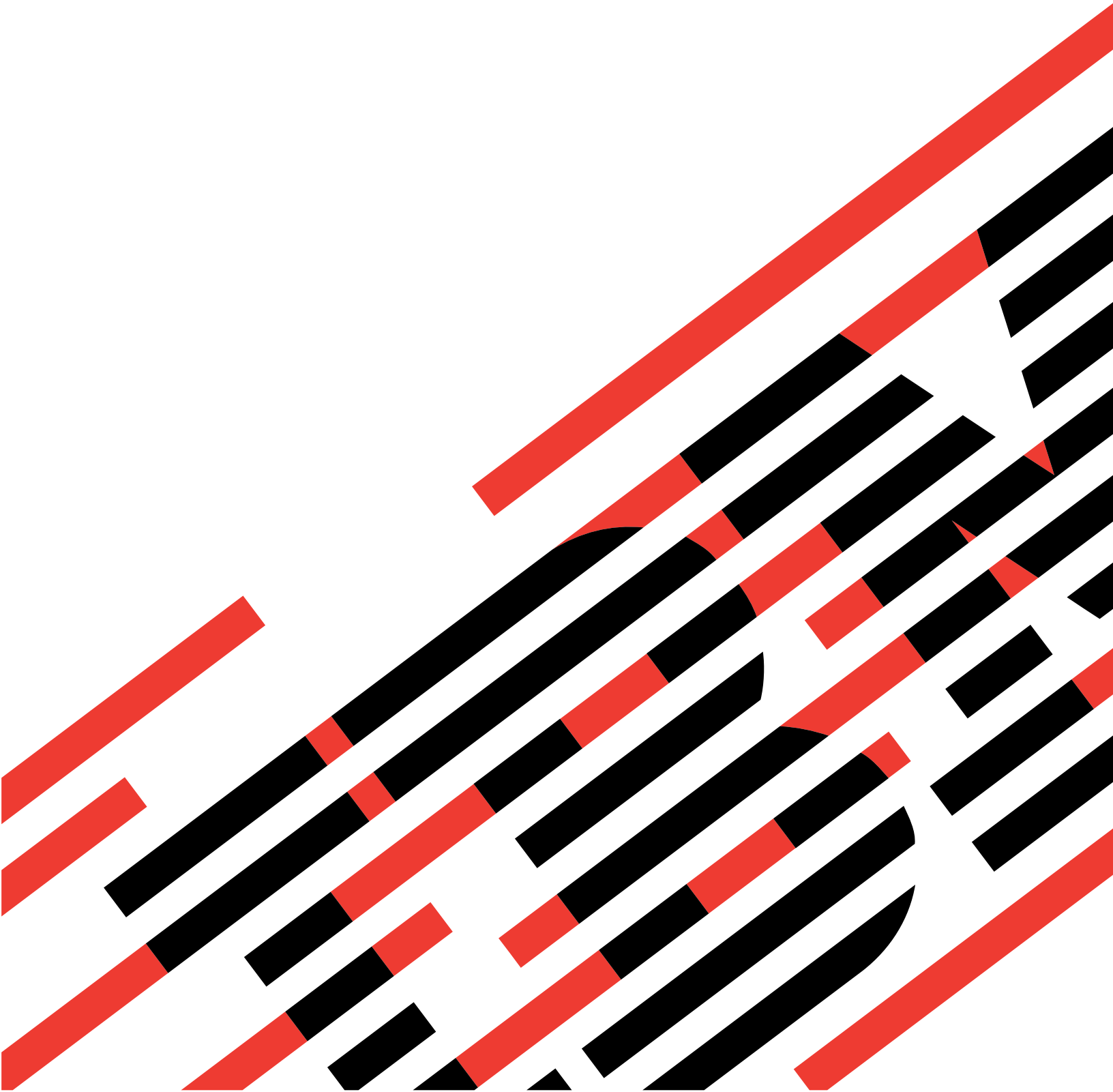
IBM Systems - iSeries

システム管理 クラスター

バージョン 5 リリース 4



IBM Systems - iSeries

システム管理 クラスター

バージョン 5 リリース 4

ご注意!

本書および本書で紹介する製品をご使用になる前に、 173 ページの『特記事項』に記載されている情報をお読みください。

本書は、IBM i5/OS (製品番号 5722-SS1) のバージョン 5、リリース 4、モディフィケーション 0 に適用されます。また、改訂版で断りがない限り、それ以降のすべてのリリースおよびモディフィケーションに適用されます。このバージョンは、すべての RISC モデルで稼働するとは限りません。また CISC モデルでは稼働しません。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

http://www.ibm.com/jp/manuals/main/mail.html

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

http://www.ibm.com/jp/manuals/ の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原 典: IBM Systems - iSeries
Systems management Clusters
Version 5 Release 4

発 行: 日本アイ・ビー・エム株式会社

担 当: ナショナル・ランゲージ・サポート

第1刷 2006.2

この文書では、平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財) 日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W7、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、平成角ゴシック体™W7

目次

クラスター

クラスターによって、iSeries サーバーを効率的にグループ化し、重要なアプリケーション、装置、およびデータに 100 % に近い可用性を提供する環境をセットアップすることができます。

また、クラスターは、単純化されたシステム管理を提供しスケーラビリティーを増大させるので、ビジネスの成長につれて、新しいコンポーネントをシームレスに追加することができます。

このコード・サンプルを使用すると、「コードに関するライセンス情報および特記事項」の条件に同意したことになります。

V5R4 の新機能

このリリースの新機能をごらんください。

クラスター管理可能ドメインのサポート

クラスター管理可能ドメインは、クラスター内の選択されたリソースに対する変更をモニターして同期します。クラスター管理可能ドメインを使えば、環境変数やユーザー・プロファイルなどの、クラスター内で共用されるリソース属性をさらに簡単に管理および同期できるようになります。クラスター管理可能ドメインの詳細は、以下のトピックを参照してください。

- 10 ページの『クラスター管理ドメイン』
- 109 ページの『クラスター管理可能ドメインの計画』
- 110 ページの『クラスター管理ドメイン・チェックリスト』
- 123 ページの『クラスター管理可能ドメインの作成』

対等クラスター・リソース・グループ (CRG) のサポート

すべての CRG インターフェースは、対等クラスター・リソース・グループ (CRG) をサポートするように機能拡張されました。対等クラスター・リソース・グループ (CRG) とは、切り替え不能な CRG のことであり、そこではリカバリー・ドメイン内のどのノードも、対等 CRG に関連したリソースの回復においては同等の役割をもっています。詳細は、以下のトピックを参照してください。

- クラスター・リソース・グループ
- 117 ページの『CRG の作成』
- 118 ページの『CRG の開始』

クラスターの機能強化

クラスター環境での電源遮断操作と問題解決を改善するためのいくつかの機能強化が行われました。そのような改善事項には次のものがあります。

- アクティブなすべてのサブシステムの終了時か、またはシステムの終了または電源遮断時に、クラスター・ノード上のクラスタリングを終了するための系統的なアプローチ。詳しくは、 123 ページの『クラスターに対するシステム・イベントの影響の仕方』を参照してください。
- アクティブなテークオーバー IP アドレスを使った新規アプリケーション CRG を構成する機能。 詳しくは、 118 ページの『アクティブなテークオーバー IP アドレスを使ったアプリケーション CRG の作成』を参照してください。

- アクティブ・ノードからクラスター全体とそれに関連した CRG を表示して、クラスター問題をトラブルシューティングする機能。詳細は、 140 ページの『クラスターに関する回復情報の収集』を参照してください。
- デバッグ・ツールとそれによって生成される結果に関する新情報が追加されました。このようなツールとその結果を使用して、クラスター内の問題の解決法を判別することができます。詳細は、以下のトピックを参照してください。
  - 141 ページの『クラスター・トレースのダンプ (DMPCLUTRC) コマンドでの問題の調査』
  - 145 ページの『CLUSTERINFO マクロを使った問題の調査』

新機能または変更点を確認する方法。

技術的に変更された箇所を見分けるには、以下の情報を用いてください。

- 新しい情報または変更された情報が始まる箇所に、 » が付けられています。
- 新しい情報または変更された情報が終わる箇所に、 « が付けられています。

このリリースでの新機能または変更点に関するその他の情報を見つけるには、「プログラム資料説明書」を参照してください。

印刷可能な PDF

これを使用して、本資料の PDF を表示して印刷してください。

本資料の PDF 版を表示またはダウンロードするには、クラスター (約 938 KB) を選択します。

Redbooks™

- Clustering and IASPs for Higher Availability  (約 6.4 MB または 330 ページ)。このレッドブックは、 クラスターおよび iSeries™ サーバーで使用可能な切り替えディスク・テクノロジーの概要を示します。

- iSeries Independent ASPs: A Guide to Moving Applications to IASPs  (約 3.4 MB)。このレッドブックは、iSeries サーバー上での独立 ASP に対する段階的アプローチを解説しています。

- Roadmap to Availability on the iSeries 400®  (約 626 KB)。この Redpaper は、iSeries サーバー上での独立 ASP に対する段階的アプローチを解説しています。

Web サイト

- High Availability and Clusters  (www.ibm.com/servers/eserver/iseries/ha)

  高可用性およびクラスターに関する IBM® サイト

PDF ファイルの保存

表示または印刷のために PDF をワークステーションに保存するには、以下のようにします。

1. ブラウザーで PDF を右マウス・ボタンでクリックする (上部のリンクを右マウス・ボタン・クリック)。
2. Internet Explorer を使用している場合は、「**対象をファイルに保存...**」をクリックする。 Netscape Communicator を使用している場合は、「**リンクを名前を付けて保存...**」をクリックする。

3. PDF の保存先のディレクトリーに移動する。
4. 「**保存**」をクリックする。

Adobe Acrobat Reader のダウンロード

これらの PDF を表示または印刷するには、Adobe Acrobat Reader が必要です。このアプリケーションは、Adobe Web サイト (www.adobe.com/products/acrobat/readstep.html) からダウンロードできます。

クラスターの概念

クラスターがどのように動作するのかに関する完全な理解が得られます。クラスターの利点、 クラスターがなぜ重要なのか、および重要なクラスタリングのいくつかの概念とそれらが一緒になってどのように要件に適合するのかについてお読みください。

iSeries クラスターとは、1 つの同じシステムとして機能する 1 つ以上のシステムまたは論理区画の集まり、すなわちグループのことです。クラスターにあるシステムのことをクラスター・ノードと呼び、協調して動作することにより単一のコンピューター・ソリューションを提供します。iSeries のクラスタリングでは、1 つのクラスターに 128 のノードまで含めることができます。この機能により、iSeries システムを効率的にグループ化し、重要なアプリケーションおよび重要なデータに 100 % に近い可用性を提供する環境をセットアップすることができます。これは、重要なシステムおよびアプリケーションが、1 日 24 時間中使用可能であることを保証するのに役立ちます。また、クラスターは、単純化されたシステム管理を提供しスケーラビリティーを増大させるので、ビジネスの成長につれて、新しいコンポーネントをシームレスに追加することができます。

クラスターの利点

クラスターは、システムが毎日 24 時間稼働し続けることが必要なビジネスで、その威力を発揮します。

| クラスタリングを使用することにより、計画外の停止回数と使用不能時間、および計画停止中の使用不能時
| 間が大幅に減少します。その結果として、システム、データ、アプリケーションの連続可用性が確実に向上
| します。

クラスターが提供する主なビジネス上の利点は次のとおりです。

連続可用性

クラスターにより、ご使用のシステム、データ、アプリケーションの連続可用性が向上します。

単純化された管理

| それぞれのシステムにサインオンしなくても、システムのグループを 1 つのシステムまたはデー
| タベースとして管理できます。クラスター管理ドメインを使用することにより、クラスター内で共
| 有されているリソースをより容易に管理することができます。

増大したスケーラビリティー

ビジネスの拡大に応じて新しい構成要素をシームレスに追加できます。

関連概念

20 ページの『フェイルオーバー』
フェイルオーバーは、システム障害が発生したときに、クラスターのサーバーが 1 つ以上のバックアップ・サーバーへ自動的に切り替わるときに発生します。

関連タスク

24 ページの『切り替え』
切り替えは、あるサーバーから別のサーバーにアクセスを手動で切り替えるときに発生します。

クラスターはどのように動作するか

i5/OS™ の一部として提供されるクラスター・インフラストラクチャーは、クラスター・リソース・サービスと呼ばれ、クリティカル・リソースの回復機能を提供します。 これらのリソースには、データ、アプリケーション、装置、および複数のクライアントからアクセス可能なリソースを含めることができます。

システムやサイトで障害が発生しても、クラスター内のシステムで提供されている機能には、クラスターで定義されているその他のシステムを介してアクセスすることができます。アクセス可能なデータにはプライマリー・バックアップ・モデル、および対等モデルの 2 種類があります。これらのモデルを基に作成できるクラスター・リソース・グループ (CRG) の詳細については、 「クラスター・リソース・グループ」を参照してください。

関連概念

20 ページの『フェイルオーバー』
フェイルオーバーは、システム障害が発生したときに、クラスターのサーバーが 1 つ以上のバックアップ・サーバーへ自動的に切り替わるときに発生します。

30 ページの『複製』
複製とは、何かのコピーをリアルタイムで作成することです。つまり、クラスター内のあるノードに含まれているオブジェクトを、同じクラスター内にある他の 1 つ以上のノードにコピーすることを意味します。

18 ページの『回復装置』
回復装置とは、装置記述などの構成オブジェクトによって表されている物理リソースのうち、クラスター内の複数のノードからアクセスできる装置を指します。

18 ページの『回復データ』
回復データとは、 クラスター内の複数のノードに複製 (コピー) されたデータのことです。

25 ページの『再結合』
再結合とは、一度メンバーから外れた後に、再びクラスターの活動メンバーになることです。

97 ページの『論理複製、切り替えディスク、およびサイト間ミラーリングの比較』
このトピックでは、高可用性を高めるためにクラスターと共に使用可能な、さまざまなデータ回復テクノロジーの概要を解説します。

関連タスク

24 ページの『切り替え』
切り替えは、あるサーバーから別のサーバーにアクセスを手動で切り替えるときに発生します。

クラスターの基本

要件を満たすようクラスターを設計しカスタマイズするためには、始める前に基本的なクラスタリングの概念を理解する必要があります。

クラスターに関連した基本的な概念が 2 つあります。クラスター・ノード、およびクラスター・リソース・グループです。クラスター・ノードとは、クラスターのメンバーになっている iSeries システムまたは論理区画のことです。クラスターを作成するとき、クラスターにノードとして組み込みたいシステム、または論理区画を指定します。クラスター・リソース・グループ (CRG) は、回復リソースのコレクションの制御オブジェクトとして機能します。 CRG ではクラスター内にノードのサブセットまたはノードのすべてが含まれている場合があります。 iSeries クラスターでは次の 4 タイプの CRG がサポートされていま

す。アプリケーション、データ、デバイス、対等です。CRG のこれらのタイプには、共通要素が 2 つあります。リカバリー・ドメインと出口プログラムです。

リカバリー・ドメインは CRG で各ノードの役割を定義します。 クラスターに CRG を作成するときに、リカバリー・ドメインに組み込むよう指定されたすべてのノードに CRG オブジェクトが作成されます。しかし、CRG オブジェクトの単一システム・イメージも提供されています。これは、CRG のリカバリー・ドメインにあるどのアクティブ・ノードからでもアクセスできるものです。つまり、CRG に行われたどんな変更も、リカバリー・ドメインにあるすべてのノードに対して行われるということです。

出口プログラムは CRG のクラスター関連イベント中に呼び出されます。このようなイベントは、1 つのノードから他のノードにアクセス・ポイントを移動します。

クラスターで作成できる CRG のモデルには 2 種類あります。プライマリー・バックアップ・モデルと対等モデルです。プライマリー・バックアップ・モデルでは、CRG のリカバリー・ドメインにあるノードを以下のように定義できます。

- プライマリー・ノードは、回復クラスター・リソースの 1 次アクセス・ポイントであるクラスター・ノードです。
- バックアップ・ノードは、現在のプライマリー・ノードで障害が発生した場合に、1 次アクセスの役割を引き継ぐクラスター・ノードです。
- 複製ノードは、クラスター・リソースのコピーを含んだクラスター・ノードですが、プライマリー・ノードやバックアップ・ノードの役割は果たしません。

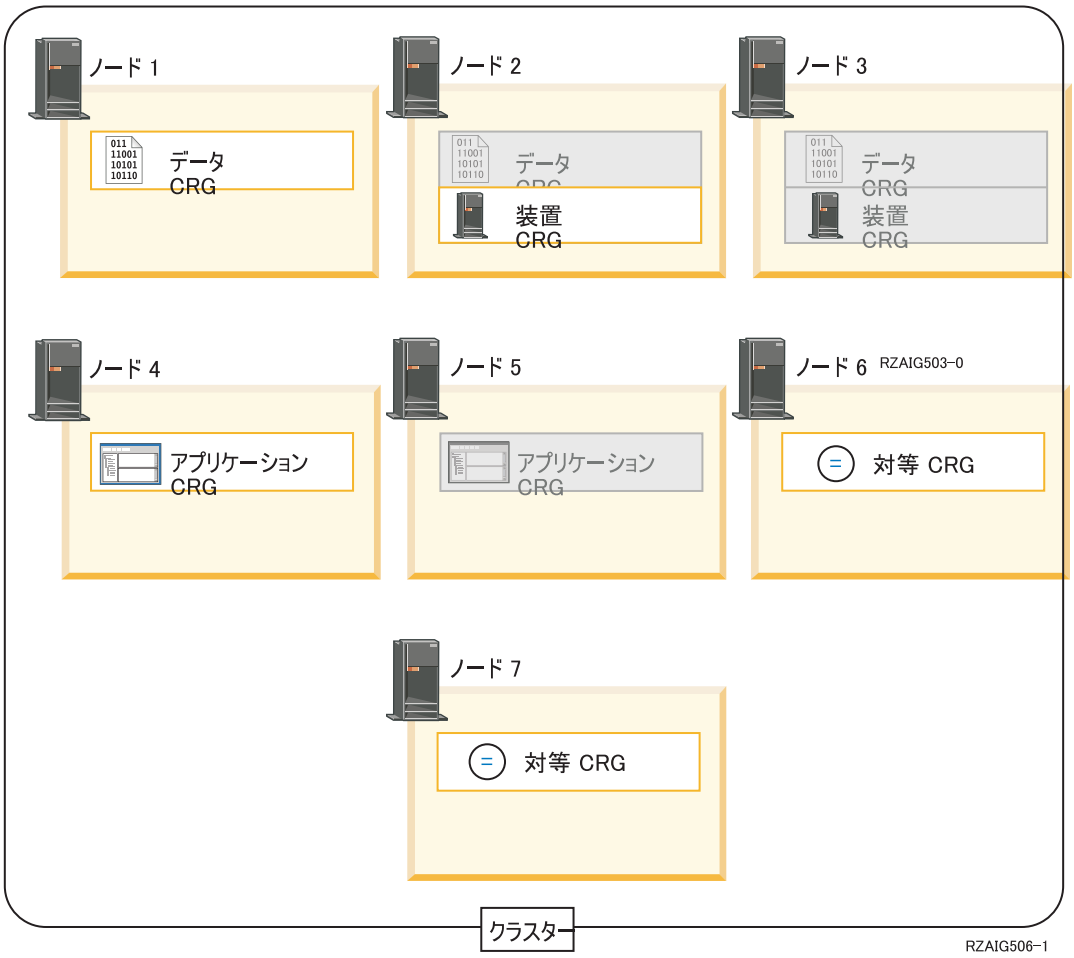
対等モデルでは、対等 CRG のリカバリー・ドメインが各ノード間の対等関係を定義します。対等 CRG のリカバリー・ドメインにあるノードは以下のように定義できます。

- 対等ノードとは、クラスター・リソースのアクティブなアクセス・ポイントになり得るクラスター・ノードです。
- 複製ノードとは、クラスター・リソースのコピーを含むクラスター・ノードです。対等 CRG で複製として定義されたノードは、クラスター・リソースのアクティブでないアクセス・ポイントを示します。

対等 CRG を用いた場合、リカバリー・ドメインのノードは、リカバリーでノードが実行する役割に等しくなります。 これは、この対等 CRG にある各ノードが基本的に同じ役割を持つためで、フェイルオーバーと切り替えの概念は適用されません。 各ノードは対等な関係で、ノードの 1 つに障害が発生すると、他の対等なノードが操作を継続します。

同様に、対等 CRG などのようなクラスター管理ドメインも作成できます。クラスター管理ドメインのノードはすべて、CRG のリカバリー・ドメインにある対等ノードになります。複製ノードはありません。

下記の例では、それぞれのタイプの CRG があります。



データ CRG

データ CRG はノード 1、ノード 2 およびノード 3 にあります。これは、データ CRG のリカバリー・ドメインが、ノード 1 (プライマリー)、ノード 2 (1 番目のバックアップ)、ノード 3 (2 番目のバックアップ) の役割を指定していることを意味しています。この例では、現在、ノード 1 が 1 次アクセス・ポイントになっています。 ノード 2 は、リカバリー・ドメイン内で 1 番目のバックアップとして定義されています。これは、ノード 2 に、論理複製により現行のリソースと同じ状態に保たれたリソースのコピーが含まれていることを意味しています。フェイルオーバーまたは切り替えが発生した場合は、ノード 2 が 1 次アクセス・ポイントになります。

アプリケーション CRG

アプリケーション CRG はノード 4 とノード 5 にあります。これは、アプリケーション CRG のリカバリー・ドメインが、ノード 4 とノード 5 を指定していることを意味します。この例では、現在、ノード 4 が 1 次アクセス・ポイントになっています。フェイルオーバーまたは切り替えが発生した場合は、ノード 5 がアプリケーションへの 1 次アクセス・ポイントになります。 テークオーバー IP アドレスが必要です。

対等 CRG

対等 CRG はノード 6 とノード 7 として表示されます。つまり、 対等 CRG のリカバリー・ドメインは、ノード 6 とノード 7 に指定されるということです。このサンプルでは、ノード 6 およびノード 7 は対等ノード、または複製ノードのいずれかになることができます。これが対等 CRG などのようなクラスター管理ドメインである場合、クラスター管理ドメインによりモニター

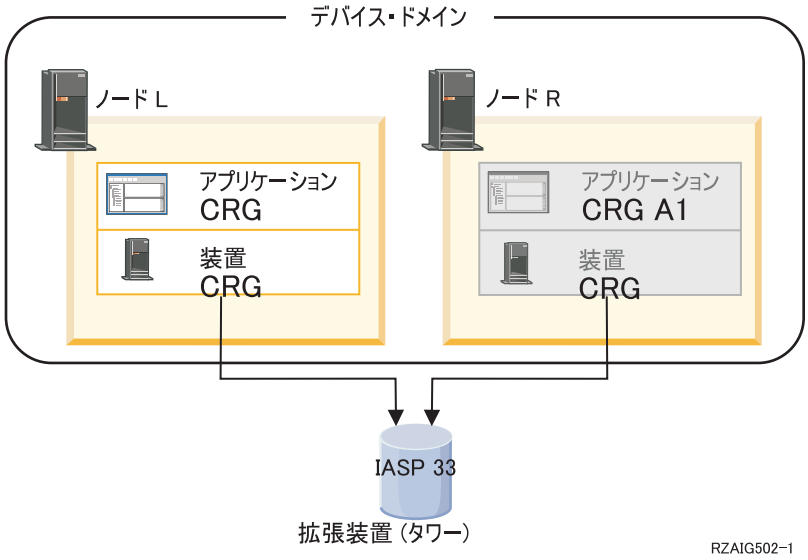
されているリソースは、変更の元になったノードとは関係なく、ノード 6 またはノード 7 として表されるドメイン全体で同期されたすべての変更が反映されます。

装置 CRG

装置 CRG はノード 2 とノード 3 にあります。 これは、装置 CRG のリカバリー・ドメインが、ノード 2 とノード 3 を指定していることを意味します。この例では、現在ノード 2 が 1 次アクセス・ポイントになっています。これは、装置 CRG に所有された回復装置が、現在、ノード 2 からアクセスできることを意味します。フェイルオーバーまたは切り替えが発生した場合は、ノード 3 が装置への 1 次アクセス・ポイントになります。

装置 CRG では、独立ディスク・プール (独立補助記憶域プールまたは独立 ASP とも呼ばれる) が外部装置、拡張装置 (タワー) または論理区画の IOP 上に構成されていることが必要です。

装置 CRG のリカバリー・ドメイン内のノードは、同じデバイス・ドメインのメンバーにもなっていなければなりません。次の例は、ノード L とノード R をリカバリー・ドメイン内に持つ装置 CRG を示しています。両方のノードが、同じデバイス・ドメインのメンバーにもなっています。



関連概念

8 ページの『クラスター・ノード』
クラスター・ノードとは、 クラスターのメンバーになっている iSeries システムまたは論理区画のことです。

8 ページの『クラスター・リソース・グループ』
クラスター・リソース・グループ (CRG) は、 i5/OS のシステム・オブジェクトであり、 クラスター化された環境で発生するイベントの管理に使用される、クラスター・リソースの一連のセット、またはグループです。 クラスター・リソース・グループはリカバリー・ドメインを記述するとともに、 特定のクラスター・イベントが発生する際に呼び出されるクラスター・リソース・グループ出口プログラムの名前を提供します。

13 ページの『リカバリー・ドメイン』
リカバリー・ドメインは、リカバリー・アクション、またはイベントの同期化を実行するなどの共通の目的を持ったクラスター内のノードを、1 つのクラスター・リソース・グループ (CRG) として集めたもののサブセットです。

12 ページの『クラスター・リソース・グループ出口プログラム』
クラスター・リソース・グループ出口プログラムは、CRG のクラスター関連イベントが発生した後、呼び出されます。

独立ディスク・プール

19 ページの『デバイス・ドメイン』
デバイス・ドメインとは、装置リソースを共用するクラスター内のノードのサブセットです。 もう少し具体的に説明すれば、デバイス・ドメイン内のノードは、回復リソースのコレクションに対するスイッチ・アクションに参加できます。

クラスターの要素

iSeries クラスターとは、1 つのシステムとして機能する 1 つ以上のシステム、または区画の集まりのことです。 この情報を使用して、各エレメント、並びにそれぞれの関係を理解します。

クラスター・ノード

クラスター・ノードとは、 クラスターのメンバーになっている iSeries システムまたは論理区画のことです。

各クラスター・ノードは、各 iSeries システムを表す 1 つ以上の IP アドレスに関連付けられた、8 文字のクラスター・ノード名により識別されます。クラスターを構成するときに、クラスターにあるノードに望む名前を付けられます。しかし、ノード名はホスト名、またはシステム名と同じにすることをお勧めします。

クラスター通信では、TCP/IP プロトコル・セットを使用して、クラスター内の各ノードにあるクラスター・サービス間の通信パスを提供します。クラスターの一部として構成されているクラスター・ノードの集合のことを、クラスター・メンバーシップ・リストといいます。

クラスター・リソース・グループ

クラスター・リソース・グループ (CRG) は、 i5/OS のシステム・オブジェクトであり、 クラスター化された環境で発生するイベントの管理に使用される、クラスター・リソースの一連のセット、またはグループです。 クラスター・リソース・グループはリカバリー・ドメインを記述するとともに、 特定のクラスター・イベントが発生する際に呼び出されるクラスター・リソース・グループ出口プログラムの名前を提供します。

クラスターではクラスター内の各ノード間の関係を定義する際、次の 2 つ選択肢を提供します。プライマリー・バックアップ・モデル、および対等モデルです。これらのモデルのいずれも、ご使用の環境の要件に応じて一緒に、または別々に使用できます。

プライマリー・バックアップ・モデル

このカテゴリーのすべてのクラスター・リソース・グループは、特定の役割 (プライマリー、バックアップ、または複製のいずれか) のリカバリー・ドメインでノードを定義します。プライマリー・ノードとバックアップ・ノードは、クラスター・リソースの使用可能なアクセス・ポイントです。ただし、指定されたポイントで 1 度にアクティブになるアクセス・ポイントは、1 つのノードのみです。これがプライマリー・ノードと呼ばれます。複製ノードがアクセス・ポイントになることはできません。複製ノードにバックアップの役割を割り当てることにより、変更することができます。プライマリー・バックアップ・モデルのクラスター・リソース・グループは、データ回復、アプリケーション回復、装置回復として定義されています。データの面での回復については、データの複数コピーがクラスター内の複数のノードで保守されるようになり、アクセス・ポイントがバックアップ・ノードに変更されるようになります。アプリケーションの面での

回復については、アプリケーション・プログラムがクラスター内の同じノードまたは別のノードで再始動できるようになります。装置の面での回復性については、装置リソースがバックアップ・ノードに移動する (切り替えられる) ことになります。

データとアプリケーションのどのクラスター・リソース・グループにも、クラスター・リソース・グループ出口プログラムが 1 つずつ関連付けられます。回復装置のクラスター・リソース・グループについては、出口プログラムを関連付けるかどうかはオプションです。

iSeries ナビゲーターでは、クラスター・リソース・グループは異なる名前で呼ばれます。

- 装置 CRG は **切り替え可能装置**と呼ばれます。
- アプリケーション CRG は**切り替え可能アプリケーション**と呼ばれます。
- データ CRG は、**切り替え可能データ・グループ**と呼ばれます。

対等モデル

このカテゴリーのすべてのクラスター・リソース・グループは、リカバリー・ドメインにあるノードを、対等または複製の役割で定義しています。対等ノードはクラスター・リソース・グループのアクセス・ポイントにもなります。対等ノードとして定義されたすべてのノードは、クラスター・リソース・グループの開始時にはアクセス・ポイントになります。複製ノードがアクセス・ポイントになることはできません。複製ノードに対等の役割を割り当てることにより、変更することができます。対等 CRG 内では、それぞれのノードに存在する複製データがノードに含まれます。対等 CRG でノードに障害が発生した場合、障害が発生した時点でクラスター内の他のノードに通知され、それらのノードが障害が発生した時点から操作を引き継ぎます。

クラスター管理ドメインとしては、対等ノードのみで作られたリカバリー・ドメインを持つ対等 CRG などが挙げられます。

関連概念

13 ページの『リカバリー・ドメイン』
リカバリー・ドメインは、リカバリー・アクション、またはイベントの同期化を実行するなどの共通の目的を持ったクラスター内のノードを、1 つのクラスター・リソース・グループ (CRG) として集めたもののサブセットです。

12 ページの『クラスター・リソース・グループ出口プログラム』
クラスター・リソース・グループ出口プログラムは、CRG のクラスター関連イベントが発生した後、呼び出されます。

クラスター・リソース・グループの処理の管理

ノードに障害が発生すると、フェイルオーバーが実行されます。 CRG の順序付けが発生すると、すべての装置 CRG が先にフェイルオーバーを実行し、続いてデータ CRG、最後にアプリケーション CRG がフェイルオーバーされます。 対等 CRG には順序はありませんが、障害が発生すると各ノードに通知されます。

クラスター・リソース・グループのフェイルオーバーまたは切り替えが完了したことを確認するには、CRG の状況をチェックします。

データが処理できるようになるまで、ブロッキングを行ってアプリケーションを保留にしておくこともできます。回復データ・クラスター・リソース・グループが処理されている間は、そのデータ CRG が表しているデータにアクセスできないようにしておくと便利な場合があります。そのようにしてアクセスをブロックする場合は、EDRS アクセスのブロック (QxdaBlockEDRS) API および EDRS ブロック状況の検査

(QxdaCheckEDRSBlock) API を使用します。フェイルオーバーまたは切り替えが実行された場合には、これらの API を使用すれば、クラスター・リソース・グループ出口プログラム内からアクセスをブロックしたり、非ブロックしたりできます。

関連概念

20 ページの『フェイルオーバー』
フェイルオーバーは、システム障害が発生したときに、クラスターのサーバーが 1 つ以上のバックアップ・サーバーへ自動的に切り替わるときに発生します。

13 ページの『リカバリー・ドメイン』
リカバリー・ドメインは、リカバリー・アクション、またはイベントの同期化を実行するなどの共通の目的を持ったクラスター内のノードを、1 つのクラスター・リソース・グループ (CRG) として集めたもののサブセットです。

12 ページの『クラスター・リソース・グループ出口プログラム』
クラスター・リソース・グループ出口プログラムは、CRG のクラスター関連イベントが発生した後、呼び出されます。

関連タスク

24 ページの『切り替え』
切り替えは、あるサーバーから別のサーバーにアクセスを手動で切り替えるときに発生します。

クラスター管理ドメイン

クラスター管理ドメインはクラスター化された環境のノード全体で、 常に保守を必要とするリソースを管理するために使用されます。

各ノードには、回復データに対するアクセス・ポイント、アプリケーション、装置などのような、各ノードで定義されなければならない特定の操作パラメーターや、構成パラメーターを作成することができます。仮にいずれかのアクセス・ポイントになり得る任意のノード上で、これらのパラメーターの 1 つに変更が加えられた場合、この変更は潜在的にアクセス・ポイントになり得る他のすべてのノードに伝搬されなければなりません。クラスター管理ドメインには、ドメイン内にあるノード全体で、常に保守を必要とするリソースを識別する機能が備わっています。そのため、クラスター管理ドメインはこれらのリソースへの変更をモニターし、アクティブなドメイン全体で、変更点の同期を取ります。クラスター管理ドメインとしては、対等 CRG などが挙げられます。クラスター管理ドメインが作成されると、システムにより対等 CRG が作成されます。クラスター管理ドメインの名前が、作成された対等 CRG の名前になります。クラスター管理ドメインを作成するノードは、対等 CRG のリカバリー・ドメインによって定義されます。すべてのノードは対等ノードです。クラスター管理ドメインでは、複製ノードは許可されていません。クラスター・ノードは、クラスター内にあるクラスター管理ドメインでのみ定義できます。クラスター管理ドメインに関連するタスクの詳細については、以下のトピックを参照してください。

1. 109 ページの『クラスター管理可能ドメインの計画』
2. 110 ページの『クラスター管理ドメイン・チェックリスト』
3. 123 ページの『クラスター管理可能ドメインの作成』
4. 124 ページの『モニター対象ソース項目の追加』
5. 118 ページの『CRG の開始』

いったんクラスター管理ドメインが作成されると、そのクラスター管理ドメインを管理するために、通常の CRG 機能が使用されます。たとえば、管理ドメインにノードを追加する場合は、対等のノード役割を持つ CRG のリカバリー・ドメインに対してノードを追加する必要があります。クラスター管理ドメインを開始するには、対等 CRG を開始します。

CRG を開始したり、終了したりすることにより、変更の同期処理を制御することができます。 CRG が終了すると、ドメイン内にあるすべてのノード上でモニターされているリソースに加えられた変更は、残りのドメインに伝搬されません。 CRG 開始後は、非アクティブ状態の間に実行されたモニター対象リソースへの変更が、残りのドメインに伝搬されます。 CRG がアクティブの場合、アクティブ・ノード上のモニター対象リソースに加えられた変更は動的に伝搬されるため、管理ドメイン全体でリソースの整合性は保たれます。詳しくは、 124 ページの『クラスター管理可能ドメインのモニター』を参照してください。

クラスター管理ドメインにノードを追加するには、対等 CRG のリカバリー・ドメインにクラスター・ノードを追加する必要があります。ドメインにノードが追加されると、管理対象になるすべてのリソースが、新規ノード上に作成され (存在しない場合)、残りの管理ドメイン全体で同期されます。

クラスター管理ドメインが削除されると、クラスター管理ドメインに定義されていたすべてのリソースが、ドメインにあるすべてのノードから除去されます。ただし、実際のリソースがシステムから除去されることはありません。詳しくは、「モニター対象リソース」を参照してください。

モニター対象リソース

モニター対象リソースとは、クラスター管理ドメインによりマージできるシステム・リソースのタイプです。 これらのリソースはクラスター管理ドメインで、 モニター対象リソース項目 (MRE) として表示されます。

クラスター管理ドメインにより同期されるリソースは、モニター対象リソース項目 (MRE) として表示されます。 MRE がクラスター管理ドメインに追加されると、クラスター管理ドメインにある任意のノード上のリソースに加えられた変更は、アクティブなドメインのすべてのノードに伝搬されます。クラスター管理ドメインで MRE を管理するには、3 種類の Integrated Operating Environments API (統合オペレーティング環境 API) を使用できます。

- モニター対象リソース項目の追加 (QfpadAddMonitoredResourceEntry) API
- モニター対象リソース項目の除去 (QfpadRmvMonitoredResourceEntry) API
- モニター対象リソース情報の検索 (QfpadRtvMonitoredResourceInfo) API

モニター対象リソース項目は以下に示すリソースのタイプのクラスター管理に追加できます。

- システム値
- ユーザー・プロファイル
- ジョブ記述
- クラス
- 独立ディスク・プール・デバイスの記述
- ネットワーク属性
- システム環境変数
- TCP/IP 属性

ドメインのすべてのノードがアクティブで、グループに関係している場合、クラスター管理ドメインに追加できるのは、MRE のみです。 MRE はクラスター管理ドメインが区画化されている場合、追加できません。いったん MRE が追加されると、対等 CRG が開始された場合、MRE などのリソースへの変更はドメイン内のすべてのアクティブ・ノードに伝搬されます。 CRG が終了していると、CRG が再び開始した際に、保留中の変更がアクティブ・ドメインに伝搬されます。

MRE に関連付けられたグローバルな状態があります。MRE などのリソースが、アクティブ・ドメイン内のすべてのノードでモニターされる属性のすべてに、同じ値が指定されている場合、リソースのグローバル

状況は整合です。クラスター管理ドメインが 1 つ以上のノードでリソースを更新しようとし、その更新が失敗した場合、そのリソースのグローバル状況は不整合になります。グローバル状況が不整合な場合、管理者は障害の原因を判別し、修正しなければなりません。クラスター管理ドメインが次にリソースが更新された際に、再同期を試みます。その可能性としては、更新が失敗する原因となった問題を修正するために、管理者がリソースを変更した場合、または、CRG が再始動された場合などです。

クラスター管理ドメイン CRG が終了している場合、すべての MRE のグローバル状況は不整合に設定されます。 これは CRG が終了している間に異なるノード上にあるモニター対象リソースに対し変更が加えられ、それによってリソースが不整合になる可能性があるためです。

MRE によって表されるリソースがシステム・オブジェクトの場合、削除、名前変更、または先に MRE を除去することなく、異なるライブラリーへの移動は避けてください。リソースが削除されたり、名前変更されたり、異なるライブラリーに移動されたりすると、 MRE のグローバル状況は不整合になり、すべてのノードのリソースに、その後に加えられた変更はクラスター管理ドメインに伝搬されなくなります。

ノードがクラスター管理ドメインに追加されると、すべての MRE はアクティブ・ドメインから新しいノードにコピーされます。 MRE によって表されるリソースで、新しいノードに存在しないものは、作成され、属性の値はアクティブなクラスター管理の値と同一に設定されます。

アクティブでないクラスター管理にノードが存在する場合、アクティブ・ドメインでリソースに加えられた変更は、それらのノードが再びアクティブ・ドメインに再結合する際、非アクティブ・ノードに伝搬されます。クラスター管理ドメインで区画が発生している場合、各区画にあるアクティブ・ノード間で引き続き変更点が同期化されます。ノードが再度マージされると、クラスター管理ドメインにより区画ごとのすべての変更が伝搬されます。こうすることにより、リソースとアクティブ・ドメインとの整合性が保たれます。同じリソースに対し、複数の変更が複数の区画に加えられると、それぞれの変更は区画がマージされた後、クラスター管理ドメインにより処理されます。この場合の順序は不確定です。

関連概念

10 ページの『クラスター管理ドメイン』

クラスター管理ドメインはクラスター化された環境のノード全体で、 常に保守を必要とするリソースを管理するために使用されます。

クラスター・リソース・グループ出口プログラム

クラスター・リソース・グループ出口プログラムは、CRG のクラスター関連イベントが発生した後、呼び出されます。

出口プログラムは、回復装置 CRG に対してはオプションとして設定できますが、他の CRG タイプに対しては必須です。クラスター・リソース・グループ出口プログラムが使用されている場合には、下記の場合も含め、クラスター全体のイベントが出現する際にそのプログラムが呼び出されます。

- 予期しないこととしてノードがクラスターからはずされる場合。
- ノードが、クラスター・ノード終了 (QcstEndClusterNode) API またはクラスター・ノード項目除去 (QcstRemoveClusterNodeEntry) API の結果としてクラスターからはずされる場合。
- クラスターが、クラスター削除 (QcstDeleteCluster) API の結果として削除される場合。
- ノードが、クラスター・ノード開始 (QcstStartClusterNode) API によって活動化される場合。
- 区画化されたノードとの通信が再確立される場合。

これらの出口プログラムは、クラスター・ミドルウェアである、IBM ビジネス・パートナーおよびクラスター対応アプリケーション・プログラム・プロバイダーによって作成または提供されます。

クラスター・リソース・グループ出口プログラムの詳細については、クラスター API 文書のクラスター・リソース・グループ出口プログラムを参照してください。アクション・コードごとに、クラスター・リソース・グループ出口プログラムにどのような情報が渡されるのかについても説明されています。

リカバリー・ドメイン

リカバリー・ドメインは、リカバリー・アクション、またはイベントの同期化を実行するなどの共通の目的を持ったクラスター内のノードを、1 つのクラスター・リソース・グループ (CRG) として集めたもののサブセットです。

ドメインは、クラスターのノードのうち、クラスター・リソースへのアクセスが可能なものを表しています。このクラスター・ノードのサブセットは、1 次アクセス・ポイント、2 次 (バックアップ) アクセス・ポイント、複製、または対等アクセス・ポイントのいずれかをサポートする特定のクラスター・リソース・グループに割り当てられます。

ノードがリカバリー・ドメインで果たすことのできる 4 つのタイプの役割は次のとおりです。

プライマリー

回復クラスター・リソースの 1 次アクセス・ポイントであるクラスター・ノード。

- データ CRG については、プライマリー・ノードにリソースの基本コピーが含まれています。
- アプリケーション CRG の場合、プライマリー・ノードは、アプリケーションが現在実行されているシステムになります。
- デバイス CRG の場合、プライマリー・ノードは、装置リソースの現在の所有者です。

  注: 遠隔ミラーリングを使用する場合、装置 CRG のリカバリー・ドメインにあるノードにはサイト名とデータ・ポート IP アドレスが必要になります。詳細は、「Site name and data port IP addresses (サイト名とデータ・ポート IP アドレス)」を参照してください。
- 対等 CRG の場合、プライマリー・ノードはサポートされていません。

CRG のプライマリー・ノードに障害が発生した場合、または手動切り替えが開始された場合、CRG の 1 次アクセス・ポイントは最初のバックアップ・ノードに移動します。

バックアップ

現在のプライマリー・ノードで障害が発生した場合、もしくは手動切り替えが開始された場合、1 次アクセスの役割を引き継ぐクラスター・ノードです。

- データ CRG については、このクラスター・ノードに、複製により現行の状態に保たれているリソースのコピーが含まれています。
- 対等 CRG の場合、バックアップ・ノードはサポートされていません。

複製 クラスター・リソースのコピーを含んだクラスター・ノードですが、プライマリー・ノードやバックアップ・ノードの役割は果たしません。複製ノードのフェイルオーバーや切り替えはできません。複製ノードをプライマリー・ノードにどうしても変更したい場合は、まず複製ノードの役割をバックアップ・ノードの役割に変更しなければなりません。

- 対等 CRG の場合、複製として定義されたノードは、クラスター・リソースの非アクティブのアクセス・ポイントを示します。

対等 クラスター・リソースのアクティブなアクセス・ポイントになり得る、順序付けされていないクラスター・ノード。 CRG が開始されると、対等として定義されているすべてのノードはアクティブなアクセス・ポイントになります。

- 対等 CRG の場合、アクセス・ポイントはシステムではなく管理アプリケーションによって、完全に制御されます。対等の役割は対等 CRG でのみサポートされます。

プライマリー・バックアップ・モデル

プライマリー・バックアップ・モデルに関係するノードの場合、リカバリー・ドメイン内の各ノードには、クラスターの現在の動作環境に関連した役割があります。この役割のことを、リカバリー・ドメインでの現行の役割といいます。ノードが終了したり、ノードが開始されたり、ノードで障害が発生したりして、クラスターの作動環境が変化すれば、ノードの現在の役割も変化していきます。リカバリー・ドメイン内の各ノードには、望ましいクラスター環境、すなわち理想的なクラスター環境としてあらかじめ設定された役割もあります。この役割のことを、リカバリー・ドメインでの優先の役割といいます。優先の役割は、クラスター・リソース・グループの作成時に最初に設定される静的な定義です。クラスター環境が変化してもこの役割は変わりませんが、優先の役割は、ノードが追加されたりリカバリー・ドメインから取り外されたり、ノードがクラスターから取り外されるときにしか変更できません。優先の役割を手作業で変更することもできます。

概念としては、プライマリー・バックアップ・モデルに含まれるリカバリー・ドメインを次のように見なすことができます。

表 1. プライマリー・バックアップ CRG のノードの役割

ノード	現行の役割	優先の役割
A	バックアップ 1	プライマリー
B	バックアップ 2	バックアップ 1
C	プライマリー	バックアップ 2
D	複製	複製

| このサンプルでは、ノード A、B、C、D がプライマリー・バックアップ・モデルである CRG の例を示しています。 ノード C が現在プライマリー・ノードになっています。なぜなら、2 番目のバックアップという優先の役割があるからです。プライマリー・ノードとしてのノード C の現行の役割は、2 回のフェイルオーバーまたは切り替えアクションによって発生します。最初のフェイルオーバーまたは切り替えでは、プライマリーの役割は、ノード B が最初のバックアップとして定義されているので、ノード A からノード B に移動しました。 2 番目のフェイルオーバーまたは切り替えにより、ノード C が 2 番目のバックアップ・ノードとして定義されていたので、ノード C がプライマリー・ノードになりました。 ノード D の現行役割、および優先役割は複製されたものです。複製ノードの役割が手動でプライマリー、またはバックアップに変更されたのでない限り、フェイルオーバー中、または切り替え中に、複製ノードをアクセス・ポイントの前提にすることはできません。

注: リカバリー・ドメインの各ノードの役割は、手動で変更することもできます。上記の例では、切り替えまたはフェイルオーバー・アクションが発生し、リカバリー・ドメインの役割指定に手動変更が行われていないときに、リカバリー・ドメインの役割がどのように変更されるかを示しています。

対等モデル

| 対等モデルでは、クラスター・リソース・グループに含まれるノードには、対等、または複製のいずれかの役割を使用できます。

| *表 2. 対等 CRG のノードの役割*

ノード	現行の役割	優先の役割
A	対等	対等
B	対等	対等
C	対等	対等

表 2. 対等 CRG のノードの役割 (続き)

ノード	現行の役割	優先の役割
D	複製	複製

ノード A、B、C はリカバリー・ドメインに対等ノードとして定義されています。ノード A で障害が発生すると、現在の役割に関わらず、リカバリー・ドメインのすべてのノードに対し通知されます。これらのノードはノード A で障害が発生した時点の操作を再開します。 ノード D にはデータがありますが、複製として定義されているため、操作を再開することはできません。

対等または複製として指定できるノードの数に制限はありません。対等ノードは順序付けされておらず、クラスター・リソースのアクティブなアクセス・ポイントになり得ます。複製は順序付けされておらず、クラスター・リソース・グループの変更 (QcstChangeClusterResourceGroup) API を使用して、役割を複製から対等に変更しない限り、クラスター・リソースのアクティブなアクセス・ポイントにすることはできません。

関連タスク

119 ページの『クラスター・リソース・グループのリカバリー・ドメインの変更』
クラスター・リソース・グループのリカバリー・ドメインにあるノードの役割の変更、 およびリカバリー・ドメインへのノードの追加、 またはリカバリー・ドメインからのノードの除去が行えます。 装置クラスター・リソース・グループの場合は、 リカバリー・ドメインにあるノードのサイト名とデータ・ポート IP アドレスの変更も行えます。

120 ページの『切り替えの実行』
切り替えを手動実行すると、現行プライマリー・ノードは、クラスター・リソース・グループのリカバリー・ドメインで定義したバックアップ・ノードに切り替わります。

クラスター・バージョン

クラスター・バージョンとは、 クラスターで実行できる機能のレベルを表す用語です。

1 つのクラスターの中に、複数のリリース・レベルのシステムを組み込み、使用可能な通信プロトコルのレベルを判別することによって、完全な相互運用を実現するための技法が、このバージョン設定です。異なるリリース・レベルのシステムを含むクラスターを使用する場合には、複数リリースのクラスターを参照してください。

クラスターには、実際のところ 2 つのバージョンがあります。

潜在クラスター・バージョン
あるノードで実行できる最新のクラスター機能レベルです。このバージョンでは、そのノードがクラスター内の他のノードと通信できるということになります。

現行クラスター・バージョン
クラスターのすべての操作で現在使用されているバージョンです。このバージョンでは、クラスター内のすべてのノード間で通信が可能です。

潜在クラスター・バージョンは、以前のクラスター・バージョンでは実行できなかった新しいクラスタリング機能を追加したオペレーティング・システムのリリースが出るたびに進んでいきます。現行クラスター・バージョンが潜在クラスター・バージョンよりも古い場合は、その新しい機能が使用できません。一部のノードが、その新しい機能に基づいた要求を認識したり処理したりすることができないからです。そのような新機能を活用するには、クラスター内のすべてのシステムを同じ潜在クラスター・バージョンでそろえ、現行クラスター・バージョンのほうもそのレベルに設定する必要があります。

あるノードがクラスターに加入しようとする場合は、そのノードの潜在クラスター・バージョンが、現行クラスター・バージョンと比較されます。その潜在クラスター・バージョンの値が、現行クラスター・バージョンの値 (N) や、その次のバージョン・レベルの値 (N+1) と同じでない場合は、そのノードの加入は認められません。現行クラスター・バージョンは、クラスター内で定義されている最初のノードを基準にして、クラスター作成 API またはコマンドで指定する値によって初期設定されます。

| たとえば、V5R3 ノードを V5R4 ノードと混在させるには、以下のいずれかの方法があります。

| • V5R3 システムでクラスターを作成し、後から V5R4 ノードを追加する。

| • V5R4 システムでクラスターを作成し (その際に、旧リリースのノードをクラスターに追加できるように
| 設定しておき)、後からクラスターに V5R3 システムを追加する。

複数のリリースのクラスターでは、必ず最も低いノード・リリース・レベルでクラスター・プロトコルが実行されることになります。この N は、最初にクラスターを作成したときに定義します。つまり、クラスター作成要求を出したノードで実行されていた潜在ノード・バージョンか、その 1 つ前のクラスター・バージョンのいずれかに設定できます。クラスター内のノードのクラスター・バージョン・レベルの差を 2 つ以上にすることはできません。

クラスター内のすべてのシステムを次のリリースにアップグレードした場合、新しい機能を実行できるようにするには、クラスター・バージョンもアップグレードしなければなりません。そのためには、クラスター・バージョンの調整を行います。

重要: 新バージョンのオペレーティング・システムが、現在のクラスター・バージョンと等しくないか、または 1 つ大きいバージョンであると、クラスター・ノードは、再始動時に失敗します。この事態からリカバリーするには、ノードを削除してから、正しいバージョンを指定して再作成する必要があります。

| **重要:** 切り替え可能な独立ディスク・プールをクラスター内で使用している場合は、複数のリリースどう
| しの間での切り替えの実行に対して制限があります。前のリリースの独立ディスク・プールを、現行リリー
| スの i5/OS を実行するシステムに切り替えて、使用可能にする必要があります。それが、現行リリースの
| i5/OS を実行するシステム上で使用可能になると、その内部コンテンツは変更されて、前のリリース・シス
| テムではもう使用できなくなります。

クラスター API 文書のクラスター・バージョンを参照してください。制限やクラスター・バージョンと i5/OS のリリースの対応に関する情報が含まれています。

関連概念

95 ページの『複数リリースのクラスター』
複数のクラスター・バージョンのノードを含むクラスターを作成する場合、クラスターの作成時に特定のステップが必要になります。

152 ページの『クラスターの一般的な問題』
ここでは、クラスターで生じる可能性のある最も一般的な問題のいくつかをリストして、 その回避方法および回復方法を示します。

関連タスク

111 ページの『クラスターの作成』
クラスターを作成および構成するには、クラスターに少なくとも 1 つのノードを組み込むことが必要で、そのクラスター内に配置されることになる最低 1 つのノードに対するアクセス権を有していなければなりません。

116 ページの『クラスターのクラスター・バージョンの調整』
クラスター・バージョンは、クラスター内のすべてのノードがアクティブに相互通信するレベルを定義します。

回復リソース

回復リソースとは、システムにクラスタリングを使用した結果、可用性が高くなったシステム・リソース (データ、装置、アプリケーションなど) のことです。

クラスター内の特定のリソース・セットの 1 次アクセス・ポイントとして使用されているクラスター・ノードに障害が発生しても、そのリソース・セットが回復力に富んでいれば、そのリソース・セットのバックアップ用として定義されている別のクラスター・ノードが新しいアクセス・ポイントとなります。

回復できるシステム・リソースは、次のとおりです。

1. ノード間で複製できるデータ。
2. IP アドレスを使用するアプリケーションで、あるノードから他のノードへ切り替えられるもの。
3. あるノードから他のノードへ切り替えられるハードウェア装置。
4. 対等リソースとは、クラスター管理ドメインでサポートされているリソースです。

回復リソースのセットに関連付けられたノードどうしの間に見られる関係の定義は、クラスター・リソース・グループ (CRG) オブジェクトに示されています。クラスター・リソース・グループは、クラスター・リソース・サービスによってクラスター内の各ノードで複製され、整合性が取られます。

関連概念

8 ページの『クラスター・リソース・グループ』
クラスター・リソース・グループ (CRG) は、 i5/OS のシステム・オブジェクトであり、 クラスター化された環境で発生するイベントの管理に使用される、クラスター・リソースの一連のセット、またはグループです。 クラスター・リソース・グループはリカバリー・ドメインを記述するとともに、 特定のクラスター・イベントが発生する際に呼び出されるクラスター・リソース・グループ出口プログラムの名前を提供します。

