインになっているノードと同じノードで、バンカーRVGをオンラインのままにしておく必要があります。

共有ディスクグループ環境では、ログ所有者ノードでバンカーRVGをオンラインにしておく必要があります。

この項では、バンカーRVGのフェールオーバーを自動化するようにエージェントを設定する方法について説明します。

専用ディスクグループ環境では、RVGリソースがフェールオーバープロセスを処理します。RVGリソースがオンラインになっているノードに障害が発生すると、RVGリソースは、クラスタ内の別のノードへフェールオーバーします。RVGリソースにより、バンカーRVGもフェールオーバーします。このためバンカーRVGはプライマリRVGと同じノードに存続します。

共有ディスクグループ環境では、RVGLogownerエージェントがバンカーRVGのフェールオーバーを処理します。ログ所有者がフェールオーバーした場合、バンカーRVGをもとのログ所有者ノードからデポートし、新しいログ所有者ノードにインポートする必要があります。

バンカーRVGの自動フェールオーバーを設定するには、アプリケーションサービスグループまたはRVGLogownerエージェントで、RVGリソースの次の属性を使用して、バンカーRVG、バンカーディスクグループ、バンカーノードを指定します。

表 2-1　　バンカーフェールオーバーを設定するための属性

属性	説明
StorageDG	バンカーRVGの名前です。
StorageRVG	バンカーRVGの名前です。
StorageHostIds	バンカーノードのホストID、またはバンカーがクラスタ化されている場合は、バンカークラスタ内の各ノードのホストIDをスペースで区切ったリストです。

この項で説明するバンカーフェールオーバー属性は、バンカーを含むRDSの属性とは異なる専用の属性です。VCSAgentの残りの設定は、他のRDSの設定と同じです。

p.36 の「例 - VCS 環境における VVR の設定」を参照してください。

バンカーレプリケーション環境での VCS エージェントのサンプル設定ファイル

次の例は、STORAGE プロトコルを使用してバンカーセカンダリをプライマリに接続するサンプル設定ファイルです。

この例では次の名前を使用します。

- seattle: プライマリクラスタノード
- london: バンカーノード
- bdg : バンカーディスクグループ名
- brvg: バンカー RVG 名

サンプル設定ファイル（フェールオーバーアプリケーション）

次のサンプルファイルは、プライマリでの VCS エージェントの設定を示しています。RVG エージェントには STORAGE バンカーの属性が含まれており、バンカーディスクグループは親 RVG と一緒にフェールオーバーします。

この例では、プライマリ上のディスクグループは共有ディスクグループではありません。

RDS のセカンダリにバンカーが関連付けられている場合は、セカンダリ上の RVG エージェントにも同様に、StorageRVG、StorageDG、StorageHostIds の各属性が含まれます。

```
group AppSG (
        ClusterList = { cluster_london = 0 }
        SystemList = { seattle = 0, london = 1 }
        Authority = 1
        AutoStartList = { seattle }
        ClusterFailOverPolicy = Manual
        )
        RVG RVG-1 (
     RVG = vcsrvg
     DiskGroup = pdg
     Primary = true
     StorageRVG = brvg
     StorageDG = bdg
     StorageHostIds = "portland"
                    )
...
```

サンプル設定ファイル（並列アプリケーション）

次のサンプルファイルは、プライマリでの VCS エージェントの設定を示しています。RVGLogowner エージェントには STORAGE バンカーの属性が含まれており、バンカー

ディスクグループはログ所有者と一緒にフェールオーバーします。この例では、プライマリ上のディスクグループは共有ディスクグループです。RDS のセカンダリにバンカーが関連付けられている場合は、セカンダリ上の RVGLogowner リソースにも同様に、StorageRVG、StorageDG、StorageHostIds の各属性が含まれます。

```
group RVGLogownerGrp (
        SystemList = { seattle = 0, london = 1 }
        AutoStartList = { seattle, london }
        )
        IP vvr_ip (
                Device = bge0
                Address = "192.168.3.13"
                )
        NIC vvr_nic (
                Device = bge0
                )
        RVGLogowner vvr_rvglogowner (
    RVG = rvg
    DiskGroup = vvrdg
    StorageRVG = brvg
    StorageDG = bdg
    StorageHostIds = "portland"
    )
  requires group RVGSharedGrp online local firm
        vvr_ip requires vvr_nic
```

IP を使用したバンカー用の VCS 設定

IP を使ったバンカー用の VCS エージェントの設定は、他のセカンダリ用の設定と同じです。

バンカー設定を行う方法

1 プライマリとセカンダリ設定は、VCS エージェントを使用した他の RDS の場合と同じです。

p.36 の「例 - VCS 環境における VVR の設定」を参照してください。

2 `vradmin addbunker` コマンドでバンカーを RDS に追加します。

詳細な手順については、『Veritas Volume Replicator 管理者ガイド』を参照してください。

3 他のセカンダリの設定と同様にして、バンカー上で VCS エージェントを設定します。IP を使ったバンカーに対して必要な特別な設定はありません。

サービスグループの管理

この項では、クラスタ Seattle の VCS サービスグループをコマンドラインから管理する方法について説明します。サービスグループの管理には、VCS Java コンソールおよび Web コンソールも使用できます。

VCS サービスグループを管理する方法

1 seattle1 で VCS エンジンを起動します。

```
# hastart
```

2 RVG リソースタイプを含むサービスグループのすべてがオンラインになっていることを確認します。

```
# hagrp -display
```

3 サービスグループをオフラインにし、すべてのリソースが停止していることを確認します。

```
# hagrp -offline hr_grp -sys seattle1
# hagrp -display
```

4 サービスグループを再びオンラインにし、すべてのリソースが利用可能であることを確認します。

```
# hagrp -online hr_grp -sys seattle1
# hagrp -display
```

5 seattle2 で VCS エンジンを起動します。

```
# hastart
```

6 VVR サービスグループを seattle2 に切り替えます。

```
# hagrp -switch hr_grp -to seattle2
```

7 RVG リソースタイプを含むサービスグループすべてが、seattle2 でオンラインになっていることを確認します。

```
# hagrp -display
```

8 クラスタ London に対して手順 1 から手順 7 を繰り返します。

9 必要に応じて、状態やエラーに関する各システムのログファイルを点検します。

/var/VRTSvcs/log/engine_A.log

/var/VRTSvcs/log/RVG_A.log

索引

あ

か

さ

た

は

ら