10 ページの『クラスター管理ドメイン』
クラスター管理ドメインはクラスター化された環境のノード全体で、 常に保守を必要とするリソースを管理するために使用されます。

回復アプリケーション:

回復アプリケーションとは、クライアントを再構成しなくても、別のクラスター・ノードで再始動できるアプリケーションのことです。

アプリケーションを回復力に富むものにする上で寄与する特性については、アプリケーション・プログラムを回復力に富むものにするを参照してください。

アプリケーションを回復力に富むものとするためには、クライアントとサーバーとの間に確立された、インターネット・プロトコル (IP) 接続が一時的に使用不能となる場合に、そのことが認識できなければなりません。クライアント・アプリケーションは、IP 接続が一時的に使用できなくなることを認識できなければならず、またフェイルオーバーを終了または開始することなく、アクセスを再試行できなければなりません。同様に、切り替えを実行する場合、サーバー・アプリケーションは IP 接続が、もはや使用できないことを認識できなければなりません。結果として、そのサーバー・アプリケーションにはエラー条件が戻されます。このエラー条件が戻された場合には、サーバー・アプリケーションがその条件を認識し、正常に終了するのが最善です。

IP アドレスのテークオーバーは、アプリケーション・サーバーで発生した障害からクライアントを保護するために使用される、高可用性機能です。**アプリケーション・テークオーバー IP アドレス**は、アプリケーションと関連した浮動アドレスです。この機能の概念は、IP アドレスの別名割り当てを使用することによって、浮動 IP アドレスを複数のアプリケーション・サーバーまたはホストに関連付けるというものです。

クラスター内のいずれかのアプリケーション・サーバーで障害が発生しても、別のクラスター・ノードがアプリケーション・サーバーの責任を引き受けるため、クライアントを再構成しなくても済みます。

もう 1 つ、IP アドレスのテークオーバーをサポートするために追加されているのが、アプリケーション・クラスター・リソース・グループ (CRG) という概念です。 アプリケーション CRG は、1 つのアプリケーション・テークオーバー IP アドレス・リソースと 1 つのリカバリー・ドメインを含むクラスター・リソース・グループです。リカバリー・ドメインには、クラスター内にあるアプリケーション・サーバーのうち、特定のアプリケーションをサポートしているもののリストが含まれています。あるリソースで障害が発生すると、クラスター・リソース・サービスはそのリソースが属しているグループにフェイルオーバーします。

関連概念

35 ページの『アプリケーション・プログラムを回復力に富むものにする』
アプリケーション・プログラムを回復力に富むものにする方法を確認する。

13 ページの『リカバリー・ドメイン』
リカバリー・ドメインは、リカバリー・アクション、またはイベントの同期化を実行するなどの共通の目的を持ったクラスター内のノードを、1 つのクラスター・リソース・グループ (CRG) として集めたもののサブセットです。

関連タスク

33 ページの『クラスター・アプリケーション』
アプリケーション回復力とは、クラスター環境の重要な要素の 1 つです。 クラスター内で可用性の高いアプリケーションを構築し、使用する場合は、これらのアプリケーションには特定の仕様がある点に留意しなければなりません。

回復データ:

回復データとは、 クラスター内の複数のノードに複製 (コピー) されたデータのことです。

リカバリー・ドメイン内の各ノードには、ある種の複製メカニズムによって保守される、回復データのコピーが含まれています。リカバリー・ドメイン内でバックアップとして定義されているノードは、回復データの 1 次アクセス・ポイントの役割を担うことができます。複製として定義されているノードも、データのコピーを含んではいますが、1 次アクセス・ポイントの役割を担うことはできません。基本的に、複製ノードにコピーされるデータは、プライマリー・ノードにおいて、バックアップや読み取り専用照会などの作業の負荷を軽減するために使用されます。

関連概念

30 ページの『複製』
複製とは、何かのコピーをリアルタイムで作成することです。つまり、クラスター内のあるノードに含まれているオブジェクトを、同じクラスター内にある他の 1 つ以上のノードにコピーすることを意味します。

回復装置:

回復装置とは、装置記述などの構成オブジェクトによって表されている物理リソースのうち、クラスター内の複数のノードからアクセスできる装置を指します。

そのようなリソースのアクセス・ポイントは、システムに障害が発生した場合に、クラスター・リソース・グループのリカバリー・ドメイン内の最初のバックアップ・ノードに切り替えられます。独立ディスク・プール、または独立 ASP は、他のシステム・ストレージに影響を与えることなく、オフラインにしたり、オンラインにしたりできる回復デバイスです。さらに、サイト間ミラーリング (XSM) の副次機能である遠隔ミラーリングを使用できます。これは i5/OS オプション 41、ハイ・アベイラビリティー切り替え可能リソ

| ースの一部です。遠隔ミラーリングは高可用性と災害時回復を提供するために、2 つのサイトで独立ディスク・プールの同一コピーを 2 つ保持する機能です。プライマリー・ノード上のコピーが実働コピーで、他のサイトのバックアップ・ノード上にあるコピーがミラー・コピーです。ユーザー操作とアプリケーションは実働コピーが置かれているプライマリー・ノード上の独立ディスク・プールにアクセスします。

回復装置クラスター・リソース・グループには、切り替え可能な装置のリストを含めることができます。そのリスト内の装置は、 それぞれ 1 つの切り替え可能な独立ディスク・プールを識別します。システムに障害が発生した場合は、その装置全体のコレクションがバックアップ・ノードに切り替えられます。さらに、切り替え/フェイルオーバー・プロセスの一部として、それらの装置をオンに構成変更することもできます。切り替え可能な装置のリストに関連した物理構成については、いくつかの制限があります。回復リソースとして定義されている独立ディスク・プール用の適切な構成をセットアップする方法の詳細は、独立ディスク・プールを参照してください。

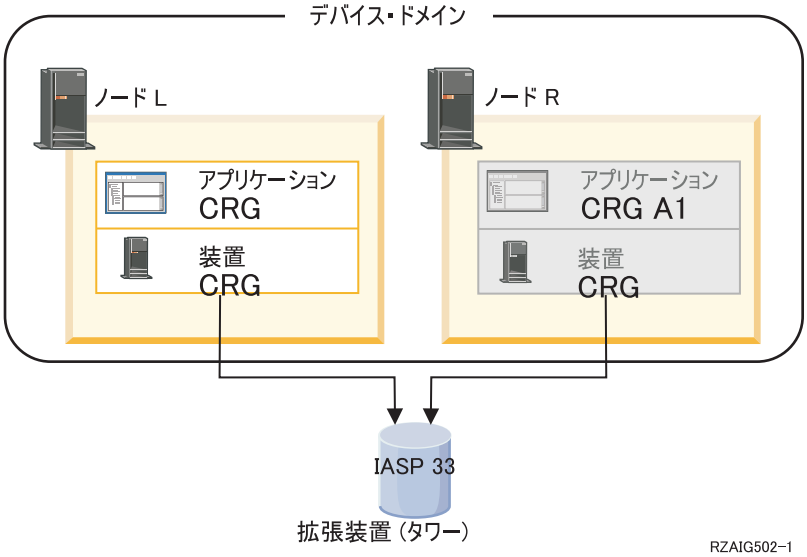
回復装置 CRG は、他の種類の CRG と基本的に変わりません。 1 つの違いは、前に述べたとおり、切り替え可能な装置のリストであり、もう 1 つの違いは、装置 CRG には出口プログラムをオプションとして設定できるという点です。環境固有の処理やデータ固有の処理が必要な場合は、CRG に出口プログラムを使用するとよいでしょう。このタイプの CRG の詳細については、「クラスター・リソース・グループの作成 (QcstCreateClusterResourceGroup) API」を参照してください。

デバイス・ドメイン

デバイス・ドメインとは、装置リソースを共用するクラスター内のノードのサブセットです。 もう少し具体的に説明すれば、デバイス・ドメイン内のノードは、回復リソースのコレクションに対するスイッチ・アクションに参加できます。

デバイス・ドメインの識別や管理は、一連のインターフェースから行います。それらのインターフェースでは、デバイス・ドメインにノードを追加したり、デバイス・ドメインからノードを削除したりできます。

デバイス・ドメインは、回復リソースをノード間で切り替えるときに必要になる、グローバル情報を管理するために使用します。デバイス・ドメイン内のすべてのノードは、装置を切り替えたときに競合が起こらないようにするために、その情報を必要とします。たとえば、切り替え可能な独立ディスク・プールのコレクションの場合、独立ディスク・プールの ID、ディスク装置の割り当て、仮想アドレスの割り当ては、デバイス・ドメイン全体で固有でなければなりません。



クラスター・ノードは、同時に複数のデバイス・ドメインに所属することはできません。装置 CRG のリカバリー・ドメインにノードを追加するには、まずそのノードをデバイス・ドメインのメンバーとして定義する必要があります。装置 CRG のリカバリー・ドメインに入るすべてのノードは、同じデバイス・ドメインに所属している必要があります。

デバイス・ドメインを作成し管理するには、オプション 41 (i5/OS - HA 切り替え可能リソース) をインストールする必要があり、システム上に有効なライセンス・キーが存在していなければなりません。

関連概念

135 ページの『例: 独立ディスク・プールを使用する切り替えディスク・クラスター』
切り替えディスク・テクノロジーを使用するクラスターは、 データを複製するための代替手段を提供します。 切り替えディスク・クラスターでは、データは独立ディスク・プール (独立 ASP とも呼ばれる) に 通常含まれています。

関連タスク

121 ページの『デバイス・ドメインへのノードの追加』
デバイス・ドメインとは、装置リソースを共用するクラスター内のノードのサブセットです。

122 ページの『デバイス・ドメインからのノードの除去』
デバイス・ドメインとは、装置リソースを共用するクラスター内のノードのサブセットです。

オプション 41 (HA 切り替え可能リソース):

デバイス・ドメインを作成し管理するには、オプション 41 (i5/OS - HA 切り替え可能リソース) をインストールする必要があり、システム上に有効なライセンス・キーが存在していなければなりません。

クラスター環境で以下のうちのいずれかを行いたい場合、このフィーチャーをインストールする必要があります。

- iSeries ナビゲーター・クラスター管理インターフェースを使用する。
- システム間で独立ディスク・プール (独立 ASP) を切り替える。
- 地理的に散在する複数のシステムで、サイトどうしの間のミラーリングを行う。

関連タスク

121 ページの『デバイス・ドメインへのノードの追加』
デバイス・ドメインとは、装置リソースを共用するクラスター内のノードのサブセットです。

122 ページの『デバイス・ドメインからのノードの除去』
デバイス・ドメインとは、装置リソースを共用するクラスター内のノードのサブセットです。

クラスター・イベント

1 つのクラスター内では、複数の種類のイベント、アクション、サービスが発生します。

フェイルオーバー

フェイルオーバーは、システム障害が発生したときに、クラスターのサーバーが 1 つ以上のバックアップ・サーバーへ自動的に切り替わるときに発生します。

あるサーバーから別のサーバーにアクセスを手動で切り替えるときに発生する切り替えと比較してください。切り替えとフェイルオーバー機能は、いったん起動すると同じです。違いは、イベントがどのように起動されるかということだけです。

フェイルオーバーが発生するとき、クラスター・リソース・グループのリカバリー・ドメインで現在プライマリー・ノードとなっているクラスター・ノードから最初のバックアップとして指定されているクラスタ

ー・ノードへアクセスが切り替えられます。切り替えの順序がどのように決定されるのかに関する情報については、リカバリー・ドメインを参照してください。

フェイルオーバー・アクションに複数のクラスター・リソース・グループ (CRG) が関係している場合、システムは、装置 CRG (切り替え可能装置)、データ CRG (切り替え可能データ・グループ)、アプリケーション CRG (切り替え可能アプリケーション) の順序で処理を実行します。

フェイルオーバー・メッセージ待ち行列は、フェイルオーバー・アクティビティーに関するメッセージを受け取ります。これを使用して、クラスター・リソース・グループのフェイルオーバー処理を制御できます。

関連概念

13 ページの『リカバリー・ドメイン』
リカバリー・ドメインは、リカバリー・アクション、またはイベントの同期化を実行するなどの共通の目的を持ったクラスター内のノードを、1 つのクラスター・リソース・グループ (CRG) として集めたもののサブセットです。

8 ページの『クラスター・リソース・グループ』
クラスター・リソース・グループ (CRG) は、 i5/OS のシステム・オブジェクトであり、 クラスター化された環境で発生するイベントの管理に使用される、クラスター・リソースの一連のセット、またはグループです。 クラスター・リソース・グループはリカバリー・ドメインを記述するとともに、 特定のクラスター・イベントが発生する際に呼び出されるクラスター・リソース・グループ出口プログラムの名前を提供します。

130 ページの『フェイルオーバー・メッセージ待ち行列』
フェイルオーバー・メッセージ待ち行列は、フェイルオーバー・アクティビティーに関するメッセージを受け取ります。

90 ページの『クラスターのハードウェア要件』
i5/OS V4R4M0 またはそれ以降を実行できる iSeries モデルは、 クラスタリングを使用したものと互換性があります。

関連タスク

24 ページの『切り替え』
切り替えは、あるサーバーから別のサーバーにアクセスを手動で切り替えるときに発生します。

例: 障害:

通常、フェイルオーバーはノード障害によって起こりますが、その他の理由でも起こることがあります。

ある問題によって単一のクラスター・リソース・グループが影響を受けると、そのクラスター・リソース・グループ (CRG) だけでなく、他の CRG でもフェイルオーバーが発生する可能性があります。

以下の表は、さまざまな障害の種類とその該当カテゴリーを示しています。

障害	一般カテゴリー
CEC ハードウェア障害 (たとえば、CPU)	2
通信アダプター、回線、 ルーターの障害; またはノードに定義されているすべての IP インターフェース・アドレスに影響を与える ENDTCPIFC	4
CEC の停電	1
オペレーティング・システムのソフトウェアによるマシン・チェック	2

障害	一般カテゴリー
ENDTCP (時間制限付きの *IMMED または *CNTRLD) が発行される	1
ENDSBS QSYSWRK(*IMMED または *CNTRLD) が発行された	1
ENDSBS(*ALL、*IMMED、または *CNTRLD) が発行された	1
ENDSYS (*IMMED または *CNTRLD) が発行された	1
PWRDWNSYS(*IMMED または *CNTRLD) が発行された	1
システム上でクラスター・リソース・サービスがアクティブの場合に初期プログラム・ロード (IPL) ボタンが押される	1
QCSTCTL ジョブの取り消しジョブ (時間制限付きの *IMMED または *CNTRLD) が発行された	1
QCSTCRGM ジョブの取り消しジョブ (時間制限付きの *IMMED または *CNTRLD) が発行された	1
クラスター・リソース・グループ・ジョブの取り消しジョブ (時間制限付きの *IMMED または *CNTRLD) が発行された	3
クラスター・ノード終了 API が呼び出される	1
クラスター・ノード除去 API が呼び出される	1
クラスター・リソース・グループ・ジョブに、そのジョブを異常終了させるソフトウェア・エラーがある	3
制御パネルから機能 8 または機能 3 を入力してシステムの電源が遮断される	2
機能 7 を入力して、区画の電源遮断が遅延される	1
アプリケーション・クラスター・リソース・グループのアプリケーション・プログラムの障害	3

一般カテゴリー:

1. ノード上のすべてのクラスター・リソース・サービス (CRS) に障害が起こり、ノード障害として検出される。そのようなノードは実際には操作可能である場合もあれば、失敗している場合 (たとえば、停電によるシステム障害) もあります。クラスター・リソース・サービスのすべてに障害が発生した場合、 CRS によって管理されるリソースにはフェイルオーバー・プロセスが実行されます。
2. ノード上のすべての CRS に障害が発生するが、クラスター区画として検出される。ノードは操作可能の場合もあれば、操作不能の場合もあります。
3. 個々のクラスター・リソース・グループ上で障害が起こる。これらの条件は常に障害として検出されます。
4. 障害は発生しても、ノードおよびクラスター・リソース・サービスが引き続き操作可能であり、障害がクラスター区画として検出される。

障害が起こる場合、クラスター・リソース・サービスが特定のクラスター・リソース・サービス・グループ用に実行するアクションは、障害の種類、およびクラスター・リソース・グループの状態によって異なります。しかし、どの場合でも、出口プログラムは呼び出されます。フェイルオーバーによって障害ノードのリストを処理しなければならない場合もあります。出口プログラムは、呼び出されたときに、処理しなければならないのが単一のノード障害だけか、それとも障害ノードのリストかを判別する必要があります。

クラスター・リソース・グループが*非アクティブ* の場合、クラスター・リソース・グループのリカバリー・ドメインで障害が発生したノードのメンバーシップ状況は、*非アクティブ* または*区画* 状態に変更されます。ただし、ノードの役割は変更されず、バックアップ・ノードはリオーダーされません。 クラスター・リソース・グループの開始 (STRCRG) コマンドまたはクラスター・リソース・グループ開始 (QcstStartClusterResourceGroup) API が呼び出されると、非アクティブのクラスター・リソース・グループ内のバックアップ・ノードの再配列が行われます。しかし、プライマリー・ノードがアクティブでない場合、クラスター・リソース・グループ開始 API は失敗します。 クラスター・リソース・グループの変更 (CHGCRG) コマンドまたはクラスター・リソース・グループ変更 (QcstChangeClusterResourceGroup) API を発行することによってアクティブ・ノードをプライマリー・ノードと指定してから、クラスター・リソース・グループ開始 API をもう一度呼び出してください。

クラスター・リソース・グループがアクティブ の場合に、障害発生ノードがプライマリー・ノード*ではない* と、フェイルオーバーによって、クラスター・リソース・グループのリカバリー・ドメイン内で障害が発生したリカバリー・ドメイン・メンバーの状況が更新されます。障害発生ノードがバックアップ・ノードの場合、アクティブ・ノードがリストの先頭になるよう、バックアップ・ノードのリストはリオーダーされます。

クラスター・リソース・グループがアクティブ で、リカバリー・ドメイン・メンバーがプライマリー・ノードである場合、発生した障害の種類に応じて、以下のアクションが実行されます。

カテゴリー 1 の障害
: フェイルオーバーが発生します。各クラスター・リソース・グループ内でプライマリー・ノードに*非アクティブ* のマークが付けられ、最後のバックアップ・ノードにされます。最初のバックアップだったノードは新しいプライマリー・ノードになります。すべての装置クラスター・リソース・グループが最初にフェイルオーバーされます。次に、すべてのデータ・クラスター・リソース・グループがフェイルオーバーされます。最後に、すべてのアプリケーション・クラスター・リソース・グループがフェイルオーバーされます。 CRG のフェイルオーバーにおいて、アクティブなバックアップ・ノードがないことが検出される場合、その CRG の状況は*未確定* に設定されます。

カテゴリー 2 の障害
: フェイルオーバーが発生しますが、プライマリー・ノードは変更されません。区画のメンバーとしてプライマリー・ノードを持たないクラスター区画内の全ノードは、アクティブなクラスター・リソース・グループを終了します。クラスター・リソース・グループのリカバリー・ドメイン内のノード状況は、1 次区画にあるノードごとに*区画* 状況に設定されます。ノードで実際に障害が発生しても、単なる区画問題として検出され、障害の発生したノードがプライマリー・ノードであれば、そのノード上のデータおよびアプリケーション・サービスはすべて失われ、フェイルオーバーは自動的に開始されません。ノードに対して失敗と宣言するか、またはノード・バックアップを起動してそのノードに対して再びクラスタリングを開始する必要があります。詳細については、 区画化ノードを障害ノードに変更するを参照してください。

カテゴリー 3 の障害
: 単一のクラスター・リソース・グループだけが影響を受ける場合、クラスター・リソース・グループは相互に独立しているので、フェイルオーバーは個別に発生します。複数のクラスター・リソース・ジョブを取り消すユーザーがいると、いくつかのクラスター・リソース・グループが同時に影響を受ける可能性があります。しかし、障害のタイプは CRG ごとに処理され、CRG 間でフェイルオーバーが調整されることはありません。プライマリー・ノードは、個々のクラスター・リソース・グループ内で*非アクティブ* のマークが付けられ、最後のバックアップ・ノードにされます。最初のバックアップ・ノードだったノードは新しいプライマリー・ノードになります。アクティブなバックアップ・ノードがない場合、クラスター・リソース・グループの状況は*未確定* に設定されます。

カテゴリー 4 の障害

このカテゴリーはカテゴリー 2 に似ています。ただし、すべてのノードおよびノード上のクラスター・リソース・サービスは引き続き操作可能ですが、すべてのノードが相互に通信できるわけではありません。クラスターは区分されますが、プライマリー・ノード (複数も可) は引き続きサービスを提供します。しかし、区画であるため、いろいろな問題に遭遇する可能性があります。たとえば、プライマリー・ノードが 1 つの区画にあり、すべてのバックアップ・ノードまたは複製ノードが別の区画にある場合、プライマリー・ノードに障害が発生すると、データの複製および保護は行われなくなります。プライマリー・ノードを収容している区画では、他の区画内のすべてのノードのクラスター・リソース・グループのリカバリー・ドメイン内のノード状況が、フェイルオーバー・プロセスによって*区画*に更新されます。プライマリー・ノードを収容していない区画では、他の区画内のすべてのノードのクラスター・リソース・グループのリカバリー・ドメイン内のノード状況は、*区画* に設定されます。

関連概念

154 ページの『区画エラー』

クラスター状態によっては、簡単に修正することができます。 クラスター区画が生じた場合の回復方法を確認してください。 このトピックでは、クラスター区画の回避方法を示し、区画をマージして元に戻す方法を例示します。

切り替え

切り替えは、あるサーバーから別のサーバーにアクセスを手動で切り替えるときに発生します。

手動切り替えを実行するのは、通常、システム保守を実行する場合です。たとえば、プログラム一時修正 (PTF) を適用したり、新しいリリースをインストールしたり、システムをアップグレードしたりする場合に、切り替えを実行します。プライマリー・ノードで障害が起こったときに自動的に発生するフェイルオーバーと比較してください。

切り替えが発生するとき、クラスター・リソース・グループのリカバリー・ドメインで現在プライマリー・ノードとなっているクラスター・ノードから最初のバックアップとして指定されているクラスター・ノードへアクセスが切り替えられます。切り替えの順序がどのように決定されるのかに関する情報については、リカバリー・ドメインを参照してください。

管理作業に関連して複数の CRG の切り替えを実行する場合は、CRG 間の関係をよく考えて順序を指定するようにしてください。たとえば、装置 CRG に関連したデータに依存するアプリケーション CRG がある場合は、切り替えの順序は、次のようになります。

1. 古いプライマリー・ノード上のアプリケーションを停止する (データに対する変更を防ぐため)。
2. 装置 CRG を新しいプライマリー・ノードに切り替える。
3. アプリケーション CRG を新しいプライマリー・ノードに切り替える。
4. 新しいプライマリー・ノードでアプリケーションを再始動する。

関連概念

20 ページの『フェイルオーバー』

フェイルオーバーは、システム障害が発生したときに、クラスターのサーバーが 1 つ以上のバックアップ・サーバーへ自動的に切り替わるときに発生します。

13 ページの『リカバリー・ドメイン』

リカバリー・ドメインは、リカバリー・アクション、またはイベントの同期化を実行するなどの共通の目的を持ったクラスター内のノードを、1 つのクラスター・リソース・グループ (CRG) として集めたもののサブセットです。

関連タスク

| 120 ページの『切り替えの実行』
| 切り替えを手動実行すると、現行プライマリー・ノードは、クラスター・リソース・グループのリカバリー・ドメインで定義したバックアップ・ノードに切り替わります。

再結合

再結合とは、一度メンバーから外れた後に、再びクラスターの活動メンバーになることです。

たとえば、一度非アクティブになったノードでクラスタリングを再始動した場合、そのクラスター・ノードはクラスターに再結合することになります。クラスター・リソース・サービスは、クラスター内ですでにアクティブなノードから開始します。クラスター・バージョン 3 から、クラスターにアクティブ・ノードがあれば、あるノードが自分自身を開始して現在アクティブなクラスターに再結合することができるようになりました。詳細については、クラスター・ノードの開始を参照してください。

ノード A、B、および C が 1 つのクラスターを構成しているとします。 ノード A に障害が発生します。アクティブなクラスターは、ノード B および C になります。障害の発生したノードが再び操作可能になると、自分自身を含めた任意のクラスター・ノードから開始された場合には、障害ノードはクラスターに再結合できます。再結合操作は、クラスター・リソース・グループ単位で行われます。すなわち、これは個々のクラスター・リソース・グループ (CRG) がクラスターに別々に加入することを意味しています。

再結合の基本機能により、CRG オブジェクトは必ず、すべてのアクティブなリカバリー・ドメイン・ノードに複製されます。再結合するノード、およびアクティブなすべての既存クラスター・ノードは、CRG オブジェクトと同じコピーを持っていなければなりません。また、一部の内部データと同じコピーも持っていなければなりません。

アクティブでないクラスター管理ドメインにノードが存在する場合、アクティブなドメインでリソースに対して加えられたの変更は、それらのノードが再びアクティブ・ドメインに再結合する際、非アクティブ・ノードに伝搬されます。

ノードに障害が生じた場合、クラスター内の残りのノードに対してクラスター・リソース・サービスを呼び出し続けると、CRG オブジェクト内のデータが変更される可能性があります。 API または後続のノード障害に関する呼び出しが行われる場合、変更は必ず起こります。単純なクラスターの場合、再結合するノードは、クラスター内で現在アクティブな特定ノードからの CRG のコピーを使って更新されます。しかし、ここに述べた事柄がすべての事例に当てはまるとは限りません。

関連タスク

| 115 ページの『クラスター・ノードの開始』
| クラスター・ノードを開始すると、クラスター内のノード上のクラスター・リソース・サービスも開始されます。 クラスター・バージョン 3 から、クラスターにアクティブ・ノードがあれば、あるノードが自分自身を開始して現在アクティブなクラスターに再結合することができるようになりました。

| 157 ページの『区画化ノードを障害ノードに変更する』
| 時々、「区画」状況が報告されているときに、実際にはノード障害が生じていることがあります。 このことは、クラスター・リソース・サービスが 1 つ以上のノードとの通信を失ったものの、 ノードが引き続き作動可能かどうかを検出できない場合に生じます。 この状況が生じた場合に、ノードに障害が起きたことを知らせるための簡単なメカニズムがあります。

例: 再結合:

このトピックでは、ノードがクラスターに再結合する際に起こり得るアクションについて解説します。

以下の図は、ノードがクラスターに再結合するときに常に実行されるアクションを示しています。また、再結合ノードの状態は、CRG のリカバリー・ドメイン内のメンバーシップ状況フィールドで、*非アクティブ*

からアクティブ に変更されます。出口プログラムは CRG のリカバリー・ドメイン内の全ノードに対して呼び出され、再結合のアクション・コードが渡されます。

表 3. 再結合操作

再結合操作			
再結合するノード		クラスター・ノード	
CRG のコピーが含まれる	CRG のコピーが含まれない	CRG のコピーが含まれる	CRG のコピーが含まれない
(1)	(2)	(3)	(4)

上記の表を使用した場合、以下の状況が考えられます。

1. 1 および 3
2. 1 および 4
3. 2 および 3
4. 2 および 4

クラスター内のあるノードに CRG のコピーがある場合、再結合の一般的な規則として、CRG はクラスター内のアクティブ・ノードから再結合ノードにコピーされます。

再結合の状態 1

クラスター内のノードからコピーされた CRG オブジェクトは、加入ノードに送信されます。結果は、以下のとおりです。

- CRG オブジェクトは、クラスターから送信されたデータを使って加入ノード上で更新されます。
- CRG オブジェクトは、加入ノードから削除される場合があります。これは、加入ノードがクラスター外にあった間に CRG のリカバリー・ドメインから加入ノードが除去された場合に起こる可能性があります。

再結合の状態 2

CRG オブジェクトのコピーが加入ノードからすべてのクラスター・ノードに送信されます。結果は、以下のとおりです。

- CRG のリカバリー・ドメインにクラスター・ノードがない場合、変化はありません。
- クラスター・ノードの 1 つ以上に CRG オブジェクトが作成される場合があります。これは、以下のシナリオで起こる可能性があります。
  - ノード A、B、C、および D が 1 つのクラスターを構成している。
  - 4 つのノードすべてが CRG のリカバリー・ドメインにある。
  - ノード A がクラスターの外にある間に、CRG が変更されてリカバリー・ドメインから B が除去される。
  - ノード C および D に障害が発生する。
  - クラスターは、CRG のコピーを持たないノード B だけである。
  - ノード A がクラスターに再結合する。
  - ノード A には CRG (しかし、現在は下位レベル) があるが、ノード B にはない。 CRG はノード B に作成されます。ノード C および D がクラスターに再結合するとき、クラスター内の CRG のコピーはノード C および D を更新し、リカバリー・ドメインからノード B を除去するための以前の変更は失われます。

再結合の状態 3

クラスター内のノードからコピーされた CRG オブジェクトは、加入ノードに送信されます。結果は、以下のとおりです。

- 加入ノードが CRG のリカバリー・ドメインにない場合、変化はありません。
- CRG オブジェクトは、加入ノードに作成される場合があります。これは、クラスター・リソース・サービスがノード上でアクティブでなく、加入ノード上で CRG が削除された場合に起こる可能性があります。

再結合の状態 4

クラスター内のノードの 1 つにある特定の内部情報が使用されて加入ノード上で情報が更新されても、可視の事柄が発生しない場合があります。

マージ

マージ操作は、区画に分割されたノードが再び通信を開始するときに発生することを除けば、再結合操作と似ています。

区画は、クラスター・リソース・サービスがすべてのノードでアクティブのままの本当の区画かもしれません。しかし、通信回線の障害のためにあるノードが他のノードと通信できない場合もあります。または、問題は、ノードに実際に障害が起こっているものの障害として検出されていないことかもしれません。

最初のケースでは、いったん通信の問題が修正されれば、区画は自動的に元のようにマージされます。これは、区画に分割されたノードと両方の区画が定期的に通信しようと試み、結果的に相互に連絡が再確立されたときに発生します。 2 番目のケースは、障害が起こったノードがクラスター内の任意のノードから始動されて、クラスター・リソース・サービスが再始動されなければなりません。

関連概念

25 ページの『再結合』
再結合とは、一度メンバーから外れた後に、再びクラスターの活動メンバーになることです。

154 ページの『区画エラー』
クラスター状態によっては、簡単に修正することができます。 クラスター区画が生じた場合の回復方法を確認してください。 このトピックでは、クラスター区画の回避方法を示し、区画をマージして元に戻す方法を例示します。

関連タスク

115 ページの『クラスター・ノードの開始』
クラスター・ノードを開始すると、クラスター内のノード上のクラスター・リソース・サービスも開始されます。 クラスター・バージョン 3 から、クラスターにアクティブ・ノードがあれば、あるノードが自分自身を開始して現在アクティブなクラスターに再結合することができるようになりました。

157 ページの『区画化ノードを障害ノードに変更する』
時々、「区画」状況が報告されているときに、実際にはノード障害が生じていることがあります。 このことは、クラスター・リソース・サービスが 1 つ以上のノードとの通信を失ったものの、 ノードが引き続き作動可能かどうかを検出できない場合に生じます。 この状況が生じた場合に、ノードに障害が起きたことを知らせるための簡単なメカニズムがあります。

例: マージ:

マージ操作はいくつかの状態で生じる可能性があります。

次の構成のいずれかの場合に、マージ操作が生じ得ます。

表 4. 1 次区画と 2 次区画の間のマージ

マージ	
1 次区画	2 次区画

表 5. 2 次区間と 2 次区画間におけるマージ

マージ	
2 次区画	2 次区画

1 次区画と 2 次区画は、クラスター・リソース・グループ (CRG) に固有のものです。 バックアップ CRG にとって、1 次区画は、1 次アクセス・ポイントと見なされるノードの入った区画と定義されます。 2 次区画は、1 次アクセス・ポイントと見なされるノードの入っていない区画と定義されます。

対等 CRG の場合、リカバリー・ドメイン・ノードがすべて 1 つの区画に含まれる場合、その区画が 1 次区画になります。リカバリー・ドメイン・ノードが区画を超える場合、1 次区画なしになります。 どちらの区画も 2 次区画になります。

クラスター管理ドメインの場合は、ドメインが 2 つ以上の区画に分かれている場合、各区画は継続して 1 つのグループとして作動します。リソースへの変更は引き続き各区画内で同期されます。区画が再度マージされ 1 つになると、システムは各区画の変更を同期します。最終結果はモニター対象のリソースがドメイン全体で整合性が取れている状態です。クラスター管理ドメインが区画化されている間は、MRE を追加または除去することができません。

表 6. 1 次区画と 2 次区画の間のマージ

マージ操作			
1 次区画		2 次区画	
CRG のコピーが含まれる	CRG のコピーが含まれない	CRG のコピーが含まれる	CRG のコピーが含まれない
(1)	(2)	(3)	(4)

上記の図の 1 次と 2 次のマージでは、以下の状況が考えられます。

1. 1 および 3
2. 1 および 4
3. 2 および 3 (主区画はプライマリー・ノードをアクティブにし、CRG のコピーを持たねばならないので、これはあり得ません。)
4. 2 および 4 (主区画はプライマリー・ノードをアクティブにし、CRG のコピーを持たねばならないので、これはあり得ません。)

1 次と 2 次のマージの状態

CRG オブジェクトのコピーが 2 次区画のすべてのノードに送信されます。結果として 2 次区画のノードで起こり得るアクションは以下のとおりです。

- 2 次ノードが CRG のリカバリー・ドメインの中にないので、何も処置が行われない。
- 2 次ノードの CRG のコピーが 1 次区画からのデータに合わせて更新される。
- 2 次ノードが CRG のリカバリー・ドメインから外れるので、CRG オブジェクトが 2 次ノードから削除される。

- CRG オブジェクトが存在しないので、2 次ノードでオブジェクトが作成される。しかし、ノードは 1 次区画から送信される CRG コピーのリカバリー・ドメインの中にある。

表 7. 2 次区画および 2 次区画のマージ・シナリオ

マージ操作			
2 次区画		2 次区画	
CRG のコピーが含まれる	CRG のコピーが含まれない	CRG のコピーが含まれる	CRG のコピーが含まれない
(1)	(2)	(3)	(4)

上記の図の 2 次と 2 次のマージでは、以下の状況が考えられます。

1. 1 および 3
2. 1 および 4
3. 2 および 3
4. 2 および 4

2 次と 2 次のマージの状態 1

プライマリー・バックアップ CRG の場合、一番最近に CRG に変更が加えられたノードが選択され、CRG オブジェクトのコピーが他の区画のすべてのノードに送信されます。一番最近に変更が加えられたノードが複数あるように見えるために複数のノードが選択された場合、リカバリー・ドメインの順序に従ってノードが選択されます。

対等 CRG の 2 つの 2 次区画をマージする場合、アクティブ状況の対等 CRG のバージョンは、他方の区画の別のノードにコピーされます。どちらの区画も対等 CRG に対して同じ状態の場合、クラスター・リソース・グループのリカバリー・ドメインにリストされている最初のノードを含む区画が、別の区画のノードにコピーされます。

プライマリー・バックアップ CRG または対等 CRG のいずれかの区画ノードを受信中に発生する可能性のあるアクションは以下のとおりです。

- ノードが CRG のリカバリー・ドメインでないので、何も処置が行われない。
- ノードが、自分の受け取った CRG オブジェクトのコピーのリカバリー・ドメインの中にあるので、ノード上に CRG が作成される。
- ノードが、自分の受け取った CRG オブジェクトのコピーのリカバリー・ドメインの中にないので、ノードから CRG が削除される。

2 次と 2 次のマージの状態 2 および 3

CRG オブジェクトのコピーを持つ区画からノードが選択され、オブジェクト・データが他の区画のすべてのノードに送信されます。ノードが CRG のリカバリー・ドメインに入っている場合、受信側の区画のノードの CRG オブジェクトが作成されることがあります。

2 次と 2 次のマージの状態 4

クラスター全体の整合性を確実なものとするため、内部データが交換されます。

続いて 1 次区画を 1 次区画と 2 次区画に区分することができます。プライマリー・ノードに障害が起こった場合、クラスター・リソース・サービス (CRS) はこれをノードの障害として検出します。 1 次区画は 2 次区画になります。同じ結果は、クラスター・ノード終了 API を使用するプライマリー・ノードを

終了させた場合にも生じることがあります。再結合またはマージ操作によって 2 次区画のプライマリー・ノードがアクティブになった場合、2 次区画は 1 次区画になります。

マージ操作の場合、出口プログラムが CRG のリカバリー・ドメイン内の全ノードに対して呼び出されます。これは、ノードがどの区画に入っているかには関係ありません。再結合の場合と同じアクション・コードが使用されます。マージの結果として役割が変化することはありませんが、CRG のリカバリー・ドメイン内のノードのメンバーシップ状況は*区画* からアクティブ に変化します。すべての区画を一緒にマージすると区画の状態はクリアされますが、CRG API はすべて使用できます。

複製

複製とは、何かのコピーをリアルタイムで作成することです。つまり、クラスター内のあるノードに含まれているオブジェクトを、同じクラスター内にある他の 1 つ以上のノードにコピーすることを意味します。

複製を行えば、システム上にまったく同じオブジェクトを作成したり保管したりできます。クラスター内の特定のノードに含まれているオブジェクトに変更を加えると、その変更は同じクラスター内にある他のノードにも複製されます。

関連概念

18 ページの『回復データ』
回復データとは、 クラスター内の複数のノードに複製 (コピー) されたデータのことです。

99 ページの『クラスター論理複製の計画』
データの複数のコピーは、論理複製を使って保守されます。 データは、クラスターのプライマリー・ノードからリカバリー・ドメインにある指定されたバックアップ・ノードに複製つまりコピーされます。プライマリー・ノードで障害が発生したときには、指定されたバックアップ・ノードが 1 次アクセス・ポイントを引き継ぐので、データは使用可能なままです。

ハートビート・モニター

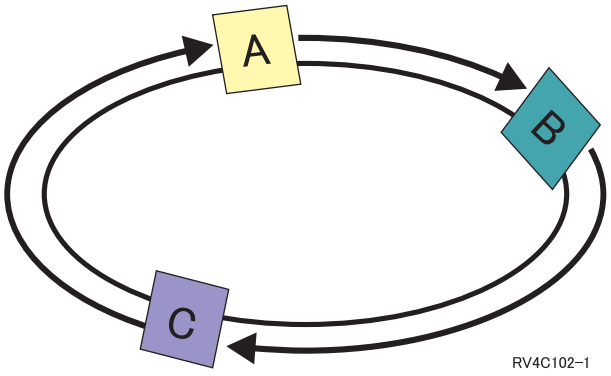
ハートビート・モニターはクラスター・リソース・サービス機能の 1 つで、クラスター内のすべてのノードからクラスター内の自分以外のすべてのノードにシグナルを送信して自分がアクティブであることを伝達してすべてのノードがアクティブであることを確証するものです。

ノードのハートビートに失敗すると、クラスター・リソース・サービスは適切なアクションを実行します。

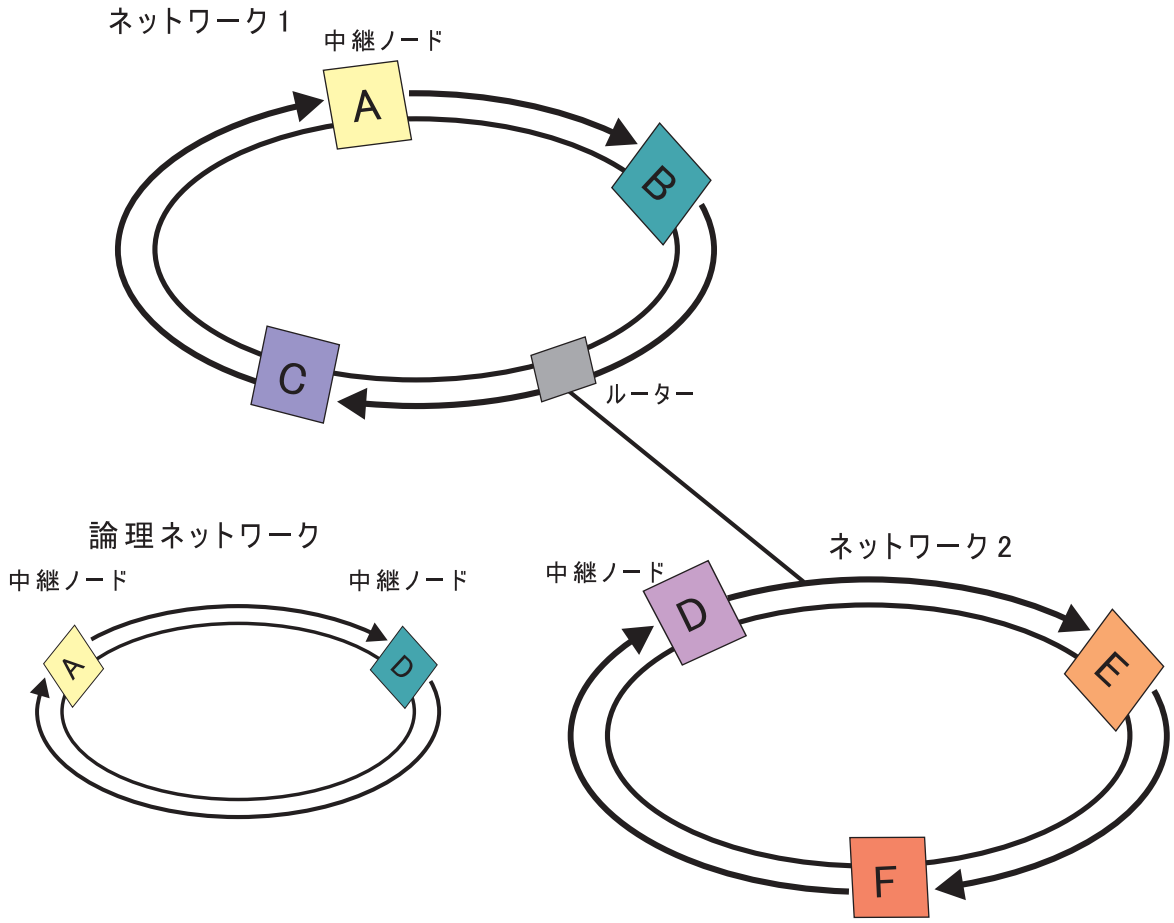
どのようにハートビート・モニターが動作するのかを理解するため、次の例を考慮してください。

例 1

ネットワーク 1



デフォルト (通常) 設定では、クラスターのすべてのノードからそれぞれのアップストリームの近隣ノードに 3 秒ごとにハートビート・メッセージが送信されます。たとえば、ネットワーク 1 において、ノード A、ノード B、ノード C を、ノード A がノード B にメッセージを送信し、ノード B がノード C にメッセージを送信し、ノード C がノード A にメッセージを送信するように構成したとします。ノード A は、ノード B に送信したハートビートに対する肯定応答が送信されてくることと、ダウンストリームにあるノード C からハートビートが送信されてくることを期待します。つまり、ハートビート送信のリングは両方向へ進むということです。ノード C からの送信されてくるべきハートビートをノード A が受信しない場合、ノード A とノード B は 3 秒ごとにハートビートを送信します。ノード C がハートビートを連続 4 回送信しない場合、ハートビート障害のシグナルが送られます。



例 2

この例に別のネットワークを追加して、どのようにルーターと中継ノードが使用されるか見てみましょう。ネットワーク 2 にノード D、ノード E、ノード F を構成します。ネットワーク 2 はルーターを使用してネットワーク 1 と接続されています。ルーターは、他のどこかにある別のルーターに通信を誘導する、さらに別の iSeries サーバーでもルーター・ボックスでもかまいません。すべてのローカル・ネットワークで中継ノードが割り当てられます。この中継ノードは、それぞれのネットワークにおいて最も小さなノード ID を持つノードに割り当てられます。ノード A がネットワーク 1 で中継ノードに割り当てられ、ノード D がネットワーク 2 で中継ノードに割り当てられているとします。そうすると、ノード A とノード D を含む論理ネットワークが作成されます。ルーターと中継ノードを使用して、これらの 2 つのネットワークは互いをモニターし、ノードの障害のシグナルを送信できます。

関連概念

113 ページの『クラスターの管理』
このトピックには、クラスター管理に関係するいくつかのタスクをカバーする情 報が含まれています。

126 ページの『クラスター・パフォーマンス』
クラスターに変更を加えると、 クラスターを管理するのに必要なオーバーヘッドに影響を与える可能性があります。

関連タスク

125 ページの『クラスター状況のモニター』
クラスター・リソース・サービスは、必要に応じて適切なアクションをとりながら、信頼性の高いメッセージ機能とハートビート・モニタリングを使って、クラスターとそのコンポーネントの基本モニターを実行します。

メッセージング機能

クラスター・リソース・サービスのメッセージング機能は、クラスター内の各ノードに注意を払い、クラスター・リソースの状態に関する整合した情報を確実にすべてのノードが保持するようにします。

信頼メッセージングは、クラスタリング固有の再試行値およびタイムアウト値を使用します。これらの値は、ほとんどの環境に適合する値に事前設定されています。しかし、これらの値は、クラスター・リソース・サービスの設定の変更インターフェースにより変更できます。メッセージの再試行値とタイムアウト値は、障害または区画状態のシグナルを送信するまでに何回メッセージをノードに送信するかを決定するために使用されます。ローカル・エリア・ネットワーク (LAN) の場合、障害または区画条件のシグナルが送信されるまで再試行が繰り返されるときの経過時間は、デフォルトの再試行値およびタイムアウト値を使用すると、およそ 45 秒です。リモート・ネットワークの場合、障害または区画条件が存在すると判断されるまでには、さらに長い時間がかかります。リモート・ネットワークの場合には、およそ 4 分 15 秒です。

関連概念

『クラスター・リソース・サービスの設定の変更』
メッセージ・タイムアウトと再試行に影響するデフォルト値は、 最も典型的なインストール・システムに合わせて設定されています。 しかし、自分の通信環境によりしっかりと適合するように、これらの値を変更することが可能です。

113 ページの『クラスターの管理』
このトピックには、クラスター管理に関係するいくつかのタスクをカバーする情 報が含まれています。

関連タスク

125 ページの『クラスター状況のモニター』
クラスター・リソース・サービスは、必要に応じて適切なアクションをとりながら、信頼性の高いメッセージ機能とハートビート・モニタリングを使って、クラスターとそのコンポーネントの基本モニターを実行します。

クラスター・リソース・サービスの設定の変更

メッセージ・タイムアウトと再試行に影響するデフォルト値は、 最も典型的なインストール・システムに合わせて設定されています。 しかし、自分の通信環境によりしっかりと適合するように、これらの値を変更することが可能です。

値は次のどちらかの方法で調整することができます。

- 自分の環境に合った一般パフォーマンス・レベルに設定する。
- 特定のメッセージ調整パラメーターの値を設定して、より明確な調整を図る。

上述の最初の方式では、メッセージ・トラフィックが 3 つの通信レベルの 1 つに調整されます。通常レベルはデフォルトであり、これについてはハートビート・モニターで詳細に説明されています。

2 番目の方式は、通常、専門家の助言が受けられる場合に限って行うべきです。

クラスター・リソース・サービス変更 (QcstChgClusterResourceServices) API の説明の中で、両方の方式が詳細に説明されています。

関連概念

30 ページの『ハートビート・モニター』
ハートビート・モニターはクラスター・リソース・サービス機能の 1 つで、クラスター内のすべてのノードからクラスター内の自分以外のすべてのノードにシグナルを送信して自分がアクティブであることを伝達してすべてのノードがアクティブであることを確証するものです。

クラスター区画

クラスター区画は、通信障害の結果として生じる、アクティブなクラスター・ノードのサブセットです。区画のメンバーは、相互の接続を維持します。

クラスター内にある 1 つ以上のノードとの通信が途絶え、かつそのノードの障害を確認できない場合には、クラスター内にクラスター区画が発生します。クラスター区画条件が検出されると、クラスター区画内のノードで実行できるアクションのタイプが、クラスター・リソース・サービス (CRS) によって制限されます。原因となった問題が修正された時点でクラスター・リソース・サービス (CRS) が区画をマージできるようにするため、区画発生中は機能が制限されます。

クラスター区画が発生している場合、特定の CRG 操作は制限されます。区画のタイプ別に制限されている操作の詳細を確認するには、「クラスター・リソース・グループ API」を参照してください。

クラスター管理ドメインで区画が発生している場合、各区画にあるアクティブ・ノード間で引き続き変更点が同期化されます。ノードが再度マージされると、クラスター管理ドメインにより区画ごとのすべての変更が伝搬されます。こうすることにより、リソースとアクティブ・ドメインとの整合性が保たれます。

関連概念

94 ページの『クラスター区画の回避』
典型的なネットワーク関連クラスター区画が発生しないようにする最も有効な手段としては、クラスター内のすべてのノードを結び付ける冗長通信パスを構成します。

154 ページの『区画エラー』
クラスター状態によっては、簡単に修正することができます。 クラスター区画が生じた場合の回復方法を確認してください。 このトピックでは、クラスター区画の回避方法を示し、区画をマージして元に戻す方法を例示します。

90 ページの『クラスターのハードウェア要件』
i5/OS V4R4M0 またはそれ以降を実行できる iSeries モデルは、 クラスタリングを使用したものと互換性があります。

クラスター・アプリケーション

アプリケーション回復力とは、クラスター環境の重要な要素の 1 つです。 クラスター内で可用性の高いアプリケーションを構築し、使用する場合は、これらのアプリケーションには特定の仕様がある点に留意しなければなりません。

回復アプリケーションをクラスターで利用することにより、クライアントを再構成しなくても、別のクラスター・ノードで再始動できます。さらに、アプリケーションに関連したデータが、切り替えまたはフェイルオーバー後も使用可能です。これは、アプリケーションのユーザーが、アプリケーションとそのデータがプ

ライマリー・ノードからバックアップ・ノードに切り替わる間に経験する中断が最小になる、またはほとんど気付かれないことを意味しています。ユーザーは、アプリケーションとデータがバック・エンドで移動したことを意識する必要がありません。

クラスターでアプリケーションの回復を実現するには、特定の仕様に合致したアプリケーションを使用しなければなりません。アプリケーションが切り替え可能になり、それゆえクラスターのユーザーにとって常に使用可能であるためには、アプリケーションに特定の特性が備わっていなければなりません。これらのアプリケーションの特性に関する詳細については、「高可用性およびクラスター (High Availability and Clusters)」を参照してください。これらの要件があるため、切り替え可能アプリケーションをクラスターに使用する際には、以下のようないくつかのオプションがあります。

1. **クラスター使用可能ソフトウェア・アプリケーションを購入する**

   クラスター使用可能のソフトウェア・プロダクトは、特定の高可用性要件に合致しています。

2. **高可用性を持つように、ユーザーのアプリケーションを作成または変更する**

   独立ソフトウェア販売会社およびアプリケーション・プログラマーが、アプリケーションをカスタマイズして iSeries クラスター環境で切り替え可能になるようにすることができます。

回復アプリケーションを手に入れたなら、クラスター内で管理する必要があります。

関連概念

17 ページの『回復アプリケーション』
回復アプリケーションとは、クライアントを再構成しなくても、別のクラスター・ノードで再始動できるアプリケーションのことです。

クラスタリング対応アプリケーションの i5/OS アーキテクチャー

エンド・ユーザーにとって利用価値が高いのは、計画された停止または予期せぬ停止が発生したときにも引き続き使用可能であるアプリケーションを認識する、高可用性を備えたアプリケーションです。

i5/OS には、レベルの異なるさまざまな高可用性アプリケーションをサポートする、アプリケーション回復力に対応したアーキテクチャーが提供されています。この多様なアプリケーションの中でハイエンドのアプリケーションは、高可用性特性を実証する統合されている機能や高可用性環境の自動化によって拡張されており、それらはクラスター管理ユーティリティーで制御されています。

これらのアプリケーションには、以下の特性があります。

- プライマリー・ノードが使用できなくなったとき、アプリケーションがバックアップ・クラスター・ノードに切り替えることが可能。
- アプリケーションにおいて回復定義および状況データ域に回復環境の定義がなされており、クラスター管理アプリケーションによりアプリケーションの自動構成と活動化が行える。
- アプリケーションが、クラスター関連イベントをハンドルするアプリケーション CRG 出口プログラムによりアプリケーション回復力を提供し、i5/OS クラスター・リソース・サービスの機能を利用している。
- アプリケーションは、ユーザーにアプリケーション・メニュー画面またはそれより進んだ画面を表示するアプリケーション再始動機能を提供する。

より厳格な可用性および再始動特性を備えたアプリケーションには、以下の特性があります。

- アプリケーションが、アプリケーション CRG 出口プログラムにより、クラスター・イベント (アクション・コード) のさらに強力なハンドリングを通して、拡張アプリケーション回復力を提供する。

- アプリケーションが、さらに高いレベルのアプリケーション再始動サポートを提供する。ホスト中心のアプリケーションの場合、コミットメント制御またはチェックポイント機能によりトランザクション境界の状態へ戻します。クライアント中心のアプリケーションの場合、最小のサービスの中断だけでシームレスにフェイルオーバーします。

関連概念

iSeries の高可用性とクラスター (iSeries High Availability and Clusters)

高可用性クラスター・アプリケーションの作成

高可用性アプリケーションとは、クラスター環境におけるシステム障害で回復できるものを言います。

いくつかのレベルのアプリケーション可用性が可能です。

1. アプリケーション・エラーが発生した場合、同じノードでアプリケーション自身が再始動しエラーの潜在的な原因 (壊れた制御データなど) を訂正する。まるでアプリケーションが初めて開始されたように見えます。
2. アプリケーションは、ある程度のチェックポイント・リスタート処理を行う。アプリケーションが障害時点に近づいたかのように見えます。
3. システム障害が発生した場合、アプリケーションはバックアップ・サーバーで再始動する。まるでアプリケーションが初めて開始されたように見えます。
4. システム障害が発生した場合、アプリケーションはバックアップ・サーバーで再始動し、ポイント・リスタート処理を行う。アプリケーションが障害時点に近づいたかのように見えます。
5. システム障害が発生した場合、アプリケーションと関連データ双方で、クラスター内の 1 つ以上の別のノードへの整合フェイルオーバーを実行する。まるでアプリケーションが初めて開始されたように見えます。
6. システム障害が発生した場合、アプリケーションと関連データ双方で、クラスター内の 1 つ以上の別のノードへの整合フェイルオーバーを実行する。アプリケーションは、複数のサーバーである程度のチェックポイント・リスタート処理を行う。アプリケーションが障害時点に近づいたかのように見えます。

注: 上記の 1 から 4 の場合、データの回復はユーザーの責任になります。

アプリケーション・プログラムを回復力に富むものにする:

アプリケーション・プログラムを回復力に富むものにする方法を確認する。

回復アプリケーションの特性として期待されるものは、次のとおりです。

- そのアプリケーションは、現在のノードまたは別のノードで再始動できる。
- そのアプリケーションは、IP アドレスを使用することによって、クライアントにアクセスできる。
- そのアプリケーションには状態情報がない (ステートレス)、あるいは状態情報が明らかになっている。
- そのアプリケーションに関連したデータは、切り替え後も使用できる。

| クラスター環境内でシステム障害が発生した場合に備えて、アプリケーションを回復力に富むものにしておくのに重要な 3 つの要素は、次のとおりです。

アプリケーションそのもの

そのアプリケーションはエラーもしくはシステム障害に対して、どれほどの耐久性を備えているでしょうか? そのアプリケーションはどれほど容易に再始動できるでしょうか?

このことは、アプリケーションにおいてクラスタリング機能を使用することによって処理できます。

関連データ

障害が発生した場合、関連データの可用性に影響しますか?

クラスタリング機能を効果的に活用するクラスター・ミドルウェアである、IBM ビジネス・パートナーの複製プロダクトにより、この問題を処理できます。あるいは、切り替え可能独立ディスク・プール (切り替え可能な独立 ASP) にデータを保管するという方法もあります。

制御機能と管理

データやアプリケーションの可用性をサポートする環境を、どれほど容易に定義できますか?

クラスタリング API を使用し、回復アプリケーションと回復データを組み合わせるサード・パーティー製のクラスター管理ソリューションにより、この問題を処理できます。

可用性の高いクラスター・アプリケーションの再始動:

アプリケーションを再始動するには、アプリケーションは、フェイルオーバーまたは切り替えの際のアプリケーションの状態を把握する必要があります。

状態情報はアプリケーションに特有のものですので、アプリケーションは必要な情報を判別する必要があります。状態情報が全くなくても、アプリケーションは PC で再始動できます。しかしそうすると、アプリケーション内の位置を再確立しなければなりません。

バックアップ・システム用にアプリケーションの状態情報を保管するのに、使用できる方法が何通りかあります。各アプリケーションは、どの方法が最もよく機能するかを判別する必要があります。

- アプリケーションはすべての状態情報を、要求を出しているクライアント・システムに転送できます。切り替えまたはフェイルオーバーが発生すると、アプリケーションはクライアント上に保管されている状態を使用して、新しいサーバーに状態を再確立します。これは、情報配布 API またはクラスター・ハッシュ・テーブル API を使用すると行えます。
- アプリケーションは、状態情報 (ジョブ情報およびアプリケーションに関連した他の制御構造など) をリアルタイムで複製できます。構造上のすべての変更に関して、アプリケーションはバックアップ・システムにその変更を送ります。
- アプリケーションはアプリケーションに関連付けられた適切な状態情報を、アプリケーション用のクラスター・リソース・グループの出口プログラムのデータ部分に保管できます。この方法は、必要な状態情報の量が少ないことが前提です。これを実行するには、クラスター・リソース・グループ変更 (QcstChangeClusterResourceGroup) API を使用できます。
- アプリケーションは、アプリケーションのデータと共に、バックアップ・システムに複製されたデータ・オブジェクト内の状態情報を保管できます。
- アプリケーションは、切り替え可能な IASP に含まれるデータ・オブジェクト内の情報を保管できます。その IASP にはアプリケーションのデータも含まれています。
- アプリケーションは、クライアントの状態情報を保管できます。
- 状態情報は全く保管されずに、回復を実行する必要があります。

注: アプリケーションがある種のチェックポイント・リスタート処理を使用する場合には、保管が必要な情報量が少なくなります。状態情報は、事前に決めたアプリケーション・チェックポイントでのみ保管されます。再始動すると、データベースのコミットメント制御処理が機能するのと同様の仕方で、最後に使用したチェックポイントまで戻してくれます。

クラスター・リソース・グループ出口プログラムの呼び出し:

クラスター・リソース・グループ出口プログラムは、クラスター環境の様々な局面で呼び出されます。

このプログラムはクラスター内のリソースに対し、環境として必要な回復性を確立します。出口プログラムは、回復装置 CRG に対してはオプションとして設定できますが、 他の CRG タイプに対しては必須です。クラスター・リソース・グループ出口プログラムが使用されている場合には、下記の場合も含め、クラスター全体のイベントが出現する際にそのプログラムが呼び出されます。

- 予期しないこととしてノードがクラスターからはずされる場合。
- ノードが、クラスター・ノード終了 (QcstEndClusterNode) API またはクラスター・ノード項目除去 (QcstRemoveClusterNodeEntry) API の結果としてクラスターからはずされる場合。
- クラスターが、クラスター削除 (QcstDeleteCluster) API の結果として削除される場合。
- ノードが、クラスター・ノード開始 (QcstStartClusterNode) API によって活動化される場合。
- 区画化されたノードとの通信が再確立される場合。

この出口プログラムは、

- 名前付き活動化グループまたは呼び出し側の活動化グループ (*CALLER) で実行されます。
- 出口プログラムで処理できない例外が生じるか、出口プログラムが取り消される場合には、再始動パラメーターを無視します。
- 取り消しハンドラーを提供します。

クラスター・リソース・グループ API が実行されると、出口プログラムは、クラスター・リソース・グループ作成 (QcstCreateClusterResourceGroup) API で指定されたユーザー・プロファイルと共に、別個のジョブから呼び出されます。その別個のジョブは、出口プログラムが呼び出されると、API により自動的に作成されます。 データ CRG の出口プログラムがうまく作動しないか異常終了した場合には、リカバリー・ドメインのうちアクション・コードが取り消しになっているすべてのアクティブ・ノードで、クラスター・リソース・グループ出口プログラムが呼び出されます。このアクション・コードによって、終了していないすべての活動がバックアウトされ、クラスター・リソース・グループの元の状態を回復できます。

アプリケーション CRG の出口プログラムがうまく作動しない、または異常終了する場合には、その CRG の状態がアクティブなら、クラスター・リソース・サービスはアプリケーションを再始動しようとします。クラスター・リソース・グループ出口プログラムは、再始動のアクション・コードによって呼び出されます。指定された最大回数を試みたもののアプリケーションを再始動できない場合には、クラスター・リソース・グループ出口プログラムはフェイルオーバーのアクション・コードにより呼び出されます。出口プログラムが開始のアクション・コードによって呼び出される場合のみ、再始動カウントがリセットされます。CRG の開始、フェイルオーバー、または切り替えの結果として、そのようにリセットされる可能性があります。

クラスター・リソース・グループが開始されると、プライマリー・ノードで呼び出されるアプリケーション CRG 出口プログラムは、アプリケーションが終了するか、エラーが生じるまでは、クラスター・リソース・サービスに制御を戻しません。アプリケーション CRG がアクティブになった後、クラスター・リソース・サービスがアプリケーション CRG 出口プログラムにイベントを通知する必要がある場合には、出口プログラムの別のインスタンスが異なるジョブで開始されます。開始または再始動を除く他のアクション・コードは、戻ってくることが期待されます。

クラスター・リソース・グループ出口プログラムが呼び出されると、処理中のクラスター・イベント、クラスター・リソースの現在の状態、およびクラスター・リソースの期待される状態を識別するパラメーターの集合が渡されます。

クラスター・リソース・グループ出口プログラムに関する完全な情報は、クラスター API の資料のクラスター・リソース・グループ出口プログラムを参照してください。アクション・コードごとに、クラスター・リソース・グループ出口プログラムにどのような情報が渡されるのかについても説明されています。出口プ

ログラムを作成する際の基礎として使用できるサンプル・ソース・コードが、 QUSRTOOL ライブラリーで提供されています。 QATTSYSC ファイルの TCSTAPPEXT メンバーを参照してください。

アプリケーション CRG の考慮事項

アプリケーション・クラスター・リソース・グループは、 アプリケーション面での回復を管理します。

アプリケーション CRG の テークオーバー IP アドレスの管理:

| クラスター・リソース・サービスを使用した、アプリケーション CRG の テークオーバー IP アドレスの
| 管理 この管理作業は手作業で行うこともできます。

アプリケーションのテークオーバー IP アドレスを、管理されているアプリケーション CRG に関連付ける方法としては、2 とおりの方法があります。最も簡単な方法がデフォルトになっていますが、それはクラスター・リソース・サービスにテークオーバー IP アドレスを管理させる方法です。この方式では、リカバリー・ドメインに後で追加されるノードも含め、リカバリー・ドメイン内の全ノードにテークオーバー IP アドレスを作成するようクラスター・リソース・サービスに指示が出されます。この方式が選択されると、その時点でテークオーバー IP アドレスはリカバリー・ドメイン内のすべてのノードで定義できなくなります。

別の方法は、テークオーバー IP アドレスをユーザー自身が管理します。この方法では、クラスター・リソース・サービスはテークオーバー IP アドレスを構成するいかなるステップも実行しないよう指示されます。ユーザーが構成に責任を持ちます。クラスター・リソース・グループを開始する前に、リカバリー・ドメイン内の全ノード (複製ノードは除く) にテークオーバー IP アドレスを追加する必要があります。 活動 CRG のリカバリー・ドメインに追加される任意のノードは、追加される前にテークオーバー IP アドレスを構成しておかなければなりません。

複数サブネット

デフォルトではすべてのリカバリー・ドメイン・ノードは同一のサブネット上にありますが、アプリケーション・テークオーバー IP アドレスを複数のサブネットで機能させることが可能です。リカバリー・ドメイン内のノードを複数のサブネットに広げる際に、アプリケーションのテークオーバー IP アドレスを構成するステップの詳細は、アプリケーション切り替えを使用可能にするを参照してください。

関連概念

40 ページの『例: アプリケーション・クラスター・リソース・グループのフェイルオーバー・アクション』
この例は、フェイルオーバーの 1 つのシナリオを示したものです。

118 ページの『アクティブなテークオーバー IP アドレスを使ったアプリケーション CRG の作成』
アプリケーション CRG の作成時に、アクティブなテークオーバー IP アドレスを許可するよう指定することができます。 それが許可されるのは、ユーザーがテークオーバー IP アドレスを構成する場合だけです。

サブネットをまたいだアプリケーション切り替えを使用可能にする:

一般的に、クラスタリングを行うには、アプリケーション・クラスター・リソース・グループのリカバリー・ドメインのすべてのクラスター・ノードが、同じ LAN 上に置かれている (同じサブネット・アドレッシングを使用している) 必要があります。

構成されたアプリケーションのテークオーバー IP アドレスを、 リカバリー・ドメインの 1 つのノードから別のノードへと切り替えるために使用される、基盤となるネットワーク・プロトコルはアドレス解決プロ

トコル (ARP) です。しかし、リカバリー・ドメインを拡張して、商用ルーターで区切られた他の LAN に存在するクラスター・ノードをそこに含めることが可能です。

この拡張は、仮想 IP アドレス・サポートを使用することと、クラスター・ノードおよびネットワークの商用ルーターで Routing Information Protocol (RIP) を利用することによって可能になります。 詳細は、『>アプリケーション切り替えを使用可能にする』を参照してください。

>アプリケーション切り替えを使用可能にする:

| クラスター・リソース・サービスは、アプリケーションの CRG の構成時に、ユーザーの構成するテークオーバー (引き継ぎ) IP アドレスをサポートします。

以下の手動構成ステップが、切り替え環境を使用可能にするために必要とされます。**この一連の指示は、リカバリー・ドメイン内のすべてのノードにおいて行わなければなりません。また、指定されたアプリケーション CRG のリカバリー・ドメイン内のノードとなる、クラスター内の他のノードに対しても、繰り返して行う必要があります。**

1. アプリケーション CRG の使用するテークオーバー IP アドレスを選択します。
   - 混乱を避けるために、このアドレスは、クラスター・ノードまたはルーターによって使用される他の既存アドレスとオーバーラップすべきではありません。たとえば、19.19.19.19 を選択した場合は、19.0.0.0 (または 19.19.0.0 など) が、システム・ルーティング・テーブルによって認識される経路であってはなりません。
   - テークオーバー・インターフェース (たとえば 19.19.19.19) を追加します。それを作成する際には、回線記述を *VIRTUALIP、サブネット・マスクを 255.255.255.255 (ホスト経路)、最大伝送単位を 1500 (576から16388 の任意の数)、および自動始動を *NO に指定します。このテークオーバー・アドレス (たとえば 19.19.19.19) は、次のステップで関連ローカル・インターフェースとしてそれを識別する前に、*VIRTUALIP アドレスとして存在している必要があります。しかし、アクティブである必要はありません。
2. クラスターを作成するか、またはクラスターにノードを追加する際に、意図するテークオーバー IP アドレスを、クラスター通信によって使用されるよう指定された IP アドレスの 1 つまたは両方と関連付けます。
   - たとえばこれは、クラスタリングのためにローカルで使用されるイーサネット・バスのクラスター・ノードの IP アドレスで、19.19.19.19 テークオーバー・アドレスを関連ローカル・インターフェースにすることを意味します。このことは、各クラスター・ノード上のそれぞれのクラスター・アドレスごとに行う必要があります。

     注: クラスター・アドレスを終了して、この変更が CFGTCP 下で達成されるようにしなければなりません。
3. クラスターを作成し、CRG を作成します。 アプリケーション CRG の場合、「引き継ぎ IP アドレスの構成」フィールドに `QcstUserCfgsTakeoverIpAddr` を指定します。どのアプリケーション CRG も開始しないでください。
4. CFGTCP 下の「TCP/IP アプリケーションの構成 (Configure TCP/IP Applications)」(オプション 20) を使用した後、 「RouteD の構成 (Configure RouteD)」(オプション 2)、さらに「RouteD 属性の変更 (Change RouteD attributes)」(オプション 1) を使用して、「供給 (Supply)」が *YES に設定されていることを確認します。設定されていなければ、*YES に設定し、それぞれのクラスター・ノード上で ROUTED (RIP または RIP-2) を開始または再開します。
   - NETSTAT オプション 3 は、現在実行中であれば、ローカル・ポートを使用して ROUTED を表示します。 ROUTED は、CRG リカバリー・ドメイン内のすべてのクラスター・ノードで経路を実行し公示している必要があります (Supply = *YES)。

5. 必ず、リカバリー・ドメイン LAN を相互接続しているネットワーク内のすべての商用ルーターが、RIP 用のホスト経路を受け入れ、公示しているようにしてください。
   - これは必ずしもルーター用のデフォルト設定ではありません。言語はルーターの製造者によって異なりますが、RIP インターフェース下では、ホスト経路を送信し、動的ホストを受信していることが期待されます。
   - このことはまた、iSeries サーバーを指すルーター・インターフェースと、ルーター間インターフェースの両方に適用されます。

   注: このシナリオでは、iSeries サーバーをルーターとして使用しないでください。ルーティングの目的で設計された商用ルーター (IBM またはそれ以外のもの) を使用してください。この機能を処理するように iSeries ルーティングを構成することはできません。
6. ここで、クラスター・ノードの 1 つでテークオーバー・アドレスを手動でアクティブにし、RIP を最大 5 分許可して経路を伝搬できるようにし、CRG リカバリー・ドメイン内のすべてのノードから、またこのアドレスを使用する LAN 上の選択されたクライアントから、テークオーバー・アドレスを PING することができます。
   - この検証テストの後、テークオーバー・アドレスが再度終了していることを確認してください。
   - CRG の開始時に、クラスタリングによって、指定されたプライマリー・ノード上のアドレスが開始されます。
7. アプリケーション CRG を開始します。
   - ここで、指定された優先ノード上でクラスタリングすることにより、テークオーバー・アドレスが開始され、RIP がリカバリー・ドメイン全体で経路を公示します。 RIP は最大 5 分かけて、ドメイン全体で経路を更新します。 RIP 機能は、CRG の開始機能とは独立しています。

重要事項:

- アプリケーション CRG リカバリー・ドメイン内のすべてのクラスター・ノードで上記の手順が実行されない場合には、切り替えプロセス中にクラスターが停止します。
- たとえレプリカ・ノードにフェイルオーバーしなくても、後にバックアップとなるときにレプリカ・ノードが変更される場合に、レプリカ・ノードで手順を実行するのはよいことです。
- 複数の仮想 IP アドレスを使用する場合には、それぞれのアドレスに、別個のアプリケーション CRG と、それらに関連付けられる別個の IP アドレスが必要です。このアドレスは、同じ物理アダプター上の別の論理 IP アドレスであっても、まったく別の物理アダプターであってもかまいません。また、ルーティング・テーブル内であいまいさがないようにするために注意が必要です。このことは、次のように行うことによって、最もよく達成されます。
  - それぞれの仮想 IP アドレスのルーティング・テーブルに *DFTROUTE を追加します。
  - このことを CFGTCP (オプション 2) で行うことができます。
  - ネクスト・ホップを含む、すべてのパラメーターを同じように設定して、選択するルーターに到達するように設定します。ただし、優先バインディング・インターフェースは、この経路によって表される仮想 IP アドレスに関連したローカル・システム IP アドレスに設定する必要があります。

例: アプリケーション・クラスター・リソース・グループのフェイルオーバー・アクション:

この例は、フェイルオーバーの 1 つのシナリオを示したものです。

再試行限界を超過したり、ジョブが取り消されたりしたために、回復アプリケーションのクラスター・リソース・グループがフェイルオーバーすると、以下のことが発生します。

- CRG のリカバリー・ドメイン内にあるすべてのアクティブ・ノードで、フェイルオーバーのアクション・コードによってクラスター・リソース・グループ出口プログラムが呼び出されます。このことは、クラスター・リソース・サービスが、アプリケーションのアクセス・ポイントを最初のバックアップにフェイルオーバーする準備を進めていることを意味します。
- クラスター・リソース・サービスが、プライマリー・ノード上のテークオーバー・インターネット・プロトコル (IP) 接続を終了します。 テークオーバー IP アドレスの詳細については、アプリケーション CRG の IP アドレスの管理を参照してください。
- クラスター・リソース・サービスが、最初のバックアップ・ノード (新しいプライマリー・ノード) 上でテークオーバー IP 接続を開始します。
- クラスター・リソース・サービスが、新しいプライマリー・ノード上でのみ開始のアクション・コードによって、クラスター・リソース・グループ出口プログラムを呼び出すジョブを投入します。このアクションによって、アプリケーションが再始動します。

上記の例は、フェイルオーバーの 1 つのシナリオを示したものです。フェイルオーバーの別のシナリオは、これとは違う動作になる場合があります。

例: アプリケーション出口プログラム:

このコード例には、アプリケーション・クラスター・リソース・グループ出口プログラムのサンプルのコードも含まれています。

このコード例は QUSRTOOL ライブラリーにあります。

このコード・サンプルを使用すると、「コードに関するライセンス情報および特記事項」の条件に同意したことになります。

```
/*****************************************************************************/
/*                                                                           */
/* Library:  QUSRTOOL                                                        */
/* File:     QATTSYSC                                                        */
/* Member:   TCSTAPPEXT                                                      */
/* Type:     ILE C                                                           */
/*                                                                           */
/* Description:                                                              */
/* This is an example application CRG exit program which gets called for    */
/* various cluster events or cluster APIs.  The bulk of the logic must      */
/* still be added because that logic is really dependent upon the unique    */
/* things that need to be done for a particular application.                */
/*                                                                           */
/* The intent of this example to to provide a shell which contains the      */
/* basics for building a CRG exit program.  Comments throughout the example*/
/* highlight the kinds of issues that need to be addressed by the real      */
/* exit program implementation.                                             */
/*                                                                           */
/* Every action code that applies to an application CRG is handled in this */
/* example.                                                                  */
/*                                                                           */
/* The tcstdtaara.h include is also shipped in the QUSRTOOL library.  See   */
/* the TCSTDTAARA member in the QATTSYSC file.                               */
/*                                                                           */
/* Change log:                                                               */
/* Flag Reason   Ver    Date   User Id   Description                         */
/* ____ _______ _____ ______ _____________________________________________ */
/* ...  D98332   v5r1m0 000509 ROCH     Initial creation.                    */
/* $A1  P9950070 v5r2m0 010710 ROCH     Dataarea fixes                       */
/* $A2  D99055   v5r2m0 010913 ROCH     Added CancelFailover action code     */
/* $A3  D98854   v5r2m0 010913 ROCH     Added VerificationPhase action code*/
/* $A4  P9A10488 v5r3m0 020524 ROCH     Added example code to wait for data*/
/*                                      CRGs on switchover action code       */
```

```
/*                                                                            */
/******************************************************************************/


/*----------------------------------------------------------------------------*/
/*                                                                            */
/* Header files                                                               */
/*                                                                            */
/*----------------------------------------------------------------------------*/
#include              /* Useful when debugging                  */
#include             /* offsetof macro                          */
#include             /* system function                         */
#include             /* String functions                        */
#include             /* Exception handling constants/structures  */
#include               /* Various cluster constants              */
#include           /* Structure of CRG information             */
#include "qusrtool/qattsysc/tcstdtaara" /* QCSTHAAPPI/QCSTHAAPPO data areas*/
#include           /* API to Retrieve contents of a data area  */
#include              /* API error code type definition          */
#include              /* mitime builtin                          */
#include           /* waittime builtin                         */


/*----------------------------------------------------------------------------*/
/*                                                                            */
/* Constants                                                                  */
/*                                                                            */
/*----------------------------------------------------------------------------*/
#define UnknownRole -999
#define DependCrgDataArea "QCSTHAAPPO"
#define ApplCrgDataArea "QCSTHAAPPI"
#define Nulls 0x0000000000000000000


/*----------------------------------------------------------------------------*/
/*                                                                            */
/* The following constants are used in the checkDependCrgDataArea()           */
/* function.  The first defines how long to sleep before checking the data */
/* area.  The second defines that maximum time to wait for the data area   */
/* to become ready before failing to start the application when the Start  */
/* CRG function is being run.  The third defines the maximum wait time for */
/* the Initiate Switchover or failover functions.                             */
/*                                                                            */
/*----------------------------------------------------------------------------*/
#define WaitSecondsIncrement 30
#define MaxStartCrgWaitSeconds 0
#define MaxWaitSeconds 900

/*----------------------------------------------------------------------------*/
/*                                                                            */
/* As this exit program is updated to handle new action codes, change the  */
/* define below to the value of the highest numbered action code that is   */
/* handled.                                                                   */
/*                                                                            */
/*----------------------------------------------------------------------------*/
#define MaxAc 21

/*----------------------------------------------------------------------------*/
/*                                                                            */
/* If the exit program data in the CRG has a particular structure to it,   */
/* include the header file for that structure definition and change the    */
/* define below to use that structure name rather than char.                  */
/*                                                                            */
/*----------------------------------------------------------------------------*/
#define EpData char

/*----------------------------------------------------------------------------*/
```

```
/*                                                                   */
/* Change the following define to the library the application resides in   */
/* and thus where the QCSTHAAPPO and QCSTHAAPPI data areas will be found.  */
/*                                                                   */
/*-------------------------------------------------------------------*/
#define ApplLib "QGPL"


/*-------------------------------------------------------------------*/
/*                                                                   */
/* Prototypes for internal functions.                                */
/*                                                                   */
/*-------------------------------------------------------------------*/
static int getMyRole(Qcst_EXTP0100_t *, int, int);
#pragma argopt(getMyRole)
static int doAction(int, int, int, Qcst_EXTP0100_t *, EpData *);
#pragma argopt(doAction)
static int createCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int startCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int restartCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int endCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int verifyPhase(int, int, Qcst_EXTP0100_t *, EpData *);
static int deleteCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int memberIsJoining(int, int, Qcst_EXTP0100_t *, EpData *);
static int memberIsLeaving(int, int, Qcst_EXTP0100_t *, EpData *);
static int switchPrimary(int, int, Qcst_EXTP0100_t *, EpData *);
static int addNode(int, int, Qcst_EXTP0100_t *, EpData *);
static int rmvNode(int, int, Qcst_EXTP0100_t *, EpData *);
static int chgCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int deleteCrgWithCmd(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoPriorAction(int, int, Qcst_EXTP0100_t *, EpData *);
static int endNode(int, int, Qcst_EXTP0100_t *, EpData *);
static int chgNodeStatus(int, int, Qcst_EXTP0100_t *, EpData *);
static int cancelFailover(int, int, Qcst_EXTP0100_t *, EpData *);
static int newActionCode(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoCreateCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoStartCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoEndCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoMemberIsJoining(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoMemberIsLeaving(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoSwitchPrimary(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoAddNode(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoRmvNode(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoChgCrg(int, int, Qcst_EXTP0100_t *, EpData *);
static int undoCancelFailover(int, int, Qcst_EXTP0100_t *, EpData *);
static void bldDataAreaName(char *, char *, char *);
#pragma argopt(bldDataAreaName)
static int checkDependCrgDataArea(unsigned int);
#pragma argopt(checkDependCrgDataArea)
static void setApplCrgDataArea(char);
#pragma argopt(setApplCrgDataArea)
static void cancelHandler(_CNL_Hndlr_Parms_T *);
static void unexpectedExceptionHandler(_INTRPT_Hndlr_Parms_T *);
static void endApplication(unsigned int, int, int, Qcst_EXTP0100_t *, EpData *);
#pragma argopt(endApplication)

/*-------------------------------------------------------------------*/
/*                                                                   */
/* Some debug routines                                               */
/*                                                                   */
/*-------------------------------------------------------------------*/
static void printParms(int, int, int, Qcst_EXTP0100_t *, EpData *);
static void printActionCode(unsigned int);
static void printCrgStatus(int);
static void printRcvyDomain(char *,
                            unsigned int,
                            Qcst_Rcvy_Domain_Array1_t *);
```

```
static void printStr(char *, char *, unsigned int);


/*--------------------------------------------------------------------------*/
/*                                                                          */
/* Type definitions                                                         */
/*                                                                          */
/*--------------------------------------------------------------------------*/

/*--------------------------------------------------------------------------*/
/*                                                                          */
/* This structure defines data that will be passed to the exception and     */
/* cancel handlers.  Extend it with information unique to your application.*/
/*                                                                          */
/*--------------------------------------------------------------------------*/
typedef struct {
  int *retCode;             /* Pointer to return code                     */
  EpData *epData;           /* Exit program data from the CRG             */
  Qcst_EXTP0100_t *crgData; /* CRG data                                   */
  unsigned int actionCode;  /* The action code                            */
  int role;                 /* This node's recovery domain role           */
  int priorRole;            /* This node's prior recovery domainrole      */
} volatile HandlerDataT;


/*--------------------------------------------------------------------------*/
/*                                                                          */
/* Function pointer array for handling action codes.  When the exit program*/
/* is updated to handle new action codes, add the new function names to     */
/* this function pointer array.                                             */
/*                                                                          */
/*--------------------------------------------------------------------------*/
static int (*fcn[MaxAc+1]) (int role,
                            int priorRole,
                            Qcst_EXTP0100_t *crgData,
                            EpData *epData) = {
  newActionCode,   /* 0 - currently reserved */
  createCrg,       /* 1  */
  startCrg,        /* 2  */
  restartCrg,      /* 3  */
  endCrg,          /* 4  */
  verifyPhase,     /* 5 - currently reserved */
  newActionCode,   /* 6 - currently reserved */
  deleteCrg,       /* 7  */
  memberIsJoining, /* 8  */
  memberIsLeaving, /* 9  */
  switchPrimary,   /* 10 */
  addNode,         /* 11 */
  rmvNode,         /* 12 */
  chgCrg,          /* 13 */
  deleteCrgWithCmd,/* 14 */
  undoPriorAction, /* 15 */
  endNode,         /* 16 */
  newActionCode,   /* 17 - applies only to a device CRG */
  newActionCode,   /* 18 - applies only to a device CRG */
  newActionCode,   /* 19 - applies only to a device CRG */
  chgNodeStatus,   /* 20 */
  cancelFailover   /* 21 */
};


/*--------------------------------------------------------------------------*/
/*                                                                          */
/* Function pointer array for handling prior action codes when called with */
/* the Undo action code.  When the exit program is updated to handle       */
/* Undo for new action codes, add the new function names to this function  */
/* pointer array.                                                           */
```

```
/*                                                                           */
/*---------------------------------------------------------------------------*/
static int (*undoFcn[MaxAc+1]) (int role,
                                int priorRole,
                                Qcst_EXTP0100_t *crgData,
                                EpData *epData) = {
  newActionCode,       /* 0 - currently reserved */
  undoCreateCrg,       /* 1  */
  undoStartCrg,        /* 2  */
  newActionCode,       /* 3  */
  undoEndCrg,          /* 4  */
  newActionCode,       /* 5 - no undo for this action code */
  newActionCode,       /* 6 - currently reserved */
  newActionCode,       /* 7  */
  undoMemberIsJoining, /* 8  */
  undoMemberIsLeaving, /* 9  */
  undoSwitchPrimary,   /* 10 */
  undoAddNode,         /* 11 */
  undoRmvNode,         /* 12 */
  undoChgCrg,          /* 13 */
  newActionCode,       /* 14 */
  newActionCode,       /* 15 */
  newActionCode,       /* 16 */
  newActionCode,       /* 17 - applies only to a device CRG */
  newActionCode,       /* 18 - applies only to a device CRG */
  newActionCode,       /* 19 - applies only to a device CRG */
  newActionCode,       /* 20 */
  undoCancelFailover   /* 21 */
};



/****************************************************************************/
/*                                                                          */
/* This is the entry point for the exit program.                            */
/*                                                                          */
/****************************************************************************/
void main(int argc, char *argv[]) {

  HandlerDataT hdlData;


/*--------------------------------------------------------------------------*/
  /*                                                                        */
  /* Take each of the arguments passed in the argv array and castit to    */
  /* the correct data type.                                                 */
  /*                                                                        */

/*--------------------------------------------------------------------------*/
  int *retCode    = (int *)argv[1];
  unsigned int *actionCode = (unsigned int *)argv[2];
  EpData *epData           = (EpData *)argv[3];
  Qcst_EXTP0100_t *crgData = (Qcst_EXTP0100_t *)argv[4];
  char *formatName         = (char *)argv[5];


/*--------------------------------------------------------------------------*/
  /*                                                                        */
  /* Ensure the format of the data being passed is what we areexpecting.  */
  /* If not, a change has been made and this exit program needs tobe      */
  /* updated to accomodate the change.  Add appropriate errorlogging for  */
  /* your application design.                                               */
  /*                                                                        */

/*--------------------------------------------------------------------------*/
  if (0 != memcmp(formatName, "EXTP0100", 8))
    abort();
```

```
/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Set up the data that will be passed to the exception andcancel   */
  /* handlers.                                                        */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  hdlData.retCode    = retCode;
  hdlData.epData     = epData;
  hdlData.crgData    = crgData;
  hdlData.actionCode = *actionCode;
  hdlData.role       = UnknownRole;
  hdlData.priorRole  = UnknownRole;
  _VBDY();  /* force changed variables to home storage location       */


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Enable an exception handler for any and all exceptions.          */
  /*                                                                  */

/*--------------------------------------------------------------------*/
#pragma exception_handler(unexpectedExceptionHandler, hdlData, ¥
                          _C1_ALL, _C2_ALL, _CTLA_INVOKE )


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Enable a cancel handler to recover if this job is canceled.     */
  /*                                                                  */

/*--------------------------------------------------------------------*/
#pragma cancel_handler(cancelHandler, hdlData)


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Extract the role and prior role of the node this exit program is  */
  /* running on.  If the cluster API or event changes the recovery domain  */
  /* (node role or membership status), the new recovery domain's offset is */
  /* passed in Offset_Rcvy_Domain_Array and the offset of the recovery     */
  /* domain as it looked prior to the API or cluster event is passed in    */
  /* Offset_Prior_Rcvy_Domain_Array.  If the recovery domain isn't changed,*/
  /* only Offset_Rcvy_Domain_Array can be used to address the recovery     */
  /* domain.                                                          */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  hdlData.role = getMyRole(crgData,
                           crgData->Offset_Rcvy_Domain_Array,
                           crgData->Number_Nodes_Rcvy_Domain);
  if (crgData->Offset_Prior_Rcvy_Domain_Array)
    hdlData.priorRole =
                getMyRole(crgData,

crgData->Offset_Prior_Rcvy_Domain_Array,

crgData->Number_Nodes_Prior_Rcvy_Domain);
  else
    hdlData.priorRole = hdlData.role;
  _VBDY();  /* force changed variables to home storage location       */


/*--------------------------------------------------------------------*/
  /*                                                                  */
```

```
  /* Enable the following to print out debug information.                */
  /*                                                                     */

/*---------------------------------------------------------------------*/
  /*
  printParms(*actionCode, hdlData.role, hdlData.priorRole, crgData,
epData);
  */


/*---------------------------------------------------------------------*/
  /*                                                                     */
  /* Do the correct thing based upon the action code.  The return code   */
  /* is set to the function result of doAction().                        */
  /*                                                                     */

/*---------------------------------------------------------------------*/
  *retCode = doAction(*actionCode,
                      hdlData.role,
                      hdlData.priorRole,
                      crgData,
                      epData);


/*---------------------------------------------------------------------*/
  /*                                                                     */
  /* The exit program job will end when control returns to the operating */
  /* system at this point.                                               */
  /*                                                                     */

/*---------------------------------------------------------------------*/
  return;

#pragma disable_handler  /* unexpectedExceptionHandler                 */
#pragma disable_handler  /* cancelHandler                              */
}  /* end main()                                                        */


/***********************************************************************/
/*                                                                     */
/* Get the role of this particular node from one of the views of the   */
/* recovery domain.                                                    */
/*                                                                     */
/* APIs and cluster events which pass the updated and prior recovery domain*/
/* to the exit program are:                                            */
/* QcstAddNodeToRcvyDomain                                             */
/* QcstChangeClusterNodeEntry                                          */
/* QcstChangeClusterResourceGroup                                      */
/* QcstEndClusterNode (ending node does not get the prior domain)      */
/* QcstInitiateSwitchOver                                              */
/* QcstRemoveClusterNodeEntry (removed node does not get the prior domain) */
/* QcstRemoveNodeFromRcvyDomain                                        */
/* QcstStartClusterResourceGroup (only if inactive backup nodes are    */
/*                                reordered)                           */
/* a failure causing failover                                          */
/* a node rejoining the cluster                                        */
/* cluster partitions merging                                          */
/*                                                                     */
/* All other APIs pass only the updated recovery domain.               */
/*                                                                     */
/***********************************************************************/
static int getMyRole(Qcst_EXTP0100_t *crgData, int offset, int
count) {

  Qcst_Rcvy_Domain_Array1_t *nodeData;
  unsigned int iter = 0;
```

```
/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Under some circumstances, the operating system may not be able to     */
  /* determine the ID of this node and passes *NONE.  An example of such a */
  /* circumstance is when cluster resource services is not active on a     */
  /* node and the DLTCRG CL command is used.                          */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  if (0 == memcmp(crgData->This_Nodes_ID, QcstNone,
sizeof(Qcst_Node_Id_t)))
    return UnknownRole;


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Compute a pointer to the first element of the recovery domain array.  */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  nodeData = (Qcst_Rcvy_Domain_Array1_t *)((char *)crgData +
offset);


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Find my node in the recovery domain array.  I will not be in the      */
  /* prior recovery domain if I am being added by the Add Node to Recovery */
  /* Domain API.                                                      */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  while (  0 != memcmp(crgData->This_Nodes_ID,
                       nodeData->Node_ID,
                       sizeof(Qcst_Node_Id_t))
         &&
           iter &lt; count
        ) {
    nodeData++;
    iter++;
  }

  if (iter &lt; count)
    return nodeData->Node_Role;
  else
    return UnknownRole;
}  /* end getMyRole()                                               */


/**********************************************************************/
/*                                                                    */
/* Call the correct function based upon the cluster action code.  The      */
/* doAction() function was split out from main() in order to clarify the   */
/* example.  See the function prologues for each called function for       */
/* information about a particular cluster action.                          */
/*                                                                    */
/* Each action code is split out into a separate function only to help     */
/* clarify this example.  For a particular exit program, some action codes */
/* may perform the same function in which case multiple action codes could */
/* be handled by the same function.                                        */
/*                                                                    */
/**********************************************************************/
static int doAction(int actionCode,
                    int role,
                    int priorRole,
                    Qcst_EXTP0100_t *crgData,
```

```
                    EpData *epData) {

/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* For action codes this exit program knows about, call a function to    */
  /* do the work for that action code.                                */
  /*                                                                  */

/*--------------------------------------------------------------------*/

  if (actionCode &lt;= MaxAc )
    return (*fcn[actionCode]) (role, priorRole, crgData, epData);
  else

/*--------------------------------------------------------------------*/
    /*                                                                */
    /* IBM has defined a new action code in a new operating system release */
    /* and this exit program has not yet been updated to handle it.  Take a*/
    /* default action for now.                                        */
    /*                                                                */

/*--------------------------------------------------------------------*/
    return newActionCode(role, priorRole, crgData, epData);
}  /* end doAction()                                                  */


/**********************************************************************/
/*                                                                    */
/* Action code = QcstCrgAcInitialize                                  */
/*                                                                    */
/* The QcstCreateClusterResourceGroup API was called.  A new cluster  */
/* resource group object is being created.                            */
/*                                                                    */
/* Things to consider:                                                */
/*   - Check that the application program and all associated objects are on*/
/*     the primary and backup nodes.  If the objects are not there,   */
/*     consider sending error/warning messages or return a failure return  */
/*     code.                                                          */
/*   - Check that required data or device CRGs are on all nodes in the     */
/*     recovery domain.                                               */
/*   - Perform any necessary setup that is required to run the       */
/*     the application on the primary or backup nodes.                */
/*   - If this CRG is enabled to use the QcstDistributeInformation API,    */
/*     the user queue needed by that API could be created at this time.    */
/*                                                                    */
/**********************************************************************/
static int createCrg(int role,
                     int doesNotApply,
                     Qcst_EXTP0100_t *crgData,
                     EpData *epData) {

  return QcstSuccessful;
}  /* end createCrg()                                                 */


/**********************************************************************/
/*                                                                    */
/* Action code = QcstCrgAcStart                                       */
/*                                                                    */
/* The QcstStartClusterResourceGroup API was called.  A cluster resource   */
/* group is being started.                                            */
/* The QcstInitiateSwitchOver API was called and this is the second action */
/* code being passed to the exit program.                             */
/* The fail over event occurred and this is the second action code being   */
/* passed to the exit program.                                        */
/*                                                                    */
```

```
/* A maximum wait time is used when checking to see if all dependent CRGs  */
/* are active.  This is a short time if the CRG is being started because of*/
/* the QcstStartClusterResourceGroup API.  It is a longer time if it is    */
/* because of a failover or switchover.  When failover or switchover are   */
/* being done, it make take a while for data or device CRGs to become      */
/* ready so the wait time is long.  If the Start CRG API is being used, the*/
/* dependent CRGs should already be started or some error occurred, the    */
/* CRGs were started out of order, etc. and there is no need for a long    */
/* wait.                                                                   */
/*                                                                         */
/* Things to consider:                                                     */
/*   - If this node's role is primary, the application should be started.  */
/*     This exit program should either call the application so that it runs*/
/*     in this same job or it should monitor any job started by this       */
/*     exit program so the exit program knows when the application job     */
/*     ends.  By far, the simplest approach is run the application in this */
/*     job by calling it.                                                  */
/*     Cluster Resource Services is not expecting this exit program to     */
/*     return until the application finishes running.                      */
/*   - If necessary, start any associated subsystems, server jobs, etc.    */
/*   - Ensure that required data CRGs have a status of active on all nodes */
/*     in the recovery domain.                                             */
/*                                                                         */
/*-------------------------------------------------------------------------*/
static int startCrg(int role,
                    int doesNotApply,
                    Qcst_EXTP0100_t *crgData,
                    EpData *epData) {

  unsigned int maxWaitTime;

  /* Start the application if this node is the primary                     */
  if (role == QcstPrimaryNodeRole) {

/*-----------------------------------------------------------------------*/
    /*                                                                     */
    /* Determine if all CRGs that this application CRG is dependent upon   */
    /* are ready.  If the check fails, return from the Start action code.  */
    /* Cluster Resource Services will change the state of the CRG to       */
    /* Inactive.                                                           */
    /*                                                                     */

/*-----------------------------------------------------------------------*/
    if (crgData->Cluster_Resource_Group_Status ==
QcstCrgStartCrgPending)
      maxWaitTime = MaxStartCrgWaitSeconds;
    else
      maxWaitTime = MaxWaitSeconds;
    if (QcstSuccessful != checkDependCrgDataArea(maxWaitTime))
      return QcstSuccessful;



/*-----------------------------------------------------------------------*/
    /*                                                                     */
    /* Just before starting the application, update the data area to       */
    /* indicate the application is running.                                */
    /*                                                                     */

/*-----------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Running);


/*-----------------------------------------------------------------------*/
    /*                                                                     */
    /* Add logic to call application here.  It is expected that control    */
    /* will not return until something causes the application to end: a    */
```

```
    /* normal return from the exit program, the job is canceled, or an    */
    /* unhandled exception occurs.  See the cancelHandler() function for   */
    /* some common ways this job could be canceled.                        */
    /*                                                                     */

/*---------------------------------------------------------------------*/



/*---------------------------------------------------------------------*/
    /*                                                                     */
    /* After the application has ended normally, update the data area to   */
    /* indicate the application is no longer running.                      */
    /*                                                                     */

/*---------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Ended);
  }
  else

/*---------------------------------------------------------------------*/
    /*                                                                     */
    /* On backup or replicate nodes, mark the status of the application in */
    /* the data area as not running.                                       */
    /*                                                                     */

/*---------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Ended);


  return QcstSuccessful;
}  /* end startCrg()
       */


/***********************************************************************/
/*                                                                     */
/* Action code = QcstCrgAcRestart                                      */
/*                                                                     */
/* The previous call of the exit program failed and set the return   */
/* code to QcstFailWithRestart or it failed due to an exception and the    */
/* exception was allowed to percolate up the call stack.  In either  */
/* case, the maximum number of times for restarting the exit program has   */
/* not been reached yet.                                               */
/*                                                                     */
/* This action code is passed only to application CRG exit programs which  */
/* had been called with the Start action code.                         */
/*                                                                     */
/***********************************************************************/
static int restartCrg(int role,
                      int doesNotApply,
                      Qcst_EXTP0100_t *crgData,
                      EpData *epData) {


/*---------------------------------------------------------------------*/
  /*                                                                     */
  /* Perform any unique logic that may be necessary when restarting the    */
  /* application after a failure and then call the startCrg() function to  */
  /* do the start functions.                                             */
  /*                                                                     */

/*---------------------------------------------------------------------*/


  return startCrg(role, doesNotApply, crgData, epData);
```

```
}  /* end restartCrg()                                                    */


/*********************************************************************/
/*                                                                   */
/* Action code = QcstCrgAcEnd                                        */
/*                                                                   */
/* The end action code is used for one of the following reasons:     */
/*   - The QcstEndClusterResourceGroup API was called.               */
/*   - The cluster has become partitioned and this node is in the secondary*/
/*     partition.  The End action code is used regardless of whether the   */
/*     CRG was active or inactive.  Action code dependent data of    */
/*     QcstPartitionFailure will also be passed.                     */
/*   - The application ended.   Action code dependent data of        */
/*     QcstResourceEnd will also be passed.  All nodes in the recovery     */
/*     domain will see the same action code (including the primary).       */
/*   - The CRG job has been canceled.  The exit program on this node will */
/*     be called with the End action code.  QcstMemberFailure will be      */
/*     passed as action code dependent data.                         */
/*                                                                   */
/*                                                                   */
/*                                                                   */
/* Things to consider:                                               */
/*   - If the CRG is active, the job running the application is canceled  */
/*     and the IP takeover address is ended AFTER the exit program is      */
/*     called.                                                       */
/*   - If subsystems or server jobs were started as a result of the        */
/*     QcstCrgAcStart action code, end them here or consolidate all logic  */
/*     to end the application in the cancelHandler() since it will be      */
/*     invoked for all Cluster Resource Services APIs which must end the   */
/*     application on the current primary.                           */
/*                                                                   */
/*********************************************************************/
static int endCrg(int role,
                  int priorRole,
                  Qcst_EXTP0100_t *crgData,
                  EpData *epData) {


/*-------------------------------------------------------------------*/
  /*                                                                 */
  /* End the application if it is running on this node.              */
  /*                                                                 */

/*-------------------------------------------------------------------*/
  endApplication(QcstCrgAcRemoveNode, role, priorRole, crgData,
epData);

  return QcstSuccessful;
}  /* end endCrg()                                                    */


/*********************************************************************/
/*                                                                   */
/* Action code = QcstCrgAcVerificationPhase                          */
/*                                                                   */
/* The verification phase action code is used to allow the exit program to */
/* do some verification before proceeding with the requested function      */
/* identified by the action code depended data. If the exit program        */
/* determines that the requested function cannot proceed it should return  */
/* QcstFailWithOutRestart.                                           */
/*                                                                   */
/*                                                                   */
/* NOTE: The exit program will NOT be called with Undo action code.        */
/*                                                                   */
/*********************************************************************/
static int verifyPhase(int role,
```

```
                      int doesNotApply,
                      Qcst_EXTP0100_t *crgData,
                      EpData *epData) {


/*--------------------------------------------------------------------------*/
  /*                                                                        */
  /* Do verification                                                        */
  /*                                                                        */

/*--------------------------------------------------------------------------*/
  if (crgData->Action_Code_Dependent_Data == QcstDltCrg) {
        /* do verification  */
        /* if ( fail ) */
          /* return QcstFailWithOutRestart */
  }

  return QcstSuccessful;
}  /* end verifyPhase()                                                     */


/****************************************************************************/
/*                                                                          */
/* Action code = QcstCrgAcDelete                                            */
/*                                                                          */
/* The QcstDeleteClusterResourceGroup or QcstDeleteCluster API was called. */
/* A cluster resource group is being deleted while Cluster Resource         */
/* Services is active.                                                      */
/* If the QcstDeleteCluster API was used, action code dependent data of     */
/* QcstDltCluster is passed.                                                */
/* If the QcstDeleteCluster API was used and the CRG is active, the exit    */
/* program job which is still active for the Start action code is canceled*/
/* after the Delete action code is processed.                               */
/*                                                                          */
/* Things to consider:                                                      */
/*   - Delete application programs and objects from nodes where they are    */
/*     no longer needed such as backup nodes.  Care needs to be exercised   */
/*     when deleting application objects just because a CRG is being        */
/*     deleted since a particular scenario may want to leave the            */
/*     application objects on all nodes.                                    */
/*                                                                          */
/****************************************************************************/
static int deleteCrg(int role,
                     int doesNotApply,
                     Qcst_EXTP0100_t *crgData,
                     EpData *epData) {

  return QcstSuccessful;
}  /* end deleteCrg()                                                       */
/****************************************************************************/
/*                                                                          */
/* Action code = QcstCrgAcReJoin                                            */
/*                                                                          */
/* One of three things is occurring-                                        */
/*  1. The problem which caused the cluster to become partitioned has been */
/*     corrected and the 2 partitions are merging back together to become   */
/*     a single cluster.  Action code dependent data of QcstMerge will be   */
/*     passed.                                                              */
/*  2. A node which either previously failed or which was ended has had     */
/*     cluster resource services started again and the node is joining the */
/*     cluster.  Action code dependent data of QcstJoin will be passed.     */
/*  3. The CRG job on a particular node which may have been canceled or    */
/*     ended has been restarted.  Action code dependent data of QcstJoin    */
/*     will be passed.                                                      */
/*                                                                          */
/* Things to consider:                                                      */
/*   - If the application replicates application state information to other*/
```

```
/*     nodes when the application is running, this state information will  */
/*     need to be resynchronized with the joining nodes if the CRG is      */
/*     active.                                                             */
/*   - Check for missing application objects on the joining nodes.         */
/*   - Ensure the required data CRGs are on the joining nodes.             */
/*   - If the application CRG is active, ensure the required data CRGs are */
/*     active.                                                             */
/*                                                                         */
/***************************************************************************/
static int memberIsJoining(int role,
                           int priorRole,
                           Qcst_EXTP0100_t *crgData,
                           EpData *epData) {


/*-------------------------------------------------------------------------*/
  /*                                                                       */
  /* Ensure the data area status on this node starts out indicating      */
  /* the application is not running if this node is not the primary.     */
  /*                                                                       */

/*-------------------------------------------------------------------------*/
  if (role != QcstPrimaryNodeRole) {
    setApplCrgDataArea(Appl_Ended);
  }


/*-------------------------------------------------------------------------*/
  /*                                                                       */
  /* If a single node is rejoining the cluster, you may do a certain set of*/
  /* actions.  Whereas if the nodes in a cluster which became partitioned  */
  /* are merging back together, you may have a different set of actions.   */
  /*                                                                       */

/*-------------------------------------------------------------------------*/
  if (crgData->Action_Code_Dependent_Data == QcstJoin) {
    /* Do actions for a node joining.                                      */
  }
    else {
    /* Do actions for partitions merging.                                  */
  }

  return QcstSuccessful;
}  /* end memberIsJoining()                                                */


/***************************************************************************/
/*                                                                         */
/* Action code = QcstCrgAcFailover                                         */
/*                                                                         */
/* Cluster resource services on a particular node(s) has failed or ended   */
/* for this cluster resource group.  The Failover action code is passed    */
/* regardless of whether the CRG is active or inactive.  Failover can      */
/* happen for a number of reasons:                                         */
/*                                                                         */
/*   - an operator canceled the CRG job on a node.  Action code dependent */
/*     data of QcstMemberFailure will be passed.                           */
/*   - cluster resource services was ended on the node (for example, the   */
/*     QSYSWRK subsystem was ended with CRS still active).  Action code    */
/*     dependent data of QcstNodeFailure will be passed.                   */
/*   - the application for an application CRG has failed on the primary    */
/*     node and could not be restarted there.  The CRG is Active.          */
/*     Action code dependent data of QcstApplFailure will be passed.       */
/*   - the node failed (such as a power failure).  Action code dependent */
/*     data of QcstNodeFailure will be passed.                             */
/*   - The cluster has become partitioned due to some communication failure*/
/*     such as a communication line or LAN failure.  The Failover action   */
```

```
/*     code is passed to recovery domain nodes in the majority partition.  */
/*     Nodes in the minority partition see the End action code.  Action    */
/*     code dependent data of QcstPartitionFailure will be passed.         */
/*   - A node in the CRG's recovery domain is being ended with the         */
/*     QcstEndClusterNode API.  The node being ended will see the End Node */
/*     action code.  All other nodes in the recovery domain will see the   */
/*     Failover action code.  Action code dependent data of QcstEndNode    */
/*     will be passed for the Failover action code.                        */
/*   - An active recovery domain node for an active CRG is being removed   */
/*     from the cluster with the QcstRemoveClusterNodeEntry API.   Action  */
/*     code dependent data of QcstRemoveNode will be passed.  If an        */
/*     inactive node is removed for an active CRG, or if the CRG is        */
/*     inactive, an action code of Remove Node is passed.                  */
/*                                                                         */
/* The exit program is called regardless of whether or not the CRG is      */
/* active.  The exit program may have nothing to do if the CRG is not      */
/* active.                                                                 */
/*                                                                         */
/* If the CRG is active and the leaving member was the primary node,       */
/* perform the functions necessary for failover to a new primary.          */
/*                                                                         */
/* The Action_Code_Dependent_Data field can be used to determine if:       */
/*  - the failure was due to a problem that caused the cluster to become   */
/*     partitioned (all CRGs which had the partitioned nodes in the        */
/*     recovery domain are affected)                                       */
/*  - a node failed or had cluster resource services ended on the node (all*/
/*    CRGs which had the failed/ended node in the recovery  domain are     */
/*    affected)                                                            */
/*  - only a single CRG was affected (for example a single CRG job was     */
/*    canceled on a node or a single application failed)                 */
/*                                                                         */
/*                                                                         */
/* Things to consider:                                                     */
/*   - Prepare the new primary node so the application can be started.     */
/*   - The application should NOT be started at this time.  The exit       */
/*     program will be called again with the QcstCrgAcStart action code if */
/*     the CRG was active when the failure occurred.                       */
/*   - If the application CRG is active, ensure the required data CRGs are */
/* active.                                                                 */
/*                                                                         */
/*************************************************************************/
static int memberIsLeaving(int role,
                           int priorRole,
                           Qcst_EXTP0100_t *crgData,
                           EpData *epData) {


/*-----------------------------------------------------------------------*/
  /*                                                                     */
  /* If the CRG is active, perform failover.  Otherwise, nothing to do.  */
  /*                                                                     */

/*-----------------------------------------------------------------------*/
  if (crgData->Original_Cluster_Res_Grp_Stat == QcstCrgActive) {


/*-----------------------------------------------------------------------*/
    /*                                                                   */
    /* The CRG is active.  Determine if my role has changed and I am now */
    /* the new primary.                                                  */
    /*                                                                   */

/*-----------------------------------------------------------------------*/

    if (priorRole != role && role == QcstPrimaryNodeRole) {
```

```
/*--------------------------------------------------------------------*/
      /*                                                                */
      /* I was not the primary but am now.  Do failover actions but don't  */
      /* start the application at this time because this exit program will */
      /* be called again with the Start action code.                    */
      /*                                                                */

/*--------------------------------------------------------------------*/


/*--------------------------------------------------------------------*/
      /*                                                                */
      /* Ensure the data area status on this node starts out indicating    */
      /* the application is not running.                                */
      /*                                                                */

/*--------------------------------------------------------------------*/
      setApplCrgDataArea(Appl_Ended);


/*--------------------------------------------------------------------*/
      /*                                                                */
      /* If the application has no actions to do on the Start action code  */
      /* and will become active as soon as the takeover IP address is      */
      /* activated, then this code should be uncommented.  This code will  */
      /* determine if all CRGs that this application CRG is dependent upon */
      /* are ready.  If this check fails, return failure from the action   */
      /* code.                                                          */
      /*                                                                */

/*--------------------------------------------------------------------*/
/*      if (QcstSuccessful != checkDependCrgDataArea(MaxWaitSeconds))  */
/*         return QcstFailWithOutRestart;                              */

    }
  }

  return QcstSuccessful;
}  /* end memberIsLeaving()                                           */


/************************************************************************/
/*                                                                      */
/* Action code = QcstCrgAcSwitchover                                    */
/*                                                                      */
/* The QcstInitiateSwitchOver API was called.  The first backup node in    */
/* the cluster resource group's recovery domain is taking over as the      */
/* primary node and the current primary node is being made the last backup.*/
/*                                                                      */
/* Things to consider:                                                  */
/*   - Prepare the new primary node so the application can be started.     */
/*   - The application should NOT be started at this time.  The exit       */
/*     program will be called again with the QcstCrgAcStart action code.   */
/*   - The job running the application is canceled and the IP takeover    */
/*     address is ended prior to the exit program being called on the      */
/*     current primary.                                                 */
/*   - Ensure required data or device CRGs have switched over and are      */
/* active.                                                              */
/*                                                                      */
/************************************************************************/
static int switchPrimary(int role,
                         int priorRole,
                         Qcst_EXTP0100_t *crgData,
                         EpData *epData) {


/*--------------------------------------------------------------------*/
```

```
  /*                                                                  */
  /* See if I am the old primary.                                     */
  /*                                                                  */

/*------------------------------------------------------------------*/
  if (priorRole == QcstPrimaryNodeRole) {

/*------------------------------------------------------------------*/
    /*                                                                */
    /* Do what ever needs to be done to cleanup the old primary before the */
    /* switch.  Remember that that job which was running the exit program  */
    /* which started the application was canceled already.              */
    /*                                                                */
    /* One example may be to clean up any processes holding locks on the   */
    /* database.  This may have been done by the application cancel       */
    /* handler if one was invoked.                                      */

/*------------------------------------------------------------------*/
  }


/*------------------------------------------------------------------*/
  /*                                                                  */
  /* I'm not the old primary.  See if I'm the new primary.            */
  /*                                                                  */

/*------------------------------------------------------------------*/
  else if (role == QcstPrimaryNodeRole) {

/*------------------------------------------------------------------*/
    /*                                                                */
    /* Do what ever needs to be done on the new primary before the      */
    /* application is started with the QcstCrgAcStart action code.      */
    /*                                                                */

/*------------------------------------------------------------------*/


/*------------------------------------------------------------------*/
    /*                                                                */
    /* Ensure the data area status on this nodes starts out indicating  */
    /* the application is not running.                                  */
    /*                                                                */

/*------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Ended);


/*------------------------------------------------------------------*/
    /*                                                                */
    /* If the application has no actions to do on the Start action code */
    /* and will become active as soon as the takeover IP address is     */
    /* activated, then this code should be uncommented.  This code will */
    /* determine if all CRGs that this application CRG is dependent upon */
    /* are ready.  If this check fails, return failure from the action  */
    /* code.                                                            */
    /*                                                                */

/*------------------------------------------------------------------*/
/*      if (QcstSuccessful != checkDependCrgDataArea(MaxWaitSeconds))   */
/*         return QcstFailWithOutRestart;                             */

  }
  else {

/*------------------------------------------------------------------*/
    /*                                                                */
```

```
    /* This node is one of the other backup nodes or it is a replicate     */
    /* node.  If there is anything those nodes must do, do it here.  If */
    /* not, remove this else block.                                        */
    /*                                                                     */

/*---------------------------------------------------------------------*/


/*---------------------------------------------------------------------*/
    /*                                                                     */
    /* Ensure the data area status on this nodes starts out indicating     */
    /* the application is not running.                                     */
    /*                                                                     */

/*---------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Ended);
  }

  return QcstSuccessful;
}  /* end switchPrimary()                                              */


/*************************************************************************/
/*                                                                       */
/* Action code = QcstCrgAcAddNode                                        */
/*                                                                       */
/* The QcstAddNodeToRcvyDomain API was called.  A new node is being added  */
/* to the recovery domain of a cluster resource group.                   */
/*                                                                       */
/* Things to consider:                                                   */
/*   - A new node is being added to the recovery domain.  See the        */
/*     considerations in the createCrg() function.                       */
/*   - If this CRG is enabled to use the QcstDistributeInformation API,  */
/*     the user queue needed by that API could be created at this time.  */
/*                                                                       */
/*************************************************************************/
static int addNode(int role,
                   int priorRole,
                   Qcst_EXTP0100_t *crgData,
                   EpData *epData) {


/*---------------------------------------------------------------------*/

  /*                                                                     */
  /* Determine if I am the node being added.                             */
  /*                                                                     */

/*---------------------------------------------------------------------*/

  if (0 == memcmp(&crgData->This_Nodes_ID,
                  &crgData->Changing_Node_ID,
                  sizeof(Qcst_Node_Id_t)))
  {

/*---------------------------------------------------------------------*/

    /*                                                                     */
    /* Set the status of the data area on this new node.                   */
    /*                                                                     */

/*---------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Ended);


/*---------------------------------------------------------------------*/
```

```
    /*                                                                  */
    /* Create the queue needed by the Distribute Information API.       */
    /*                                                                  */

/*------------------------------------------------------------------*/

    if (0 == memcmp(&crgData->DI_Queue_Name,
                    Nulls,
                    sizeof(crgData->DI_Queue_Name)))
    {
    }
  }

  return QcstSuccessful;
}  /* end addNode()
       */


/**********************************************************************/
/*                                                                    */
/* Action code = QcstCrgAcRemoveNode                                  */
/*                                                                    */
/* The QcstRemoveNodeFromRcvyDomain or the QcstRemoveClusterNodeEntry */
/* API was called.  A node is being removed from the recovery domain of */
/* a cluster resource group or it is being removed entirely from the  */
/* cluster.                                                           */
/*                                                                    */
/* This action code is seen by:                                       */
/*  For the QcstRemoveClusterNodeEntry API:                           */
/*    - If the removed node is active and the CRG is Inactive, all nodes in*/
/*      the recovery domain including the node being removed see this */
/*      action code.  The nodes NOT being removed see action code dependent*/
/*      data of QcstNodeFailure.                                      */
/*    - If the removed node is active and the CRG is Active, the node being*/
/*      removed sees the Remove Node action code.  All other nodes in the */
/*      recovery domain see an action code of Failover and action code */
/*      dependent data of QcstNodeFailure.                            */
/*    - If the node being removed is not active in the cluster, all nodes */
/*      in the recovery domain will see this action code.             */
/*  For the QcstRemoveNodeFromRcvyDomain API:                         */
/*    - All nodes see the Remove Node action code regardless of whether or */
/*      not the CRG is Active.  Action code dependent data of         */
/*      QcstRmvRcvyDmnNode will also be passed.                       */
/*                                                                    */
/* Things to consider:                                                */
/*   - You may want to cleanup the removed node by deleting objects no */
/*     longer needed there.                                           */
/*   - The job running the application is canceled and the IP takeover */
/*     address is ended after the exit program is called if this is the */
/*     primary node and the CRG is active.                            */
/*   - If subsystems or server jobs were started as a result of the   */
/*     QcstCrgAcStart action code, end them here or consolidate all logic */
/*     to end the application in the cancelHandler() since it will be */
/*     invoked for all Cluster Resource Services APIs which must end the */
/*     application on the current primary.                            */
/*                                                                    */
/**********************************************************************/
static int rmvNode(int role,
                   int priorRole,
                   Qcst_EXTP0100_t *crgData,
                   EpData *epData) {


/*------------------------------------------------------------------*/

  /*                                                                  */
  /* Determine if I am the node being removed.                        */
```

```
  /*                                                                  */

/*------------------------------------------------------------------*/

  if (0 == memcmp(&crgData->This_Nodes_ID,
                  &crgData->Changing_Node_ID,
                  sizeof(Qcst_Node_Id_t)))
  {

/*------------------------------------------------------------------*/
      /*                                                            */
      /* End the application if it is running on this node.         */
      /*                                                            */

/*------------------------------------------------------------------*/
      endApplication(QcstCrgAcRemoveNode, role, priorRole, crgData,
epData);

  }
  return QcstSuccessful;
}  /* end rmvNode                                                  */


/********************************************************************/
/*                                                                  */
/* Action code = QcstCrgAcChange                                    */
/*                                                                  */
/* The QcstChangeClusterResourceGroup API was called.  Some attribute     */
/* or information stored in the cluster resource group object is being    */
/* changed.  Note that not all changes to the CRG object cause the exit   */
/* program to be called.  As of V5R1M0, only these changes will cause the */
/* exit program to be called-                                       */
/*   - the current recovery domain is being changed                 */
/*   - the preferred recovery domain is being changed               */
/*                                                                  */
/* If any of the above changes are being made but additionally the exit   */
/* program is being changed to *NONE, the exit program is not called.     */
/*                                                                  */
/* Things to consider:                                              */
/*   - None unless changing the recovery domain affects information or    */
/*     processes for this cluster resource group.  Note that the primary  */
/*     node cannot be changed with the QcstChangeClusterResourceGroup API */
/*     if the CRG is active.                                        */
/*                                                                  */
/********************************************************************/
static int chgCrg(int role,
                  int priorRole,
                  Qcst_EXTP0100_t *crgData,
                  EpData *epData) {

  return QcstSuccessful;
}  /* end chgCrg()                                                 */


/********************************************************************/
/*                                                                  */
/* Action code = QcstCrgAcDeleteCommand                             */
/*                                                                  */
/* The Delete Cluster Resource Group (DLTCRG) CL command has been called  */
/* to delete a cluster resource group object, the QcstDeleteCluster API   */
/* has been called, or the QcstRemoveClusterNodeEntry API has been called. */
/* In each case, cluster resource services is not active on the cluster   */
/* node where the command or API was called.  Thus, this function is not  */
/* distributed cluster wide but occurs only on the node where the CL      */
/* command or API was called.                                       */
/*                                                                  */
/* If the QcstDeleteCluster API was used, action code dependent data of   */
```

```
/* QcstDltCluster is passed.                                                 */
/*                                                                           */
/* See the considerations in the deleteCrg() function                        */
/*                                                                           */
/*****************************************************************************/
static int deleteCrgWithCmd(int role,
                            int doesNotApply,
                            Qcst_EXTP0100_t *crgData,
                            EpData *epData) {

  return QcstSuccessful;
}  /* end deleteCrgWithCmd()                                                 */


/*****************************************************************************/
/*                                                                           */
/* Action code = QcstCrgEndNode                                              */
/*                                                                           */
/* The QcstEndClusterNode API was called or a CRG job was canceled.       */
/*                                                                           */
/* The QcstCrgEndNode action code is passed to the exit program only on the*/
/* node being ended or where the CRG job was canceled.  On the node where */
/* a Cluster Resource Services job is canceled, action code dependent data*/
/* of QcstMemberFailure will be passed.                                      */
/* When Cluster Resource Services ends on this node or the CRG job ends, it*/
/* will cause all other nodes in the cluster to go through failover        */
/* processing.  The action code passed to all other nodes will be          */
/* QcstCrgAcFailover.   Those nodes will see action code dependent data of */
/* QcstMemberFailure if a CRG job is canceled or QcstNodeFailure if the   */
/* node is ended.                                                            */
/*                                                                           */
/* Things to consider:                                                       */
/*   - The job running the application is canceled and the IP takeover    */
/*     address is ended after the exit program is called if this is the    */
/*     primary node and the CRG is active.                                   */
/*   - If subsystems or server jobs were started as a result of the        */
/*     QcstCrgAcStart action code, end them here.                            */
/*                                                                           */
/*****************************************************************************/
static int endNode(int role,
                   int priorRole,
                   Qcst_EXTP0100_t *crgData,
                   EpData *epData) {


/*---------------------------------------------------------------------------*/
  /*                                                                         */
  /* End the application if it is running on this node.                      */
  /*                                                                         */

/*---------------------------------------------------------------------------*/
  endApplication(QcstCrgEndNode, role, priorRole, crgData, epData);

  return QcstSuccessful;
}  /* end endNode()                                                          */


/*****************************************************************************/
/*                                                                           */
/* Action code = QcstCrgAcChgNodeStatus                                      */
/*                                                                           */
/* The QcstChangeClusterNodeEntry API was called.  The status of a node    */
/* is being changed to failed.  This API is used to inform cluster resource*/
/* services that the node did not partition but really failed.               */
/*                                                                           */
/* Things to consider:                                                       */
/*   - The exit program was called previously with an action code of       */
```

```
/*     QcstCrgAcEnd if the CRG was active or an action code of            */
/*     QcstCrgAcFailover if the CRG was inactive because cluster resource  */
/*     services thought the cluster had become partitioned.  The user is   */
/*     now telling cluster resource services that the node really failed   */
/*     instead of partitioned.  The exit program has something to do only  */
/*     if it performed some action previously that needs to be changed now */
/*     that node failure can be confirmed.                                 */
/*                                                                         */
/*************************************************************************/
static int chgNodeStatus(int role,
                         int priorRole,
                         Qcst_EXTP0100_t *crgData,
                         EpData *epData) {

  return QcstSuccessful;
}  /* end chgNodeStatus()                                                 */


/*************************************************************************/
/*                                                                       */
/* Action code = QcstCrgAcCancelFailover                                 */
/*                                                                       */
/* Cluster resource services on the primary node has failed or ended     */
/* for this cluster resource group.  A message was sent to the failover  */
/* message queue specified for the CRG, and the result of that message   */
/* was to cancel the failover.  This will change the status of the CRG to */
/* inactive and leave the primary node as primary.                       */
/*                                                                       */
/* Things to consider:                                                   */
/*   - The primary node is no longer participating in cluster activities. */
/*     The problem which caused the primary node to fail should be fixed */
/*     so that the CRG may be started again.                             */
/*                                                                       */
/*************************************************************************/
static int cancelFailover(int role,
                          int priorRole,
                          Qcst_EXTP0100_t *crgData,
                          EpData *epData) {

  return QcstSuccessful;
}  /* end cancelFailover()                                               */


/*************************************************************************/
/*                                                                       */
/* Action code = exit program does not know it yet                       */
/*                                                                       */
/* A new action code has been passed to this exit program.  This can occur */
/* after a new i5/OS release has been installed and some new cluster API  */
/* was called or some new cluster event occurred.  The logic in this exit */
/* program has not yet been updated to understand the new action code.   */
/*                                                                       */
/* Two different strategies could be used for the new action code.  The  */
/* correct strategy is dependent upon the kinds of things this particular */
/* exit program does for the application.                                */
/*                                                                       */
/* One strategy is to not do anything and return a successful return code. */
/* This allows the new cluster API or event to run to completion.  It    */
/* allows the function to be performed even though this exit program     */
/* did not understand the new action code.  The risk, though, is that the */
/* exit program should have done something and it did not.  At a minimum, */
/* you may want to log some kind of error message about what happened so  */
/* that programming can investigate and get the exit program updated.    */
/*                                                                       */
/* The opposite strategy is to return an error return code such as       */
/* QcstFailWithRestart.  Of course doing this means that the new cluster */
/* API or event cannot be used until the exit program is updated for the */
```

```
/* new action code.  Again, logging some kind of error message for        */
/* programming to investigate would be worthwhile.                        */
/*                                                                        */
/* Only the designer of the exit program can really decide which is the   */
/* better course of action.                                               */
/*                                                                        */
/**************************************************************************/
static int newActionCode(int role,
                         int doesNotApply,
                         Qcst_EXTP0100_t *crgData,
                         EpData *epData) {


/*------------------------------------------------------------------------*/
  /*                                                                      */
  /* Add logic to log an error somewhere - operator message queue, job    */
  /* log, application specific error log, etc. so that the exit program   */
  /* gets updated to properly handle the new action code.                 */
  /*                                                                      */
  /* Note that if this is left coded as it is, this is the "don't do      */
  /* anything" strategy described in the prologue above.                  */
  /*                                                                      */

/*------------------------------------------------------------------------*/

  return QcstSuccessful;
}  /* end newActionCode()                                                 */


/**************************************************************************/
/*                                                                        */
/* Action code = QcstCrgAcUndo                                            */
/*                                                                        */
/* Note: The exit program is never called with an undo action code for    */
/* any of these prior action codes:                                       */
/*   QcstCrgAcChgNodeStatus                                               */
/*   QcstCrgAcDelete                                                      */
/*   QcstCrgAcDeleteCommand                                               */
/*   QcstCrgEndNode                                                       */
/*   QstCrgAcRemoveNode (If the node being removed is active in the       */
/*                       cluster and the API is Remove Cluster Node.      */
/*                       The Remove Node From Recovery Domain will call   */
/*                       with Undo and the Remove Cluster Node API will   */
/*                       call with Undo if the node being removed is      */
/*                       inactive.                                        */
/*   QcstCrgAcRestart                                                     */
/*   QcstCrgAcUndo                                                        */
/*                                                                        */
/* APIs that call an exit program do things in 3 steps.                   */
/*   1. Logic which must be done prior to calling the exit program.       */
/*   2. Call the exit program.                                            */
/*   3. Logic which must be done after calling the exit program.          */
/*                                                                        */
/* Any errors that occur during steps 2 or 3 result in the exit program   */
/* being called again with the undo action code.  This gives the exit     */
/* program an opportunity to back out any work performed when it was first */
/* called by the API.  The API will also be backing out any work it       */
/* performed trying to return the state of the cluster and cluster objects */
/* to what it was before the API was called.                              */
/*                                                                        */
/* It is suggested that the following return codes be returned for the    */
/* specified action code as that return code will result in the most      */
/* appropriate action being taken.                                        */
/*                                                                        */
/*   QcstCrgAcInitialize: QcstSuccessful; The CRG is not created.         */
/*   QcstCrgAcStart:      QcstSuccessful; The CRG is not started.         */
/*   QcstCrgAcEnd:        QcstFailWithOutRestart; The CRG is set to Indoubt*/
```

```
/*                                      The cause of the failure needs to*/
/*                                      investigated.                    */
/*   QcstCrgAcReJoin:     QcstFailWithOutRestart; The CRG is set to Indoubt*/
/*                                      The cause of the failure needs to*/
/*                                      investigated.                    */
/*   QcstCrgAcFailover:   QcstFailWithOutRestart; The CRG is set to Indoubt*/
/*                                      The cause of the failure needs to*/
/*                                      investigated.                    */
/*   QcstCrgAcSwitchover: QcstFailWithOutRestart; The CRG is set to Indoubt*/
/*                                      The cause of the failure needs to*/
/*                                      investigated.                    */
/*   QcstCrgAcAddNode:    QcstSuccessful; The node is not added.          */
/*   QcstCrgAcRemoveNode: QcstFailWithOutRestart; The CRG is set to Indoubt*/
/*                                      The cause of the failure needs to*/
/*                                      investigated.                    */
/*   QcstCrgAcChange:     QcstSuccessful; The recovery domain is not      */
/*                                      changed.                         */
/*                                                                       */
/*************************************************************************/
static int undoPriorAction(int role,
                           int priorRole,
                           Qcst_EXTP0100_t *crgData,
                           EpData *epData) {


/*-----------------------------------------------------------------------*/
  /*                                                                     */
  /* The prior action code defines what the exit program was doing when  */
  /* it failed, was canceled, or returned a non successful return code.  */
  /*                                                                     */

/*-----------------------------------------------------------------------*/
  if (crgData->Prior_Action_Code &lt;= MaxAc )
    return (*undoFcn[crgData-&lt;Prior_Action_Code])
                                       (role, priorRole, crgData,
epData);
  else

/*-----------------------------------------------------------------------*/
    /*                                                                   */
    /* IBM has defined a new action code in a new operating system release */
    /* and this exit program has not yet been updated to handle it.  Take a*/
    /* default action for now.                                           */
    /*                                                                   */

/*-----------------------------------------------------------------------*/
    return newActionCode(role, priorRole, crgData, epData);
}  /* end undoPriorAction()                                            */


/*************************************************************************/
/*                                                                       */
/* Action code = QcstCrgAcUndo                                           */
/*                                                                       */
/* Prior action code = QcstCrgAcInitialize                               */
/*                                                                       */
/* Things to consider:                                                   */
/*   The CRG will not be created.  Objects that might have been created  */
/*   on nodes in the recovery domain should be deleted since a subsequent */
/*   create could fail if those objects already exist.                   */
/*                                                                       */
/*************************************************************************/
static int undoCreateCrg(int role,
                         int doesNotApply,
                         Qcst_EXTP0100_t *crgData,
                         EpData *epData) {
```

```
  return QcstSuccessful;
}  /* end undoCreateCrg()                                                  */


/*************************************************************************/
/*                                                                       */
/* Action code = QcstCrgAcUndo                                           */
/*                                                                       */
/* Prior action code = QcstCrgAcStart                                    */
/*                                                                       */
/* Things to consider:                                                   */
/*   Cluster Resource Services failed when it was finishing the Start CRG  */
/*   API after it had already called the exit program with the Start     */
/*   Action code.                                                        */
/*                                                                       */
/*   On the primary node, the exit program job which is running the      */
/*   application will be canceled.  The exit program will then be called  */
/*   with the Undo action code.                                          */
/*                                                                       */
/*   All other nodes in the recovery domain will be called with the Undo  */
/*   action code.                                                        */
/*                                                                       */
/*************************************************************************/
static int undoStartCrg(int role,
                        int doesNotApply,
                        Qcst_EXTP0100_t *crgData,
                        EpData *epData) {

  return QcstSuccessful;
}  /* end undoStartCrg()                                                  */


/*************************************************************************/
/*                                                                       */
/* Action code = QcstCrgAcUndo                                           */
/*                                                                       */
/* Prior action code = QcstCrgAcEnd                                      */
/*                                                                       */
/* Things to consider:                                                   */
/*   The CRG will not be ended.  If the exit program did anything to bring */
/*   down the application it can either restart the application or it can  */
/*   decide to not restart the application.  If the application is not    */
/*   restarted, the return code should be set to QcstFailWithOutRestart so */
/*   the status of the CRG is set to Indoubt.                            */
/*                                                                       */
/*************************************************************************/
static int undoEndCrg(int role,
                      int doesNotApply,
                      Qcst_EXTP0100_t *crgData,
                      EpData *epData) {

  return QcstFailWithOutRestart;
}  /* end undoEndCrg()                                                    */


/*************************************************************************/
/*                                                                       */
/* Action code = QcstCrgAcUndo                                           */
/*                                                                       */
/* Prior action code = QcstCrgAcReJoin                                   */
/*                                                                       */
/* Things to consider:                                                   */
/*   An error occurred which won't allow the member to join this CRG     */
/*   group.  Anything done for the Join action code needs to be looked at  */
/*   to see if something must be undone if this member is not an active  */
/*   member of the CRG group.                                            */
/*                                                                       */
```

```
/*********************************************************************/
static int undoMemberIsJoining(int role,
                               int doesNotApply,
                               Qcst_EXTP0100_t *crgData,
                               EpData *epData) {

  return QcstFailWithOutRestart;
}  /* end undoMemberIsJoining()                                      */


/*********************************************************************/
/*                                                                   */
/* Action code = QcstCrgAcUndo                                       */
/*                                                                   */
/* Prior action code = QcstCrgAcFailover                             */
/*                                                                   */
/* Things to consider:                                               */
/*   This does not mean that the node failure or failing member is being   */
/*   undone.  That failure is irreversible.  What it does mean is that the */
/*   exit program returned an error from the Failover action code or       */
/*   Cluster Resource Services ran into a problem after it called the exit */
/*   program.  If the CRG was active when Failover was attempted, it is    */
/*   not at this point.  End the resilient resource and expect a human to  */
/*   look into the failure.  After the failure is corrected, the CRG will  */
/*   must be started with the Start CRG API.                          */
/*                                                                   */
/*                                                                   */
/*********************************************************************/
static int undoMemberIsLeaving(int role,
                               int doesNotApply,
                               Qcst_EXTP0100_t *crgData,
                               EpData *epData) {

  return QcstFailWithOutRestart;
}  /* end undoMemberIsLeaving()                                      */


/*********************************************************************/
/*                                                                   */
/* Action code = QcstCrgAcUndo                                       */
/*                                                                   */
/* Prior action code = QcstCrgAcSwitchover                           */
/*                                                                   */
/* Things to consider:                                               */
/*   Some error occurred after the point of access was moved from the      */
/*   original primary and before it could be brought up on the new primary.*/
/*   The IP address was ended on the original primary before moving the    */
/*   point of access but is started on the original primary again.  Cluster*/
/*   Resource Services will now attempt to move the point of access back   */
/*   to the original primary.  The application exit program and IP takeover*/
/*   address will be started on the original primary.                      */
/*                                                                   */
/*                                                                   */
/*********************************************************************/
static int undoSwitchPrimary(int role,
                             int doesNotApply,
                             Qcst_EXTP0100_t *crgData,
                             EpData *epData) {

  return QcstFailWithOutRestart;
}  /* end undoSwitchPrimary()                                        */


/*********************************************************************/
/*                                                                   */
/* Action code = QcstCrgAcUndo                                       */
/*                                                                   */
```

```
/* Prior action code = QcstCrgAcAddNode                                   */
/*                                                                        */
/* Things to consider:                                                    */
/*   If objects were created on the new node, they should be removed so   */
/*   that a subsequent Add Node to aRecovery Domain does not fail if it   */
/*   attempts to create objects again.                                    */
/*                                                                        */
/*                                                                        */
/**************************************************************************/
static int undoAddNode(int role,
                       int doesNotApply,
                       Qcst_EXTP0100_t *crgData,
                       EpData *epData) {

  return QcstSuccessful;
}  /* end undoAddNode()                                                   */


/**************************************************************************/
/*                                                                        */
/* Action code = QcstCrgAcUndo                                            */
/*                                                                        */
/* Prior action code = QcstCrgAcRemoveNode                                */
/*                                                                        */
/* Things to consider:                                                    */
/*   The node is still in the recovery domain.  If objects were removed   */
/*   from the node, they should be added back.                            */
/*                                                                        */
/**************************************************************************/
static int undoRmvNode(int role,
                       int doesNotApply,
                       Qcst_EXTP0100_t *crgData,
                       EpData *epData) {

  return QcstFailWithOutRestart;
}  /* end undoRmvNode()                                                   */


/**************************************************************************/
/*                                                                        */
/* Action code = QcstCrgAcUndo                                            */
/*                                                                        */
/* Prior action code = QcstCrgAcChange                                    */
/*                                                                        */
/* Things to consider:                                                    */
/*   Changes to the CRG will be backed out so that the CRG and its        */
/*   recovery domain look just like it did prior to the attempted change. */
/*   Any changes the exit program made should also be backed out.         */
/*                                                                        */
/**************************************************************************/
static int undoChgCrg(int role,
                      int doesNotApply,
                      Qcst_EXTP0100_t *crgData,
                      EpData *epData) {

  return QcstSuccessful;
}  /* end undoChgCrg()                                                    */


/**************************************************************************/
/*                                                                        */
/* Action code = QcstCrgAcUndo                                            */
/*                                                                        */
/* Prior action code = QcstCrgAcCancelFailover                            */
/*                                                                        */
/* Things to consider:                                                    */
/*   This does not mean that the node failure or failing member is being  */
```

```
/*   undone.  That failure is irreversible.  What it does mean is that     */
/*   Cluster Resource Services ran into a problem after it called the exit */
/*   program.  The CRG will be InDoubt regardless of what is returned from */
/*   this exit program call.  Someone will need to manually look into the  */
/*   the failure.  After the failure is corrected, the CRG will must be */
/*   started with the Start CRG API.                                       */
/*                                                                         */
/*                                                                         */
/*************************************************************************/
static int undoCancelFailover(int role,
                              int doesNotApply,
                              Qcst_EXTP0100_t *crgData,
                              EpData *epData) {

  return QcstSuccessful;
}  /* end undoCancelFailover()                                            */


/*************************************************************************/
/*                                                                       */
/* A simple routine to take a null terminated object name and a null     */
/* terminated library name and build a 20 character non-null terminated  */
/* qualified name.                                                       */
/*                                                                       */
/*************************************************************************/
static void bldDataAreaName(char *objName, char* libName, char *qualName) {

  memset(qualName, 0x40, 20);
  memcpy(qualName, objName, strlen(objName));
  qualName += 10;
  memcpy(qualName, libName, strlen(libName));
  return;
}  /* end bldDataAreaName                                                */


/*************************************************************************/
/*                                                                       */
/* The data area is checked to see if all the CRGs that this application */
/* is dependent upon are ready.  If they are not ready, a wait for a     */
/* certain amount of time is performed and the data area is checked again. */
/* This check, wait loop continues until all dependent CRGs become ready or*/
/* until the maximum wait time has been reached.                         */
/* The length of the wait can be changed to some other value if a        */
/* particular situation would be better with shorter or longer wait times. */
/*                                                                       */
/*                                                                       */
/*************************************************************************/
static int checkDependCrgDataArea(unsigned int maxWaitTime) {

  Qus_EC_t errCode = { sizeof(Qus_EC_t), 0 };
  char dataAreaName[20];
  struct {
    Qwc_Rdtaa_Data_Returned_t stuff;
    char ready;
  } data;


/*-----------------------------------------------------------------------*/
  /*                                                                     */
  /* This is an accumulation of the time waited for the dependent CRGs to  */
  /* become ready.                                                       */
  /*                                                                     */

/*-----------------------------------------------------------------------*/
  unsigned int timeWaited = 0;
```

```
/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Build definition of the amount of time to wait.                  */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  _MI_Time   timeToWait;
  int hours     = 0;
  int minutes   = 0;
  int seconds   = WaitSecondsIncrement;
  int hundreths = 0;
  short int options = _WAIT_NORMAL;
  mitime( &timeToWait, hours, minutes, seconds, hundreths );


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Build the qualified name of the data area.                       */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  bldDataAreaName(DependCrgDataArea, ApplLib, dataAreaName);


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Get the data from the data area that indicates whether or not the     */
  /* CRGs are all ready.  This data area is updated by the High       */
  /* Availability Business Partners when it is ok for the application to   */
  /* proceed.                                                         */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  QWCRDTAA(&data,
           sizeof(data),
           dataAreaName,
           offsetof(Qcst_HAAPPO_t,Data_Status)+1,  /* API wants a 1 origin */
           sizeof(data.ready),
           &errCode);


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* If the dependent CRGs are not ready, wait for a bit and check again.  */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  while (data.ready != Data_Available) {


/*--------------------------------------------------------------------*/
    /*                                                                */
    /* If the dependent CRGs have not become ready by the time we have     */
    /* waited our maximum wait time, return an error.  Consider logging    */
    /* some message to describe why the application did not start so that  */
    /* the problem can be looked into.                                  */
    /*                                                                */

/*--------------------------------------------------------------------*/
    if (timeWaited >= maxWaitTime)
      return QcstFailWithOutRestart;


/*--------------------------------------------------------------------*/
    /*                                                                */
    /* Wait to allow the data CRGs to become ready.                     */
    /*                                                                */
```

```
/*--------------------------------------------------------------------*/
    waittime(&timeToWait, options);
    timeWaited += WaitSecondsIncrement;


/*--------------------------------------------------------------------*/
    /*                                                                */
    /* Get information from the data area again to see if the data CRGs are*/
    /* ready.                                                         */
    /*                                                                */

/*--------------------------------------------------------------------*/
    QWCRDTAA(&data,
             sizeof(data),
             dataAreaName,
             offsetof(Qcst_HAAPPO_t,Data_Status)+1,  /* API wants a 1 origin */
             sizeof(data.ready),
             &errCode);
  }

  return QcstSuccessful;
}  /* end checkDependCrgDataArea                                     */


/**********************************************************************/
/*                                                                    */
/* The application CRG data area is updated to indicate that the      */
/* application is running or to indicate it is not running.  This data area*/
/* information is used by the High Availability Business Partners to  */
/* coordinate the switchover activities between CRGs that have dependencies*/
/* on each other.                                                     */
/*                                                                    */
/**********************************************************************/
static void setApplCrgDataArea(char status) {

  char cmd[54];
  char cmdEnd[3] = {0x00, ')', 0x00};


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Set up the CL command string with the data area library name, the data*/
  /* area name, and the character to put into the data area.  Then run the */
  /* CL command.                                                      */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  memcpy(cmd, "CHGDTAARA DTAARA(", strlen("CHGDTAARA DTAARA(")+1);
  strcat(cmd, ApplLib);
  strcat(cmd, "/");
  strcat(cmd, ApplCrgDataArea);
  strcat(cmd, " (425 1)) VALUE(");                          /* @A1C */
  cmdEnd[0] = status;
  strcat(cmd, cmdEnd);

  system(cmd);

  return;
}  /* end setApplCrgDataArea                                        */


/**********************************************************************/
/*                                                                    */
/* This function is called any time the exit program receives an exception */
/* not specifically monitored for by some other exception handler.  Add    */
/* appropriate logic to perform cleanup functions that may be required.    */
```

```
/* A failure return code is then set and control returns to the operating  */
/* system.  The job this exit program is running in will then end.         */
/*                                                                         */
/* When this function gets called, myData->role may still contain the      */
/* UnknownRole value if an exception occurred before this node's role      */
/* value was set.  To be completely correct, the role should be tested     */
/* for UnknownRole before making any decisions based upon the value of     */
/* role.                                                                   */
/*                                                                         */
/*************************************************************************/
static void unexpectedExceptionHandler(_INTRPT_Hndlr_Parms_T
*exData) {


/*-------------------------------------------------------------------------*/
  /*                                                                       */
  /* Get a pointer to the structure containing data we passed to the       */
  /* exception handler.                                                    */
  /*                                                                       */

/*-------------------------------------------------------------------------*/
  HandlerDataT *myData = (HandlerDataT *)exData->Com_Area;


/*-------------------------------------------------------------------------*/
  /*                                                                       */
  /* Perform as much cleanup function as necessary.  Some global state     */
  /* information may must be kept so the exception handler knows what   */
  /* steps were completed before the failure occurred and thus knows what  */
  /* cleanup steps must be performed.  This state information could be     */
  /* kept in the HandlerDataT structure or it could be kept in some other  */
  /* location that this function can address.                              */
  /*                                                                       */

/*-------------------------------------------------------------------------*/


/*-------------------------------------------------------------------------*/
  /*                                                                       */
  /* If this is the primary node and the application was started, end it.  */
  /* The application is ended because the exit program will be called again*/
  /* with the Restart action code and we want the restartCrg() function to */
  /* always work the same way.  In addition, ending the application may    */
  /* clear up the condition that caused the exception that got us here.    */
  /* If possible, warn users and have them stop using the application so   */
  /* things are done in an orderly manner.                                 */
  /*                                                                       */

/*-------------------------------------------------------------------------*/
  endApplication(myData->actionCode,
                 myData->role,
                 myData->priorRole,
                 myData->crgData,
                 myData->epData);


/*-------------------------------------------------------------------------*/
  /*                                                                       */
  /* Set the exit program return code.                                     */
  /*                                                                       */

/*-------------------------------------------------------------------------*/
  *myData->retCode = QcstFailWithRestart;


/*-------------------------------------------------------------------------*/
  /*                                                                       */
```

```
  /* Let the exception percolate up the call stack.                 */
  /*                                                                */

/*----------------------------------------------------------------------*/
  return;
}  /* end unexpectedExceptionHandler                                 */


/************************************************************************/
/*                                                                      */
/* This function is called any time the job this exit program is running in*/
/* is canceled.  The job could be canceled due to any of the following   */
/* (the list is not intended to be all inclusive)-                      */
/*   - an API cancels an active application CRG.  The End CRG, Initiate    */
/*     Switchover, End Cluster Node, Remove Cluster Node or Delete Cluster */
/*     API cancels the job which was submitted when the exit program was   */
/*     called with a Start action code.                                 */
/*   - operator cancels the job from some operating system display such as */
/*     Work with Active Jobs                                            */
/*   - the subsystem this job is running in is ended                    */
/*   - all subsystems are ended                                         */
/*   - the system is powered down                                       */
/*   - an operating system machine check occurred                       */
/*                                                                      */
/* When this function gets called, myData->role may still contain the      */
/* UnknownRole value if cancelling occurred before this node's role        */
/* value was set.  To be completely correct, the role should be tested     */
/* for UnknownRole before making any decisions based upon the value of     */
/* role.                                                                */
/*                                                                      */
/************************************************************************/
static void cancelHandler(_CNL_Hndlr_Parms_T *cnlData) {


/*----------------------------------------------------------------------*/
  /*                                                                */
  /* Get a pointer to the structure containing data we passed to the       */
  /* cancel handler.                                                */
  /*                                                                */

/*----------------------------------------------------------------------*/
  HandlerDataT *myData = (HandlerDataT *)cnlData->Com_Area;


/*----------------------------------------------------------------------*/
  /*                                                                */
  /* Perform as much cleanup function as necessary.  Some global state     */
  /* information may must be kept so the cancel handler knows what      */
  /* steps were completed before the job was canceled and thus knows if   */
  /* the function had really completed successfully or was only partially  */
  /* complete and thus needs some cleanup to be done.  This state          */
  /* information could be kept in the HandlerDataT structure or it could   */
  /* be kept in some other location that this function can address.        */
  /*                                                                */

/*----------------------------------------------------------------------*/


/*----------------------------------------------------------------------*/

  /*                                                                */
  /* This job is being canceled.  If I was running the application as a   */
  /* result of the Start or Restart action codes, end the application now. */
  /* This job is being canceled because a Switch Over or some other       */
  /* Cluster Resource Services API was used which affects the primary node */
  /* or someone did a cancel job with a CL command, from a system display, */
  /* etc.                                                           */
```

```
/*--------------------------------------------------------------------*/
  endApplication(myData->actionCode,
                 myData->role,
                 myData->priorRole,
                 myData->crgData,
                 myData->epData);


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Set the exit program return code.                                */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  *myData->retCode = QcstSuccessful;


/*--------------------------------------------------------------------*/
  /*                                                                  */
  /* Return to the operating system for final ending of the job.      */
  /*                                                                  */

/*--------------------------------------------------------------------*/
  return;
}  /* end cancelHandler                                             */


/**********************************************************************/
/*                                                                    */
/* A common routine used to end the application by various action code */
/* functions, the exception handler, and the cancel handler.          */
/*                                                                    */
/**********************************************************************/
static void endApplication(unsigned int actionCode,
                           int role,
                           int priorRole,
                           Qcst_EXTP0100_t *crgData,
                           EpData *epData) {

  if (  role == QcstPrimaryNodeRole
      &&
        crgData->Original_Cluster_Res_Grp_Stat == QcstCrgActive)
{

/*--------------------------------------------------------------------*/
    /*                                                                */
    /* Add logic to end the application here.  You may need to add logic */
    /* to determine if the application is still running because this  */
    /* function could be called once for an action code and again from */
    /* the cancel handler (End CRG is an example).                    */
    /*                                                                */

/*--------------------------------------------------------------------*/




/*--------------------------------------------------------------------*/
    /*                                                                */
    /* After the application has ended, update the data area to indicate */
    /* the application is no longer running.                          */
    /*                                                                */

/*--------------------------------------------------------------------*/
    setApplCrgDataArea(Appl_Ended);
```

```
  }

  return;
}  /* end endApplication                                                    */


/*************************************************************************/
/*                                                                       */
/* Print out the data passed to this program.                            */
/*                                                                       */
/*************************************************************************/
static void printParms(int actionCode,
                       int role,
                       int priorRole,
                       Qcst_EXTP0100_t *crgData,
                       EpData *epData) {

  unsigned int i;
  char *str;

  /* Print the action code.                                              */
  printf("%s", "Action_Code = ");
  printActionCode(actionCode);

  /* Print the action code dependent data.                               */
  printf("%s", "    Action_Code_Dependent_Data = ");
  switch (crgData->Action_Code_Dependent_Data) {
    case QcstNoDependentData:  str = "QcstNoDependentData";
                               break;
    case QcstMerge:            str = "QcstMerge";
                               break;
    case QcstJoin:             str = "QcstJoin";
                               break;
    case QcstPartitionFailure: str = "QcstPartitionFailure";
                               break;
    case QcstNodeFailure:      str = "QcstNodeFailure";
                               break;
    case QcstMemberFailure:    str = "QcstMemberFailure";
                               break;
    case QcstEndNode:          str = "QcstEndNode";
                               break;
    case QcstRemoveNode:       str = "QcstRemoveNode";
                               break;
    case QcstApplFailure:      str = "QcstApplFailure";
                               break;
    case QcstResourceEnd:      str = "QcstResourceEnd";
                               break;
    case QcstDltCluster:       str = "QcstDltCluster";
                               break;
    case QcstRmvRcvyDmnNode:   str = "QcstRmvRcvyDmnNode";
                               break;
    case QcstDltCrg:           str = "QcstDltCrg";
                               break;
    default: str = "unknown action code dependent data";
  }
  printf("%s ¥n", str);


  /* Print the prior action code.                                        */
  printf("%s", "  Prior_Action_Code = ");
  if (crgData->Prior_Action_Code)
    printActionCode(crgData->Prior_Action_Code);
  printf("¥n");

  /* Print the cluster name.                                             */
    printStr("  Cluster_Name = ",
           crgData->Cluster_Name, sizeof(Qcst_Cluster_Name_t));
```

```
  /* Print the CRG name.                                              */
  printStr("  Cluster_Resource_Group_Name = ",
           crgData->Cluster_Resource_Group_Name,
sizeof(Qcst_Crg_Name_t));

  /* Print the CRG type.                                              */
  printf("%s ¥n", "  Cluster_Resource_Group_Type =
QcstCrgApplResiliency");

  /* Print the CRG status.                                            */
  printf("%s", "  Cluster_Resource_Group_Status = ");
  printCrgStatus(crgData->Cluster_Resource_Group_Status);

  /* Print the CRG original status.                                   */
  printf("%s", "  Original_Cluster_Res_Grp_Stat = ");
  printCrgStatus(crgData->Original_Cluster_Res_Grp_Stat);

  /* Print the Distribute Information queue name.                     */
  printStr("  DI_Queue_Name = ",
           crgData->DI_Queue_Name,
sizeof(crgData->DI_Queue_Name));
  printStr("  DI_Queue_Library_Name = ",
           crgData->DI_Queue_Library_Name,
           sizeof(crgData->DI_Queue_Library_Name));

  /* Print the CRG attributes.                                        */
  printf("%s", "  Cluster_Resource_Group_Attr = ");
  if (crgData->Cluster_Resource_Group_Attr &
QcstTcpConfigByUsr)
    printf("%s", "User Configures IP Takeover Address");
  printf("¥n");

  /* Print the ID of this node.                                       */
  printStr("  This_Nodes_ID = ",
           crgData->This_Nodes_ID, sizeof(Qcst_Node_Id_t));

  /* Print the role of this node.                                     */
  printf("%s %d ¥n", "  this node's role = ", role);

  /* Print the prior role of this node.                               */
  printf("%s %d ¥n", "  this node's prior role = ", priorRole);

  /* Print which recovery domain this role comes from.                */
  printf("%s", "  Node_Role_Type = ");
  if (crgData->Node_Role_Type == QcstCurrentRcvyDmn)
    printf("%s ¥n", "QcstCurrentRcvyDmn");
  else
    printf("%s ¥n", "QcstPreferredRcvyDmn");

  /* Print the ID of the changing node (if any).                      */
  printStr("  Changing_Node_ID = ",
           crgData->Changing_Node_ID, sizeof(Qcst_Node_Id_t));

  /* Print the role of the changing node (if any).                    */
  printf("%s", "  Changing_Node_Role = ");
  if (crgData->Changing_Node_Role == -3)
    printf("%s ¥n", "*LIST");
  else if (crgData->Changing_Node_Role == -2)
   printf("%s ¥n", "does not apply");
  else
    printf("%d ¥n", crgData->Changing_Node_Role);

  /* Print the takeover IP address.                                   */
  printStr("  Takeover_IP_Address = ",
           crgData->Takeover_IP_Address,
sizeof(Qcst_TakeOver_IP_Address_t));
```

```
  /* Print the job name.                                              */
  printStr("  Job_Name = ", crgData->Job_Name, 10);

  /* Print the CRG changes.                                           */
  printf("%s ¥n", "  Cluster_Resource_Group_Changes = ");
  if (crgData->Cluster_Resource_Group_Changes &
QcstRcvyDomainChange)
    printf("      %s ¥n", "Recovery domain changed");
  if (crgData->Cluster_Resource_Group_Changes &
QcstTakeOverIpAddrChange)
    printf("      %s ¥n", "Takeover IP address changed");

  /* Print the failover wait time.                                    */
  printf("%s", "Failover_Wait_Time = ");
  if (crgData->Failover_Wait_Time == QcstFailoverWaitForever)
     printf("%d %s ¥n", crgData->Failover_Wait_Time, "Wait
forever");
  else if (crgData->Failover_Wait_Time == QcstFailoverNoWait)
     printf("%d %s ¥n", crgData->Failover_Wait_Time, "No wait");
  else
     printf("%d %s ¥n", crgData->Failover_Wait_Time, "minutes");

  /* Print the failover default action.                               */
  printf("%s", "Failover_Default_Action = ");
  if (crgData->Failover_Default_Action == QcstFailoverProceed)
     printf("%d %s ¥n", crgData->Failover_Default_Action,
"Proceed");
  else
     printf("%d %s ¥n", crgData->Failover_Default_Action,
"Cancel");

  /* Print the failover message queue name.                           */
  printStr("  Failover_Msg_Queue = ",
           crgData->Failover_Msg_Queue,
sizeof(crgData->Failover_Msg_Queue));
  printStr("  Failover_Msg_Queue_Lib = ",
           crgData->Failover_Msg_Queue_Lib,
           sizeof(crgData->Failover_Msg_Queue_Lib));

  /* Print the cluster version.                                       */
  printf("%s %d ¥n",
         "  Cluster_Version = ", crgData->Cluster_Version);

  /* Print the cluster version mod level                              */
  printf("%s %d ¥n",
         "  Cluster_Version_Mod_Level = ",
         crgData->Cluster_Version_Mod_Level);

  /* Print the requesting user profile.                               */
  printStr("  Req_User_Profile = ",
           crgData->Req_User_Profile,
sizeof(crgData->Req_User_Profile));

  /* Print the length of the data in the structure.                   */
  printf("%s %d ¥n",
         "  Length_Info_Returned = ",
crgData->Length_Info_Returned);

  /* Print the offset to the recovery domain array.                   */
  printf("%s %d ¥n",
         "  Offset_Rcvy_Domain_Array = ",
crgData->Offset_Rcvy_Domain_Array);

  /* Print the number of nodes in the recovery domain array.          */
  printf("%s %d ¥n",
         "  Number_Nodes_Rcvy_Domain = ",
```

```
crgData->Number_Nodes_Rcvy_Domain);

  /* Print the current/new recovery domain.                              */
  printRcvyDomain("   The recovery domain:",
                  crgData->Number_Nodes_Rcvy_Domain,
                  (Qcst_Rcvy_Domain_Array1_t *)
                  ((char *)crgData +
crgData->Offset_Rcvy_Domain_Array));

  /* Print the offset to the prior recovery domain array.                */
  printf("%s %d ¥n",
         "  Offset_Prior_Rcvy_Domain_Array = ",
         crgData->Offset_Prior_Rcvy_Domain_Array);

  /* Print the number of nodes in the prior recovery domain array.       */
  printf("%s %d ¥n",
         "  Number_Nodes_Prior_Rcvy_Domain = ",
         crgData->Number_Nodes_Prior_Rcvy_Domain);

  /* Print the prior recovery domain if one was passed.                  */
  if (crgData->Offset_Prior_Rcvy_Domain_Array) {
    printRcvyDomain("   The prior recovery domain:",
                    crgData->Number_Nodes_Prior_Rcvy_Domain,
                    (Qcst_Rcvy_Domain_Array1_t *)
                ((char *)crgData +
crgData->Offset_Prior_Rcvy_Domain_Array));
  }

  return;
}  /* end printParms                                                     */


/*****************************************************************************/
/*                                                                           */
/* Print a string for the action code.                                       */
/*                                                                           */
/*****************************************************************************/
static void printActionCode(unsigned int ac) {

  char *code;
  switch (ac) {
    case QcstCrgAcInitialize: code = "QcstCrgAcInitialize";
                              break;
    case QcstCrgAcStart:      code = "QcstCrgAcStart";
                              break;
    case QcstCrgAcRestart:    code = "QcstCrgAcRestart";
                              break;
    case QcstCrgAcEnd:        code = "QcstCrgAcEnd";
                              break;
    case QcstCrgAcDelete:     code = "QcstCrgAcDelete";
                              break;
    case QcstCrgAcReJoin:     code = "QcstCrgAcReJoin";
                              break;
    case QcstCrgAcFailover:   code = "QcstCrgAcFailover";
                              break;
    case QcstCrgAcSwitchover: code = "QcstCrgAcSwitchover";
                              break;
    case QcstCrgAcAddNode:    code = "QcstCrgAcAddNode";
                              break;
    case QcstCrgAcRemoveNode: code = "QcstCrgAcRemoveNode";
                              break;
    case QcstCrgAcChange:     code = "QcstCrgAcChange";
                              break;
    case QcstCrgAcDeleteCommand: code = "QcstCrgAcDeleteCommand";
                              break;
    case QcstCrgAcUndo:       code = "QcstCrgAcUndo";
                              break;
```

```
    case QcstCrgEndNode:      code = "QcstCrgEndNode";
                              break;
    case QcstCrgAcAddDevEnt:  code = "QcstCrgAcAddDevEnt";
                              break;
    case QcstCrgAcRmvDevEnt:  code = "QcstCrgAcRmvDevEnt";
                              break;
    case QcstCrgAcChgDevEnt:  code = "QcstCrgAcChgDevEnt";
                              break;
    case QcstCrgAcChgNodeStatus: code = "QcstCrgAcChgNodeStatus";
                              break;
    case QcstCrgAcCancelFailover: code = "QcstCrgAcCancelFailover";
                              break;
    case QcstCrgAcVerificationPhase: code =
"QcstCrgAcVerificationPhase";
                                        break;
    default:                  code = "unknown action code";
                              break;
  }
  printf("%s", code);

    return;
}  /* end printActionCode                                                  */


/*****************************************************************************/
/*                                                                           */
/* Print the CRG status.                                                     */
/*                                                                           */
/*****************************************************************************/
static void printCrgStatus(int status) {

  char * str;
  switch (status) {
    case QcstCrgActive:               str = "QcstCrgActive";
                                      break;
    case QcstCrgInactive:             str= "QcstCrgInactive";
                                      break;
    case QcstCrgIndoubt:              str = "QcstCrgIndoubt";
                                      break;
    case QcstCrgRestored:             str = "QcstCrgRestored";
                                      break;
    case QcstCrgAddnodePending:       str =
"QcstCrgAddnodePending";
                                      break;
    case QcstCrgDeletePending:        str = "QcstCrgDeletePending";
                                      break;
    case QcstCrgChangePending:        str = "QcstCrgChangePending";
                                      break;
    case QcstCrgEndCrgPending:        str = "QcstCrgEndCrgPending";
                                      break;
    case QcstCrgInitializePending:    str =
"QcstCrgInitializePending";
                                      break;
    case QcstCrgRemovenodePending:    str =
"QcstCrgRemovenodePending";
                                      break;
    case QcstCrgStartCrgPending:      str =
"QcstCrgStartCrgPending";
                                      break;
    case QcstCrgSwitchOverPending:    str =
"QcstCrgSwitchOverPending";
                                      break;
    case QcstCrgDeleteCmdPending:     str =
"QcstCrgDeleteCmdPending";
                                      break;
    case QcstCrgAddDevEntPending:     str =
"QcstCrgAddDevEntPending";
```

```
                                    break;
    case QcstCrgRmvDevEntPending:     str =
"QcstCrgRmvDevEntPending";
                                    break;
    case QcstCrgChgDevEntPending:     str =
"QcstCrgChgDevEntPending";
                                    break;
    case QcstCrgChgNodeStatusPending: str =
"QcstCrgChgNodeStatusPending";
                                    break;
    default: str = "unknown CRG status";
  }
  printf("%s ¥n", str);

  return;
}  /* end printCrgStatus                                                    */


/*****************************************************************************/
/*                                                                           */
/* Print the recovery domain.                                                */
/*                                                                           */
/*****************************************************************************/
static void printRcvyDomain(char *str,
                            unsigned int count,
                            Qcst_Rcvy_Domain_Array1_t *rd) {

  unsigned int i;
  printf("¥n %s ¥n", str);
  for (i=1; i&lt;=count; i++) {
    printStr("    Node_ID = ", rd->Node_ID,
sizeof(Qcst_Node_Id_t));
    printf("%s %d ¥n", "      Node_Role = ", rd->Node_Role);
    printf("%s", "      Membership_Status = ");
    switch (rd->Membership_Status) {
      case 0: str = "Active";
              break;
      case 1: str = "Inactive";
              break;
      case 2: str = "Partition";
              break;
      default: str = "unknown node status";
    }
    printf("%s ¥n", str);
    rd++;
  }
  return;
}  /* end printRcvyDomain                                                   */

/*****************************************************************************/
/*                                                                           */
/* Concatenate a null terminated string and a non null terminated string   */
/* and print it.                                                             */
/*                                                                           */
/*****************************************************************************/
static void printStr(char *s1, char *s2, unsigned int len) {

  char buffer[132];
  memset(buffer, 0x00, sizeof(buffer));
  memcpy(buffer, s1, strlen(s1));
  strncat(buffer, s2, len);
  printf("%s ¥n", buffer);
  return;
}  /* end printStr                                                          */
```

クラスターの計画

iSeries サーバーにクラスターをセットアップする前に行う必要があることが説明されています。 クラスターの前提条件およびクラスターを設計する上でのヒントがあります。さらに、ネットワークのセットアップを行う上でのヒントおよびクラスターのパフォーマンスを改善するためのいくつかのヒントもあります。

このトピックでは、クラスタリングを使用する前に必要な要件を取り上げます。クラスタリング・ソリューションを設計するための概念、要件、考慮事項がそれぞれ以下のトピックで説明されています。

クラスターの構成および管理用のソリューション

クラスター・リソース・サービス (CRS) は基本的なクラスター・インフラストラクチャーを提供します。クラスター・リソース・サービスによって提供されるクラスタリング機能の利点を活用できるようにする方法がいくつかあります。

iSeries 上の i5/OS クラスター・リソース・サービスは、クラスターを使用するための基本的なインフラストラクチャーとして機能します。クラスター・リソース・サービスにより、クラスター・トポロジーの保守、ハートビートの実行、およびクラスター構成とクラスター・リソース・グループの作成や管理を可能にする統合サービスが提供されます。またクラスター・リソース・サービスは、クラスター内の各ノードのトラックを保持する信頼メッセージ機能を提供し、全ノードがクラスター・リソースに関する整合性の取れた情報を有するようにします。

クラスター・リソース・サービスは基本的なインフラストラクチャーを備えていますが、そうしたクラスタリング機能の利点を活用できるようにする方法がいくつかあります。各方法には、他とは異なる利点や機能があります。

重要: これらのソリューションのいずれかのみを専一に使用します。複数のソリューションを使用してクラスターを作成して管理しようとすると、競合や問題、および不測の事態が生じる可能性があります。iSeries Information Center に掲載されている情報には、iSeries ナビゲーターおよびクラスター・リソース・サービスの CL コマンドや API に特有の手順が解説されています。クラスター・ミドルウェアの IBM ビジネス・パートナー・ソリューションを使用する場合は、プロダクトに付属している資料で、タスクの実行に関する手順を参照してください。

iSeries ナビゲーター・クラスター管理

IBM は、iSeries ナビゲーターから利用できて、オプション 41 (i5/OS - HA 切り替え可能リソース) を通してアクセスできるシンプル・クラスター管理インターフェースを提供しています。

このインターフェースを使うと、切り替え可能な独立ディスク・プール (切り替え可能な独立 ASP) を使用するクラスターを作成して管理し、データの可用性を保つことができます。また、クラスター、CRG、クラスター管理可能ドメイン、およびリソースを作成して管理することもできます。

重要: iSeries ナビゲーター・クラスター管理インターフェースは、クラスター・リソース・サービスによって提供される機能をすべて備えているとは限りません。 iSeries ナビゲーターはクラスターを構成して管理するのに必要な多くの機能を提供しますが、アプリケーションによっては、クラスター・コマンドおよび API を通してのみ使用可能な機能や、おそらくはクラスター・ミドルウェアの IBM ビジネス・パートナー・アプリケーションを通してのみ使用可能な機能もあることに注意してください。たとえば、iSeries のクラスタリング・アーキテクチャーでは 1 つのクラスターに 128 のノードまで収容でできますが、 iSeries ナビゲーター・インターフェースは 1 つのクラスターにわずか 4 つのノードしか収容できません。iSeries ナビゲーター・クラスター管理の特徴は、 4 つのノードからなる単純なクラスターの作成を段階的にたどるウィザードにあります。それ以上の数のノードが

クラスターで必要なら、IBM クラスター・コマンドおよび API の使用か、またはクラスター・ミドルウェアの IBM ビジネス・パートナーのプロダクトの使用を検討してみてください。

また、iSeries ナビゲーターを使って、他のクラスター関連タスクを実行することもできます。そのようなタスクには、以下のものがあります。

- 既存のクラスターへのノードの追加
- クラスターへの切り替え可能デバイスの追加
- クラスターへの切り替え可能アプリケーションの追加
- クラスターへの切り替え可能なデータ・グループの追加
- リカバリー・ドメインにあるノードの役割の変更
- クラスター記述の変更
- クラスターの削除
- クラスタリングの開始
- クラスタリングの停止
- クラスター活動に関するメッセージの表示
- クラスター管理可能ドメインの作成
- モニター対象リソース項目の追加

iSeries ナビゲーターで利用できるクラスター関連タスクの総合リストは、クラスターのオンライン・ヘルプを参照してください。

注: iSeries ナビゲーター・クラスター管理インターフェースは、論理オブジェクト複製をサポートしません。複製の場合、高可用性ソリューションを提供している IBM ビジネス・パートナーから入手可能なクラスタリング製品の使用を検討する必要があります。

関連概念

iSeries ナビゲーター

82 ページの『クラスター・コマンドおよび API』
i5/OS クラスター・リソース・サービスからは、一連の制御言語 (CL) コマンドおよびアプリケーション・プログラム・インターフェース (API) が提供され、さらに、iSeries アプリケーション・プロバイダーまたはカスタマーが、アプリケーションの可用性を拡張するために使用できる諸機能も提供されます。

89 ページの『クラスター・ミドルウェアの IBM ビジネス・パートナーおよび使用可能なクラスタリング・プロダクト』
クラスタリングに不可欠な論理複製機能を提供するプロダクトや、クラスターを簡単に作成および管理するためのプロダクトは、 IBM クラスター・ミドルウェアのビジネス・パートナーから購入することができます。

152 ページの『クラスターの一般的な問題』
ここでは、クラスターで生じる可能性のある最も一般的な問題のいくつかをリストして、 その回避方法および回復方法を示します。

関連資料

162 ページの『iSeries ナビゲーター・クラスター管理についてよく尋ねられる質問』
クラスターの作成および管理を行う iSeries ナビゲーターのグラフィカル・ユーザー・インターフェースに関する質問と答え。

クラスター・コマンドおよび API

i5/OS クラスター・リソース・サービスからは、一連の制御言語 (CL) コマンドおよびアプリケーション・プログラム・インターフェース (API) が提供され、さらに、iSeries アプリケーション・プロバイダーまたはカスタマーが、アプリケーションの可用性を拡張するために使用できる諸機能も提供されます。

クラスター制御言語 (CL) コマンドおよびアプリケーション・プログラミング・インターフェース (API) を使用することにより、独自のカスタム・アプリケーションを作成して、クラスターを構成し管理することができます。これらのコマンドおよび API は、 i5/OS の一部として提供されるクラスター・リソース・サービスによって備えられるテクノロジーを活用できます。

QUSRTOOL

クラスター・リソース・サービスでは、QUSRTOOL ライブラリー内に、コマンド・インターフェースがサポートされていない API にマップされる コマンド例も用意されています。環境によっては、QUSRTOOL コマンドが役立つこともあるかもしれません。たとえば、ハートビートを変更したり、クラスターに関する情報を送信することもできます。これらのコマンド例の詳細については、ファイル QUSRTOOL/QATTINFO のメンバー TCSTINFO を参照してください。 アプリケーション CRG 出口プログラムの例も QUSRTOOL ライブラリーに含まれています。サンプル・ソース・コードは、出口プログラムを作成する際の基礎として使用できます。 QATTSYSC ファイルのサンプル・ソース TCSTDTAEXT には、 QCSTHAAPPI および QCSTHAAPP0 データ域、および QACSTOSDS (オブジェクト指定子) ファイルを作成するためのプログラムのソースが含まれています。

関連タスク

114 ページの『クラスターへのノードの追加』
iSeries ナビゲーターまたはコマンドを使用して、クラスターにノードを追加できます。

クラスター CL コマンドと API の説明:

クラスター、クラスター・ノード、クラスター・リソース・グループの構成、活動化、管理に使用できる API および CL コマンドが多数あります。

以下の表に、クラスターやクラスター・リソース・グループを制御するために使用可能な、制御言語 (CL) コマンドおよび API の名前と要旨を示します。クラスター CL コマンドは、OS/400® V5R2M0 以降でのみ使用可能です。

| 最初の表では、**クラスター**とクラスター内の**ノード**を構成、活動化、および管理するためのコマンドと
| API を示します。 2 番目の表では、クラスター内の**クラスター・リソース・グループ**を構成、活動化、お
| よび管理するためのコマンドと API を示します。 3 番目の表では、**クラスター管理ドメイン**を構成およ
| び管理するためのコマンドを示します。 4 番目の表では、クラスター管理ドメインでモニター対象リソー
| ス項目を追加したり、除去したりするために使用できる、統合オペレーティング・システム API について
| 解説しています。

表 8. クラスター制御 CL コマンドおよび API の説明

説明	クラスター制御 CL コマンド	クラスター制御 API 名
クラスター・ノード項目の追加 既存のクラスターのメンバーシップ・リストにノードを追加します。クラスター通信に使用される IP インターフェース・アドレスも割り当てます。	ADDCLUNODE	クラスター・ノード項目追加 (QcstAddClusterNodeEntry)
デバイス・ドメイン項目の追加 ノードをドメイン・メンバーシップ・リストに追加し、ノードが回復装置のリカバリー・アクションに参加できるようにします。デバイス・ドメインに最初のノードを追加すると、そのデバイス・ドメインが作成されることになります。	ADDDEVDMNE	デバイス・ドメイン項目追加 (QcstAddDeviceDomainEntry)
クラスター・バージョンの調整、クラスター・バージョンの変更 現行のクラスター・バージョンを次のレベルに調整します。たとえば、クラスター内で新しい機能を使えるようにする場合などです。	CHGCLUVER	クラスター・バージョンの調整 (QcstAdjustClusterVersion)
クラスター・ノード項目の変更 クラスター・ノード項目のフィールドを変更します。たとえば、クラスター通信に使用される IP インターフェース・アドレスを変更できます。	CHGCLUNODE	クラスター・ノード項目の変更 (QcstChangeClusterNodeEntry)
クラスター・リソース・サービスの変更、クラスター構成の変更 クラスターのパフォーマンスおよび構成調整パラメーターを、クラスター通信に使用されるネットワークの通信環境に合わせて調整します。	CHGCLUCFG	クラスター・リソース・サービス変更 (QcstChgClusterResourceServices)
クラスターの作成 1 つ以上のノードに新しいクラスターを作成します。	CRTCLU	クラスターの作成 (QcstCreateCluster)
クラスターの削除 既存のクラスターを削除します。クラスター・リソース・サービスはアクティブなすべてのクラスター・ノード上で終了し、クラスターから除去されます。	DLTCLU	クラスターの削除 (QcstDeleteCluster)

表 8. クラスター制御 CL コマンドおよび API の説明 (続き)

説明	クラスター制御 CL コマンド	クラスター制御 API 名
クラスター・ノードの終了 既存のクラスターのメンバーシップ・リストに載っている 1 つまたはすべてのノードで、クラスター・リソース・サービスを終了させます。そのノードは、クラスター・ノードの開始インターフェースを使用して再始動するまで、クラスターから使用不能になります。	ENDCLUNOD	クラスター・ノード終了 (QcstEndClusterNode)
クラスター情報のリスト、クラスター情報の表示 クラスターについての情報を検索します。たとえば、完全なクラスター・メンバーシップ・リストを戻すことができます。	DSPCLUINF	クラスター情報のリスト (QcstListClusterInfo)
デバイス・ドメイン情報のリスト、クラスター情報の表示 クラスターのデバイス・ドメイン情報をリストします。たとえば、現在定義されているデバイス・ドメインのリストを戻すことができます。	DSPCLUINF	デバイス・ドメイン情報リスト (QcstListDeviceDomainInfo)
クラスター・ノード項目の除去 クラスターのメンバーシップ・リストからノードを除去します。ノードはあらゆるリカバリー・ドメインから除去され、そのノード上のクラスター操作は終了します。また、そのノードからすべてのクラスター・リソース・サービス・オブジェクトが削除されます。	RMVCLUNODE	クラスター・ノード項目の除去 (QcstRemoveClusterNodeEntry)
デバイス・ドメイン項目の除去 デバイス・ドメイン・メンバーシップ・リストからノードを除去します。デバイス・ドメインから最後のノードを除去すると、クラスターからデバイス・ドメインも削除されます。	RMVDEVDMNE	デバイス・ドメイン項目の除去 (QcstRemoveDeviceDomainEntry)
クラスター情報の検索、クラスター情報の表示 クラスターについての情報を検索します。たとえば、クラスター名と現行のクラスターのバージョンを戻します。	DSPCLUINF	クラスター情報の検索 (QcstRetrieveClusterInfo)

表 8. クラスター制御 CL コマンドおよび API の説明 (続き)

説明	クラスター制御 CL コマンド	クラスター制御 API 名
クラスター・リソース・サービス情報検索、クラスター情報の表示 クラスター・リソース・サービスのパフォーマンス調整および構成パラメーターの情報を検索します。	DSPCLUINF	クラスター・リソース・サービス情報検索 (QcstRetrieveCRSInfo)
クラスター・ノードの開始 クラスターの一部ではあるものの、現在アクティブでないノード上で、クラスター・リソース・サービスを開始します。	STRCLUNOD	クラスター・ノードの開始 (QcstStartClusterNode)
クラスターとの作業 クラスター・ノードとオブジェクトの表示、およびその作業。	WRKCLU	なし

表 9. クラスター・リソース・グループ CL コマンドおよび API の説明

説明	クラスター・リソース・グループ CL コマンド	クラスター・リソース・グループ API
クラスター・リソース・グループの装置項目の追加 クラスター・リソース・グループに新しい装置項目を追加します。装置は、切り替え可能な装置のグループのメンバーになります。	ADDCRGDEVE	クラスター・リソース・グループの装置項目の追加 (QcstAddClusterResourceGroupDevice)
リカバリー・ドメインへのノードの追加、クラスター・リソース・グループの装置項目の追加 既存のクラスター・リソース・グループのリカバリー・ドメインに新しいノードを追加します。	ADDCRGNODE	リカバリー・ドメインへのノードの追加 (QcstAddNodeToRcvyDomain)
クラスター・リソース・グループの変更 クラスター・リソース・グループの属性を変更します。たとえば、アプリケーション CRG 用のテークオーバー IP アドレスを変更できます。	CHGCRG	クラスター・リソース・グループ変更 (QcstChangeClusterResourceGroup)
クラスター・リソース・グループの装置項目の変更 クラスター・リソース・グループの装置項目を変更します。たとえば、切り替えまたはフェイルオーバー時の構成オブジェクトをオンラインで変更するオプションを変更できます。	CHGCRGDEVE	クラスター・リソース・グループの装置項目変更 (QcstChangeClusterResourceGroupDev)

表 9. クラスター・リソース・グループ CL コマンドおよび API の説明 (続き)

説明	クラスター・リソース・グループ CL コマンド	クラスター・リソース・グループ API
クラスター・リソース・グループの作成 クラスター・リソース・グループ・オブジェクトを作成します。クラスター・リソース・グループ・オブジェクトはリカバリー・ドメインを識別します。リカバリー・ドメインは、回復に関与するクラスター内のノードの集合です。	CRTCRG	クラスター・リソース・グループの作成 (QcstCreateClusterResourceGroup)
クラスター・リソース・グループの削除 ローカル・ノードでのみ、クラスター・リソース・グループ (CRG) を削除します。ローカル・クラスター・リソース・グループの削除には、クラスター・リソース・サービスを非活動化する必要があります。	DLTCRG	なし
クラスター・リソース・グループの削除、CRG クラスターの削除 クラスターからクラスター・リソース・グループを削除します。リカバリー・ドメインの中のすべてのアクティブ・ノードから CRG オブジェクトが削除されます。	DLTCRGCLU	クラスター・リソース・グループの削除 (QcstDeleteClusterResourceGroup)
情報の配布 CRG のリカバリー・ドメインの中のノードから、 同じ CRG のリカバリー・ドメインの別のノードに情報を送信します。	なし	情報配布 (QcstDistributeInformation)
クラスター・リソース・グループの終了 指定されたクラスター・リソース・グループの回復性を使用不可にします。この API が正常に完了すると、クラスター・リソース・グループの状況は非アクティブに設定されます。	ENDCRG	クラスター・リソース・グループの終了 (QcstEndClusterResourceGroup)

表 9. クラスター・リソース・グループ CL コマンドおよび API の説明 (続き)

説明	クラスター・リソース・グループ CL コマンド	クラスター・リソース・グループ API
切り替えの開始、クラスター・リソース・グループ・プライマリーの変更 クラスター・リソース・グループのために管理切り替えを実行します。リカバリー・ドメインは変更され、現行のプライマリー・ノードは最後のバックアップ・ノードになり、現行の最初のバックアップ・ノードは新しいプライマリー・ノードになります。	CHGCRGPRI	切り替えの開始 (QcstInitiateSwitchover)
クラスター・リソース・グループのリスト、クラスター・リソース・グループ情報の表示 現行のクラスター・リソース・グループと、クラスター内のクラスター・リソース・グループについての情報のリストを生成します。	DSPCRGINF	クラスター・リソース・グループのリスト (QcstListClusterResourceGroups)
クラスター・リソース・グループ情報のリスト、クラスター・リソース・グループ情報の表示 クラスター・リソース・グループ・オブジェクトの内容を戻します。たとえば、リカバリー・ドメインと現行のノードの役割を戻すことができます。	DSPCRGINF	クラスター・リソース・グループのリスト (QcstListClusterResourceGroupInf)
クラスター・リソース・グループの装置項目の除去 クラスター・リソース・グループから装置項目を除去します。装置は切り替え可能リソースではなくなります。	RMVCRGDEVE	クラスター・リソース・グループの装置項目除去 (QcstRemoveClusterResourceGroupDev)
リカバリー・ドメインからのノードの除去、クラスター・リソース・グループのノード項目の除去 既存のクラスター・リソース・グループのリカバリー・ドメインからノードを除去します。ノードは、そのリソース・グループのリカバリー・アクションには関与しなくなります。	RMVCRGNODE	リカバリー・ドメインからのノードの除去 (QcstRemoveNodeFromRcvyDomain)
クラスター・リソース・グループの開始 指定されたクラスター・リソース・グループの回復性を使用可能にします。クラスターの中でクラスター・リソース・グループがアクティブになります。	STRCRG	クラスター・リソース・グループの開始 (QcstStartClusterResourceGroup)

注: クラスター・リソース・サービスでは、QUSRTOOL ライブラリー内に、 上述の CL コマンドと API にマップされるコマンド例も用意されています。環境によっては、QUSRTOOL コマンドが役立つこともあるかもしれません。たとえば、クラスター対応のアプリケーションをテストするために、簡単にクラスターをセットアップできるコマンドもあります。これらのコマンド例の詳細については、ファイル QUSRTOOL/QATTINFO のメンバー TCSTINFO を参照してください。

表 10. 管理ドメイン CL コマンドの説明

説明	管理ドメイン CL コマンド	管理ドメイン API
管理ドメインの作成 クラスター管理ドメインを表す対等 CRG を作成します。 1 度クラスター管理ドメインが作成されると、リソースの変更を同期するためドメインにモニター対象リソース項目 (MRE) を追加できるようになります。 **注:** クラスター管理ドメインが作成された後は、ドメインを管理するために 85 ページの表 9 で CRG コマンドを使用できるようになります。	CRTADMDMN	なし
管理ドメインの削除 クラスター管理ドメインを表す対等 CRG を削除します。 ドメインからすべての MRE の除去が完了すると、モニターされていたリソースへの変更は、それ以後伝搬されなくなります。	DLTADMDMN	なし

表 11. 統合オペレーティング・システム API の説明: これらのクラスター管理ドメイン CL コマンドの他にも、モニター対象リソース項目の追加、および削除を実行できる、統合オペレーティング・システム・アプリケーション・プログラミング・インターフェース (API) があります。

説明	CL コマンド[1]	統合オペレーティング・システム API
モニター対象リソース項目の追加 システム・リソースとその属性に、モニター対象リソース項目を追加します。	なし	モニター対象リソース項目の追加 (QfpadAddMonitoredResourceEntry)
モニター対象リソース項目の除去 モニター対象ディレクトリーから、モニター対象リソース項目 (MRE) を除去します。	なし	モニター対象リソース項目の除去 (QfpadRmvMonitoredResourceEntry)

表 11. 統合オペレーティング・システム API の説明 (続き): これらのクラスター管理ドメイン CL コマンドの他にも、モニター対象リソース項目の追加、および削除を実行できる、統合オペレーティング・システム・アプリケーション・プログラミング・インターフェース (API) があります。

説明	CL コマンド[1]	統合オペレーティング・システム API
モニター対象リソース情報検索 モニター対象リソースに関する情報を戻します。	なし	モニター対象リソース情報検索 (QfpadRtvMonitoredResourceInfo)

注:

1. この機能に相当する CL コマンドとしてサポートされているものはありません。サポートされていないコマンドおよび呼び出し処理プログラム (CPP) のソースは、QUSRTOOL ライブラリー内に用意されています。このコマンド・ソース、および CPP の詳細は、ファイル QATTINFO のメンバー QFPADINFO を参照してください。

関連資料

クラスター API

クラスター・ミドルウェアの IBM ビジネス・パートナーおよび使用可能なクラスタリング・プロダクト

クラスタリングに不可欠な論理複製機能を提供するプロダクトや、クラスターを簡単に作成および管理するためのプロダクトは、 IBM クラスター・ミドルウェアのビジネス・パートナーから購入することができます。

IBM クラスター・ミドルウェアのビジネス・パートナーは、複製専用のおよびクラスター管理機能用のソフトウェア・ソリューションを提供します。　クラスタリングに不可欠な複製機能を提供するプロダクトや、クラスターを簡単に作成および管理するためのプロダクトの購入をご希望の方は、IBM 営業担当員またはビジネス・パートナーにお問い合わせください。 IBM クラスター・ミドルウェアのビジネス・パートナーから提供されるクラスタリング対応プロダクトの総合リストが用意されています。

クラスター・ミドルウェアの IBM ビジネス・パートナーのクラスター管理プロダクト:

- クラスター構成を定義および保守するためのユーザー・インターフェースを提供します。
- 装置クラスター・リソース・グループ、データ・クラスター・リソース・グループ、アプリケーション・クラスター・リソース・グループを定義および管理するためのユーザー・インターフェースを提供します。
- クラスター API を使用することによって、クラスター内に定義されているクラスター・リソース・グループについての情報、および必要とされている関係についての情報を保守します。
- 装置クラスター・リソース・グループ、データ・クラスター・リソース・グループ、アプリケーション・クラスター・リソース・グループを作成します。

クラスター・ミドルウェアの IBM ビジネス・パートナーの複製プロダクト:

- 回復力を備える必要のあるデータおよびオブジェクトを特定するためのミドルウェアの制御構造を構築します。
- 重要なデータのためのクラスター・リソース・グループを作成し、そのオブジェクトを制御構造に関連付けます。
- データ・クラスター・リソース・グループのための出口プログラムを提供します。

関連タスク

114 ページの『クラスターへのノードの追加』
iSeries ナビゲーターまたはコマンドを使用して、クラスターにノードを追加できます。

クラスター要件

ここでは、クラスターを使用するのに必要なハードウェア、ソフトウェア、および通信の諸要件の概略を述べています。

クラスターの使用に関する要件は、どのクラスター機能を選んで使用するかによって異なります。たとえば、論理複製の利点をいかすために、2 つのノードからなる単純なクラスターを選んで使用することもあれば、切り替えディスクや切り替え可能独立ディスク・プールの利点をいかす設計のクラスターを選んで使用することもあります。

関連概念

133 ページの『例: クラスター構成』
典型的なクラスターのインプリメンテーション例を使って、いつ、なぜ、そしてどのようにクラスターを使用するのが有益であるかを解明します。

クラスターのハードウェア要件

i5/OS V4R4M0 またはそれ以降を実行できる iSeries モデルは、 クラスタリングを使用したものと互換性があります。

さらに、外部の無停電電源装置などの装置によって、停電時の保護手段を設けるようにしてください。保護手段がない場合は、クラスター・ノードで突然の停電があったときに、フェイルオーバーではなく、クラスター区画状態が発生する可能性があります。

クラスタリングを実施する際には、インターネット・プロトコル (IP) マルチキャスト機能を使用します。物理メディアによっては、マルチキャストが正常にマップを行わない場合があります。各自のハードウェアに当てはまるマルチキャストの制約事項の詳細は、TCP/IP のセットアップ (TCP/IP Setup) を参照してください。

クラスターで独立ディスク・プールを使用する予定の場合、独立ディスク・プールに対するハードウェア要件のトピックを参照してください。ミラー保護または装置パリティー保護により、ディスクを保護することもできます。これらのソリューションをプライマリー・システムで使用すれば、フェイルオーバーが実行されても、保護されているディスクに障害が発生してしまうのを防ぐことができます。フェイルオーバーが実行される場合には、バックアップ・システムでも、これらのソリューションを使用するようお勧めします。詳細については、ディスク保護を参照してください。

注: クラスターの構成の前に必要なその他の要件の詳細は、 103 ページの『クラスター構成チェックリスト』を参照してください。

関連概念

無停電電源装置

33 ページの『クラスター区画』
クラスター区画は、通信障害の結果として生じる、アクティブなクラスター・ノードのサブセットです。 区画のメンバーは、相互の接続を維持します。

20 ページの『フェイルオーバー』
フェイルオーバーは、システム障害が発生したときに、クラスターのサーバーが 1 つ以上のバックアップ・サーバーへ自動的に切り替わるときに発生します。

クラスターのソフトウェアおよびライセンス交付要件

クラスタリングを使用するためには、正しいソフトウェアとライセンスが必要です。

1. OS/400 V4R4M0 またはそれ以降 (TCP/IP (TCP/IP Connectivity Utilities) を使って構成されたもの)。

2. クラスターの構成および管理ソフトウェア・ソリューション。これは、次のいずれかで構いません。
   - iSeries ナビゲーター・クラスター管理
   - クラスター・ミドルウェア IBM ビジネス・パートナー・ソリューション
   - クラスター・リソース・サービス・コマンドおよび API を使用する、ユーザー自身が作成したクラスター管理アプリケーション・プログラム
3. 103 ページの『クラスター構成チェックリスト』を参照してください。

重要: 切り替え可能装置の利点をいかすために独立ディスク・プールを使用する予定の場合、さらに別の要件があります。詳細については、 独立ディスク・プールの計画を参照してください。

関連概念

80 ページの『クラスターの構成および管理用のソリューション』
クラスター・リソース・サービス (CRS) は基本的なクラスター・インフラストラクチャーを提供します。 クラスター・リソース・サービスによって提供されるクラスタリング機能の利点を活用できるようにする方法がいくつかあります。

15 ページの『クラスター・バージョン』
クラスター・バージョンとは、 クラスターで実行できる機能のレベルを表す用語です。

クラスターの通信要件

任意のタイプの通信メディアをクラスタリング環境で使用します。ただし、そのタイプのメディアがインターネット・プロトコル (IP) をサポートしていることを前提とします。

クラスター・リソース・サービスは、ノードとの通信を確立するのに TCP/IP プロトコルだけを使用します。ローカル・エリア・ネットワーク (LAN)、広域ネットワーク (WAN)、OptiConnect システム・エリア・ネットワーク (SAN)、またはこれらの接続装置の組み合わせがサポートされています。以下の要因に基づいて選択を行ってください。

- トランザクションの量
- 応答時間の要件
- ノード間の距離
- コストについての考慮事項

これらの考慮事項は、リソースのプライマリー・ロケーションと、バックアップ・ロケーションとを接続するために使用される接続メディアを決定する際に当てはまります。クラスターの計画段階では、サイトが失われるような災害を切り抜けるために、リモート・ロケーションにある 1 つ以上のバックアップ・ノードを指定するようお勧めします。

容量の不足が原因で生じ得るパフォーマンスの問題を回避するためには、ノードからノードへと送られる多量の情報を処理するのに使用されている通信メディアを評価する必要があります。トークンリング、イーサネット、非同期転送モード (ATM)、SPD OptiConnect、高速リンク (HSL) OptiConnect、仮想 OptiConnect (論理区画間の高速内部接続) など、使用したい物理メディアを選択することができます。

HSL OptiConnect は、i5/OS ソフトウェア (i5/OS オプション 23 - i5/OS OptiConnect) に備わったテクノロジーです。これを使用して、高可用性ソリューションを構成できます。HSL OptiConnect は、 高速リンク (HSL) ループ・テクノロジーを使用してクラスター・ノード間の高速、2 地点間接続を提供するシステム・エリア・ネットワークです。HSL OptiConnect は、標準 HSL ケーブルを必要としますが、他のハードウェアは必要としません。

切り替え可能ハードウェア (回復装置 CRG とも呼ばれる) の場合は、それぞれの環境の中に切り替え可能な独立ディスク・プールが必要になります。論理区画環境において、これは、論理区画で共用されるバス上にあるディスク装置のコレクション、または 1 つの入出力プールに割り当てられた入出力プロセッサーに接続されたディスク装置のコレクションです。マルチシステム環境では、リカバリー・ドメインにあるシステムも含む HSL ループ上に適切に 構成された 1 つ以上の切り替え可能な拡張装置 (タワー) になります。 切り替え可能なタワーは、LPAR 環境でも使用できます。切り替え可能ハードウェアおよび独立ディスク・プールの計画に関する情報については、 独立ディスク・プールの計画を参照してください。

注: TCP/IP **だけ**を使用し、システム・ネットワーク体系 (SNA) や IPX を使用しない 2810 LAN アダプターを使用する場合は、回線記述の処理 (WRKLIND) コマンドを使って、該当する回線記述に対して TCP/IP の場合に使用可能 (*YES) を指定すれば、V4R5M0 サーバーのアダプターのパフォーマンスを向上することができます。 TCP/IP の場合に使用可能 (*YES) は、V5R1M0 およびそれ以降のリリースでは自動的に設定されます。

関連情報

OptiConnect for i5/OS

クラスターの設計

どのようにクラスターを設計するかを決定するために必要なことを示します。

何を成し遂げたいかによりクラスタリングの使用方法は変わってくるので、クラスターの設計方法を決定するために、どのような要件があるのかを判断する時間を取ることは大切です。

クラスターに適したネットワークの設計

クラスタリングの準備としてネットワークを構成する前に、TCP/IP に関係したクラスタリング前の初期構成作業を注意深く計画し、 実行に移す必要があります。

クラスターを構成する前に、下記のトピックを読んでおくようお勧めします。これらのトピックでは、以下のことを行うタイミングと方法が説明されています。

- IP アドレスの設定
- TCP/IP 構成属性の設定
- クラスター区画の回避

冗長通信パスのセットアップとクラスタリングを実施するための専用ネットワークが必要かどうかに関する情報については、クラスターに使用するネットワークの専用化を参照してください。

IP アドレスの設定:

クラスター・リソース・サービスは、 他のクラスター・ノードとの通信を確立するのに IP *だけ*を使用するので、 すべてのクラスター・ノードが *IP に到達*できなければなりません。

このことは、クラスター内のノードを接続するために、IP インターフェースを構成しなければならないことを意味します。これらの IP アドレスは、ネットワーク管理者が各クラスター・ノード上の TCP/IP ルーティング・テーブルにおいて手動で設定するか、ネットワーク内のルーター上で実行されているルーティング・プロトコルによって生成します。この TCP/IP ルーティング・テーブルは、各ノードを検出するためにクラスタリングが使用するマップです。したがって、各ノードには*固有な* IP アドレスが必要です。 1 つのノードには最多で 2 つの IP アドレスを割り当てることができます。これらのアドレスは、いかなる手段によっても、他のネットワーク通信アプリケーションが変更してはなりません。各アドレスを割り当てる際には、どのアドレスがどのタイプの通信回線を使用するかに注意してください。特定のタイプの通信メディアを優先して使用したい場合には、優先して使用したいそのメディアによって、1 番目の IP アドレスを

構成してください。 1 番目の IP アドレスは、信頼メッセージ機能およびハートビート・モニターによって、優先的に扱われる IP アドレスです。 ノード上のすべての IP アドレスは、クラスター内の他のすべての IP アドレスに到達可能でなければなりません。 ping をして、UDP メッセージの経路追跡を両方向で使用できれば、アドレスからアドレスへの到達が可能であるということです。

注: ループバック・アドレス (127.0.0.1) がアクティブでなければ、クラスタリングは実施できません。デフォルトでは、このアドレス (メッセージをローカル・ノードに戻すために使用される) はアクティブであるのが普通です。しかし、何らかの間違いによって終了した場合には、このアドレスを再びアクティブにしない限り、クラスターのメッセージ機能は実行できません。

関連概念

32 ページの『メッセージング機能』
クラスター・リソース・サービスのメッセージング機能は、クラスター内の各ノードに注意を払い、クラスター・リソースの状態に関する整合した情報を確実にすべてのノードが保持するようにします。

30 ページの『ハートビート・モニター』
ハートビート・モニターはクラスター・リソース・サービス機能の 1 つで、クラスター内のすべてのノードからクラスター内の自分以外のすべてのノードにシグナルを送信して自分がアクティブであることを伝達してすべてのノードがアクティブであることを確証するものです。

TCP/IP 構成属性の設定:

クラスター・リソース・サービスを使用可能にするには、 ネットワークの TCP/IP 構成でいくつかの属性を設定しなければなりません。

これらの属性を設定しないと、クラスターにノードを追加することはできません。

- 他のネットワークと通信するためのルーターとして iSeries サーバーを使用する予定の場合に、そのサーバー上で他のルーティング・プロトコルが実行されていなければ、CHGTCPA (TCP/IP 属性の変更) コマンドを使用して IP データグラムの転送を *YES に設定します。
- INETD サーバーを START に設定します。 INETD サーバーの開始方法については、INETD サーバーを参照してください。
- CHGTCPA (TCP/IP 属性の変更) コマンドを使用して UDP チェックサムを *YES に設定します。
- ブリッジを使用してトークンリング・ネットワークに接続する場合は、MCAST の転送を *YES に設定します。
- i5/OS に対して OptiConnect を使用してクラスター・ノード間の通信を行う場合は、STRSBS(QSOC/QSOC) を指定して QSOC サブシステムを開始します。

ヒント: クラスター通信:

通信パスをセットアップする際には、以下のヒントを参考にしてください。

- クラスタリング・ハートビート機能に関係した非クラスター活動を処理し、増大する活動をモニターし続けられるようにするため、通信回線には十分な帯域幅を持たせてください。
- 最善の信頼性を得るために、1 つ以上のノードにリンクしている通信パスを、1 つだけ構成することは避けてください。
- ノードとの通信を維持するために使用する回線に、過度の負担をかけないでください。
- 1 箇所の障害が重大な影響を及ぼすような構成は避けてください。たとえば、2 本の通信回線が 1 つのアダプター、同じ入出力プロセッサー (IOP)、または同じタワーに接続することはできるだけ避けてください。

- 通信回線を通して極端に大量のデータを送信する場合は、データ複製とハートビート・モニターを別々のネットワークで行ってください。
- インターネット・プロトコル (IP) マルチキャストを使用する場合には、TCP/IP 構成および解説書を参照してください。使用する物理メディアによっては、マルチキャストに関する制限があります。
- 望ましいプロトコルはユーザー・データグラム・プロトコル (UDP) マルチキャストで、クラスター通信インフラストラクチャーがクラスター内のノード間でクラスター管理情報を送信するために使用します。物理メディアがマルチキャスト機能をサポートする場合、クラスター通信は UDP マルチキャストを使用して、管理メッセージを特定のノードから、同じサブネット・アドレスをサポートするローカル・クラスター・ノードすべてに送信します。リモート・ネットワーク上のノードに送信されるメッセージは常に、UDP 2 地点間機能を使って送信されます。マルチキャスト・メッセージの場合、クラスター通信はルーティング機能に依存しません。
- クラスター管理メッセージをサポートするマルチキャスト・トラフィックは、変動しやすい性質を持っています。特定の LAN (共通サブネット・アドレスをサポートする) 上のノード数、およびクラスター管理者が選択したクラスター管理構造の複雑さに応じて、クラスターに関連したマルチキャスト・パケットは秒当たり 40 パケットを超えることも珍しくありません。この種の変動は、旧式のネットワーク装置に悪影響を与えることがあります。一例として、すべての UDP マルチキャスト・パケットを評価しなければならない Simple Network Management Protocol (SNMP) エージェントの役目を果たす LAN における装置上の問題があります。旧式のネットワーク装置の中には、この種のトラフィックに見合う適切な帯域幅を持っていないものがあります。ユーザー自身またはネットワーク管理者が UDP マルチキャスト・トラフィックの処理に関するネットワークの容量を検討し、クラスタリングによってネットワークのパフォーマンスが低下しないことを確かめる必要があります。

  関連概念

  99 ページの『クラスター論理複製の計画』
  データの複数のコピーは、論理複製を使って保守されます。 データは、クラスターのプライマリー・ノードからリカバリー・ドメインにある指定されたバックアップ・ノードに複製つまりコピーされます。プライマリー・ノードで障害が発生したときには、指定されたバックアップ・ノードが 1 次アクセス・ポイントを引き継ぐので、データは使用可能なままです。

  32 ページの『メッセージング機能』
  クラスター・リソース・サービスのメッセージング機能は、クラスター内の各ノードに注意を払い、クラスター・リソースの状態に関する整合した情報を確実にすべてのノードが保持するようにします。

  関連情報

  TCP/IP セットアップ

クラスター区画の回避:

典型的なネットワーク関連クラスター区画が発生しないようにする最も有効な手段としては、クラスター内のすべてのノードを結び付ける冗長通信パスを構成します。

冗長通信パスを構成することは、クラスター内の 2 つのノードの間に 2 本の回線を構成することを意味します。万が一、最初の通信パスに障害が発生しても、ノード間で実行されている通信を 2 番目の通信パスが引き継ぐため、クラスター内の 1 つ以上のノードがクラスター区画に入る可能性を最小限に抑えることができます。これらのパスを構成する際に考慮するべき事柄は、たとえ 2 本の通信回線を用意したとしても、それらの回線をどちらもシステム上の同じアダプターに接続してしまうのであれば、障害がそのアダプターで発生した場合に、それらの回線が 2 本とも使用不可になる危険性があるということです。

ただし、クラスター区画を常に阻止できるとは限らないことに注意しなければなりません。システムで停電が発生した場合や、ハードウェア障害が発生した場合、クラスターが区画に分割されることがあります。

関連概念

33 ページの『クラスター区画』
クラスター区画は、通信障害の結果として生じる、アクティブなクラスター・ノードのサブセットです。 区画のメンバーは、相互の接続を維持します。

93 ページの『ヒント: クラスター通信』
通信パスをセットアップする際には、以下のヒントを参考にしてください。

154 ページの『区画エラー』
クラスター状態によっては、簡単に修正することができます。 クラスター区画が生じた場合の回復方法を確認してください。 このトピックでは、クラスター区画の回避方法を示し、区画をマージして元に戻す方法を例示します。

クラスターに使用するネットワークの専用化:

通常の操作時は、基本的なクラスタリング通信トラフィックは最小限に抑えられています。 とはいえ、クラスター内のノードごとに冗長通信パスを構成するよう強くお勧めします。

2 つの回線を構成すると、一方の回線をクラスタリング・トラフィック専用に、もう一方の回線は通常のトラフィックの処理用に割り当てることができます。後者の回線は、クラスタリング用の専用回線がダウンした場合のバックアップ回線とすることもできます。

関連概念

94 ページの『クラスター区画の回避』
典型的なネットワーク関連クラスター区画が発生しないようにする最も有効な手段としては、クラスター内のすべてのノードを結び付ける冗長通信パスを構成します。

複数リリースのクラスター

複数のクラスター・バージョンのノードを含むクラスターを作成する場合、クラスターの作成時に特定のステップが必要になります。

デフォルトでは、現行クラスター・バージョンには、クラスターに最初に追加されたノードの潜在クラスター・バージョンが設定されます。このノードがクラスターに存在する最も低いバージョン・レベルである場合、この方法は適切であると言えます。しかし、このノードがそれよりも新しいバージョン・レベルであった場合、その後、それよりも低いバージョン・レベルのノードは追加できません。代わりに、クラスターの作成時のターゲット・クラスター・バージョン値を使用して、現行クラスター・バージョンを、クラスターに最初に追加されたノードの潜在クラスター・バージョンよりも 1 低い値に設定します。

たとえば、2 つのノードからなるクラスターを作成する場合を考慮してみましょう。このクラスターのノードは次のようになります。

ノード ID	リリース	潜在クラスター・バージョン
ノード A	V5R3	4
ノード B	V5R4	5

クラスターがノード B から作成された場合、これは混合リリース・クラスターになることを必ず示すようにしてください。ターゲット・クラスター・バージョンは、通信を要求するノードの潜在ノード・バージョンよりも 1 小さいバージョンでクラスターのノードが通信することを示すように設定されなければなりません。

関連概念

15 ページの『クラスター・バージョン』
クラスター・バージョンとは、 クラスターで実行できる機能のレベルを表す用語です。

クラスターに含めるサーバーの識別

クラスターに含めるサーバーを識別するためには、ビジネスを運営する上で必要なデータやアプリケーションのバックアップを十分に提供できるサーバーは、 どれであるかを見極める必要があります。

以下のシステムを決定する必要があります。

- 重要なデータやアプリケーションを含めるサーバー
- それらのサーバーをバックアップするシステム

上記のサーバーが決定したならば、それらのサーバーをクラスターに含めます。

プライマリー・バックアップ・モデルと対等モデルとの比較

プライマリー・バックアップ CRG および対等 CRG は、クラスター内のリソースに回復性を提供します。 ただし、それらの違い、使用方法について理解することが重要です。

クラスターでは、ご使用の環境で 2 種類の CRG モデルを定義できるようになっています。役割はプライマリー・バックアップ・モデル、対等モデルのいずれでも定義されます。プライマリー・バックアップ・モデルでは、順序を定義する必要あります。バックアップ・ノードとして定義されているノードは、ノードに障害が発生した場合にプライマリー・ノードに存在するノードへのアクセスを提供します。対等モデルの場合、役割の中の各ノードは等しく、リソースへのアクセスを提供できます。ただし、順序という概念はありません。

プライマリー・バックアップ・モデル

プライマリー・バックアップ・モデルの場合、ノードをプライマリーかバックアップのいずれか、あるいは複製役割として定義しなければなりません。これらの役割はリカバリー・ドメインで定義、管理されます。ノードがリソースのプライマリー・アクセス・ポイントとして定義されている場合、その他のノードはプライマリー・ノードに障害が発生した場合にバックアップとしての機能を提供します。

対等モデル

対等モデル CRG は、順序付きのリカバリー・ドメインを定義する必要性を除去します。対等モデルの場合、対等、または複製のいずれかとしてノードを定義できます。ノードが対等として定義された場合、リカバリー・ドメインにあるすべてのノードは等しく、リソースへのアクセス・ポイントを提供できます。

クラスターに含めるアプリケーションの識別

すべてのアプリケーションが、クラスタリングを使用できることを有効に利用できるわけではありません。

クラスタリングにより提供される切り替えおよびフェイルオーバー機能を利用するためには、アプリケーションは回復できなければなりません。アプリケーション回復力により、アプリケーションを使用するクライアントを再構成しなくても、アプリケーションをバックアップ・ノードで再始動できます。そのため、クラスタリングにより提供される機能を十分活用するためには、使用するアプリケーションは特定の要件を満たしていなければなりません。回復アプリケーションに関する詳細は、クラスター・アプリケーションを参照してください。

データ回復の計画

ユーザーまたはアプリケーションが常にデータを使用できれば、そのデータには回復力があることになります。 論理複製や切り替え可能独立ディスク・プールを使用して、データ回復力を達成することができます。

どのデータが回復されるべきかを判断する:

どのタイプのデータを回復できるように考慮すべきなのかを理解してください。

回復する必要があるのはどのデータであるかを判断することは、担当システムのバックアップ戦略および回復戦略を準備する際に、どのタイプのデータを、バックアップおよび保管する必要があるかを考慮することに似ています。環境内のデータのうち、ビジネスが稼働状態になる上で重要なデータはどのデータであるかを見極める必要があります。

たとえば、Web 上でビジネスを実行する場合、次のようなデータが重要なデータとしてあげられます。

- 本日分の注文
- 在庫
- 顧客レコード

変更の頻度が少ない情報や日常の業務であまり使わないデータは、通常、回復する必要はありません。

関連概念

バックアップおよび回復方針の計画

論理複製、切り替えディスク、およびサイト間ミラーリングの比較:

このトピックでは、高可用性を高めるためにクラスターと共に使用可能な、さまざまなデータ回復テクノロジーの概要を解説します。

データ回復性は、元々そのデータをホストしているシステムに障害が発生した場合にも、アプリケーションおよびユーザーに対し、データを使用できる状態を保ちます。ビジネス全体での継続性戦略という背景において、データ回復性テクノロジーをいかに正しく選択するかということは、複雑かつ困難になる可能性があります。複数のシステム環境において、可用性を高めるために使用される、さまざまなデータ回復ソリューションを理解することは重要です。ニーズに応じて、あるソリューションを単独で使用することも、複数のテクノロジーを組み合わせることも可能です。

これらのソリューションの詳細については、「Data Resilience Solutions for IBM i5/OS High Availability Clusters」を参照してください。「Data Resilience Solutions for IBM i5/OS High Availability Clusters」というセクションでは、これらのテクノロジーの属性を詳細に比較しています。

論理複製

論理複製とは、クラスター内のあるノードに含まれているオブジェクトを、同じクラスター内にある他の 1 つ以上のノードにコピーし、すべてのシステムにあるオブジェクトを同一にする処理です。

複製されたリソースにより、アプリケーションおよびそのデータなどのオブジェクトを、クラスター内のあるノードから同じクラスター内にある 1 つ以上の別のノードにコピーできます。この処理は、リソースのリカバリー・ドメインにあるすべてのサーバーのオブジェクトを同一に保ちます。クラスター内の特定のノードに含まれているオブジェクトに変更を加えると、その変更は同じクラスター内にある他のノードにも複製されます。その後、フェイルオーバーまたは切り替えが発生すると、バックアップ・ノードがシームレスにプライマリー・ノードの役割を引き継ぎます。バックアップとして機能する 1 つ以上のサーバーを、リ

カバリー・ドメイン内に定義します。リカバリー・ドメインにあるプライマリー・ノードとして定義されているサーバーで障害が発生し、切り替えまたはフェイルオーバーが開始されると、リカバリー・ドメインでバックアップとして指定されているノードがそのリソースへの 1 次アクセス・ポイントになります。

複製では、ユーザー作成アプリケーションまたはクラスター・ミドルウェア・ビジネス・パートナー作成のソフトウェア・アプリケーションのどちらかを使用することが必要になります。詳しくは、「論理複製の計画」を参照してください。

切り替え可能ディスク

切り替え可能ディスクにより、データおよびアプリケーションなどのリソースで、論理区画の共用バスまたは入出力プールの拡張装置や入出力プロセッサー (IOP) にあるものは、クラスターのプライマリー・ノードとバックアップ・ノードの間で切り替えられるようになります。この機能により、ディスク装置のセットを現在使用しているサーバーが障害を経験し、フェイルオーバーまたは切り替えが発生したとき、それらのディスク装置のセットは、クラスター・リソース・グループのリカバリー・ドメインのバックアップ・ノードとして定義されている 2 次サーバーによりアクセスできるようになります。

クラスターの切り替え可能リソースを利用するには、独立ディスク・プールを使用する必要があります。詳細については、独立ディスク・プールの計画を参照してください。

サイト間ミラーリング

サイト間ミラーリングは、遠隔ミラーリング機能と組み合わせることにより、地理的にかなり離れている複数の場所にあるディスク上にデータをミラーリングすることができます。このテクノロジーを使用すると、物理的なコンポーネント接続の制限を超えて、装置クラスター・リソース・グループの機能を拡張することが可能になります。遠隔ミラーリングには、独立ディスク・プールの実動コピーに加えた変更を、その独立ディスク・プールのミラー・コピーに複製する機能があります。独立ディスク・プールの実動コピーにデータが書き込まれると、オペレーティング・システムはそのデータを別のシステムを介して、独立ディスク・プールの 2 番目のコピーにミラーリングします。このプロセスにより、同じ内容のデータを複数のコピーとして保持できます。

万が一フェイルオーバーまたは切り替えが発生したときには、装置 CRG を介して、バックアップ・ノードがシームレスにプライマリー・ノードの役割を引き継ぎます。バックアップとして機能する 1 つ以上のサーバーを、リカバリー・ドメイン内に定義します。バックアップ・ノードを置く物理的位置は、プライマリー・ノードと同じでも違ってもかまいません。リカバリー・ドメインでプライマリー・ノードとして定義されているサーバーで障害が発生し、切り替えまたはフェイルオーバーが開始されると、リカバリー・ドメインでバックアップとして指定されているノードがそのリソースへの 1 次アクセス・ポイントになり、次いで独立ディスク・プールの実動コピーを所有します。このようにして、切り替え可能リソースに関連した Single Point of Failure (単一障害地点) の問題から保護されます。

表 12. クラスターと共に使用可能なデータ回復テクノロジーの比較 : ご使用のクラスターに最善のソリューションを決定しやすくするため、さまざまなデータ回復テクノロジーの特徴について解説します。

項目	複製	切り替え可能ディスク	サイト間ミラーリング
柔軟性	数十台のサーバー	2 台のサーバー	4 台のサーバー
1 箇所の障害が重大な影響を及ぼす構成 (Single Point of Failure)	なし	ディスク・サブシステム	なし

表 12. クラスターと共に使用可能なデータ回復テクノロジーの比較 (続き)： ご使用のクラスターに最善のソリューションを決定しやすくするため、さまざまなデータ回復テクノロジーの特徴について解説します。

項目	複製	切り替え可能ディスク	サイト間ミラーリング
コスト	• ディスク容量を追加することが必要。 • ビジネス・パートナー複製ソフトウェア。 • ディスクの複写	• 切り替え可能 I/O 拡張タワーまたは IOP • オプション 41	• 独立ディスク・プールのミラー・コピー用の追加ディスク • 任意選択で切り替え可能な I/O 拡張装置 • オプション 41
パフォーマンス	複製のオーバーヘッド	ほとんど低下しない	遠隔ミラーリング・オーバーヘッド
リアルタイム・カバレッジ	ジャーナル・オブジェクト	独立ディスク・プールに含まれているオブジェクト	独立ディスク・プールに含まれているオブジェクト
地理的な分散	パフォーマンスの考慮によって限定	サーバーと拡張装置は HSL OptiConnect ループ (最大 250 メートル) で接続されなければならないので、接続距離により限定	パフォーマンスの考慮に関する制限 (システムに貸される制限はありません。しかし、選択した通信回線での応答時間とスループットのために、現実的には否応なしに何らかの制限が生じる場合があります。)
災害時回復保護	あり	なし	あり
並行バックアップ	あり	なし	なし
セットアップ	• 複製環境。 • 複製するものの決定。	• 独立ディスク・プール環境。 • 独立ディスク・プールの導入。	• 独立ディスク・プール環境(遠隔ミラーリングのセットアップを含む) • 独立ディスク・プールの導入。

関連概念

『クラスター論理複製の計画』
データの複数のコピーは、論理複製を使って保守されます。 データは、クラスターのプライマリー・ノードからリカバリー・ドメインにある指定されたバックアップ・ノードに複製つまりコピーされます。 プライマリー・ノードで障害が発生したときには、指定されたバックアップ・ノードが 1 次アクセス・ポイントを引き継ぐので、データは使用可能なままです。

100 ページの『切り替え可能独立ディスク・プールおよびサイト間ミラーリングの計画 (XSM)』
データの単一コピーが切り替え可能ハードウェア (拡張装置 (タワー) または論理区画環境の IOP のいずれか) で保守されます。

クラスター論理複製の計画:

データの複数のコピーは、論理複製を使って保守されます。 データは、クラスターのプライマリー・ノードからリカバリー・ドメインにある指定されたバックアップ・ノードに複製つまりコピーされます。 プライマリー・ノードで障害が発生したときには、指定されたバックアップ・ノードが 1 次アクセス・ポイントを引き継ぐので、データは使用可能なままです。

複製とは、何かのコピーをリアルタイムで作成することです。つまり、クラスター内のあるノードに含まれているオブジェクトを、同じクラスター内にある他の 1 つ以上のノードにコピーする処理のことです。複

製を行えば、システム上にまったく同じオブジェクトを作成したり保管したりできます。クラスター内の特定のノードに含まれているオブジェクトに変更を加えると、その変更は同じクラスター内にある他のノードにも複製されます。

論理複製に使用するソフトウェア・テクノロジーを決めなければなりません。クラスターで論理複製を行うときは、次のソリューションを使用することができます。

- **IBM ビジネス・パートナーのプロダクト**

  承認を受けた IBM クラスター・ビジネス・パートナー製のデータ複製ソフトウェアを使って、複数のノードに対してオブジェクトを複製することができます。詳しくは、 89 ページの『クラスター・ミドルウェアの IBM ビジネス・パートナーおよび使用可能なクラスタリング・プロダクト』を参照してください。

- **ユーザー作成複製アプリケーション**

  IBM ジャーナル管理は、システムのオブジェクトの活動を記録する手段となります。ジャーナル管理の利点をいかして、論理複製を実現するアプリケーションを作成することができます。ジャーナル管理がどのように動作するかに関する詳細については、 iSeries ジャーナル管理を参照してください。

  関連概念

  ジャーナル管理

論理複製に使用するシステムの決定:

論理複製にどのシステムを使用するかの決定では、いくつかの考慮事項があります。

そのような考慮事項には、次のものがあります。

- パフォーマンス容量
- ディスク容量
- 重要なデータ
- 災害時対策

システムがフェイルオーバーした場合には、プライマリー・システムおよびバックアップ・システムで、実行していたデータおよびアプリケーションは何であったかを、把握しなければなりません。フェイルオーバーが実行される事態を想定すると、最も処理能力の高いシステムに重要なデータを置きたいと思うかもしれません。とはいえ、ディスク・スペースがいっぱいになっても困ります。プライマリー・システムのスペースに余裕がなくなってフェイルオーバーが実行されると、ディスク・スペースの不足のため、バックアップ・システムもフェイルオーバーが実行される可能性が非常に高くなります。洪水、台風、竜巻などの自然災害によってデータ・センターが完全に破壊されてしまうことがないよう、複製の対象となるシステムは別の遠隔地に設置するのが最善です。

切り替え可能独立ディスク・プールおよびサイト間ミラーリングの計画 (XSM):

データの単一コピーが切り替え可能ハードウェア (拡張装置 (タワー) または論理区画環境の IOP のいずれか) で保守されます。

プライマリー・ノードで障害が発生したときには、切り替え可能ハードウェア上のデータへのアクセスは指定されたバックアップ・ノードへ切り替えられます。さらに、サイト間ミラーリング (XSM) 環境で独立ディスク・プールを使用することもできます。その場合、利用および保護を目的として、起点サイトから地理的に遠く離れたシステムにも、独立ディスク・プールのミラー・コピーを保存することができます。

切り替え可能な独立ディスク・プール上に置かれた切り替え可能リソースまたはサイト間ミラーリングの利点をいかす予定の場合は、慎重な計画が必要です。

関連概念

独立ディスク・プールを計画する

クラスター・セキュリティー

クラスタリングをシステムで使用する計画を立てる際に考慮すべき、 セキュリティーの問題について考えます。

ノードをクラスターに追加できるようにする

ノードをクラスターに追加するためには、 クラスターへの追加可能 (ALWADDCLU) ネットワーク属性の値を設定する必要があります。

クラスター・ノードとして設定するサーバーに対して、ネットワーク属性変更 (CHGNETA) コマンドを使用します。ネットワーク属性変更 (CHGNETA) コマンドは、システムのネットワーク属性を変更します。ALWADDCLU ネットワーク属性では、他のシステムにクラスターのノードとしてそのノード追加させるかどうかを指定します。

注: ネットワーク属性 ALWADDCLU を変更するには、*IOSYSCFG 権限がなければなりません。

以下の値のいずれかを指定します。

***SAME**
値は変更されません。システム出荷時の設定値は *NONE です。

***NONE**
他のシステムは、このシステムをクラスターのノードとして追加できません。

***ANY** 他のどんなシステムも、このシステムをクラスターのノードとして追加できます。

***RQSAUT**
クラスター追加要求が認証された後ならば、他のどんなシステムもこのシステムをクラスターのノードとして追加できます。

ALWADDCLU ネットワーク属性を検査すれば、追加しようとしているノードをクラスターの一部とすることが許可されているかどうか、および X.509 ディジタル証明書を使用することによって、クラスター要求の妥当性を検査する必要があるかどうかが分かります。**ディジタル証明書**は、電子的に検査することができる身分証明書のような形式になっています。妥当性検査が必要とされた場合、要求を出しているノードと追加しようとしているノードには、以下のものがシステムにインストールされていなければなりません。

- i5/OS オプション 34 (ディジタル証明書マネージャー)
- 暗号化アクセス・プロバイダー

*RQSAUT を選択した場合は、i5/OS クラスター・セキュリティー・サーバー・アプリケーションの認証局信頼リストを正しく設定する必要があります。このサーバー・アプリケーションの ID は、QIBM_QCST_CLUSTER_SECURITY です。少なくとも、クラスターへの加入を許可するノードについては、認証局を追加してください。

関連概念

ディジタル証明書管理

152 ページの『クラスターの一般的な問題』
ここでは、クラスターで生じる可能性のある最も一般的な問題のいくつかをリストして、 その回避方法および回復方法を示します。

関連資料

ネットワーク属性変更 (CHGNETA) コマンド

クラスター全体への情報の配布

クラスター全体の情報の使用と管理に関するセキュリティー上の意義の考察

情報配布 (QcstDistributeInformation) API を使用すれば、クラスター・リソース・グループのリカバリー・ドメイン内の 1 つのノードから、そのリカバリー・ドメイン内の他のノードにメッセージを送信することができます。これは、出口プログラムの処理で役に立つことがあります。ただし、その情報については、暗号化がないことに注意しなければなりません。セキュリティーが確保されているネットワークを使用している場合を除き、セキュリティーが必要な情報をこのメカニズムで送信することは避けてください。

非永続的データも、クラスター・ハッシュ・テーブル API を使用してクラスター・ノード間で共用したり複製できます。このデータは、非永続記憶装置に保管されます。これは、そのクラスター・ノードがクラスタリングされたハッシュ・テーブルの一部になっている間しかデータを取り出せないことを意味します。これらの API は、クラスタリングされたハッシュ・テーブル・ドメインに定義されているクラスター・ノードからでなければ使用できません。クラスター・ノードは、クラスター内でアクティブになっていなければなりません。

クラスター・メッセージ機能で配布される他の情報も、やはりセキュリティー保護が十分ではありません。これには、低レベルのクラスター・メッセージ機能も含まれます。低レベルの場合は、出口プログラムのデータに変更が加えられたときでも、そのデータを含むメッセージの暗号化が行われません。

関連資料

情報の配布 (QcstDistributeInformation) API

クラスター・ハッシュ・テーブル API

すべてのノードでユーザー・プロファイルを保守する

クラスター内のどのノードでも、ユーザー・プロファイルの保守には 2 通りのメカニズムを使用することができます。

一方のメカニズムは、クラスター内のすべてのノードを通して共用リソースをモニターするためのクラスター管理可能ドメインを作成するためのメカニズムです。クラスター管理可能ドメインは、ユーザー・プロファイルに加えていくつかのタイプのリソースをモニターできるので、すべてのノードを通して共用されているリソースの管理を簡単に行うことができます。そのようなリソースの詳細は、モニター対象リソースを参照してください。クラスター管理ドメインがアクティブになっているときに、ユーザー・プロファイルを更新すると、変更内容は自動的に他のノードに伝搬されます。クラスター管理可能ドメインがアクティブになっていない場合、変更内容は、クラスター管理可能ドメインがアクティブになってから伝搬されます。

注: クラスター内でパスワード同期を活用するユーザー・プロファイルを共有する予定の場合、サーバー・セキュリティー保持 (QRETSVRSEC) システム値を 1 に設定する必要があります。

もう一方のメカニズムでは、管理者は iSeries ナビゲーターの「マネージメント・セントラル」も併用して、複数のシステムおよびシステム・グループを対象に各種機能を実行することができます。このサポートには、オペレーターがクラスター内の複数のシステムにまたがって実行しなければならない、いくつかの一般的なユーザー管理タスクも含まれています。マネージメント・セントラルを使用すれば、システム・グル

ープに対してユーザー・プロファイル機能を実行できます。管理者であれば、ユーザー・プロファイルの作成時に、ターゲット・システムで伝搬後のコマンドが実行されるように設定することもできます。

関連概念

129 ページの『ジョブ構造とユーザー待ち行列』
クラスターを管理するには、ジョブ構造とユーザー待ち行列についての知識が必要です。

10 ページの『クラスター管理ドメイン』
クラスター管理ドメインはクラスター化された環境のノード全体で、 常に保守を必要とするリソースを管理するために使用されます。

ファイアウォールを備えたクラスターの使用に関する考慮事項

ファイアウォールを使用しているネットワーク内でクラスタリングを使用する場合、いくつかの制限事項と要件に配慮する必要があります。

ファイアウォールを備えたクラスタリングを使用する場合、どのノードでも、他のクラスター・ノードとのやりとりのために、アウトバウンド・メッセージの送信と、インバウンド・メッセージの受信を行えるようにする必要があります。各ノード上の各クラスター・アドレスが、他のどのノードのどのクラスター・アドレスとでも通信できるように、ファイアウォールに開口部を設けなければなりません。ネットワーク上を行き来する IP パケットは、多様なトラフィック・タイプをとることがあります。タイプが ICMP のクラスタリングは、ping を使用しますが、UDP と TCP も使用します。ユーザーがファイアウォールを構成するときに、そのようなタイプをベースにトラフィックをフィルターすることができます。クラスタリングの稼働のためには、ICMP、UDP、および TCP のトラフィックをファイアウォールで許可する必要があります。アウトバウンド・トラフィックは任意のポートを通して送信できるのに対して、インバウンド・トラフィックはポート 5550 および 5551 で受信されます。

クラスター構成チェックリスト

クラスター構成チェックリストを完了して、環境が正しく準備されていることを確認してから、クラスターの構成を開始してください。

表 13. クラスターの TCP/IP 構成チェックリスト

TCP/IP 要件	
__	クラスターに組み込むすべてのノードで、 TCP/IP の開始 (STRTCP) コマンドを使用して TCP/IP を開始します。
__	TCP ループバック・アドレス (127.0.0.1) を構成し、状況がアクティブ であることを確認します。クラスター内のすべてのノードで TCP/IP ネットワーク状況の処理 (WRKTCPSTS) コマンドを使用して検査してください。
__	所定のノードのクラスタリングに使用した IP アドレスの状況が*活動中* になっていることを、対象ノードで TCP/IP ネットワーク状況の処理 (WRKTCPSTS) コマンドを使用して確認します。
__	STRTCPSVR *INETD コマンドを使用するか、または iSeries ナビゲーターを使用して次のようなステップを実行することにより、INETD がクラスターのすべてのノードでアクティブになっていることを確認します。 1. iSeries ナビゲーターで、「**ネットワーク**」を展開します。 2. 「**サーバー**」を展開します。 3. 「**TCP/IP**」を展開します。 4. 「**INETD**」を右クリックして、「**開始**」を選択します。 これは、対象ノードのアクティブなジョブ・リストに QTOGINTD (ユーザー QTCP) が存在することによって検査できます。

表 13. クラスターの TCP/IP 構成チェックリスト (続き)

TCP/IP 要件	
__	/QIBM/ProdData/OS400/INETD/inetd.config に指定されている INETD のユーザー・プロファイルが、最低限のものより大きい権限をもっていないことを確認します。このユーザー・プロファイルがもっている権限が最低限のものより大きい場合、クラスター・ノードの開始 (STRCLUNOD) コマンドは失敗します。デフォルトでは、INETD のユーザー・プロファイルとして QUSER が指定されます。
__	クラスター内の各 IP アドレスが、UDP データグラムへの経路指定が可能であり、クラスター内の他の各 IP アドレスにこのデータグラムを送信できることを確認します。ローカル IP アドレスを指定する PING コマンドを使用し、UDP メッセージを指定する TRACEROUTE コマンドを使用します。
__	ポート 5550 と 5551 が他のアプリケーションによって使用されていないことを確認します。これらのポートは IBM クラスタリング用に予約済みです。ポートの使用状況は、TCP/IP ネットワーク状況の処理 (WRKTCPSTS) コマンドを使って表示することができます。 ポート 5550 は、クラスタリングの INETD が開始されると、オープンされて「接続待機」状態になります。

クラスターに切り替え可能装置を使用する場合、以下の要件が満たされていなければなりません。

表 14. クラスター用の回復装置構成チェックリスト

回復装置要件	
__	デバイス・ドメインに収容されるすべてのクラスター・ノードにオプション 41 (HA 切り替え可能リソース) がインストール済みであり、有効なライセンス・キーが存在することを確認します。 iSeries ナビゲーター・クラスター管理インターフェースを使用するためには、このオプションが必要であることに注意してください。
__	iSeries ナビゲーターのディスク管理機能を使用するためには、保守ツール・サーバー (STS) を DST アクセスおよびユーザー・プロファイル付きで構成しなければなりません。詳細については、通信のセットアップを参照してください。
__	1 つのシステム上の複数の論理区画の間で回復装置を切り替えている場合、論理区画の管理のために HMC 以外のものを使用しているのであれば、区画の仮想 OptiConnect を有効にしてください。これは、専用保守ツール (DST) にサインオンして行います。詳細については仮想 OptiConnect を参照してください。 区画の管理にハードウェア管理コンソールを使用している場合には、「OptiConnect」タブで区画プロファイル・プロパティーを変更することによって、切り替え可能構成に属する区画ごとに仮想 OptiConnect を有効にしてください。変更内容を実際に反映させるには、区画プロファイルを活動化する必要があります。
__	HSL OptiConnect ループのタワーが 2 つのシステム間で切り替えられ、システムのうちの 1 つに論理区画がある場合、その区画に対して HSL OptiConnect を使用可能にします。 論理区画の管理に HMC 以外のものを使用している場合、これは専用保守ツール (DST) にサインオンして行います。 区画の管理にハードウェア管理コンソールを使用している場合には、「OptiConnect」タブで区画プロファイル・プロパティーを変更することによって、切り替え可能構成に含まれる区画ごとに HSL OptiConnect を有効にしてください。変更内容を実際に反映させるには、区画プロファイルを活動化する必要があります。
__	複数の論理区画の間で回復装置を切り替えている場合、論理区画の管理のために HMC 以外のものを使用しているのであれば、それらの区画の間でバスを共有するように構成するか、または入出力プールを構成しなければなりません。バスを 1 つの区画で「共用バスを所有 (own bus shared)」として構成し、装置切り替えに参加する他のすべての区画では「共用バスを使用 (use bus shared)」として構成する必要があります。 区画の管理にハードウェア管理コンソールを使用している場合には、入出力プロセッサー、入出力アダプター、および接続されているすべてのリソースから成る入出力プールを構成することによって、1 つの独立ディスク・プールを複数の区画の間で切り替え可能にしてください。区画ごとに、その入出力プールにアクセス可能でなければなりません。詳細は、ハードウェアを切り替え可能にする (Make your hardware switchable) を参照してください。 切り替え可能なデバイスの設備計画の要件に関する詳細は、設備計画の要件 (Physical planning requirements) を参照してください。

表 14. クラスター用の回復装置構成チェックリスト (続き)

回復装置要件	
__	HSL ループにあるタワーを異なる 2 つのシステム間で切り替えるときには、タワーは切り替え可能として構成されていなければなりません。詳細については、 ハードウェアを切り替え可能にする (Make your hardware switchable) を参照してください。
__	既存の HSL ループにタワーを追加するときには、その同じループにあるすべてのサーバーを再始動します。
__	通信パスの最大伝送単位 (MTU) は、調整できるクラスター通信パラメーター、メッセージ・フラグメント・サイズより大きくなければなりません。クラスター IP アドレスの MTU は、対象ノードで TCP/IP ネットワーク状況の処理 (WRKTCPSTS) コマンドを使用して確認できます。 MTU は、通信パス全体にわたって、すべてのステップで検査されます。通信パスの MTU の値を増やすよりも、クラスターを作成するときにメッセージ・フラグメント・サイズの値を減らすほうが容易に行えると思われます。メッセージ・フラグメント・サイズの詳細については、 調整可能なクラスター通信パラメーターを参照してください。クラスター・リソース・サービス情報検索 (QcstRetrieveCRSInfo) API を使用して調整パラメーターの現行設定値を表示し、 クラスター・リソース・サービス変更 (QcstChgClusterResourceServices) API を使用して設定値を変更することができます。

表 15. クラスターのセキュリティー構成チェックリスト

セキュリティー要件	
__	リモート・ノードを開始するには、ターゲット・ノードで ALWADDCLU (クラスターへの追加可能) ネットワーク属性が適切に設定されていなければなりません。これは、環境に応じて *ANY または *RQSAUT に設定されていなければなりません。*RQSAUT に設定されている場合、i5/OS オプション 34 (デジタル証明書マネージャー) および Cryptographic Access Provided プロダクトがインストールされていなければなりません。ALWADDCLU ネットワーク属性の設定値に関する詳細については、 ノードをクラスターに追加できるようにするを参照してください。
__	/QIBM/ProdData/OS400/INETD/inetd.config に指定されている INETD のユーザー・プロファイルの状況を使用可能にします。このユーザー・プロファイルには、*SECADM または *ALLOBJ 特殊権限を付与することはできません。デフォルトでは、INETD のユーザー・プロファイルとして QUSER が指定されます。
__	クラスター・リソース・サービス API を呼び出すユーザー・プロファイルが、すべてのクラスター・ノードに存在しかつ *IOSYSCFG 権限を持っていることを確認します。
__	クラスター・リソース・グループ (CRG) の出口プログラムを実行させるユーザー・プロファイルが、すべてのリカバリー・ドメイン・ノードに存在していることを確認します。

表 16. クラスターのジョブ構成チェックリスト

ジョブ考慮事項	
__	クラスター・リソース・サービス API により、要求を処理するようにジョブを投入できます。ジョブは、クラスター・リソース・グループを作成するときに指定された出口プログラムを実行するユーザー・プロファイルや、API (回復装置 CRG 内の装置のみをオンに構成変更する場合) を要求するユーザー・プロファイルのどちらでも実行されます。ユーザー・プロファイルに関連したジョブ待ち行列を保守するサブシステムが、ジョブ待ち行列から実行できるジョブの数を *NOMAX で構成されていることを確認してください。
__	ジョブは、CRG 用に定義されたユーザー・プロファイルから取得されたジョブ記述により指定されたジョブ待ち行列に投入されます。デフォルトのジョブ記述では、ジョブは QBATCH ジョブ待ち行列に送られます。このジョブ待ち行列は多くのユーザーに使用されるため、出口プログラム・ジョブはタイミングよく実行されないかもしれません。固有ユーザー待ち行列を使用した固有ジョブ記述を考慮するべきです。

表 16. クラスターのジョブ構成チェックリスト (続き)

ジョブ考慮事項	
__	出口プログラム・ジョブは実行されると、ジョブ記述からのルーティング・データを使用して、使用する主記憶域プールと実行時属性を選択します。デフォルト値では、他のバッチ・ジョブのあるプールで実行されているジョブの実行優先度は 50 になります。これらは、出口プログラム・ジョブの望ましいパフォーマンスをもたらしません。出口プログラム・ジョブを開始するサブシステム (固有ジョブ待ち行列を使用する同じサブシステム) は、同じサブシステムまたは別のサブシステムで開始された他のジョブで使用されることのないプールを出口プログラム・ジョブに割り当てるべきです。さらに、出口プログラム・ジョブには優先度 15 を割り当て、ほとんどすべての他のユーザー・ジョブよりも先に実行されるようにしなければなりません。
__	QMLTTHDACN システム値は、1 または 2 に設定する必要があります。

クラスターの構成と管理で使用可能ないくつかのソフトウェア・ソリューションがあります。これらのソリューションの 1 つは、iSeries ナビゲーター・クラスター管理です。 iSeries ナビゲーターを使用する場合は、以下の要件が満たされていなければなりません。

表 17. クラスターの iSeries ナビゲーター構成チェックリスト

iSeries ナビゲーター・クラスター管理の考慮事項	
__	アクティブ・ドメインに収容されるすべてのクラスター・ノード上にオプション 41 (i5/OS - HA 切り替え可能リソース) がインストール済みであり、有効なライセンス・キーが存在しなければなりません。
__	すべてのホスト・サーバーが、STRHOSTSVR (ホスト・サーバーの開始) コマンド: STRHOSTSVR SERVER(*ALL) を使用して開始されることを確認します。
__	マネージメント・セントラル・サーバーは、STRTCPSVR (TCP/IP サーバーの開始) コマンド: STRTCPSVR SERVER(*MGTC) を使用して開始されることを確認します。

関連概念

80 ページの『iSeries ナビゲーター・クラスター管理』
IBM は、iSeries ナビゲーターから利用できて、オプション 41 (i5/OS - HA 切り替え可能リソース) を通してアクセスできるシンプル・クラスター管理インターフェースを提供しています。

『INETD サーバー』
ノードを追加または開始したり、区画のマージ処理を行うためには、 インターネット・デーモン (INETD) サーバーを開始しなければなりません。

107 ページの『調整可能なクラスター通信パラメーター』
クラスター・リソース・サービス変更 (QcstChgClusterResourceServices) API を使用すれば、 クラスター・トポロジー・サービスおよびクラスター通信のパフォーマンス・パラメーターと、 構成パラメーターの一部を、 クラスタリングが行われる多数の固有なアプリケーション環境およびネットワーク環境により良く適合するように調整することができます。 この API は、 クラスター・バージョン 2 またはそれ以降のバージョンで実行されるあらゆるクラスターで使用できます。

関連資料

110 ページの『クラスター管理ドメイン・チェックリスト』
クラスター管理ドメインを作成する前に完了しなければならないすべての前提条件が網羅されています。

INETD サーバー

ノードを追加または開始したり、区画のマージ処理を行うためには、 インターネット・デーモン (INETD) サーバーを開始しなければなりません。

INETD サーバーをクラスター内で常時実行することをお勧めします。

iSeries ナビゲーターを使用する場合

これを実行するには、オプション 41 (i5/OS- HA 切り替え可能リソース) がインストールされていて、ライセンス交付を受けている必要があります。

INETD サーバーを開始するには、以下の手順で行います。

1. iSeries ナビゲーターで、「**ネットワーク**」を展開します。
2. 「**サーバー**」を展開します。
3. 「**TCP/IP**」を展開します。
4. 「**INETD**」を右クリックして、「**開始**」を選択します。

CL コマンドおよび API を使用する場合

INETD サーバーは、STRTCPSVR (TCP/IP サーバー開始) コマンドに *INETD パラメーターを指定して開始することもできます。 INETD サーバーが開始すると、対象ノードのアクティブなジョブのリストに QTOGINTD (ユーザー QTCP) が現れます。

関連概念

152 ページの『クラスターの一般的な問題』
ここでは、クラスターで生じる可能性のある最も一般的な問題のいくつかをリストして、 その回避方法および回復方法を示します。

関連資料

TCP/IPサーバーの開始 (STRTCPSVR)

調整可能なクラスター通信パラメーター

クラスター・リソース・サービス変更 (QcstChgClusterResourceServices) API を使用すれば、 クラスター・トポロジー・サービスおよびクラスター通信のパフォーマンス・パラメーターと、 構成パラメーターの一部を、 クラスタリングが行われる多数の固有なアプリケーション環境およびネットワーク環境により良く適合するように調整することができます。 この API は、 クラスター・バージョン 2 またはそれ以降のバージョンで実行されるあらゆるクラスターで使用できます。

クラスター構成調整の変更 (CHGCLUCFG) コマンド は基本レベルのチューニングに対応するのに対して、QcstChgClusterResourceServices API は、基本レベルと拡張レベルのどちらのチューニングにも対応します。

QcstChgClusterResourceServices API と CHGCLUCFG コマンドは、クラスターのパフォーマンスと構成を調整するために使用できます。これらの API とコマンドは、基本レベルの調整をサポートしています。このレベルの調整では、タイムアウトとメッセージ送信間隔について、高、中、低のレベルの値がそれぞれ事前に定義されているので、クラスターをその事前定義の値に調整できます。拡張レベルの調整が必要な場合は、通常 IBM サポート担当員の援助を受けて API を使用し、個々のパラメーターを事前定義値の範囲で調整できます。個々のパラメーターに不適切な変更を行うと、クラスターのパフォーマンスが低下することがあります。

いつ、どのようにクラスター・パラメーターを調整するか?

CHGCLUCFG コマンドと QcstChgClusterResourceServices API には、詳細を理解していなくてもクラスターのパフォーマンスと構成のパラメーターを設定できるよう、高速パスが用意されています。この基本レベルの調整は主に、ハートビート感度およびクラスター・メッセージ・タイムアウト値に影響します。基本レベルの調整サポートの有効値は以下のとおりです。

1 (高タイムアウト値/より低頻度のハートビート)

2 (デフォルト値)
クラスター通信のパフォーマンス・パフォーマンスおよび構成パラメーターに、通常のデフォルト値を使用します。この設定値を使用すると、すべてのパラメーターをオリジナルのデフォルト値に戻すことができます。

3 (低タイムアウト値/より高頻度のハートビート)
ハートビート間隔を小さくし、各種のメッセージ・タイムアウト値を小さくするように、クラスター通信に調整を行います。ハートビートをより頻繁にし、タイムアウト値を短くすると、クラスターは、通信障害に対する対応が早く (感度が高く) なります。

ノードの区画化が発生するようなハートビート障害の場合について、結果的に得られる応答時間の例を次の表に示しています。

	1 (より低感度)			**2 (デフォルト)**			**3 (より高感度)**		
	ハートビート問題の検出	分析	合計	ハートビート問題の検出	分析	合計	ハートビート問題の検出	分析	合計
単一サブネット	00:24	01:02	01:26	00:12	00:30	00:42	00:04	00:14	00:18
複数サブネット	00:24	08:30	08:54	00:12	04:14	04:26	00:04	02:02	02:06

注: 時刻は、分:秒のフォーマットで指定します。

通常のネットワーク負荷および使用する特定の物理メディアに応じて、クラスター管理者は、ハートビート感度およびメッセージ・タイムアウトのレベルの調整を行う場合があります。高速で信頼性の高いトランスポート、たとえば、共通 OptiConnect バス上にクラスターのすべてのシステムをもつ OptiConnect などでは、迅速な検出によってより早くフェイルオーバーを実行させるため、さらに高感度の環境を確立したいことがあります。この場合、オプション 3 を選択します。負荷が大きい 10Mbs のイーサネット・バスで実行しているときに、デフォルトが設定されているために、ネットワーク・ピーク・ロードだけが原因となって区画化が時折起きる場合は、オプション 1 を選択します。それによって、ピーク・ロードに対するクラスタリングの感度を軽減することができます。

クラスター・リソース・サービス変更 API を使用すれば、個々のネットワーク環境が持つ固有の状態に合わせて、特定のパラメーターを個別に調整することも可能です。たとえば、OptiConnect バスに共通なすべてのノードをもつクラスターをもう一度考えてみましょう。 Message Fragment Size パラメーターを最大 32,500 バイトに設定することにより、OptiConnect 最大伝送単位 (MTU) サイズにより合致させると、 デフォルトの 1,464 バイトの場合よりもクラスター・メッセージのパフォーマンスを大幅に拡張することができます。これによって、大きなメッセージのフラグメント化および再組み立てのオーバーヘッドが削減されます。当然、この利点は、クラスター・アプリケーションと、それらのアプリケーションの結果行われるクラスター・メッセージングの使用量に応じて変わってきます。他のパラメーターは API 資料に定義されており、それらのパラメーターを使用して、クラスター・メッセージングのパフォーマンスを調整したり、区画化に対するクラスターの感度を変更したりすることができます。

関連概念

127 ページの『クラスター・パフォーマンスの調整』
それぞれの通信環境には大きな違いがあり得るので、クラスター通信に影響を与える変数を各環境に合わせて最適な値に調整するための機能が用意されています。

クラスター構成解除チェックリスト

クラスターまたは CRG を削除する必要がある場合、各種のクラスター・コンポーネントを徹底的に除去して、必ず完全に構成解除されるようにしなければなりません。

表 18. クラスターの独立ディスク・プールの構成解除チェックリスト

	独立ディスク・プールの要件
__	独立ディスク・プール・グループのサブセットの除去、または切り替え可能装置内の最後の独立ディスク・プールの除去を行う予定の場合、まず CRG を終了する必要があります。 クラスター・リソース・グループの終了 (ENDCRG)コマンドを使用します。
__	クラスターに属している独立ディスク・プールを削除したい場合、まず、装置クラスター・リソース・グループ (CRG) とも呼ばれる切り替え可能装置から、ディスク・プールの構成オブジェクトを除去するよう強くお勧めします。ディスク・プールの構成を切り替え可能装置から削除するには、次のステップを行います。 ディスク・プールの構成を切り替え可能装置から削除するステップは次のとおりです。 1. iSeries ナビゲーターで、「**マネージメント・セントラル**」 → 「**クラスター**」を展開します。 2. *切り替え可能装置を収容しているクラスターの名前* → 「**切り替え可能装置**」を拡張表示します。 3. 切り替え可能装置の名前をクリックします。 4. iSeries ナビゲーターの右側のペインで、ディスク・プールを右マウス・ボタンでクリックし、「**削除**」を選択します。 また、CRG 装置項目の除去 (RMVCRGDEVE) コマンドを使って、 独立ディスク・プールの構成オブジェクトを CRG から除去することもできます。
__	独立ディスク・プールの構成オブジェクトをクラスター切り替え可能装置から除去し終わってから、独立ディスク・プールを削除することができます。
__	以下のタスクを実行して、独立ディスク・プールの装置記述を削除します。 1. コマンド行インターフェースで WRKDEVD DEVD(*ASP) と入力し、Enter キーを押します。 2. ページ送りして、削除したい独立ディスク・プールの装置記述を見つけ出します。 3. 装置記述の名前のそばのオプション 4 (「削除」) を選択し、Enter キーを押します。

表 19. クラスターのクラスター・リソース・グループの構成解除のチェックリスト

	クラスター・リソース・グループの要件
__	以下のタスクのいずれかを実行して、クラスター・リソース・グループを削除します。 1. ノード上でクラスタリングがアクティブになっていない場合、コマンド行インターフェースに DLTCRG CRG(CRGNAME) と入力します。 CRGNAME は、削除しようとしている CRG の名前です。 Enter を押します。 2. ノード上でクラスタリングがアクティブになっている場合、コマンド行インターフェースに DLTCRGCLU CLUSTER(CLUSTERNAME) CRG(CRGNAME) と入力します。 CLUSTERNAME は、クラスターの名前です。 CRGNAME は、削除しようとしている CRG の名前です。 Enter を押します。

クラスター管理可能ドメインの計画

クラスター管理可能ドメインの場合、クラスター管理可能ドメイン内のノードで共用されるリソースを管理するための計画をたてる必要があります。

クラスター管理可能ドメインを作成すると、そのドメインを表わす対等 CRG が自動的に作成されます。クラスター管理可能ドメインを管理するには、API、CL コマンド、およびiSeries ナビゲーターを使用します。

クラスター管理者は、クラスター管理可能ドメインを作成してから、各ノードによって共用されるモニター対象リソースを追加することができます。 i5/OS クラスターには、モニター対象リソース項目 (MRE) というシステム・リソースのリストが用意されていて、クラスター管理可能ドメイン内の各ノードは、そのリストを共用することができます。モニターできるシステム・リソースの総合リストは、モニター対象リソースを参照してください。

クラスター管理可能ドメインを設計するときは、次のような設問に対する答を出す必要があります。

どのリソースを共用するか。
どのシステム・リソースを共用する必要があるかを判別する必要があります。そのようなリソースのそれぞれに属性を選択し、各ノードによって共用される対象をカスタマイズすることができます。複数のノード上で実行されるアプリケーションは、正しく稼働するのに特定の環境変数を必要とすることがあります。また、いくつかのノード上に散在するデータの場合も、特定のユーザー・プロファイルへのアクセスが必要になることがあります。どのリソースを共用するかを決めるときは、事前にアプリケーションとデータの操作上の要件に注意を払う必要があります。

クラスター管理可能ドメインにどのノードを組み込むか。
クラスター内のどのノードがクラスター管理可能ドメインによって管理されるかを決める必要があります。複数のクラスター管理可能ドメインにノードを収容することはできません。たとえば、4 つのノードがクラスター内にあるとします (ノード A、ノード B、ノード C、およびノード D)。ノード A および B を 1 つのクラスター管理可能ドメイン内に置き、ノード C と D を別のドメインに置くことができます。ただし、ノード B および C を別のクラスター管理可能ドメイン内に置くことはできません。

クラスター管理可能ドメインの命名規則をどうするか。
クラスター環境の複雑さとサイズによっては、対等 CRG とクラスター管理可能ドメインに対して標準命名規則を設定したほうがよい場合があります。クラスター管理可能ドメインを表わすために対等 CRG を作成した場合、クラスター内のリソースをモニターするものからその対等 CRG を差異化するのが得策です。たとえば、クラスター管理可能ドメインを表す対等 CRG を *ADMDMN1* や *ADMDMN2* などと命名し、他の対等 CRG を *PEER1* と命名することができます。また、クラスター・リソース・グループ情報のリスト (QcstListClusterResourceGroupIn) API を使って、対等 CRG がクラスター管理可能ドメインとして使用されているかどうかを判別することができます。

クラスター管理ドメイン・チェックリスト

クラスター管理ドメインを作成する前に完了しなければならないすべての前提条件が網羅されています。

表 20. クラスター管理ドメイン・チェックリスト

クラスター管理ドメインの要件	
__	クラスターが構成されていることを確認する。「クラスター構成チェックリスト」を参照してください。
__	クラスター内でパスワード同期を使用するユーザー・プロファイルをモニターする場合は、「サーバー機密保護の保存」(QRETSVRSEC) システム値を 1 に設定する必要があります。
__	クラスター管理ドメインにリソースを追加するには、クラスター管理ドメイン内のすべてのノードが活動化されていること、それらのノードがグループのメンバーであること、区画化されていないことが必須になります。

クラスターの構成

クラスターの構成方法について理解できます。

IBM および IBM クラスター・ミドルウェアのビジネス・パートナーはチームとして互いに協力しあいながら、クラスター管理用のグラフィカル・ユーザー・インターフェース (GUI) を伴った最先端のクラスター・リソース・サービス機能を提供してきました。i5/OS クラスター・リソース・サービスにより、クラスター・トポロジーの保守、ハートビートの実行、およびクラスター構成とクラスター・リソース・グループの作成や管理を可能にする統合サービスが提供されます。またクラスター・リソース・サービスは、クラスター内の各ノードのトラックを保持する信頼メッセージ機能を提供し、全ノードがクラスター・リソースに関する整合性の取れた情報を有するようにします。それ以外に、クラスター・リソース・サービスからは、一連の制御言語 (CL) コマンドおよびアプリケーション・プログラム・インターフェース (API) が提供され、さらに、iSeries アプリケーション・プロバイダーまたはカスタマーが、アプリケーションの可用性を拡張するために使用できる諸機能も提供されます。また、iSeries ナビゲーター・クラスター管理機能または IBM ビジネス・パートナーのクラスター・ミドルウェア・プロダクトから提供されるグラフィカル・ユーザー・インターフェース・ソリューションを通して、クラスター・リソース・サービス機能にアクセスすることもできます。

開始

クラスターを構成するには、以下の手順で行います。

1. **ソフトウェア・ソリューションを選択します。**

   クラスターの構成および管理用のオプションについて十分に考察するには、 80 ページの『クラスターの構成および管理用のソリューション』を参照してください。

2. **ハードウェア、ソフトウェアおよび通信に関する要件を満たします。**

   クラスターの計画にあるクラスター要件を検討してください。

3. **クラスター用のネットワークおよびサーバー環境をセットアップします。**

   103 ページの『クラスター構成チェックリスト』を使用して、ご使用の環境でクラスターを構成する準備が整っていることを確認してください。

4. **クラスターを構成します。**

関連概念

169 ページの『クラスター・サポートについての問い合わせ先』
クラスターの問題に関して IBM に問い合わせる必要がある場合は、このトピックを参照してください。

クラスターの作成

クラスターを作成および構成するには、クラスターに少なくとも 1 つのノードを組み込むことが必要で、そのクラスター内に配置されることになる最低 1 つのノードに対するアクセス権を有していなければなりません。

| 指定されるノードが 1 つだけの場合、そのノードは現在アクセス中のサーバーでなければなりません。クラスターの作成要件の総合リストは、 103 ページの『クラスター構成チェックリスト』を参照してください。

クラスター内で切り替え可能な装置を使用する場合には、切り替え可能な装置を用いないクラスターにはない、追加の要件があります。切り替え可能な装置が組み込まれているクラスター環境をセットアップするには、クラスターの競合を避けるために注意が必要です。切り替え可能な装置を使うクラスターを構成する手順については、切り替え可能な独立ディスク・プールの作成を参照してください。

iSeries ナビゲーターを使用する場合

これを行うためには、オプション 41 (HA 切り替え可能リソース) がインストールされ、ライセンス交付を受けていることが必要です。

iSeries ナビゲーター・クラスター管理は、1 つか 2 つのノードからなる単純なクラスターを構成して開始するまでのステップを手引きするウィザードを提供します。 1 つまたは 2 つのノードからなるクラスターを作成すると、そのクラスターにノードを追加できます。 iSeries ナビゲーターで作成し管理するクラスターは、4 つのノードを含めることができます。このウィザードは、組み込むサーバーを指定して、クラスター・リソース・グループを作成するステップをガイドします。単純なクラスターを作成する場合、ノードの 1 つはクラスターを作成するサーバーでなければなりません。

iSeries ナビゲーターの「新規クラスター」ウィザードを使用して単純なクラスターを作成するには、以下の手順で行います。

1. iSeries ナビゲーターで、「**マネージメント・セントラル**」を展開します。
2. 「**クラスター**」を右マウス・ボタン・クリックしてから、「**新規クラスター**」を選択します。
3. ウィザードの指示に従って、クラスターを作成します。

クラスターを作成したなら、以下の事柄を確認してください。

1. クラスターに組み込みたいすべてのノードを追加します。 iSeries ナビゲーターで作成し管理するクラスターの場合には、最高 4 つのノードを追加できます。
2. 要望のノードをデバイス・ドメインに追加します (切り替え可能なハードウェア・グループおよび独立ディスク・プールで使用するため)。
3. 切り替え可能リソースを作成して開始します (切り替え可能なデバイス、切り替え可能なアプリケーション、および切り替え可能なデータ)。

iSeries ナビゲーターのオンライン・ヘルプには、こうしたタスクを完了するための段階的な手順が解説されています。

CL コマンドおよび API を使用する場合

CL コマンドまたは API を使用してクラスターを作成することもできます。

1. **クラスターを構成する。**
   クラスターの作成 (CRTCLU) コマンド
   クラスターの作成 (QcstCreateCluster) API
2. **アクティブなノードからクラスターにノードを追加する。**
   クラスター・ノード項目の追加 (ADDCLUNODE) コマンド
   クラスター・ノード項目の追加 (QcstAddClusterNodeEntry) API
3. **クラスター・ノードの開始。**
   クラスター・ノードの開始 (STRCLUNOD) コマンド
   クラスター・ノード開始 (QcstStartClusterNode) API
4. **デバイス・ドメインを定義する。** 切り替え可能装置の使用を計画している場合、希望するノードをデバイス・ドメインに組み込む必要があります。
   デバイス・ドメイン項目の追加 (ADDDEVDMNE) コマンド
   デバイス・ドメイン項目の追加 (QcstAddDeviceDomainEntry) API
5. **クラスター・リソース・グループ (CRG) を作成する。**
   クラスター・リソース・グループの作成 (CRTCRG) コマンド

クラスター・リソース・グループの作成 (QcstCreateClusterResourceGroup) API

6. **クラスター・リソース・グループ (CRG) を開始する。**
   クラスター・リソース・グループの開始 (STRCRG) コマンド
   クラスター・リソース・グループの開始 (QcstStartClusterResourceGroup) API

クラスターの管理

このトピックには、クラスター管理に関係するいくつかのタスクをカバーする情 報が含まれています。

クラスターの管理にどのタイプのインターフェースを使用するかまだ考慮していない場合、先に進む前にクラスターの構成および管理用のソリューションをご覧ください。

いったん構成したクラスターに行えるいくつかの変更は以下のとおりです。

クラスター・タスク

- クラスターへのノードの追加
- クラスターからのノードの除去
- クラスター・ノードの開始
- クラスター・ノードの終了
- 最新レベルへのクラスターのクラスター・バージョンの調整
- クラスターの削除
- クラスター・ノードの変更

クラスター・リソース・グループ・タスク

- 新規クラスター・リソース・グループの作成
- 既存クラスター・リソース・グループの削除
- クラスター・リソース・グループの開始
- クラスター・リソース・グループへのノードの追加
- クラスター・リソース・グループからのノードの除去
- クラスター・リソース・グループの終了
- クラスター・リソース・グループのリカバリー・ドメインの変更
- 切り替えの実行
- デバイス・ドメインへのノードの追加
- デバイス・ドメインからのノードの除去

このトピックには、クラスター構成の保管を行うのに役立つ情報もあります。クラスター・リソース・サービス・ジョブがどのように構造化され、クラスター API がユーザー待ち行列をどのように使用するかに関しての情報を読むことができます。クラスター・ジョブの終了を正しく行う方法およびクラスター状況のモニター方法について読んでください。また、どのように信頼メッセージ機能とハートビート・モニターがクラスターの状況を更新し続けるのかについても学んでください。

クラスター管理可能ドメインのタスク

- クラスター管理可能ドメインの作成
- モニター対象リソースの追加
- クラスター管理可能ドメインの削除

関連概念

32 ページの『メッセージング機能』
クラスター・リソース・サービスのメッセージング機能は、クラスター内の各ノードに注意を払い、クラスター・リソースの状態に関する整合した情報を確実にすべてのノードが保持するようにします。

30 ページの『ハートビート・モニター』
ハートビート・モニターはクラスター・リソース・サービス機能の 1 つで、クラスター内のすべてのノードからクラスター内の自分以外のすべてのノードにシグナルを送信して自分がアクティブであることを伝達してすべてのノードがアクティブであることを確証するものです。

クラスターへのノードの追加

iSeries ナビゲーターまたはコマンドを使用して、クラスターにノードを追加できます。

iSeries ナビゲーターを使用する場合

これを実行するためには、オプション 41 (HA 切り替え可能リソース) がインストールされ、ライセンス交付を受けていることが必要です。

iSeries ナビゲーターがサポートするシンプル・クラスターは、最大 4 ノードで構成できます。すでにクラスターにノードが 4 つ存在する場合、「**ノードの追加**」 オプションは使用不可になります。クラスタリングで 5 ノード以上必要な場合、128 ノードまでサポートするクラスター・コマンドまたは API を使用するか、クラスター・ミドルウェアである IBM ビジネス・パートナー・プロダクトを使用しなければなりません。

既存クラスターへノードを追加するには、以下の手順で行います。

1. iSeries ナビゲーターで、「マネージメント・セントラル」を展開します。
2. 「**クラスター**」を展開します。
3. ノードを追加したいクラスターを展開します。
4. 「**ノード**」を右マウス・ボタンでクリックし、「**ノードの追加...**」を選択します。

クラスター・コマンドおよび API を使用する場合

以下のものを使用して、クラスターにノードを追加できます。

- クラスター・ノード項目の追加 (ADDCLUNODE) コマンド
- クラスター・ノード項目の追加 (QcstAddClusterNodeEntry) API

関連概念

82 ページの『クラスター・コマンドおよび API』
i5/OS クラスター・リソース・サービスからは、一連の制御言語 (CL) コマンドおよびアプリケーション・プログラム・インターフェース (API) が提供され、さらに、iSeries アプリケーション・プロバイダーまたはカスタマーが、アプリケーションの可用性を拡張するために使用できる諸機能も提供されます。

89 ページの『クラスター・ミドルウェアの IBM ビジネス・パートナーおよび使用可能なクラスタリング・プロダクト』
クラスタリングに不可欠な論理複製機能を提供するプロダクトや、クラスターを簡単に作成および管理するためのプロダクトは、 IBM クラスター・ミドルウェアのビジネス・パートナーから購入することができます。

クラスター・ノードの開始

クラスター・ノードを開始すると、クラスター内のノード上のクラスター・リソース・サービスも開始されます。 クラスター・バージョン 3 から、クラスターにアクティブ・ノードがあれば、あるノードが自分自身を開始して現在アクティブなクラスターに再結合することができるようになりました。

iSeries ナビゲーターを使用する場合

これを実行するためには、オプション 41 (HA 切り替え可能リソース) がインストールされ、ライセンス交付を受けていることが必要です。

指定されたノードでクラスター・リソース・サービスが正常に開始されると、ノードの状況は*開始済み* に設定されます。

ノードでクラスタリングを開始するには、以下の手順で行います。

1. iSeries ナビゲーターで、「**マネージメント・セントラル**」を展開します。
2. 「**クラスター**」を展開します。
3. クラスタリングを開始するノードが中に入っているクラスターを展開します。
4. 「**ノード**」をクリックします。
5. クラスターを開始したいノードを右マウス・ボタンでクリックし、「**クラスター**」 → 「**開始**」を選択します。

クラスター

CL コマンドおよび API を使用する場合

ノードを開始するために CL コマンドまたは API を使用することもできます。指定されたノードでクラスター・リソース・サービスが正常に開始されると、ノードの状況はアクティブ に設定されます。

- クラスター・ノードの開始 (STRCLUNOD) コマンド
- クラスター・ノード開始 (QcstStartClusterNode) API

関連タスク

128 ページの『クラスター・ジョブの終了』
クラスター・ジョブは、決して直接終了しないでください。

159 ページの『クラスター・ジョブ障害からの回復』
クラスター・リソース・サービス・ジョブの障害は、通常、何か他の問題があることを示しています。

クラスター・ノードの終了

ノードを停止または終了すると、そのノード上のクラスター・リソース・サービスが停止します。

iSeries ナビゲーターを使用する場合

これを実行するためには、オプション 41 (HA 切り替え可能リソース) がインストールされ、ライセンス交付を受けていることが必要です。

指定したノード上でクラスター・リソース・サービスが正常に停止すると、ノードの状況が*停止* に設定されます。

ノード上のクラスタリングを終了するには、以下の手順で行います。

1. iSeries ナビゲーターで、「**マネージメント・セントラル**」を展開します。

2. 「**クラスター**」を展開します。
3. クラスタリングを停止するノードが中に入っているクラスターを展開します。
4. 「**ノード**」をクリックします。
5. クラスターを終了したいノードを右マウス・ボタンでクリックし、「**クラスター**」 → 「**停止**」を選択します。

CL コマンドおよび API を使用する場合

CL コマンドまたは API を使用してノードを終了することもできます。指定したノード上でクラスター・リソース・サービスが正常に終了すると、ノードの状況が*非アクティブ* に設定されます。

- クラスター・ノードの終了 (ENDCLUNOD) コマンド
- クラスター・ノード終了 (QcstEndClusterNode) API

関連タスク

128 ページの『クラスター・ジョブの終了』
クラスター・ジョブは、決して直接終了しないでください。

159 ページの『クラスター・ジョブ障害からの回復』
クラスター・リソース・サービス・ジョブの障害は、通常、何か他の問題があることを示しています。

クラスターのクラスター・バージョンの調整

クラスター・バージョンは、クラスター内のすべてのノードがアクティブに相互通信するレベルを定義します。

1 つのクラスターの中に、複数のリリース・レベルのシステムを組み込み、使用可能な通信プロトコルのレベルを判別することによって、完全な相互運用を実現するための技法が、このクラスター・バージョン設定です。

クラスター・バージョンを変更するには、クラスター内のすべてのノードは同じ潜在バージョンでなければなりません。そうすると、クラスター・バージョンを潜在バージョンと一致するように変更することができます。これにより、新機能が使用できるようになります。バージョンは 1 つずつしか増やすことができません。バージョンを減らすには、クラスターを削除した後、低いバージョンで再作成する以外にありません。現行のクラスター・バージョンは、クラスター内で定義されている最初のノードを基準にして初期設定されます。それ以降クラスターに追加されるノードは、現行のクラスター・バージョンと同等またはその次のレベルのバージョンにならなければなりません。そうでないと、クラスターに追加できません。

iSeries ナビゲーターを使用する場合

これを実行するためには、オプション 41 (HA 切り替え可能リソース) がインストールされ、ライセンス交付を受けていることが必要です。

クラスターのクラスター・バージョンを調整するには、以下の手順で行います。

1. iSeries ナビゲーターで、「**マネージメント・セントラル**」を展開します。
2. 「**クラスター**」を展開します。
3. クラスターを右クリックして、「**プロパティー**」を選択します。
4. クラスター・バージョンを希望の設定に変更します。

クラスター・コマンドおよび API を使用する場合

以下を使用してクラスターのクラスター・バージョンを調整することもできます。

- クラスター・バージョンの変更 (CHGCLUVER) コマンド
- クラスター・バージョン調整 (QcstAdjustClusterVersion) API

関連概念

15 ページの『クラスター・バージョン』
クラスター・バージョンとは、 クラスターで実行できる機能のレベルを表す用語です。

152 ページの『クラスターの一般的な問題』
ここでは、クラスターで生じる可能性のある最も一般的な問題のいくつかをリストして、 その回避方法および回復方法を示します。

関連タスク

『クラスターの削除』
クラスターを削除すると、クラスター・リソース・サービスはアクティブなすべてのクラスター・ノード上で終了し、クラスターから除去されます。

クラスターの削除

クラスターを削除すると、クラスター・リソース・サービスはアクティブなすべてのクラスター・ノード上で終了し、クラスターから除去されます。

| **重要:** クラスター内に独立ディスク・プールがある場合は、クラスターを削除する前にデバイス・ドメイン
| 項目の削除 (RMVDEVDMNE) コマンドを使用して、まずデバイス・ドメインから各ノードを削除し
| ます。

iSeries ナビゲーターを使用する場合

これを実行するためには、オプション 41 (HA 切り替え可能リソース) がインストールされ、ライセンス交付を受けていることが必要です。

クラスターを削除するには、以下の手順で行います。

1. iSeries ナビゲーターで、「**マネージメント・セントラル**」を展開します。
2. 「**クラスター**」を展開します。
3. 削除したいクラスターを右マウス・ボタンでクリックして、「**削除**」を選択します。

CL コマンドおよび API を使用する場合

CL コマンドまたは API を使用してクラスターを削除することもできます。

| • クラスター削除 (DLTCLU) コマンド
| • クラスター削除 (QcstDeleteCluster) API

関連タスク

116 ページの『クラスターのクラスター・バージョンの調整』
クラスター・バージョンは、クラスター内のすべてのノードがアクティブに相互通信するレベルを定義します。

| CRG の作成

| アプリケーション、データ、装置、および対等 CRG などの、いくつかのタイプの CRG を作成することができます。

| クラスター内で CRG を作成するには、次のようなステップを行います。

1. iSeries ナビゲーターで、「**マネージメント・セントラル**」 → 「**クラスター**」を展開します。
2. CRG の追加先のクラスターを拡張表示します。
   a. 装置 CRG を作成したい場合、「**切り替え可能ハードウェア**」を右マウス・ボタンでクリックしてから、「**新規グループ**」を選択します。 注: 「**新規グループ**」オプションを使用できるのは、リカバリー・ドメイン内のすべてのノードが開始済みの場合のみです。詳細は、クラスター・ノードの開始を参照してください。
   b. アプリケーション CRG を作成したい場合、「**切り替え可能ソフトウェア**」を右マウス・ボタンでクリックしてから、「**プロダクトの追加**」を選択します。
   c. データ CRG を作成したい場合、「**切り替え可能データ**」を右マウス・ボタンでクリックしてから、「**新規グループ**」を選択します。
   d. 対等 CRG を作成したい場合、「**対等リソース**」を右マウス・ボタンでクリックしてから、「**新規対等 CRG** 」を選択します。

CL コマンドおよび API を使用する場合

以下のコマンドおよび API を使用して、CRG を作成することができます。

- クラスター・リソース・グループの作成 (CRTCRG) コマンド
- クラスター・リソース・グループの作成 (QcstCreateClusterResourceGroup) API

アクティブなテークオーバー IP アドレスを使ったアプリケーション CRG の作成

アプリケーション CRG の作成時に、アクティブなテークオーバー IP アドレスを許可するよう指定することができます。 それが許可されるのは、ユーザーがテークオーバー IP アドレスを構成する場合だけです。

これまでは、アクティブなテークオーバー IP アドレスがユーザーによって構成されていた場合に限って、そのアドレスを使ってアプリケーション CRG を作成することができました。ただし、テークオーバー IP アドレスがすでにアクティブになっている場合は、アプリケーション CRG を開始することはできませんでした。現在は、アプリケーション CRG の作成時に、アクティブなテークオーバー IP アドレスを許可するよう指定することができます。アクティブなテークオーバー IP アドレスが許可されたアプリケーション CRG を開始すると、その CRG の開始が許可されます。

アプリケーション CRG の作成時に、アクティブなテークオーバー IP アドレスを許可するには、次のようなステップを行います。

1. コマンド行インターフェースで、次のように入力します。

```
CRTCRG CLUSTER(MYCLUSTER) CRG(MYCRG) CRGTYPE(*APP) EXITPGM(QDEVELOP/EXITPGM)
USRPRF(USER) RCYDMN((NODE1 *PRIMARY)(NODE2 *BACKUP)) TKVINTNETA('10.1.2.1') CFGINTNETA(*USR *YES)
```

   TKVINTNETA パラメーターは、使用するテークオーバー IP アドレスを特定し、**CFGINTNETA** パラメーターは、ユーザーがテークオーバー IP アドレスを構成し、さらに CRG の開始時にそのアドレスをアクティブにできることを指定します。

アクティブなテークオーバー IP アドレスを許可するアプリケーション CRG の作成が完了したら、CRG を開始することができます。

CRG の開始

アプリケーション、データ、装置、および対等 CRG などの、いくつかのタイプの CRG を開始することができます。

CRG を開始するには、次のようなタスクを行います。

1. iSeries ナビゲーターで、「**マネージメント・セントラル**」 → 「**クラスター**」を展開します。
2. CRG を開始する場所であるクラスターを拡張表示します。
   a. 装置 CRG を開始したい場合、「**切り替え可能ハードウェア**」をクリックし、開始しようとしている切り替え可能ハードウェア・グループを右マウス・ボタンでクリックしてから、「**開始**」を選択します。
   b. アプリケーション CRG を開始したい場合、「**切り替え可能ソフトウェア**」をクリックし、開始しようとしている切り替え可能ソフトウェア・プロダクトを右マウス・ボタンでクリックしてから、「**開始**」を選択します。
   c. データ CRG を開始したい場合、「**切り替え可能データ**」をクリックし、開始しようとしている切り替え可能データ・グループを右マウス・ボタンでクリックしてから、「**開始**」を選択します。
   d. 対等 CRG を開始したい場合、「**対等リソース**」をクリックして、すべての対等 CRG を一覧で表示し、開始しようとしている対等 CRG を右マウス・ボタンでクリックしてから、「**開始**」を選択します。

CL コマンドおよび API を使用する場合

以下のコマンドおよび API を使用して、CRG を開始することができます。

- クラスター・リソース・グループの開始 (STRCRG) コマンド
- クラスター・リソース・グループの開始 (QcstStartClusterResourceGroup) API

クラスター・リソース・グループのリカバリー・ドメインの変更

クラスター・リソース・グループのリカバリー・ドメインにあるノードの役割の変更、 およびリカバリー・ドメインへのノードの追加、 またはリカバリー・ドメインからのノードの除去が行えます。 装置クラスター・リソース・グループの場合は、 リカバリー・ドメインにあるノードのサイト名とデータ・ポート IP アドレスの変更も行えます。

iSeries ナビゲーターを使用する場合

これを実行するためには、オプション 41 (HA 切り替え可能リソース) がインストールされ、ライセンス交付を受けていることが必要です。

クラスター・リソース・グループの回復ノードにあるノードの役割の変更 (切り替え可能ハードウェア、切り替え可能ソフトウェア、または切り替え可能データ)、また、リカバリー・ドメインへのノードの追加やリカバリー・ドメインからのノードの除去を行うには、以下の手順で行います。

1. iSeries ナビゲーターで、「**マネージメント・セントラル**」を展開します。
2. 「**クラスター**」を展開します。
3. リカバリー・ドメインを変更したい切り替え可能ハードウェア、ソフトウェア、またはデータを含むクラスターを展開します。
4. 切り替え可能ハードウェア、ソフトウェア、またはデータを展開します。
5. 切り替え可能ハードウェア、ソフトウェア、またはデータを右クリックして、「**プロパティー**」を選択します。
6. 「**リカバリー・ドメイン**」ページを選択します。

「リカバリー・ドメイン」ページの「ヘルプ」をクリックして、役割を変更する方法、またはノードを追加、除去する方法の説明を参照してください。

CL コマンドおよび API を使用する場合

リカバリー・ドメインにあるノードの役割の変更、またはノードの追加や除去を行うには、以下の CL コマンドおよび API を使用してください。

- クラスター・リソース・グループ・ノード項目の追加 (ADDCRGNODE) コマンド
- リカバリー・ドメインへのノードの追加 (QcstAddNodeToRcvyDomain) API
- クラスター・リソース・グループの変更 (CHGCRG) コマンド
- クラスター・リソース・グループの変更 (QcstChangeClusterResourceGroup) API
- クラスター・リソース・グループ・ノード項目の削除 (RMVCRGNODE) コマンド
- リカバリー・ドメインからのノードの除去 (QcstRemoveNodeFromRcvyDomain) API

関連概念

13 ページの『リカバリー・ドメイン』
リカバリー・ドメインは、リカバリー・アクション、またはイベントの同期化を実行するなどの共通の目的を持ったクラスター内のノードを、1 つのクラスター・リソース・グループ (CRG) として集めたもののサブセットです。

切り替えの実行

切り替えを手動実行すると、現行プライマリー・ノードは、クラスター・リソース・グループのリカバリー・ドメインで定義したバックアップ・ノードに切り替わります。

切り替えが実行されると、クラスター・リソース・グループのリカバリー・ドメインにあるノードの現行役割は次のように変わります。

- 現行プライマリー・ノードに、最後のアクティブ・バックアップの役割が割り当てられる。
- 現行の最初のバックアップに、プライマリー・ノードの役割が割り当てられる。
- それ以降のバックアップは、バックアップの順序が 1 つ上に移動する。

切り替えを実行できるのは、状況が ACTIVE になっているプライマリー・バックアップ・モデルの CRG においてのみです。

注: 切り替え可能装置 (装置 CRG とも呼ばれる) で切り替えを実行する場合、パフォーマンス上の理由で、ユーザー・プロファイル名、UID、および GID を同期する必要があります。

iSeries ナビゲーターを使用する場合

これを実行するためには、オプション 41 (HA 切り替え可能リソース) がインストールされ、ライセンス交付を受けていることが必要です。

リソース (切り替え可能ハードウェア・グループ、切り替え可能アプリケーション、または切り替え可能データ・グループ) をプライマリー・ノードからリカバリー・ドメインのバックアップ・ノードに切り替えるには、リソースの状況が**開始済み**になっていなければなりません。

リソースで切り替えを実行するには、以下の手順のようにします。

1. iSeries ナビゲーターで、「**マネージメント・セントラル**」を展開します。
2. 「**クラスター**」を展開します。
3. 希望のリソースを含むクラスターを展開します。
4. 「**切り替え可能ハードウェア**」、「**切り替え可能ソフトウェア**」、または「**切り替え可能データ**」をクリックします。

5. 希望のリソースを右クリックして、「**切り替え...**」を選択します。

クラスター API の使用

以下を使用して切り替えを実行することもできます。

- クラスター・リソース・グループ・プライマリーの変更 (CHGCRGPRI) コマンド
- 切り替えの開始 (QcstInitiateSwitchOver) API

関連概念

13 ページの『リカバリー・ドメイン』

リカバリー・ドメインは、リカバリー・アクション、またはイベントの同期化を実行するなどの共通の目的を持ったクラスター内のノードを、1 つのクラスター・リソース・グループ (CRG) として集めたもののサブセットです。

関連タスク

24 ページの『切り替え』

切り替えは、あるサーバーから別のサーバーにアクセスを手動で切り替えるときに発生します。

ユーザー・プロファイル名、UID、および GID を同期する

デバイス・ドメインへのノードの追加

デバイス・ドメインとは、装置リソースを共用するクラスター内のノードのサブセットです。

装置クラスター・リソース・グループ (CRG) のリカバリー・ドメインにノードを追加するには、まずそのノードをデバイス・ドメインのメンバーとして定義する必要があります。装置 CRG のリカバリー・ドメインに入るすべてのノードは、同じデバイス・ドメインに所属している必要があります。クラスター・ノードは、同時に複数のデバイス・ドメインに所属することはできません。

デバイス・ドメインを作成し、管理するには、オプション 41 (HA 切り替え可能リソース)をインストールする必要があり、デバイス・ドメイン内にあるすべてのクラスター・ノードにライセンス・キーが存在していなければなりません。

iSeries ナビゲーターを使用する場合

iSeries ナビゲーターでデバイス・ドメインにノードを追加するには、以下の手順で行います。

1. iSeries ナビゲーターで、「**マネージメント・セントラル**」を展開します。
2. 「**クラスター**」を展開します。
3. デバイス・ドメインに追加したいノードを含むクラスターを展開します。
4. 「**ノード**」をクリックします。
5. デバイス・ドメインに追加したいノードを右クリックし、「**プロパティー**」を選択します。
6. 「**クラスタリング**」ページで、ノードを追加したいデバイス・ドメインの名前を「**デバイス・ドメイン**」フィールドに指定します。

CL コマンドおよび API を使用する場合

以下を使用してデバイス・ドメインにノードを追加することもできます。

- デバイス・ドメイン項目の追加 (ADDDEVDMNE) コマンド
- デバイス・ドメイン項目の追加 (QcstAddDeviceDomainEntry) API

関連概念

19 ページの『デバイス・ドメイン』
デバイス・ドメインとは、装置リソースを共用するクラスター内のノードのサブセットです。 もう少し具体的に説明すれば、デバイス・ドメイン内のノードは、回復リソースのコレクションに対するスイッチ・アクションに参加できます。

関連タスク

『デバイス・ドメインからのノードの除去』
デバイス・ドメインとは、装置リソースを共用するクラスター内のノードのサブセットです。

デバイス・ドメインからのノードの除去

デバイス・ドメインとは、装置リソースを共用するクラスター内のノードのサブセットです。

重要:

デバイス・ドメインからノードを除去するときは、注意が必要です。デバイス・ドメインからノードを除去する場合、そしてそのノードがすべての独立ディスク・プールに対する現行の 1 次アクセス・ポイントになっている場合、ノードは除去されますがそれらの独立ディスク・プールは残ります。これは、デバイス・ドメイン内の残りのノードからそれらの独立ディスク・プールにアクセスできなくなることを意味します。

いったんノードをデバイス・ドメインから除去したら、 1 つ以上の既存のクラスター・ノードがまだその同じデバイス・ドメインに属している場合、ノードを再びその同じデバイス・ドメインに追加することはできません。ノードを再びデバイス・ドメインに追加するには、以下の手順で行います。

1. デバイス・ドメインに追加されるノードが現在所有している独立ディスク・プールを削除する。
2. ノードでシステム再始動 (IPL) を実行する。
3. デバイス・ドメインにノードを追加する。デバイス・ドメインへのノードの追加を参照してください。
4. ステップ 1 で削除した独立ディスク・プールの再作成

iSeries ナビゲーターを使用する場合

これを実行するためには、オプション 41 (HA 切り替え可能リソース) がインストールされ、ライセンス交付を受けていることが必要です。

iSeries ナビゲーターでデバイス・ドメインからノードを除去するには、以下の手順で行います。

1. iSeries ナビゲーターで、「**マネージメント・セントラル**」を展開します。
2. 「**クラスター**」を展開します。
3. デバイス・ドメインから除去したいノードを含むクラスターを展開します。
4. 「**ノード**」をクリックします。
5. デバイス・ドメインから除去したいノードを右クリックし、「**プロパティー**」を選択します。
6. 「クラスタリング」ページで、「**デバイス・ドメイン**」フィールドの項目を除去します。

CL コマンドおよび API を使用する場合

以下を使用してデバイス・ドメインからノードを除去することもできます。

- デバイス・ドメイン項目の除去 (RMVDEVDMNE) コマンド
- デバイス・ドメイン項目の除去 (QcstRemoveDeviceDomainEntry) API

関連概念

19 ページの『デバイス・ドメイン』
デバイス・ドメインとは、装置リソースを共用するクラスター内のノードのサブセットです。 もう少し具体的に説明すれば、デバイス・ドメイン内のノードは、回復リソースのコレクションに対するスイッチ・アクションに参加できます。

関連タスク

121 ページの『デバイス・ドメインへのノードの追加』
デバイス・ドメインとは、装置リソースを共用するクラスター内のノードのサブセットです。

ディスク装置またはディスク・プールを追加する

クラスターに対するシステム・イベントの影響の仕方

システム電源遮断 (PWRDWNSYS)、システム終了 (ENDSYS)、およびサブシステム終了 (ENDSBS) コマンドなどの、システム機能を終了する一部のコマンドによってクラスターが突然終了し、それが原因でクラスター区画が発生することがあります。

V5R4 では、PWRDWNSYS、ENDSYS、および ENDSBS コマンドの機能強化が行われました。ノード上でクラスタリングがアクティブのときに、上記のようなコマンドが実行された場合、クラスター・ノード終了 (QcstEndClusterNode) APIが発行されます。

上記のようなコマンドを完了したい場合、OPTION(*CNTRLD) を使用し、該当する遅延時間を DELAY パラメーターに指定してください。そうしないと、クラスター・ノード終了 API が完了した後でシステム終了機能に制御が戻されるという順序にならない可能性があります。

注: ユーザーが **OPTION(*IMMED)** を指定すると、クラスター・ノード終了 (QcstEndClusterNode) API の完了に 約 30 秒かけてからシステムが終了します。この場合、結果としてクラスター・ノード終了ではなく、フェイルオーバーが行われることがあります。

クラスター管理可能ドメインの作成

クラスター管理可能ドメインは、iSeries ナビゲーターで作成できますが、クラスター管理可能ドメインの作成 (CRTADMDMN) コマンドを使って作成することもできます。

クラスター管理可能ドメインを作成して管理するには、作成する CRG、CRG コマンド、および QCLUSTER ユーザー・プロファイルに対する許可がユーザーになければなりません。

iSeries ナビゲーターを使用する場合

クラスター管理可能ドメインを作成するには、以下の手順で行います。

1. iSeries ナビゲーターで、「**マネージメント・セントラル**」 → 「**クラスター**」を展開します。
2. クラスター管理可能ドメインを追加する対象のノードを拡張表示します。
3. 「**対等リソース**」を右マウス・ボタンでクリックし、「**新規の管理可能ドメイン**」を選択します。

CL コマンドおよび API を使用する場合

以下のコマンドおよび API を使用して、クラスター管理可能ドメインを作成することができます。

- クラスター管理可能ドメインの作成 (CRTADMDMN) コマンド
- クラスター管理可能ドメインを作成する API はありません。

関連概念

82 ページの『クラスター・コマンドおよび API』
i5/OS クラスター・リソース・サービスからは、一連の制御言語 (CL) コマンドおよびアプリケーション・プログラム・インターフェース (API) が提供され、さらに、iSeries アプリケーション・プロバイダーまたはカスタマーが、アプリケーションの可用性を拡張するために使用できる諸機能も提供されます。

モニター対象ソース項目の追加

ノード間で共有されるリソースを表すクラスター管理ドメインに、モニター対象リソース項目を追加できます。

モニター対象リソース項目を追加するには、以下の手順で行います。

1. iSeries ナビゲーターで、「**マネージメント・セントラル**」 → 「**クラスター**」を展開します。
2. モニター対象リソース項目を追加するクラスターを拡張します。
3. 「**対等リソース**」を拡張し、クラスターにあるすべての対等リソースのリストを表示します。
4. モニター対象リソース項目を追加するクラスター管理ドメインを拡張します。
5. モニター対象リソース・タイプを右クリックし、「**モニター対象リソース項目の追加**」を選択します。
6. モニター対象リソース項目としてモニターする属性を選択し、「**OK**」をクリックします。

注: モニター対象リソース項目としてパスワード同期を使用するユーザー・プロファイルを追加する場合は、「サーバー機密保護の保存」(QRETSVRSEC) システム値を 1 に設定する必要があります。

CL コマンドおよび API を使用する場合

モニター対象リソースを追加するには、以下のコマンド、および API を使用できます。

- この機能に相当する CL コマンドとしてサポートされているものはありません。サポートされていないコマンドおよび呼び出し処理プログラム (CPP) のソースは、QUSRTOOL ライブラリー内に用意されています。このコマンド・ソース、および CPP の詳細は、ファイル QATTINFO のメンバー QFPADINFO を参照してください。
- モニター対象リソース項目の追加 (QfpadAddMonitoredResourceEntry) API

クラスター管理可能ドメインのモニター

クラスター管理可能ドメインを作成して、適切なモニター対象リソース項目を追加したら、クラスター管理者は、管理可能ドメイン内のアクティビティーをモニターして、モニター対象リソースの一貫性が保たれるようにする必要があります。

モニター対象リソース状況のグローバル状況に不整合が生じた場合、管理者は、リソースが不整合になった理由を判別し、問題を解決し、リソースを再同期するのに必要な処置をとる必要があります。

1 つ以上のノード上で更新が失敗したことが原因でリソースに不整合が生じる場合に備えて、障害の原因を特定するのに役立つ MRE の情報が保存さます。障害が発生したノードでは、失敗した更新の原因である MRE にメッセージが記録されます。それ以外のノードでは、障害の発生を知らせるメッセージが、更新が失敗したノードのリストと一緒に記録されます。

不整合の原因を特定したら、障害の発生場所であるノード上で更新操作を行うか、または管理可能ドメインの終了と再始動を介して、リソースを再同期することができます。

| ドメイン内のいずれかのノード上でリソースの削除、名前変更、または移動を行った場合、モニター対象リソースのグローバル状況は常に不整合に設定されます。そのような場合、クラスター管理可能ドメインによってリソースはもう同期されないので、MRE を除去する必要があります。

iSeries ナビゲーターを使用する場合

クラスター管理可能ドメインをモニターするには、以下の手順で行います。

1. iSeries ナビゲーターで、「**マネージメント・セントラル**」 → 「**クラスター**」を展開します。
2. クラスター管理可能ドメインを関連付ける対象のクラスターを拡張表示します。
3. 「**対等リソース**」を拡張表示し、「**新規の管理可能ドメイン**」を右マウス・ボタンでクリックしてから、「**プロパティー**」を選択します。 アクティブなクラスター管理可能ドメイン全体でリソースに指定できる状況値は次のとおりです。

   整合 システムでモニターされるすべてのリソースの属性の値は、クラスター管理可能ドメイン内のどのアクティブ・ノードでも同じです。

   不整合 リソースの属性のうちシステムでモニターされるものすべての値がクラスター管理可能ドメイン内の全アクティブ・ノードで同じになっていないか、またはクラスター管理可能ドメインがアクティブではありません。

   保留中 モニター対象の属性の値は、クラスター管理可能ドメイン全体において同期処理の進行中にあります。

   追加済み
   モニター対象リソース項目は、クラスター管理可能ドメイン内のモニター対象ディレクトリーに追加されましたが、まだ同期されていません。

CL コマンドおよび API を使用する場合

以下のコマンドおよび API を使用して、クラスター管理可能ドメインをモニターすることができます。

- この機能に相当する CL コマンドとしてサポートされているものはありません。サポートされていないコマンドおよび呼び出し処理プログラム (CPP) のソースは、QUSRTOOL ライブラリー内に用意されています。このコマンド・ソース、および CPP の詳細は、ファイル QATTINFO のメンバー QFPADINFO を参照してください。
- モニター対象リソース情報の検索 (QfpadRtvMonitoredResourceInfo) API

クラスター状況のモニター

クラスター・リソース・サービスは、必要に応じて適切なアクションをとりながら、信頼性の高いメッセージ機能とハートビート・モニタリングを使って、クラスターとそのコンポーネントの基本モニターを実行します。

また、クラスターおよびそのコンポーネントの状況を手動でモニターすることもできます。

iSeries ナビゲーターを使用する場合

これを実行するためには、オプション 41 (HA 切り替え可能リソース) がインストールされ、ライセンス交付を受けていることが必要です。

iSeries ナビゲーターでクラスターの状況をモニターするには、以下の手順で行います。

1. iSeries ナビゲーターで、「マネージメント・セントラル」を展開します。
2. 「**クラスター**」を展開します。

3. iSeries ナビゲーター・フォルダー内の希望のクラスターにナビゲートし、クラスター、そのノード、およびリソースの状況を、iSeries ナビゲーション・リストにある「状況」列を使って表示します。オンライン・ヘルプに、「状況」列に入り得る値が説明されています。クラスターの構成要素を右クリックして、「**プロパティー**」を選択することによってクラスターに関する情報を表示することもできます。

CL コマンドおよび API を使用する場合

クラスター状況をモニターするために、以下のコマンドおよび API を使用することができます。

クラスター情報

クラスター内のノード、各ノードで使用されているアダプター IP アドレス、またクラスター内の各ノードの状況などの、クラスターに関する情報を取得します。

- クラスター情報の表示 (DSPCLUINF) コマンド
- クラスター情報のリスト (QcstListClusterInfo) API
- デバイス・ドメイン情報リスト (QcstListDeviceDomainInfo) API
- クラスター・リソース・サービス情報検索 (QcstRetrieveCRSInfo) API
- クラスター情報の検索 (QcstRetrieveClusterInfo) API

クラスター・リソース・グループ情報

クラスター・リソース・グループのリストおよびクラスター内のクラスター・リソース・グループに関する情報 (たとえばクラスター内の各 CRG のプライマリー・ノード名など) を生成します。

- クラスター・リソース・グループ情報の表示 (DSPCRGINF) コマンド
- クラスター・リソース・グループのリスト (QcstListClusterResourceGroups) API
- クラスター・リソース・グループのリスト (QcstListClusterResourceGroupInf) API

関連概念

32 ページの『メッセージング機能』
クラスター・リソース・サービスのメッセージング機能は、クラスター内の各ノードに注意を払い、クラスター・リソースの状態に関する整合した情報を確実にすべてのノードが保持するようにします。

30 ページの『ハートビート・モニター』
ハートビート・モニターはクラスター・リソース・サービス機能の 1 つで、クラスター内のすべてのノードからクラスター内の自分以外のすべてのノードにシグナルを送信して自分がアクティブであることを伝達してすべてのノードがアクティブであることを確証するものです。

クラスター・パフォーマンス

クラスターに変更を加えると、 クラスターを管理するのに必要なオーバーヘッドに影響を与える可能性があります。

クラスタリングが必要とするリソースは、ハートビート・モニターを実行すること、クラスター・リソース・グループとクラスター・ノードを管理すること、およびクラスター・リソース・グループとクラスター・ノードとの間で行われるメッセージ交換を処理することだけです。クラスタリング環境が作動可能になったなら、オーバーヘッドが増えるのは、クラスターに変更を加えたときだけです。

通常の作動環境にある間は、クラスタリング活動によるクラスター・システムへの影響は最小限に抑えられます。

関連概念

30 ページの『ハートビート・モニター』
ハートビート・モニターはクラスター・リソース・サービス機能の 1 つで、クラスター内のすべてのノ

ードからクラスター内の自分以外のすべてのノードにシグナルを送信して自分がアクティブであることを伝達してすべてのノードがアクティブであることを確証するものです。

152 ページの『クラスターの一般的な問題』
ここでは、クラスターで生じる可能性のある最も一般的な問題のいくつかをリストして、 その回避方法および回復方法を示します。

クラスターでのネットワーク負荷の平衡

クラスター内のノードを相互に結び付けるために使用する各通信回線に処理を分割すると、ネットワーク負荷の平衡が取れます。

処理を分散させてリソースの使用率を低くすればするほど、システムはよりスムーズに稼働する結果となります。

バックアップ・ノードの CPU 負荷:

バックアップ・システムを最大限に活用してください。ただし、フェイルオーバーが行われた場合は、余分なワークロードがバックアップ・ノードに転嫁される可能性があることに注意する必要があります。

業務上何が重要で何が重要でないのかを識別することは、非常に重要です。非常に重要なアプリケーションで障害が発生してノード変更が行われた場合には、バックアップ・ノードでの中央演算処理装置 (CPU) の負荷が高くなりすぎて、その重要なアプリケーションを実行できなくなってしまわないように注意しなければなりません。

クラスター・パフォーマンスの調整

それぞれの通信環境には大きな違いがあり得るので、クラスター通信に影響を与える変数を各環境に合わせて最適な値に調整するための機能が用意されています。

デフォルト値は、一般的な環境のほとんどに適用されますが、デフォルト値が合わない環境の場合は、それぞれの環境に合わせてクラスター通信を調整できます。この調整については、2 つのレベルがあります。

初級レベルの調整

初級レベルの調整。タイムアウトとメッセージ送信間隔については、高、中、低のレベルの値がそれぞれ事前に定義されているので、その事前定義の値に調整パラメーターを設定できます。中レベルを選択すると、クラスター通信のパフォーマンス・パラメーターと構成パラメーターには、デフォルト値が使用されます。低レベルを選択した場合は、クラスタリング機能のハートビート間隔値と各種メッセージ・タイムアウト値が増えます。したがって、ハートビートの数が減り、タイムアウト時間が長くなるので、クラスターによる通信障害の検出能力が落ちることになります。高レベルを選択した場合は、クラスタリング機能のハートビート間隔値と各種メッセージ・タイムアウト値が減ります。したがって、ハートビートの数が増え、タイムアウト時間が短くなるので、クラスターによる通信障害の検出能力が高まることになります。

上級レベルの調整

上級レベルの調整。値の範囲は事前に定義されていますが、その範囲内で個々のパラメーターを自由に調整できます。したがって、通信環境内のいろいろな特殊事情に合わせて、きめの細かい調整ができます。上級レベルの調整を望む場合は、IBM サポート担当者などの援助を受けることをお勧めします。個々のパラメーターの設定が不適切であれば、パフォーマンスの低下につながります。

関連概念

107 ページの『調整可能なクラスター通信パラメーター』
クラスター・リソース・サービス変更 (QcstChgClusterResourceServices) API を使用すれば、 クラスタ

ー・トポロジー・サービスおよびクラスター通信のパフォーマンス・パラメーターと、 構成パラメーターの一部を、 クラスタリングが行われる多数の固有なアプリケーション環境およびネットワーク環境により良く適合するように調整することができます。 この API は、 クラスター・バージョン 2 またはそれ以降のバージョンで実行されるあらゆるクラスターで使用できます。

関連資料

クラスター・リソース・サービス変更 (QcstChgClusterResourceServices) API

クラスター・ジョブの終了

クラスター・ジョブは、決して直接終了しないでください。

クラスター環境内で実行されているものを停止する必要がある場合は、以下の手順で行います。

1. クラスター・ノードの終了。
2. 問題を修正する。
3. クラスター・ノードの開始。

関連タスク

115 ページの『クラスター・ノードの終了』
ノードを停止または終了すると、そのノード上のクラスター・リソース・サービスが停止します。

115 ページの『クラスター・ノードの開始』
クラスター・ノードを開始すると、クラスター内のノード上のクラスター・リソース・サービスも開始されます。 クラスター・バージョン 3 から、クラスターにアクティブ・ノードがあれば、あるノードが自分自身を開始して現在アクティブなクラスターに再結合することができるようになりました。

リソースのモニターと制御 (RMC)

リソースのモニターと制御 (RMC) は、物理的または論理的なシステム・エンティティーなどのリソースの管理、モニター、および操作のために汎用化されたフレームワークです。

RMC は、ハードウェア管理コンソール (HMC) にサポート・イベントをレポートするための通信メカニズムとして使用されます。 RMC がアクティブになっていないと、サービス・イベントは HMC にはレポートされません。 以下のリストは、RMC に関連付けられているサービスを説明しています。

CAS デーモン

用途: RMC の認証サーバーとして機能します。

ジョブ名: QCSTCTCASD

RMC デーモン

用途: リソース・マネージャーとの通信によって、リソースをモニターします。

ジョブ名: QCSTCTRMCD

SRC デーモン

用途: 他の RMC ジョブの状況をモニターします。特定のジョブが不意に終了した場合、ジョブを再始動します。

ジョブ名: QCSTSRCD

リソース・マネージャー (RM)

リソース・マネージャー (RM) とは、RMC と実際の物理または論理エンティティーを結び付けるインターフェースを管理し提供するジョブのことです。 RMC は、物理または論理エンティティーのリソース・クラス、リソース、および属性などの基本的な抽象を提供しますが、それ自身は実際のエンティティーを表わ

しません。実際のエンティティーは、RM によって RMC の抽象にマップされます。以下のリストは、サポートされている RMC 用のさまざまなリソース・マネージャーを説明しています。

監査ログ RM

用途: システム操作に関する情報を記録する機能を提供します。

ジョブ名: QYUSALRMD

CSMAgent RM

用途: 管理サーバーを表すリソース・クラスを提供します。それは HMC です。

ジョブ名: QYUSCMCRMD

ホスト RM

用途: 個々のマシンを表すリソース・クラスを提供します。

ジョブ名: QCSTCTHRMD

サービス RM

用途: 問題情報を管理し、HMC へのデリバリーのための準備を行います。

ジョブ名: QSVRMSERMD

RMC の開始または終了

RM ジョブを含むすべての RMC ジョブは、QSYSWRK サブシステム内に置かれ、サブシステムの開始時に自動的に開始します。始動を完了するには、TCP/IP がアクティブになっていなければなりません。RMC デーモンの場合、TCP/IP がアクティブになっている必要があります。 TCP/IP が非アクティブになると、RMC デーモンは終了します。 TCP/IP が再びアクティブになると、RMC デーモンは SRC デーモンによって自動的に再始動されます。通常の条件下では、ユーザーが処理を行う必要はありません。 RMC を手動で開始する必要が生じた場合、次のようなコマンドを実行してください。

```
SBMJOB CMD(CALL PGM(QSYS/QCSTCTSRCD)) JOBD(QSYS/QCSTSRCD) PRTDEV(*JOBD) OUTQ(*JOBD)
USER(*JOBD) PRTTXT(*JOBD) RTGDTA(RUNPTY50)
```

RMC を手動で終了する必要が生じた場合、ENDJOB コマンドを使って QCSTSRCD ジョブを終了してください。このコマンドは、すべての RMC ジョブを終了するはずです。ジョブがすべては終了しなかった場合、上記のジョブを 1 つずつ手動で終了してください。

ジョブ構造とユーザー待ち行列

クラスターを管理するには、ジョブ構造とユーザー待ち行列についての知識が必要です。

クラスター・リソース・サービス・ジョブ構造

クラスター・リソース・サービスは、一連のマルチスレッド・ジョブで構成されます。サーバー上でクラスタリングがアクティブになっていると、QSYS ユーザー・プロファイルの下、QSYSWRK サブシステムで以下のジョブが実行されます。ジョブは QDFTSVR ジョブ記述を使って実行されますが、ジョブ・ログが生成されるロギング・レベルに設定されます。

- クラスター制御は、QCSTCTL という名前の 1 つのジョブにより構成される。
- クラスター・リソース・グループ管理機能は、QCSTCRGM という名前の 1 つのジョブにより構成される。
- クラスター・リソース・グループは、クラスター・リソース・グループ・オブジェクトにつき 1 つのジョブにより構成される。ジョブ名は、クラスター・リソース・グループ名と同じになります。ここには、クラスター管理可能ドメインも含まれます。

- 回復装置 CRG 内の 1 つ以上の装置リスト項目が切り替えまたはフェイルオーバー時にオンラインになるように設定されると、追加ジョブが投入され、オンに変更する機能を実行します。

QCSTCTL および QCSTCRGM ジョブは、クラスターの重要なジョブです。つまり、ノードがクラスター内でアクティブになるためには、これらのジョブが実行されなければなりません。

クラスター・リソース・グループの作成時に指定されたユーザー・プロファイルを使用する大抵のクラスター・リソース・グループ API は、ジョブを個別に投入します。クラスター・リソース・グループ内で定義される出口プログラムは、投入されたジョブの中で呼び出されます。デフォルトでは、ジョブは QBATCH ジョブ待ち行列に送信されます。一般に、このジョブ待ち行列は実動バッチ・ジョブで使用され、出口プログラムの完了を遅延または妨害します。 API を効果的に実行できるようにするには、クラスター・リソース・グループで使用されるユーザー・プロファイル、ジョブ記述、およびジョブ待ち行列を個別に作成します。作成するすべてのクラスター・リソース・グループに対して新しいユーザー・プロファイルを指定します。クラスター・リソース・グループで定義されたリカバリー・ドメイン内のすべてのノード上で同一プログラムが処理されます。

クラスター回復変更 (CHGCLURCY) コマンド を使って、ノード上でクラスタリングの終了と再始動を行わないで終了したクラスター・リソース・グループのジョブを再始動することができます。

クラスター API によるユーザー待ち行列の使用

結果情報パラメーターを持つ API によって実行される機能は非同期で実行され、 API が処理を終了したらその結果をユーザー待ち行列に送信します。ユーザー待ち行列は、API を呼び出す前に作成しておかなければなりません。ユーザー待ち行列は、 ユーザー待ち行列作成 (QUSCRTUQ) API を使用して作成できます。待ち行列は、キー付き待ち行列として作成しなければなりません。ユーザー待ち行列のキーは、ユーザー待ち行列項目の形式で記述されます。ユーザー待ち行列名は、API に渡されます。クラスター API 資料には、クラスター API と一緒にユーザー待ち行列を使用する方法の例が記載されています。

情報配布 (QcstDistributeInformation) API が使用されると、ノード間で送信される情報は、CRG の作成時に指定されたユーザー待ち行列に保管されます。リカバリー・ドメイン内のすべてのアクティブ・ノード上のユーザーは、情報配布 API を使用する前にこの待ち行列を作成しなければなりません。情報配布待ち行列が存在しなければならないときの詳細は、クラスター・リソース・グループ作成 (QcstCreateClusterResourceGroup) API を参照してください。

フェイルオーバー・メッセージ待ち行列は、フェイルオーバー・アクティビティーに関するメッセージを受け取ります。

関連概念

102 ページの『すべてのノードでユーザー・プロファイルを保守する』
クラスター内のどのノードでも、ユーザー・プロファイルの保守には 2 通りのメカニズムを使用することができます。

139 ページの『クラスターの問題が存在するかどうかの判別』
クラスターに関する問題を診断するには、ここから始めてください。

関連タスク

159 ページの『クラスター・ジョブ障害からの回復』
クラスター・リソース・サービス・ジョブの障害は、通常、何か他の問題があることを示しています。

フェイルオーバー・メッセージ待ち行列

フェイルオーバー・メッセージ待ち行列は、フェイルオーバー・アクティビティーに関するメッセージを受け取ります。

フェイルオーバー・メッセージ待ち行列を使うことにより、管理者はフェイルオーバーが発生する前に通知を受け取ることができます。この時点でフェイルオーバーを避けるのが望ましければ、管理者はそれを取り消すことができます。

フェイルオーバー・メッセージ待ち行列は、クラスターの作成 (QcstCreateCluster) API を使用してクラスター・リソース・グループを作成する際に定義されます。また、クラスター・リソース・グループを変更するための CL コマンドと API を使って修正することも可能です。フェイルオーバー・メッセージ待ち行列を iSeries ナビゲーター・クラスター管理インターフェースから使用することはできません。

フェイルオーバー・メッセージ待ち行列の詳細は、クラスター・リソース・グループ APIの中で説明されています。フェイルオーバー・メッセージ待ち行列の使用法の詳細は、次の説明を参照してください。

CL コマンド

- クラスター・リソース・グループの作成 (CRTCRG) コマンド
- クラスター・リソース・グループの変更 (CHGCRG) コマンド

API

- クラスター・リソース・グループの作成 (QcstCreateClusterResourceGroup) API
- クラスター・リソース・グループの変更 (QcstChangeClusterResourceGroup) API

すべてのノードでユーザー・プロファイルを保守する

クラスター内のどのノードでも、ユーザー・プロファイルの保守には 2 通りのメカニズムを使用することができます。

一方のメカニズムは、クラスター内のすべてのノードを通して共用リソースをモニターするためのクラスター管理可能ドメインを作成するためのメカニズムです。クラスター管理可能ドメインは、ユーザー・プロファイルに加えていくつかのタイプのリソースをモニターできるので、すべてのノードを通して共用されているリソースの管理を簡単に行うことができます。そのようなリソースの詳細は、モニター対象リソースを参照してください。クラスター管理ドメインがアクティブになっているときに、ユーザー・プロファイルを更新すると、変更内容は自動的に他のノードに伝搬されます。クラスター管理可能ドメインがアクティブになっていない場合、変更内容は、クラスター管理可能ドメインがアクティブになってから伝搬されます。

注: クラスター内でパスワード同期を活用するユーザー・プロファイルを共有する予定の場合、サーバー・セキュリティー保持 (QRETSVRSEC) システム値を 1 に設定する必要があります。

もう一方のメカニズムでは、管理者は iSeries ナビゲーターの「マネージメント・セントラル」も併用して、複数のシステムおよびシステム・グループを対象に各種機能を実行することができます。このサポートには、オペレーターがクラスター内の複数のシステムにまたがって実行しなければならない、いくつかの一般的なユーザー管理タスクも含まれています。マネージメント・セントラルを使用すれば、システム・グループに対してユーザー・プロファイル機能を実行できます。管理者であれば、ユーザー・プロファイルの作成時に、ターゲット・システムで伝搬後のコマンドが実行されるように設定することもできます。

クラスターのバックアップおよび回復

システムにクラスタリングを使用する場合でも、データを保護するためのバックアップ戦略および回復戦略を計画することは依然として重要です。

バックアップ戦略の一環としてクラスタリングの使用を計画し、バックアップ時に一方のシステムがダウンしたとき、もう一方のシステムが稼働中であるようにする場合は、クラスター内に最低でも 3 つ以上のシ

ステムを含めるようお勧めします。クラスター内に 3 つのシステムがあれば、常に 1 つのシステムは切り替えが可能で、いずれかのシステムで障害が発生した場合に安心です。

クラスター・リソース・グループの保管および復元

クラスターがアクティブかどうかに関係なく、クラスター・リソース・グループはいつでも保管できます。クラスター・リソース・グループの復元については、以下の制限があてはまります。

- そのクラスターに認識されているクラスター・リソース・グループは、復元できません。
- ノードがクラスターに構成されていない場合、クラスター・リソース・グループは復元できません。

アクティブなクラスターにクラスター・リソース・グループが認識されておらず、ノードがクラスター・リソース・グループのリカバリー・ドメイン内に含まれており、かつそのクラスター名がクラスター・リソース・グループ内のクラスター名と一致している場合、そのクラスター・リソース・グループは復元できます。クラスターが構成されているものの、そのノードではアクティブではなく、しかしそのノードがそのクラスター・リソース・グループのリカバリー・ドメイン内に含まれている場合には、そのクラスター・リソース・グループを復元できます。

災害のための準備

災害が起きたら、クラスターを再構成する必要があります。そのようなシナリオに備えるために、クラスター構成情報を保管して、その情報のハードコピー印刷出力を取っておくことをお勧めします。

1. クラスター構成に変更を加えた後、 構成の保管 (SAVCFG) コマンドまたはシステム保管 (SAVSYS) コマンドを使用して、復元された内部クラスター情報が現行のものとし、クラスター内のその他のノードと整合性をとる。 SAVCFG または SAVSYS の実行について詳しくは、構成情報を保管するを参照してください。
2. クラスター構成情報を変更するたびにそのコピーを出力する。クラスター構成の印刷には、クラスター情報の表示 (DSPCLUINF) コマンドを使用できます。バックアップ・テープと一緒にコピーを取っておき、クラスター全体の再構成が必要になるような災害の際に使用してください。

関連概念

162 ページの『バックアップ・テープからのクラスターの復元』
通常の操作時には、バックアップ・テープから復元する必要が生じることはありません。

構成情報を保管する

161 ページの『完全なシステム消失後のクラスターの回復』
サーバーでの想定外の電力喪失が原因で完全なシステム消失が起きた場合、ここに記載されている情報と、「バックアップおよび回復の手引き」の中の該当するチェックリストを一緒に使用して、システム全体を回復してください。

133 ページの『クラスター構成の保管』
クラスター・リソース・グループ・オブジェクトを保管するために、 コマンドを使用することができます。

162 ページの『災害後のクラスターの回復』
災害が起きてすべてのノードが失われた場合、クラスターを再構成する必要があります。

関連タスク

バックアップおよび回復方針の計画

システム情報の印刷

クラスター構成の保管

クラスター・リソース・グループ・オブジェクトを保管するために、 コマンドを使用することができます。

SAVSYS (システム保管) コマンドを使用すると、構成済みクラスターだけでなく、システム全体を保管することができます。構成済みシステムを保管するには、SAVCFG (構成の保管) コマンドを使用します。

SAVOBJ(QUSRSYS/*ALL) OBJTYPE (*CRG)

注: クラスター・リソース・グループ・オブジェクトは、現行リリースでのみ保管できます。

関連タスク

131 ページの『クラスターのバックアップおよび回復』
システムにクラスタリングを使用する場合でも、データを保護するためのバックアップ戦略および回復戦略を計画することは依然として重要です。

関連資料

システム保管 (SAVSYS)

構成の保管 (SAVCFG)

例: クラスター構成

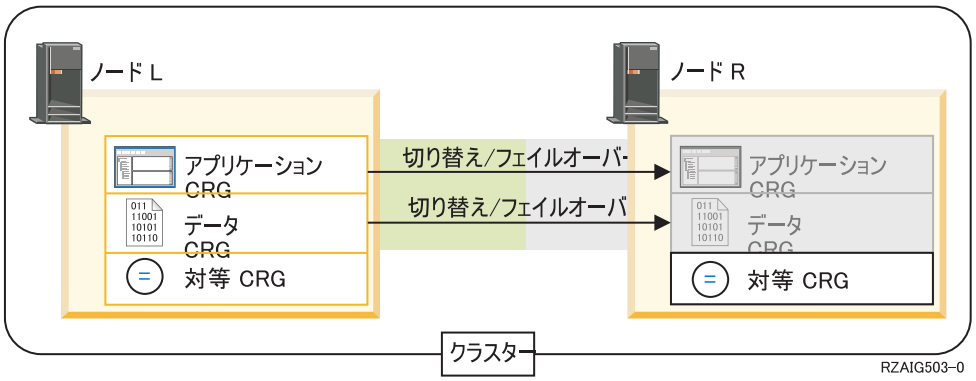
典型的なクラスターのインプリメンテーション例を使って、いつ、なぜ、そしてどのようにクラスターを使用するのが有益であるかを解明します。

例: シンプル 2 ノード・クラスター

この構成例は、2 つのノードからなる基本クラスターを説明しています。

この構成例では、以下が用意されています。

- 片方向複製およびフェイルオーバー
- 2 層の環境
- アプリケーションとデータの並行した移動
- データのオフライン処理で使用されるバックアップ
- 対等 CRG に対する連続稼働



この例のノード L は現在、 アプリケーション CRG、データ CRG という 2 つのクラスター・リソース・グループのプライマリー・ノードとして機能しています。また、いずれかのノードでの連続稼働に対応する対等 CRG も組み入れられています。 アプリケーション CRG に関して 2 つの出口プログラムがノ

| ード L 上で定期的に実行されています。 2 つの出口プログラムが同時に実行されているのは、CRG 開始
| API を呼び出すと、出口プログラムが開始され、アプリケーション CRG がアクティブの間継続的に実行
| されるためです。アプリケーション CRG に対して CRG 終了 API を呼び出すと、もう 1 つの出口プロ
| グラムが開始されます。 ノード R は、各クラスター・リソース・グループのリカバリー・ドメインで指
| 定された、最初で唯一のバックアップ・ノードです。データ CRG に関連付けられたデータと、アプリケ
| ーション CRG に関連付けられた該当アプリケーション情報は、ノード L からノード R に複製されま
| す。ノード L で障害が発生するか、または管理上の理由で停止する必要が生じた場合、フェイルオーバー
| または切り替えが開始され、ノード R が、アプリケーションとデータ CRG の両方に対するプライマリ
| ー・ノードになります。ノード R はアプリケーション CRG に定義されたインターネット・プロトコル
| (IP) を引き継ぎます。

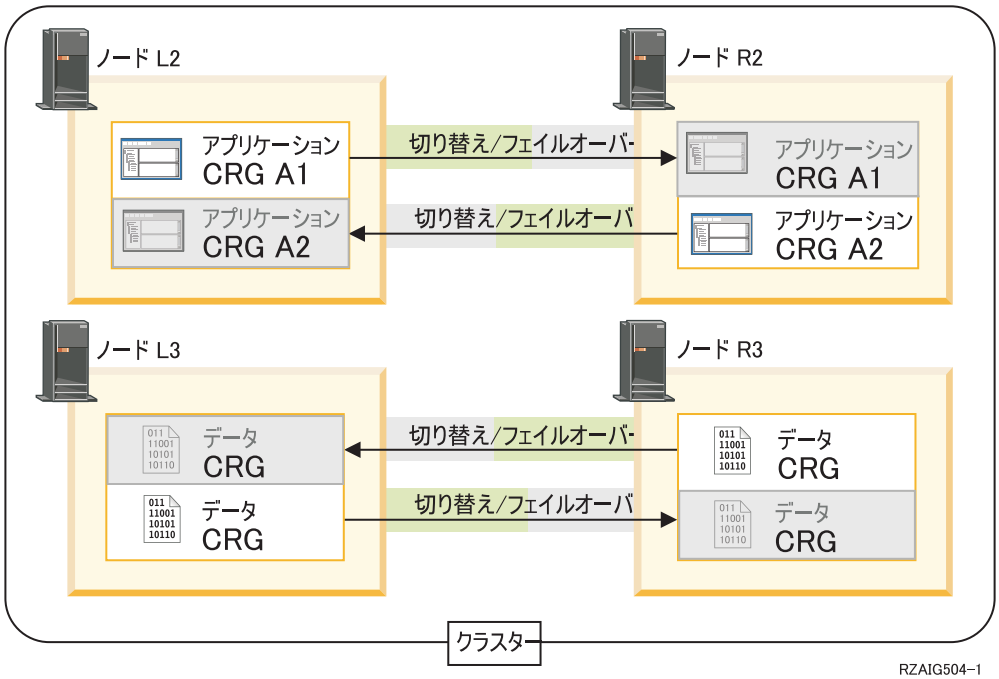
注: ノード L の停止中、ノード R にも障害が生じた場合にバックアップが存在しないので、システムの可用性は無くなります。ノード L が回復し、クラスターに再結合する際、両方のクラスター・リソース・グループのバックアップが作成されます。その時点で、ノード R からノード L への複製が行われます。プライマリー・ノードの役割を再びノード L に与えたい場合、管理切り替えを実行する必要があります。

例: 4 ノード・クラスター

この例を使って、4 つのノードからなるさらに複雑なクラスターを作成します。

この構成例では、以下が用意されています。

- 両方向の複製およびフェイルオーバー
- 3 層の環境
- 独立して移動するアプリケーションおよびデータ
- ワークロードの異なる通常の実動に対してバックアップが使用される



4 ノードの例は、iSeries クラスターで柔軟性がどのようにさらに増強されるかを示します。 2 つのアプリケーション・クラスター・リソース・グループ (A1 と A2) および 2 つのクラスター・リソース・グループ (D1 と D2) があります。 D1 に関連したデータは、A1 に関連したアプリケーションの重要なデータ

です。 D2 に関連したデータは、A2 に関連したアプリケーションの重要なデータです。これは 3 層の環境なので、アプリケーションは 2 番目の層 (ノード L2 とノード R2) に存在し、データは 3 番目の層 (ノード L3 とノード R3) に分離されています。

クラスター・リソース・グループ (CRG)	プライマリー	バックアップ
アプリケーション CRG A1	L2	R2
アプリケーション CRG A2	R2	L2
データ CRG D1	R3	L3
データ CRG D2	L3	R3

これにより、アプリケーション・レベルおよびデータ・レベルの両方で、相互のテークオーバー機能が使用可能になります。 4 つのノードはすべて、通常の実動で使用されます。それらはクラスター内の他のシステムをバックアップするためにも使用されています。 2 つのアプリケーションおよび関連データは、このクラスター内で常に使用可能でなければなりません。単一のノードが停止しても、その可用性は中断されません。さらに、アプリケーション・レベルのノードとデータ・レベルのノードとが同時に停止しても、その可用性は中断されません。

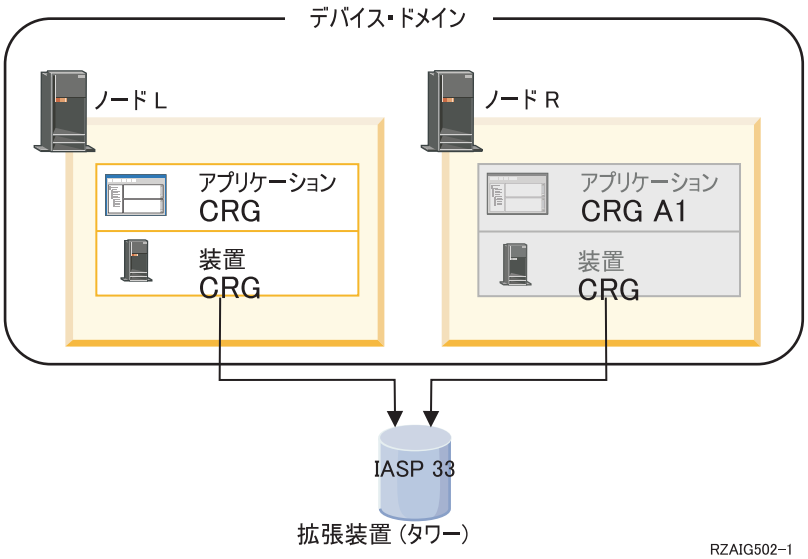
注: どちらの場合も、ノードのダウン中には一部のクラスター・リソースが複製されないという危険性のある状態で、クラスターが実行されます。重要なクラスター・リソースについては複数のバックアップを行うことで、この問題に対処できます。

例: 独立ディスク・プールを使用する切り替えディスク・クラスター

切り替えディスク・テクノロジーを使用するクラスターは、 データを複製するための代替手段を提供します。 切り替えディスク・クラスターでは、データは独立ディスク・プール (独立 ASP とも呼ばれる) に通常含まれています。

この構成例では、以下が用意されています。

- 使用されていない待機サーバーを持つ切り替え可能な独立ディスク・プール。独立ディスク・プールは、切り替え可能なディスク装置の集合に含まれています。
- 2 層の環境
- アプリケーションとデータの並行した移動
- このアプリケーションのデータに関連しない異なるワークロードで使用されるバックアップ
- データは複製されない。データのコピー 1 つだけがこのクラスター内に存在する。



この例を使用すると、ノード L とノード R は同じデバイス・ドメインに属しています。 ノード L は現在 2 つのクラスター・リソース・グループ (アプリケーション CRG および装置 CRG) のプライマリー・ノードとして機能しています。ノード R は両方のクラスター・リソース・グループに対する最初の (そして唯一の) バックアップです。 装置 CRG に関連したデータは、外部拡張装置 (タワー) などの切り替え可能なリソースに含まれています。アプリケーション CRG に関連した適切なアプリケーション情報は、そのタワーに保管されているか、またはノード L からノード R に複製されています。 ノード L に障害が生じたか、または管理のために取り外す必要が生じた場合、ノード R が両方のクラスター・グループに対してプライマリー・ノードとなります。ノード R はアプリケーション CRG に定義されたインターネット・プロトコル (IP) を引き継ぎます。 ノード R は装置 CRG に定義された切り替え可能なリソースの所有権も受け入れます。

注: ノード L の停止中、ノード R にも障害が生じた場合にバックアップが存在しないので、システムの可用性は無くなります。ノード L が回復し、クラスターに再結合する際、両方のクラスター・リソース・グループのバックアップが作成されます。プライマリー・ノードの役割を再びこのノードに与えたい場合、管理切り替えを実行する必要があります。

関連概念

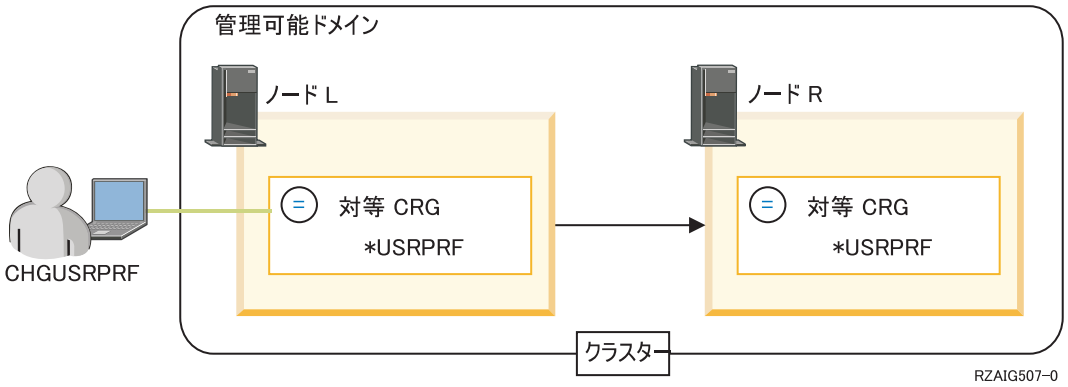
例: 独立ディスク・プールの構成

例: 対等リソースを管理するクラスター管理可能ドメイン

クラスター内のリソースをモニターするのに使用されるクラスター管理可能ドメインの構成例を示しています。

この構成例では、以下が用意されています。

- 2 ノード・クラスター
- ドメイン・ノード・リスト中に 2 つのノードをもったクラスター管理可能ドメイン
- ドメイン内で同期される、ユーザー・プロファイル用のモニター対象リソース (MRE)



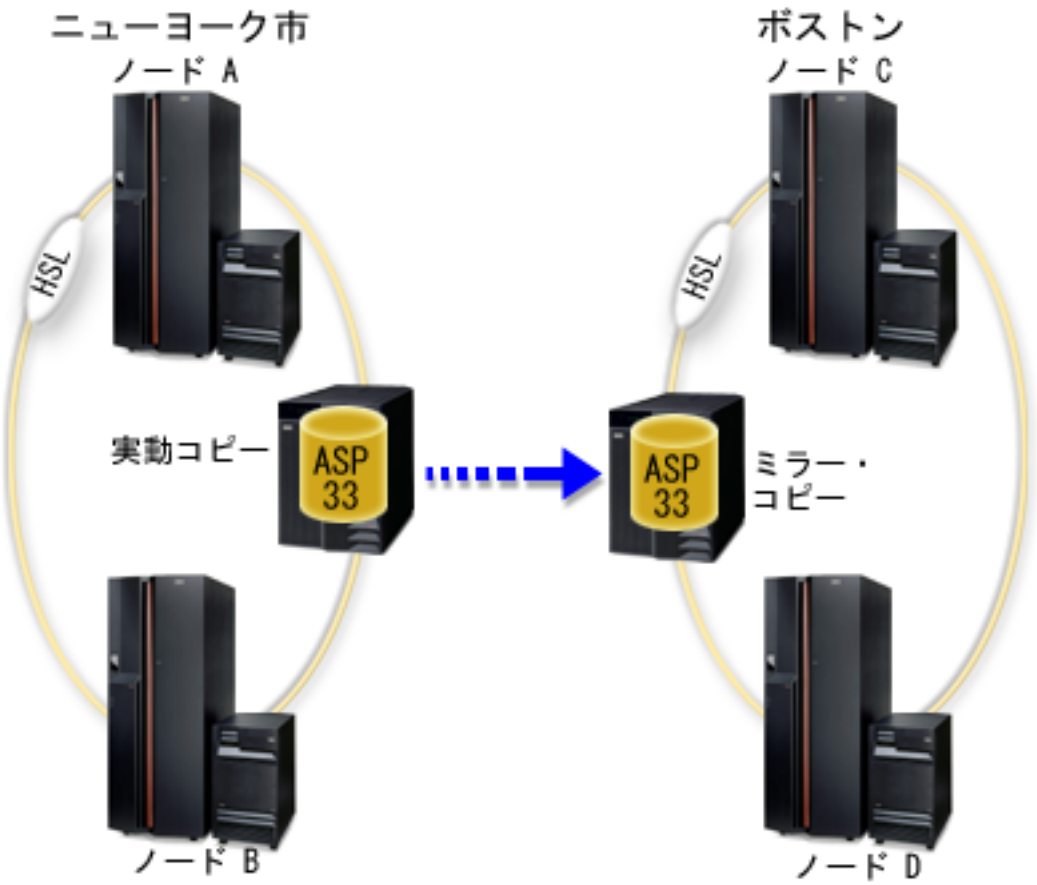
この例では、管理者は、クラスター全体を通してユーザー・プロファイルの整合性を保って、作成したクラスター管理可能ドメインで、ユーザー・プロファイルに加えられた変更をモニターして同期したいと考えています。クラスター管理可能ドメインは、ノード L とノード R からなる対等 CRG で表わされています。モニター対象リソース項目が、ユーザー・プロファイル用にクラスター管理可能ドメインに追加されます。この例では、すべてのユーザー・プロファイル属性は、MRE の追加時に指定されたものです。したがって、ノード L またはノード R でユーザー・プロファイルのいずれかの属性が変更されると、CRG の開始後にその変更は、ドメイン内のアクティブ・ノードに自動的に伝搬されます。

管理者がこの例をセットアップするのに行ったステップは次のとおりです。

1. ノード L およびノード R でクラスターを作成します。
2. ノード L およびノード R でクラスター管理可能ドメインを作成します。
3. ユーザー・プロファイルを表す MRE を追加します。
4. クラスター管理可能ドメインを表す対等 CRG を開始します。
5. ノード L またはノード R でユーザー・プロファイルを変更します。もう一方のノード上のユーザー・プロファイルは、クラスター管理可能ドメインによって自動的に変更されます。変更が正常に完了にすれば、モニター対象リソースのグローバル状況は整合した状態になります。

例: 地理的ミラーリングを使用する独立ディスク・プール

以下の例は、地理的ミラーリングを構成する 1 つの方法を示しています。 ノード A とノード B はニューヨーク市に置かれています。 ノード C とノード D はボストンに置かれています。 4 つのノードはすべて、同じリカバリー・ドメイン内で構成されています。ノード A と B の間で、実動コピーを切り替えることができます。ノード C と D の間で、ミラー・コピーを切り替えることができます。すべてのノードが同じリカバリー・ドメイン内にあるため、ニューヨークの起動システムは、ボストンの受動システムと役割を交換して、ボストンが実動コピーをホストすることもできます。



この会社は、リカバリー・ドメイン内のノードに対して次のような役割を定義しました。

ノード	役割
ノード A	プライマリー
ノード B	バックアップ 1
ノード C	バックアップ 2
ノード D	バックアップ 3

ニューヨークで自然災害が発生した場合、ボストンのノード C は、そのミラー・コピーを実動コピーにアップグレードしてプライマリー・ノードになります。 ノード C は、地理的ミラーリングの起動システムになりますが、地理的ミラーリングは中断されます。その理由は、ニューヨークの自然災害が原因でターゲット・ノードはなくなるからです。ニューヨークのサイトが回復したら、ノード A はバックアップ・ノードになり、その旧実動コピーはミラー・コピーになります。

クラスターのトラブルシューティング

クラスターに特有の問題のエラー回復ソリューションがあります。

クラスターが正常に機能していないと思える状況が生じることがあります。このトピックでは、クラスターに関して生じる可能性のある問題についての情報を示します。

クラスターの問題が存在するかどうかの判別

クラスターに関する問題を診断するには、ここから始めてください。

クラスターが正常に機能していないと思える状況が生じることがあります。問題が存在すると思われる場合、以下のステップを使用して問題が存在するかどうか、および問題の本質を判別することができます。

- **システム上でクラスタリングがアクティブかどうかの判別**

  クラスター・リソース・サービスがアクティブかどうかを判別するには、QSYSWRK サブシステム内で 2 つのジョブ - QCSTCTL および QCSTCRGM - を探してください。これらのジョブがアクティブであれば、クラスター・リソース・サービスもアクティブです。 iSeries ナビゲーターの実行管理機能を使用して、サブシステム内のジョブを表示できますが、WRKACTJOB (活動ジョブ処理) コマンドを使用して同じことを行うこともできます。また、DSPCLUINF (クラスター情報の表示) コマンドを使用して、クラスターの状況情報を表示することもできます。

  – 追加のジョブ・クラスター・リソース・サービスもアクティブであることがあります。 クラスター・リソース・サービス・ジョブ構造を使うと、クラスター・リソース・サービス・ジョブがどのようにフォーマットされるかに関する情報が提供されます。

- **問題を示しているメッセージを探します。**
  – QSYSOPR 内で応答を待っている照会メッセージを探します。
  – QSYSOPR 内でクラスター問題を示しているエラー・メッセージを探します。通常、これらは CPFBB00 から CPFBBFF までの範囲にあります。
  – 活動記録ログを表示して (DSPLOG CL コマンド)、クラスター問題を示すメッセージを探します。通常、これらは CPFBB00 から CPFBBFF までの範囲にあります。
- **クラスター・ジョブのジョブ・ログを調べて、重大エラーを探します。**

  これらのジョブは最初にロギング・レベルが (4 0 *SECLVL) に設定されているので、必要なエラー・メッセージを見つけることができます。これらのジョブおよび出口プログラム・ジョブのロギング・レベルが適切に設定されていることを確認してください。クラスタリングがアクティブではない場合でも、クラスター・ジョブおよび出口プログラム・ジョブのスプール・ファイルを探すことができます。

- **ある種のハング状態が疑われる場合、クラスター・ジョブの呼び出しスタックを調べます。**

  何かの種類の DEQW (デキュー待機) に問題がないかどうかを判別してください。問題がある場合、各スレッドの呼び出しスタックを調べて、いずれかのスレッドの呼び出しスタックに getSpecialMsg が含まれていないかを確かめます。

- **クラスター垂直ライセンス内部コード (VLIC) のログ項目を調べます。**

  これらのログ項目には 4800 メジャー・コードがあります。

- **NETSTAT コマンドを使用して、通信環境に異常がないかどうかを判別します。**

NETSTAT は TCP/IP ネットワーク経路、インターフェース、TCP 接続、 およびシステム上の UDP ポートの状態に関する情報を戻します。

- NETSTAT オプション 1 (TCP/IP インターフェース状況の処理) を使用して、クラスタリングに使用するために選択した IP アドレスが「活動中」の状況であることを確認します。さらに、LOOPBACK アドレス (127.0.0.1) も活動中であることを確認します。
- NETSTAT オプション 3 (TCP/IP 接続状況の処理) を使用して、ポート番号を表示します (F14)。 ローカル・ポート 5550 は「接続待機」状況となっているはずです。このポートは、STRTCPSVR *INETD コマンドを介して開く必要があります。このことは、QTOGINTD (ユーザー QTCP) ジョブがアクティブのジョブ・リスト内に存在することによって証明されます。クラスタリングがノード上で開始した場合、ローカル・ポート 5551 は開いていて、「*UDP」状況でなければなりません。クラスタリングが開始していない場合、ポート 5551 は閉じている必要があります。これが開いていると、対象ノードでクラスタリングが正常に開始できなくなります。

- ping を使用します。クラスター・ノードの開始を試みたときにそのノードを ping できない場合、内部クラスタリング・エラー (CPFBB46) が表示されます。
- **CLUSTERINFO マクロを使用して、クラスター内のノード、さまざまなクラスター・リソース・グループ内のノード、および現在使用されているクラスター IP アドレスについてのクラスター・リソース・サービスのビューを表示します。**

クラスターが見込みどおりに機能しない場合に、ここで矛盾が見つかれば、問題領域を正確に見極めるのに役立ちます。 CLUSTERINFO マクロの使用法と解釈の詳細は、 145 ページの『CLUSTERINFO マクロを使った問題の調査』を参照してください。

関連概念

129 ページの『ジョブ構造とユーザー待ち行列』
クラスターを管理するには、ジョブ構造とユーザー待ち行列についての知識が必要です。

関連タスク

サブシステム内のジョブの表示

関連資料

活動ジョブ処理 (WRKACTJOB)

クラスター情報の表示 (DSPCLUINF)

クラスターに関する回復情報の収集

クラスターの処理 (WRKCLU) コマンドを使って、クラスターの全体像に関する情報を収集することができます。 その情報は、エラーの解決で活用することができます。

クラスターの処理 (WRKCLU) コマンドは、クラスターのノードとオブジェクトの表示と処理に使用します。このコマンドを実行すると、「クラスターの処理」画面が表示されます。クラスター内のノードとクラスター情報の表示以外に、クラスター情報を表示してクラスターに関するデータを収集するのにも、このコマンドを使用することができます。

エラー回復情報を収集するには、次のようなステップを行います。

1. 文字ベース・インターフェースで、`WRKCLU OPTION(OPTION)` と入力します。次のようなオプションを指定して、どのクラスター状況情報を処理したいかを指示することができます。

   ***SELECT**
   「クラスターの処理」メニューを表示します。

***NODE**
クラスター内のノードのリストを示した「クラスター情報」パネルを表示します。

***CFG** クラスターの全構成パラメーターを表示します。また、クラスター・リソース・グループに関する詳細情報を入手する機能も備えています。

***CRG** クラスター内のクラスター・リソース・グループのリストを表示します。

***SERVICE**
クラスター内のすべてのクラスター・リソース・サービス・ジョブに関連したトレースおよびデバッグの情報を収集します。この情報は、各クラスター・リソース・サービス・ジョブごとにメンバーと一緒にファイルに書き込まれます。このオプションは、サービス・プロバイダーから指定された場合のみ使用してください。これは、クラスター・トレースのダンプ (DMPCLUTRC)のプロンプト・パネルを表示します。

クラスター・トレースのダンプ (DMPCLUTRC) コマンドでの問題の調査

クラスターでの問題を判別して解決するには、クラスター・トレースのダンプ (DMPCLUTRC) コマンドを使用します。

クラスター・トレースのダンプ (DMPCLUTRC) コマンドは、クラスター・ジョブが完了したかどうかや、ジョブが現在処理している内容を判別するのに役立ちます。このコマンドは、クラスター関連のトレースとデバッグの情報をファイルにダンプします。その情報は、1 つ以上のクラスター・ノードでローカル・ダンプされます。このコマンドを使って、1 つまたはすべてのクラスター・リソース・サービス (CRS) ジョブをダンプすることができます。ダンプされる各 CRS ジョブは、そのファイル内にファイル・メンバーをもっています。ファイル・メンバーの名前は、CRS ジョブの名前になります。このコマンドが出力を生成するには、クラスタリングがアクティブになっていなければなりません。アクティブな CRS ジョブをもったノードだけが、出力を生成します。ダンプされる情報は、ユーザー・トレースから取り出され、他の情報はクラスター・オブジェクトから取り込まれます。ダンプされる情報量は、ダンプ・レベルで決まります。そのようなダンプ・レベルには、基本情報、エラー情報、通知情報、および冗長情報があります。ダンプ・レベルによって、ファイルに送信される情報量が決まります。多くの場合、各ユーザーのニーズに基づいて IBM サービス技術員からどのレベルを指定すればよいかが指示されますが、たいていのトラブルシューティングのケースでは LEVEL(*ERROR) で十分です。各自のケースにはどのレベルが適しているか確かめたい場合、IBM サービス技術員にお問い合わせください。

トレース結果の解釈

トレース結果を分析すれば、どのクラスター・ジョブが原因でプロトコルが待機中になっているかといった、クラスターが行っている作業内容を知ることができます。ユーザー・トレースから得た出力には、一連の等号 (=) から成る区切り線が示されます。 DMPCLUTRC を何回発行したかによって、このファイルに表示される区切り線の数が影響を受けます。 1 つのファイル内で複数回の DMPCLUTRC 呼び出しがあることがあります。最後の一連のスタック・ダンプに、最新情報が収められています。場合によっては、CRG ジョブが 2 つのグループ分かれることがあります。それぞれのグループごとに、別々のダンプ・セクションがファイル内に設けられます。

以下のクラスター・トレース・ダンプ結果の例では、MYCLUSTER というクラスター内に 2 つのノード (SYSTEM1 および SYSTEM2) があります。 これには、MYCRG という名前の CRG が 1 つあります。どちらのノードも、MYCRG のリカバリー・ドメイン内に置かれています。あるユーザーが STRCRG CL コマンドを発行しましたが、プロセスから結果が戻されるのに長時間かかっています。このユーザーは、別のワークステーションでコマンド行インターフェースに DMPCLUTRC CLUSTER(MYCLUSTER)CRG(*ALL) LEVEL(*ERROR) FILE(MYFILE) と入力しました。

この例の場合、DMPCLUTRC コマンドからの出力は、メンバー MYCRG 内の MYFILE というファイルに入れられます。 メンバー MYCRG の内容を分かりやすく説明するために、それはいくつかのセクションに分割されました。説明されている情報を特定できるよう、すべてのセクションを通して番号は括弧付きで強調表示されています。クラスター問題のトラブルシューティングの際に、このような詳細を参考にすることができます。

注: 縦の省略符号は、トレースの一部が除去されて、出力には表示されないことを示します。

DMPCLUTRC の結果のセクション 1。

```
User Trace Dump for job 073586/QSYS/MYCRG. Size: 300K, Wrapped 0 times.

--- 08/22/2005 16:43:32 ---
(1a) 00000006:658536 Main thread handle 2
(1b) 00000008:748016 Work thread 1 handle 13
(1b) 00000007:754576 Work thread 2 handle 11
--- 08/22/2005 16:46:04 ---
 00000008:269608 CSTDAMBR 1115: WaitForMsg 4 1005 CPFBB3C
--- 08/22/2005 16:48:17 ---
       00000006:925112
(1c)        DMPCLUTRC  Node SYSTEM1  Group MYCRG
    ================================================
```

最初のセクションには、クラスター・ジョブのスレッド番号とハンドルが示されます。クラスター・ジョブには、複数のスレッドが付帯していることがあります。この例では、すべての作業の入り口であるメイン・スレッド (1a) 用に 1 つと、作業スレッド (1b) 用に 2 つあります。また、このセクションには、どのシステムからこのトレースが取り出されたかと、それがどのクラスター・ジョブに属するかに関する情報 (1c) も示されます。

DMPCLUTRC の結果のセクション 2。

```
00000006:925168 Stack Dump For Target Thread: Handle 2 (0x00000002)
       00000006:925192 Stack:
 (2aa) Main Thread Stack MYCRG
       00000006:925256 Stack:  Library    / Program     Module      Stmt    Procedure
       00000006:933432 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   0     : _CXX_PEP__Fv
       00000006:933488 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   46    : main
       00000006:933536 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   65    : completeStartup__FP8CstDAMbr
       00000006:933584 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   26    : mainQueueProcessLoop__FP8Cs
                                  DAMbr
       00000006:933616 Stack:  QSYS       / QCSTCMN     CSTDAMBR    57    : processQueueMsgs__8CstDAMbrF
                                  Q2_8CstDAMbr13CstQueueIndex
       00000006:933664 Stack:  QSYS       / QCSTCMN     CSTDAMBR    53    : processMsg__8CstDAMbrFP6CstM
                                  sg
       00000006:933712 Stack:  QSYS       / QCSTCMN     CSTDAMBR    17    : callFnPtr__8CstDAMbrFPQ2_8Cs
                                  tDAMbr19MsgFunctionPtrEntryP6
       00000006:933744 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   94    : crgDump__FP6CstMsgP8CstDAMbr
       00000006:933792 Stack:  QSYS       / QCSTCMN     CSTACK      95    : CstAckQueryMsg
       00000006:933832 Stack:  QSYS       / QP0ZCPA     QP0ZUDBG    3     : Qp0zDumpTargetStack
00000006:933864 Stack:  QSYS       / QP0ZSCPA    QP0ZSDBG    12    : Qp0zSUDumpTargetStack
       00000006:934016 Stack:  Exception         In Stack Dump Code
       00000006:934040 Stack:  thread is likely terminated or no longer running the same code as the
captured stack
       00000006:934080
  (2a) Work Thread Index 1  Group MYCRG      Last or current values
  (2e) 00000006:934112   Request handle 8E3E1002 EE3218A1 824F0004 AC000456
  (2c) 00000006:934136   SPI name QcstStartClusterResourceGroup
       00000006:934160 (2g)   POF 10, Completed ack round 1 (2i)
       00000006:934176  (2o) In waitForJobEnd QDFTJOBD  MYCLUSTER     073590
       00000006:934216         Node         Ack Status      POF     (2bb) Nack Msg Id
       00000006:934240 (2n)    SYSTEM1   (2cc) Ready
       00000006:934272     SYSTEM2   Ack    10 (2k)
       00000006:934296   Messages
```

```
|         00000006:934320 Stack Dump For Target Thread: Handle 13 (0x0000000d)
|         00000006:934344 Stack:  Work Thread 1 Stack MYCRG
|         00000006:934792 Stack:  Library    / Program     Module      Stmt    Procedure
|         00000006:934840 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   9     : workThreadRoutine__FPv
|         00000006:934888 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   28    : workQueueProcessLoop__FP8Cst
|                                    DAMbr
|         00000006:941688 Stack:  QSYS       / QCSTCMN     CSTDAMBR    57    : processQueueMsgs__8CstDAMbrF
|                                    Q2_8CstDAMbr13CstQueueIndex
|         00000006:941696 Stack:  QSYS       / QCSTCMN     CSTDAMBR    33    : processMsg__8CstDAMbrFP6CstM
|                                    sg
|         00000006:941712 Stack:  QSYS       / QCSTCMN     CSTDAMBR    17    : callFnPtr__8CstDAMbrFPQ2_8Cs
|                                    tDAMbr19MsgFunctionPtrEntryP6
|         00000006:941728 Stack:  QSYS       / QCSTCMN     CSTACK      3     : CstStripOffHeaderMsgPart
|         00000006:941736 Stack:  QSYS       / QCSTCMN     CSTDAMBR    53    : processMsg__8CstDAMbrFP6CstM
|                                    sg
|         00000006:941752 Stack:  QSYS       / QCSTCMN     CSTDAMBR    17    : callFnPtr__8CstDAMbrFPQ2_8Cs
|                                    tDAMbr19MsgFunctionPtrEntryP6
|         00000006:970888 Stack:  QSYS       / QCSTCRGS2   CSTCRGSS    39    : startCrg
|         00000006:970912 Stack:  QSYS       / QCSTCRGS2   CSTCRGSS    344   : doMessageProcessing__FP6CstM
|                                    sgP8CstDAMbr
|         00000006:970928 Stack:  QSYS       / QCSTCRGS2   CSTCRGSS    57    : doExitPgmPhase__FP6CstMsgP8C
|                                    stDAMbr
|         00000006:981984 Stack:  QSYS       / QCSTCMN     CSTDAMBR    52    : waitForJobEnd__8CstDAMbrFPA2
|                                    6_ci
|         00000006:982000 Stack:  QSYS       / QCSTCMN     CSTDAMBR    73    : waitForSpecialMsg__8CstDAMbr
|                                    FP17CstSpecialMsgListPA8_ciT3
|         00000006:982016 Stack:  QSYS       / QC2UTIL1    QC2MI3      1     : (2dd) deq
|         00000006:982136 Stack:  Exception         In Stack Dump Code
|         00000006:982136 Stack:  thread is likely terminated or no longer running the same code as the
|  captured stack
|         00000006:982160
|  (2b)Work Thread Index 2  Group MYCRG      Last or current values
|  (2f)00000006:982176   Request handle D9C3C8C3 E2E3F5F2 0003 0000
|  (2cc)00000006:982176   SPI name
|         00000006:982184 (2h)  POF 0, (2d)Completed ack (2j)round 0
|         00000006:982184   In getNextWorkMsg
|         00000006:982208   Node         Ack Status      POF     Nack Msg Id
|  (2l) 00000006:982208     SYSTEM1   Ready
|  (2l) 00000006:982232     SYSTEM2   Ready
|         00000006:982248   Messages
|         00000006:982256 Stack Dump For Target Thread: Handle 11 (0x0000000b)
|         00000006:982256 Stack:  Work Thread 2 Stack MYCRG
|         00000006:982344 Stack:  Library    / Program     Module      Stmt    Procedure
|         00000006:982360 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   9     : workThreadRoutine__FPv
|         00000006:982376 Stack:  QSYS       / QCSTCRGJOB  CSTCRGJOB   28    : workQueueProcessLoop__FP8Cst
|                                    DAMbr
|         00000006:982392 Stack:  QSYS       / QCSTCMN     CSTDAMBR    51    : processQueueMsgs__8CstDAMbrF
|                                    Q2_8CstDAMbr13CstQueueIndex
|  (2m) 00000006:982400 Stack:  QSYS   / QCSTCMN     CSTDAMBR    105   : getNextWorkMsg__8CstDAMbrFv
|         00000006:982416 Stack:  QSYS       / QC2UTIL1    QC2MI3      1     : deq
|         00000006:982480 Stack:  Exception        In Stack Dump Code
|         00000006:982480 Stack:  thread is likely terminated or no longer running the same code as the
|  captured stack
```

2 番目のセクションには、クラスター・ジョブに属する各スレッドの呼び出しスタックが示されます。たいていは、メイン・スレッドには、完了したばかりの DMPCLUTRC が示されます (2aa)。 作業スレッド (2a および 2b) には、クラスター・ジョブで何が起きているかを判別するのに役立つトレース情報が収容されます。このセクションには、呼び出しスタックに関する詳細情報が示されます。そのような詳細には、SPI 名 (2c)、完了済み確認通知 (ACK)(2d)、関連 API のハンドル (2e)、または最後に完了した要求ハンドル (2f)、現在の障害地点 (POF) (2g および 2h)、現在の確認通知ラウンド (ACK) (2i および 2j)、確認通知されたノード (ACK) (2k および 2l) があります。

現在の障害点 (POF) は、コードのどこに現在のプロトコルがあるかを示す内部値ですが、これは、必ずしも障害が起きたことを示すわけではありません。 Ack は、プロトコルのこの部分をノードで正常に完了

| し、他のすべてのノードからの確認応答 (ACK) または Nack 待ちになっていることを意味します。 Nack は、プロトコルのこの部分をノードで正常に完了できないので、他のすべてのノードからの応答待ちになっていることを意味します。 Nack のメッセージ ID は、次の列に付記されます (2bb)。これは、発信元の RIQ に送信されるのと同じメッセージです。プロトコルの途中でノードに障害が起きた場合、その状況は Fail と表示され、さらに、プロトコルとノードに応じて、Nack とみなされることもみなされないこともあります。非アクティブの Ack 状況とは、ノードがプロトコルに関与しなかったことを意味します。 Ready の値は、ノードからまだ応答がないことを意味します。スレッドが getNextWorkMsg (2m) にあるとは、そのスレッドが処理する作業待ちになっていることを意味します。

| プロシージャー名の読み取りを最下部から開始し、呼び出しスタックを上方にたどっていきます。このサンプル・ファイルには、waitForSpecialMsg、waitForJobEnd、および doExitPgmPhase の付いた deq が示されます (2dd)。これは、プロトコルが、処理を先に進める前に、出口プログラムの完了待ちになっていることを示します。 Ack 状況 (2k) から、プロトコルがどのノード待ちになっているかを判別することができます。この例では、ノード SYSTEM1 待ちになっています (2n)。修飾ジョブ名 (2e) は、システムが待ち受けているジョブを示します。ジョブ名を確かめたら、ジョブを処理して、遅延の原因をトラブルシューティングすることができます。この場合、ジョブはジョブ待ち行列内でずっと待機中であるか、ジョブは稼働しているが処理に時間がかかっているか、またはジョブはロックされているオブジェクト待ちになっていることがその原因と考えられます。

| この例では、プロトコルは出口プログラムの完了待ちになっています。プロトコルが出口プログラムまたは変更ジョブの完了待ちになっているかどうかを判別するもっと簡単な方法としては、最初のセクションを調べて、waitForJobEnd (2o) になっているかどうかを確認します。待機の対象になっているジョブ名が、同じ行上にあります。このようにすれば、スタックを順にたどって調べる必要がなくなります。

DMPCLUTRC の結果のセクション 3。

```
5722SS1 V5R4M0  060210    AS/400 DUMP    073586/QSYS/MYCRG       08/22/05 16:48:18        PAGE    1
DMPSYSOBJ PARAMETERS
(3a)OBJ- MYCRGAIX                        CONTEXT- QTEMP
TYPE- *ALL SUBTYPE-*ALL
OBJECT TYPE-           INDEX                                          *CRGM
NAME-        MYCRGAIX                        TYPE-          0E   SUBTYPE-          A5
LIBRARY-     QTEMP     006B8A19B00C9E807000  TYPE-          04   SUBTYPE-          C1
CREATION-    08/22/05  16:43:32              SIZE-          0000007000
ATTRIBUTES-          0000                    ADDRESS-       C7FE286F04 000000
.
.
.
.POINTERS-
  NONE
OIR DATA-
  NONE
END OF DUMP
                                  * * * * *  E N D  O F  L I S T I N G  * * * * *
```

| ここに示した 3 番目のセクションは、クラスター・ジョブに関する情報を示す内部オブジェクトです。この例では、それは MYCRGAIX (3a) という内部索引です。ここに示される情報は、上記のセクション 2 のものよりはるかに読みやすい情報です。

DMPCLUTRC の結果のセクション 4。

```
5722SS1 V5R4M0  060210   AS/400 DUMP     073586/QSYS/USER     08/22/05 16:48:18        PAGE    1
DMPSYSOBJ PARAMETERS
(4a)OBJ- MYCRGTQ                         CONTEXT- QTEMP
TYPE- 0A   SUBTYPE-EF
OBJECT TYPE-           QUEUE                                          *QTQ
NAME-        MYCRGTQ                         TYPE-          0A   SUBTYPE-          EF
LIBRARY-     QTEMP     006B8A19B00C9E807000  TYPE-          04   SUBTYPE-          C1
```

```
| CREATION-    08/22/05  16:43:32             SIZE-          000002C000
| ATTRIBUTES-          0000                   ADDRESS-       CC6765CAA2 000000
| QUEUE ATTRIBUTES-
| .
| .
| .
| .POINTERS-
|   000010   SPP 1A EF MYCRG     QSYS      073586                                                  00002160  0000
|   000020   SPP 1A EF MYCRG     QSYS      073586                                                  00001540  8000
|   000030   SPP 1A EF MYCRG     QSYS      073586                                                  000016E0  0000
|   000040   SPP 1A EF MYCRG     QSYS      073586                                                  00001690  0000
|   000050   SPP 1A EF MYCRG     QSYS      073586                                                  000016A0  0000
|   000070   SPP 1A EF MYCRG     QSYS      073586                                                  00002160  0000
| OIR DATA-
|   NONE
| END OF DUMP
|                                    * * * * *  E N D  O F  L I S T I N G  * * * * *
```

| ここに示した 4 番目のセクションは、トレース待ち行列 (4a) と呼ばれます。 この場合、それは MYCRGTQ という名称です。ここには、ジョブを実行させるためにクラスターが何をし、各ジョブがその要求にどのように応答したかに関する情報が示されます。

| **注:** どのメッセージについても、ここでは完全には説明されていません。

DMPCLUTRC の結果のセクション 5。

```
| 5722SS1 V5R4M0  060210  AS/400 DUMP  073586/QSYS/MYCRG     08/22/05 16:48:18        PAGE    1
| DMPSYSOBJ PARAMETERS
| (5a) OBJ- MYCRG                          CONTEXT- QUSRSYS
| TYPE- 19   SUBTYPE-2C
| OBJECT TYPE-           SPACE                                        *CRG
| NAME-        MYCRG                          TYPE-          19   SUBTYPE-          2C
| LIBRARY-     QUSRSYS                        TYPE-          04   SUBTYPE-          01
| CREATION-    08/17/05  07:16:40             SIZE-          0000002000
| OWNER-       MYCLUSTER                          TYPE-          08   SUBTYPE-          01
| ATTRIBUTES-          0800                   ADDRESS-       1EC687A1F3 000000
| SPACE ATTRIBUTES-
| .
| .
| .
| END OF DUMP
|                                    * * * * *  E N D  O F  L I S T I N G  * * * * *
```

| 5 番目のセクションには、CRG オブジェクト (5a) に関する情報が示されます。

CLUSTERINFO マクロを使った問題の調査

CLUSTERINFO マクロは、ノード、CRG 、およびアクティブな IP アドレスに関してクラスター・リソース・サービスが収容している情報を表示します。

CLUSTERINFO マクロは、現行クラスターに関する情報のスナップショットを作成します。コマンドは、クラスタリング・オブジェクトを順にナビゲートし、クラスターの説明をローカル・ノード上で作成します。 CLUSTERINFO マクロは、さまざまなクラスター・オブジェクトのフライト・レコーダーとして働き、クラスター内の問題源の判別に役立ちます。 CLUSTERINFO マクロにアクセスするには、次のようなステップを行います。

1. 文字ベースのインターフェースで、STRSST と入力します。
2. 保守ツールのユーザー・プロファイルを使ってサインオンします。
3. 「保守ツールの開始」で、オプション 1 (保守ツールの開始) を選択します。
4. オプション 4 (表示/変更/ダンプ) を選択します。

5. オプション 2 (プリンターへのダンプ) を選択します。
6. オプション 2 (ライセンス内部コード (LIC) データ) を選択します。
7. オプション 14 (拡張分析) を選択します。
8. CLUSTERINFO マクロ・オプションの前に 1 を入力します。 Enter を押します。

CLUSTERINFO マクロを表示した後、-H オプションを使って、このマクロ内で使用可能なすべてのオプションのヘルプを表示します。 以下の使用法ダイアグラムは、CLUSTERINFO マクロで利用できる各オプションを説明しています。

表 21. CLUSTERINFO マクロのオプション

オプション	説明
-H	オプションのヘルプ画面を表示します。
-A	すべての情報を表示します。
-FR	フライト・レコーダー項目を表示します。
-HB	ハートビート情報を表示します。
-PERF	パフォーマンス・カウンターを表示します。
-Q	出力メッセージ待ち行列の状況を表示します。
-G	すべての CC ブロードキャスト・グループの表示の抑止を解除します。
-T	チューニング・パラメーターを表示します。
-M	すべての DA のマトリックスを表示します。
-DP	dataPort 情報を表示します。

CLUSTERINFO マクロの結果の解釈

この例では、-A オプションが指定されているので、すべてのフィールドがダンプされます。デバッグのメイン・ツールは、フライト・レコーダーです。このようなフライト・レコーダーは、クラスター・ノードの終了または削除の時点で削除されることに注意してください。問題分析の場合、クラスターの終了または削除より前に CLUSTERINFO マクロを実行する必要があります。場合によっては、クラスターの削除または終了の時点で、フライト・レコーダーは vlog に書き込まれることがあります。フライト・レコーダーは、クラスターの構造とパフォーマンスに影響を与えるさまざまなイベントを記録します。フライト・レコーダーのデータの詳細な解釈は、本書では取り上げていません。

注: 縦の省略符号は、結果の一部が除去されて、出力には表示されないことを示します。

CLUSTERINFO マクロ結果のセクション 1。

```
DISPLAY/ALTER/DUMP    CLUSTERINFO -NEW2     08/23/05  13:36:37   PAGE      1
Running macro: CLUSTERINFO               -A
Use -H for command information
Cluster Name   : MYCLUSTER
Local Node Name: SYSTEM1
CC/CTS Version : 5
Macro Timestamp: 08/23/05 13:36:37.079
```

セクション 1 には、クラスター名、クラスター・バージョン、およびレポートの生成時のタイム・スタンプなどの、クラスターに関する一般情報が示されます。この例では、クラスター名は MYCLUSTER、ローカル・ノード名は SYSTEM1 です。

CLUSTERINFO マクロ結果のセクション 2。

```
Cluster Object Addresses
CstcClusterServices Address: DBF08681C9161580
Cluster Address            : FC5B04B0D4001000
Cluster Task Address       : B00010000E932000
Cluster Task Q Address     : DBF08681C9169A00
Clue Group Services Address: CDAB6D0339001000
CC Services Address        : FC5B04B0D4008000
```

セクション 2 では、主要クラスター・オブジェクトのロケーションを指すポインターが示されます。

CLUSTERINFO マクロ結果のセクション 3。

```
Message Statistics
Number of non-fragmented messages: 250
Number of fragmented messages    : 1
Number of fragments              : 7
Number of acks                   : 148
```

セクション 3 には、フラグメント数や確認通知 (acks) 数などの、クラスターに関するメッセージング統計が記載されています。

CLUSTERINFO マクロ結果のセクション 4。

```
Node Map
Node ID : SYSTEM1
 GenesisSubnetId : 9.5.251.0
 CCNode *   : FC5B04B0D4007000
 CCSrvNode *: FC5B04B0D404F000
 Adapter 1  : 9.5.251.46 Primary
  Status    : 0x01 Reachable
  Line Type : 0x09 Ethernet
Node ID : SYSTEM2
 GenesisSubnetId : 9.5.251.0
 CCNode *   : FC5B04B0D4060000
 CCSrvNode *: FC5B04B0D4061000
 Adapter 1  : 9.5.251.47 Primary
  Status    : 0x01 Reachable
  Line Type : 0x09 Ethernet
```

セクション 4 には、ノード・マップ中の現在アクティブなクラスター・ノードがすべて一覧で示されます。この例では、SYSTEM1 および SYSTEM2 の 2 つのアクティブ・ノードがあります。 T

CLUSTERINFO マクロ結果のセクション 5。

```
Subnet Map
Subnet ID: 127.0.0.0
 CCSubnet *            : FC5B04B0D4006000
 CCSrvSubnet*          : FC5B04B0D400C000
 Retries               : 0
 Msg Timeouts          : 0
 Bad Msg Counter       : 0
 Failed Default Address: 0
Subnet ID: 226.5.5.5
 CCSubnet *            : FC5B04B0D405B100
 CCSrvSubnet*          : FC5B04B0D405C000
 Retries               : 0
 Msg Timeouts          : 0
 Bad Msg Counter       : 0
 Failed Default Address: 0
```

セクション 5 には、クラスター内のすべてのサブネット・オブジェクトのリストが示されます。

CLUSTERINFO マクロ結果のセクション 6。

```
Group Map
Group ID: 0x0000000000000001
 Name  : CTS
 CCGroup *   : FC5B04B0D405FF00
 CCSrvGroup *: FC5B04B0D4064B00
 Member Nodes
  SYSTEM1
  SYSTEM2
Group ID: 0x0000000000000002
 Name  : CTS
 CCGroup *   : FC5B04B0D4055100
 CCSrvGroup *: FC5B04B0D4055200
 Member Nodes
  SYSTEM1
  SYSTEM2
.
.
.
```

セクション 6 には、すべての現行クラスター・グループが一覧で示されます。各グループは、分散アクティビティー・グループと呼ばれます。このようなグループは、クラスター内の各アクティブ・ノード上のグループ相互の通信で使用されます。これらのグループの大半が取り扱うのは、ライセンス内部コード (LIC) に関連した処理です。 それは、CTS および BADA というグループ名で識別できます。 また、CCTL (オペレーティング・システムの QCSTCTL ジョブ)、CRGM (オペレーティング・システムの QCSTCRGM ジョブ)、および各クラスター・リソース・グループ (CRG) ジョブ用のグループも示されます。 CRG ジョブのグループには、グループ名が付きません。どのグループも、メンバー・ノードを持ちます。メンバー・ノードとは、該当するグループに代わって互いに通信するノードのことです。

CLUSTERINFO マクロ結果のセクション 7。

```
Partition Map
Partition Map is empty
```

セクション 7 には、SLIC 区画リスト内のすべてのノードが一覧で示されます。

注: これは、XPF 区画ノードと同じ概念ではありません。

CLUSTERINFO マクロ結果のセクション 8。

```
CTS Client List
CTS Client List is empty
```

セクション 8 には、データ・ポートなどの、すべての登録クラスター・クライアントのリストが示されます。

CLUSTERINFO マクロ結果のセクション 9。

```
Flight Recorder        : CSTCSVFR
Flight Recorder Address: DBF08681C9161620
```

セクション 9 には、IPL の時点までシステム上に残るクラスター・サービスのフライト・レコーダー (CSTCSVFR) が示されます。

CLUSTERINFO マクロ結果のセクション 10。

```
Message Statistics
Number of non-fragmented messages: 250
Number of fragmented messages    : 1
Number of fragments              : 7
Number of acks                   : 148
```

```
Time Stamp: 08/18/05 14:00:15.329
Trace Point: 0x0010 CstcClusterServicesTracePtCreatedFlightRecorder
C3D9C5C1E3C5C6D9          <CREATEFR>
Time Stamp: 08/22/05 16:43:28.912
Trace Point: 0x0020 CstcClusterServicesTracePtCreatedClusterObject
D4D6D9C5E8404040 4040C5F8D3770500  <MYCLUSTER   E8L...>
1000             <..        >
Time Stamp: 08/23/05 13:33:40.935
Trace Point: 0x0030 CstcClusterServicesTracePtDeletedClusterObject
D4D6D9C5E8404040 404040E2E3       <MYCLUSTER      ST  >
Time Stamp: 08/23/05 13:33:41.204
Trace Point: 0x0030 CstcClusterServicesTracePtDeletedClusterObject
C3D4D7E3                          <CMPT            >
Time Stamp: 08/23/05 13:33:55.122
Trace Point: 0x0020 CstcClusterServicesTracePtCreatedClusterObject
D4D6D9C5E8404040 4040FC5B04B0D400 <MYCLUSTER   ....M.>
1000             <..         >
```

セクション 10 には、CSTCCCFR のフライト・レコーダーが示されます。このクラスター・フライト・レコーダーは、このノード上でクラスタリングが終了するまでシステム上に残ります。

CLUSTERINFO マクロ結果のセクション 11。

```
Flight Recorder        : CSTCCLFR
Flight Recorder Address: FC5B04B0D4001E80
--------------------------------------------------------------------------------
Time Stamp: 08/23/05 13:33:54.944
Trace Point: 0x1010 CstcClusterTracePtCreatedSubnetObject
7F000000FC5B04B0 D4006000         <........M.-.    >
Time Stamp: 08/23/05 13:33:55.062
Trace Point: 0x1000 CstcClusterTracePtCreatedNodeObject
C3E2E3D9D9C3C8C3 E2E3F5F2FC5B04B0 <CSTRSYSTEM1....>
D4007000                          <M...            >
Time Stamp: 08/23/05 13:33:55.122
Trace Point: 0x1020 CstcClusterTracePtCreatedMCGroupObject
0000000000000001 00000000D9C3C8C3 <............RCHC>
E2E3F5F2                          <ST52            >
.
.
.
```

セクション 11 には、クラスター通信のフライト・レコーダー (CSTECLFR) が示されます。このクラスター・フライト・レコーダーは、このノード上でクラスタリングが終了するまでシステム上に残ります。

CLUSTERINFO マクロ結果のセクション 12。

```
Flight Recorder        : CSTCCCFR
Flight Recorder Address: FC5B04B0D4006380
--------------------------------------------------------------------------------
Time Stamp: 08/23/05 13:33:55.080
Trace Point: 0x3000 CstcCCScamTracePtScamOpen
FC5B04B0D400E480 0000000000000000 <....M.U.........>
Time Stamp: 08/23/05 13:33:55.097
Trace Point: 0x3010 CstcCCScamTracePtScamBind
FC5B04B0D400E480 0000000000000000 <....M.U.........>
Time Stamp: 08/23/05 13:33:55.100
Trace Point: 0x3000 CstcCCScamTracePtScamOpen
FC5B04B0D400E480 0000000000000000 <....M.U.........>
D6E4E3                            <OUT             >
Time Stamp: 08/23/05 13:33:55.100
Trace Point: 0x3010 CstcCCScamTracePtScamBind
FC5B04B0D400E480 0000000000000000 <....M.U.........>
.
.
.
```

セクション 12 には、このノード上でクラスタリングが終了するまでシステム上に残る clue フライト・レコーダー (CSTCCCFR) が示されます。

CLUSTERINFO マクロ結果のセクション 13。

```
Time Stamp: 08/23/05 13:33:55.201
C3A2A385C7E27A7A C3A2A385C7E24082 <CsteGS::CsteGS b>
85878995A2                        <egins           >
Time Stamp: 08/23/05 13:33:55.201
C3A2A385C4C14083 9695A2A399A483A3 <CsteDA construct>
85847A40C2C1C4C1 404040404040     <ed: BADA        >
Time Stamp: 08/23/05 13:33:55.201
C3A2A385C7E27A7A C3A2A385C7E24081 <CsteGS::CsteGS a>
8484408281848140 A39640C4C16D9389 <dd bada to DA_li>
A2A3                              <st              >
.
.
.
```

セクション 13 には、送信待ち行列およびアクティブ・メッセージ待ち行列の内容が表示されます。このセクションがかなりの期間空でない場合、クラスター内に問題があることを示します。

CLUSTERINFO マクロ結果のセクション 14。

```
Flight Recorder        : CSTECLF2
Flight Recorder Address: CDAB6D0339001300
--------------------------------------------------------------------------------
Time Stamp: 08/23/05 13:33:55.201
C3A2A385C4C17A7A C3A2A385C4C16B40 <CsteDA::CsteDA, >
83998581A385D4C3 C79996A49740C9C4 <createMCGroup ID>
407E40F0A7F1F5                    < = 0x15         >
Time Stamp: 08/23/05 13:33:55.209
C3A2A385E2C3D985 977A7A84859389A5 <CsteSCRep::deliv>
85994094A287E3A8 97857EF0A7F140A2 <er msgType=0x1 s>
A482E3A897857EF0 A7F240C4C17EC2C1 <ubType=0x2 DA=BA>
C4C1404040404040                  <DA              >
Time Stamp: 08/23/05 13:33:55.209
C3A2A385C4C17A7A A58985A66B409985 <CsteDA::view, re>
9496A585D4C3C799 96A497D485948285 <moveMCGroupMembe>
99A240C9C4407E40 F0A7F1F540969384 <rs ID = 0x15 old>
6D959684856D9389 A2A340A289A98540 <_node_list size >
7E40F0A7F0                        <= 0x0           >
.
.
.
```

セクション 14 には、フライト・レコーダー情報が示されます。

CLUSTERINFO マクロ結果のセクション 15。

```
Message Queues
Send Queues:
Send Queue: 00 Messages: 00 MessageQueue*: FC5B04B0D400BF80
Send Queue: 01 Messages: 00 MessageQueue*: FC5B04B0D400DF80
Send Queue: 02 Messages: 00 MessageQueue*: FC5B04B0D400E600
Send Queue: 03 Messages: 00 MessageQueue*: FC5B04B0D400E680
Send Queue: 04 Messages: 00 MessageQueue*: FC5B04B0D400E700
Send Queue: 05 Messages: 00 MessageQueue*: FC5B04B0D400E780
Send Queue: 06 Messages: 00 MessageQueue*: FC5B04B0D400E800
Send Queue: 07 Messages: 00 MessageQueue*: FC5B04B0D400E880
Send Queue: 08 Messages: 00 MessageQueue*: FC5B04B0D400E900
Send Queue: 09 Messages: 00 MessageQueue*: FC5B04B0D400E980
Send Queue: 10 Messages: 00 MessageQueue*: FC5B04B0D400EA00
Send Queue: 11 Messages: 00 MessageQueue*: FC5B04B0D400EA80
Send Queue: 12 Messages: 00 MessageQueue*: FC5B04B0D400EB00
```

```
Send Queue: 13 Messages: 00 MessageQueue*: FC5B04B0D400EB80
Send Queue: 14 Messages: 00 MessageQueue*: FC5B04B0D400EC00
Send Queue: 15 Messages: 00 MessageQueue*: FC5B04B0D400EC80
Send Queue: 16 Messages: 00 MessageQueue*: FC5B04B0D400ED00
Send Queue: 17 Messages: 00 MessageQueue*: FC5B04B0D400ED80
Send Queue: 18 Messages: 00 MessageQueue*: FC5B04B0D400EE00
Send Queue: 19 Messages: 00 MessageQueue*: FC5B04B0D400EE80
Active Message Queues:
Active Message Queue: 00 Messages: 00 MessageQueue*: FC5B04B0D4008570
Active Message Queue: 01 Messages: 00 MessageQueue*: FC5B04B0D4008640
Active Message Queue: 02 Messages: 00 MessageQueue*: FC5B04B0D4008710
Active Message Queue: 03 Messages: 00 MessageQueue*: FC5B04B0D40087E0
Active Message Queue: 04 Messages: 00 MessageQueue*: FC5B04B0D40088B0
Active Message Queue: 05 Messages: 00 MessageQueue*: FC5B04B0D4008980
Active Message Queue: 06 Messages: 00 MessageQueue*: FC5B04B0D4008A50
Active Message Queue: 07 Messages: 00 MessageQueue*: FC5B04B0D4008B20
Active Message Queue: 08 Messages: 00 MessageQueue*: FC5B04B0D4008BF0
Active Message Queue: 09 Messages: 00 MessageQueue*: FC5B04B0D4008CC0
Active Message Queue: 10 Messages: 00 MessageQueue*: FC5B04B0D4008D90
Active Message Queue: 11 Messages: 00 MessageQueue*: FC5B04B0D4008E60
Active Message Queue: 12 Messages: 00 MessageQueue*: FC5B04B0D4008F30
Active Message Queue: 13 Messages: 00 MessageQueue*: FC5B04B0D4009000
Active Message Queue: 14 Messages: 00 MessageQueue*: FC5B04B0D40090D0
Active Message Queue: 15 Messages: 00 MessageQueue*: FC5B04B0D40091A0
Active Message Queue: 16 Messages: 00 MessageQueue*: FC5B04B0D4009270
Active Message Queue: 17 Messages: 00 MessageQueue*: FC5B04B0D4009340
Active Message Queue: 18 Messages: 00 MessageQueue*: FC5B04B0D4009410
Active Message Queue: 19 Messages: 00 MessageQueue*: FC5B04B0D40094E0
--------------------------------------------------------------------------------
Tuning Values
cstcRcvSendTimerRatio        : 2 Default: 2
cstcMcastRelayTimerRatio     : 8 Default: 8
cstcMcastRelayHBTimerRatio   : 4 Default: 4
cstcHeartbeatBaseTimer       : 12288000000 Default: 12288000000
cstcHeartbeatBasePrecision   : 4096000000 Default: 4096000000
cstcRetryPrecision           : 4096000000 Default: 4096000000
cstcRetryTimerVal            : 4096000000 Default: 4096000000
cstcCDATBaseTimer            : 491520000000 Default: 491520000000
cstcCDATBasePrecision        : 40960000000 Default: 40960000000
cstcRecoveryBaseTimer        : 3686400000000 Default: 3686400000000
cstcRecoveryBasePrecision    : 491520000000 Default: 491520000000
cstcMaxRetryTime             : 32768000000 Default: 32768000000
cstcCCFragmentSize           : 1464 Default: 1464
cstcCCSendQOverflow          : 1024 Default: 1024
cstcBadMsgCtrThreshold       : 3 Default: 3
cstcUnreachableHBAckThreshold: 1 Default: 1
cstcReachableHBAckThreshold  : 3 Default: 3
cstcUnreachableHBThreshold   : 4 Default: 4
cstcReachableHBThreshold     : 4 Default: 4
cstcMaxHBThreshold           : 16 Default: 16
cstcDisableMsgTimer          : 0 Default: 0
cstcRepeatAckThreshold       : 10 Default: 10
     DISPLAY/ALTER/DUMP    CLUSTERINFO -NEW2   08/23/05  13:36:37   PAGE     87
cstcDelayedAckTimer          : 409600000 Default: 409600000
cstcDelayedAckPrecision      : 40960000 Default: 40960000
cstcCCSendWindow             : 2 Default: 2
cstcCCEnableMcast            : 1 Default: 1
cstcCCPerfClass              : 2 Default: 2
--------------------------------------------------------------------------------
****** END OF DUMP ******
```

セクション 15 にはチューニング値が示されます。チューニング値は、クラスターのパフォーマンス情報および構成情報と同等です。それらの情報については、クラスター・リソース・サービス情報検索 (QcstRetrieveCRSInfo) API に説明されています。 セクション 14 には、これらのフィールドの現行値とデフォルト値が示されます。

クラスターの一般的な問題

ここでは、クラスターで生じる可能性のある最も一般的な問題のいくつかをリストして、 その回避方法および回復方法を示します。

以下の共通問題は容易に回避可能であるか、または容易に修正可能です。

クラスター・ノードを始動または再始動できない。

この状況は通常、通信環境で問題が生じていることに原因があります。この状況を回避するには、ループバック・アドレス、INETD 設定値、 ALWADDCLU 属性、およびクラスター通信の IP アドレスを含むネットワーク属性が正しく設定されていることを確認してください。

- リモート・ノードを始動しようとする場合、 ALWADDCLU ネットワーク属性がターゲット・ノードに正しく設定されている必要があります。この属性は環境に応じて、*ANY または *RQSAUT のいずれかに設定する必要があります。
- ローカル上およびターゲット・ノード上でクラスタリングに使用するために選択した IP アドレスは、*「活動中」*の状況を示していなければなりません。
- ローカル上およびターゲット・ノード上の LOOPBACK アドレス (127.0.0.1) も、活動中でなければなりません。
- ローカル・ノードおよびすべてのリモート・ノードは、ネットワーク・ルーティングがアクティブであることを確認するために、クラスタリングに使用する IP アドレスを使用しての PING が可能でなければなりせん。
- ターゲット・ノード上で INETD がアクティブである必要があります。 INETD がアクティブなら、ターゲット・ノード上のポート 5550 は*「接続待機」*状況になっているはずです。 INETD サーバーの開始方法については、『INETD サーバー』の項を参照してください。
- ノードを始動しようとする前は、始動するノード上のポート 5551 は閉じている必要があります。これが開いていると、対象ノードでクラスタリングが正常に開始できなくなります。

結合されていない複数の単一ノード・クラスターが生じた。

これは始動するノードが他のクラスター・ノードと通信できない場合に生じることがあります。通信パスを調べてください。

出口プログラムからの応答が遅い。

この状況が生じる一般的な原因は、出口プログラムで使用されるジョブ記述の設定が正しくないことです。MAXACT パラメーターの設定値が低すぎるために、たとえば任意の時点でアクティブの出口プログラムのインスタンスが 1 つだけに制限されることがあります。 この値は *NOMAX に設定することをお勧めします。

一般的なパフォーマンスが遅い。

この症状については、いくつかの一般的な原因があります。

- 最も可能性の高い原因は、共用通信回線でのトラフィックが大きいことです。
- 可能性のある別の原因は、通信環境とクラスター・メッセージ調整パラメーターとの間に不整合があることです。 クラスター・リソース・サービス情報検索 (QcstRetrieveCRSInfo) API を使用して調整パラメーターの現行設定値を表示し、 クラスター・リソース・サービス変更 (QcstChgClusterResourceServices) API を使用して設定値を変更することができます。旧式のアダプター・ハードウェアを使用している場合、デフォルトのクラスター調整パラメーター設定値では、クラスター

のパフォーマンスが低下することがあります。「*旧式*」の定義に含まれるアダプター・ハードウェアのタイプには、2617、2618、2619、2626、および 2665 があります。この場合、「パフォーマンス・クラス」調整パラメーターの設定値を「*Normal*」にすることをお勧めします。

- この状況を生じる別の一般的な原因は、 IP マルチキャスト・グループに問題があることです。複数のノードのプライマリー・クラスター・アドレス (クラスターの作成時またはノードの追加時に指定のノードに入力された最初のアドレス) が共通 LAN 上に存在する場合、クラスターは IP マルチキャスト機能を利用します。 NETSTAT コマンドを使用して、プライマリー・クラスター・アドレスがマルチキャスト・ホスト・グループの 226.5.5.5 を示していることを確認してください。これは、対象アドレスに対してオプション 14 「*マルチキャスト・グループの表示*」を使用することによって表示できます。マルチキャスト・グループが存在しない場合、クラスター・リソース・サービス情報検索 (QcstRetrieveCRSInfo) API を使用して、*Enable multicast* クラスター調整パラメーターがデフォルトの設定値「TRUE」に設定されたままであることを確認してください。
- クラスターのノードすべてがローカル LAN 上にあるか、またはネットワーク経路全体をとおして 1,464 バイトよりも大きい最大転送単位 (MTU) のパケット・サイズを処理できるルーティング機能を持っている場合、*Message fragment size* のクラスター調整パラメーター値を、経路 MTU にもっと適した値に増加すれば、大きなクラスター・メッセージ転送 (1,536 KB よりも大きい) を大幅に高速化することができます。

新しいリリースの機能をいずれも使用できない。

新リリースの機能を使用しようとしたときにエラー・メッセージ CPFBB70 が表示された場合、クラスター・バージョンは以前のバージョン・レベルに設定されたままであるということです。すべてのクラスター・ノードを新規のリリース・レベルにアップグレードしてから、クラスター・バージョンの調整インターフェースを使用して現行のクラスター・バージョンを新規のレベルに設定する必要があります。詳細は、『クラスターのクラスター・バージョンの調整』の項を参照してください。

ノードをデバイス・ドメインに追加することも、iSeries ナビゲーター・クラスター管理インターフェースにアクセスすることもできない。

iSeries ナビゲーター・クラスター管理インターフェースへのアクセスや切り替え可能な装置の使用を可能にするには、システムに i5/OS オプション 41 (HA 切り替え可能リソース) をインストールする必要があります。さらに、そのオプションのための有効なライセンス・キーも必要です。

クラスター PTF を適用したが機能しているように見えない。

| PTF を適用した後、以下のタスクを実行したことを確認してください。

1. クラスターを終了する
2. サインオフしてからサインオンする

   以前のプログラムは活動化グループが破棄されるまでアクティブのままです。すべてのクラスター・コード (クラスター API を含む) は、デフォルトの活動化グループで実行します。
3. クラスターを開始する

   ほとんどのクラスター PTF の場合、PTF を活動化するには、ノード上でクラスタリングをいったん終了してから再始動する必要があります。

出口プログラムのジョブ・ログに CEE0200 がある。

このエラー・メッセージで、呼び出し元モジュールは QLEPM、呼び出し元プロシージャーは Q_LE_leBdyPeilog です。出口プログラムが呼び出すプログラムは、*CALLER または名前の指定された活

動化グループ内で実行しなければなりません。この状態を修正するには、出口プログラムまたはエラーの生じたプログラムを変更しなければなりません。

クラスター・リソース・サービスのジョブ・ログに CPD000D とその後に続く CPF0001 とがある。

このエラー・メッセージが表示されたら、QMLTTHDACN のシステム値が 1または 2 に設定されていることを確認してください。

クラスターがハングしたと思われる。

クラスター・リソース・グループの出口プログラムが未解決であることを確認してください。出口プログラムを検査するには、WRKACTJOB ((活動ジョブ処理) コマンドを使用して、「機能」列に PGM-QCSTCRGEXT があるかどうかを確かめます。

関連概念

101 ページの『ノードをクラスターに追加できるようにする』
ノードをクラスターに追加するためには、 クラスターへの追加可能 (ALWADDCLU) ネットワーク属性の値を設定する必要があります。

126 ページの『クラスター・パフォーマンス』
クラスターに変更を加えると、 クラスターを管理するのに必要なオーバーヘッドに影響を与える可能性があります。

15 ページの『クラスター・バージョン』
クラスター・バージョンとは、 クラスターで実行できる機能のレベルを表す用語です。

80 ページの『iSeries ナビゲーター・クラスター管理』
IBM は、iSeries ナビゲーターから利用できて、オプション 41 (i5/OS - HA 切り替え可能リソース) を通してアクセスできるシンプル・クラスター管理インターフェースを提供しています。

関連タスク

116 ページの『クラスターのクラスター・バージョンの調整』
クラスター・バージョンは、クラスター内のすべてのノードがアクティブに相互通信するレベルを定義します。

区画エラー

クラスター状態によっては、簡単に修正することができます。 クラスター区画が生じた場合の回復方法を確認してください。 このトピックでは、クラスター区画の回避方法を示し、区画をマージして元に戻す方法を例示します。

クラスター内でクラスター区画が発生するのは、クラスター内にある 1 つ以上のノードどうしの連絡が途絶えたときに、その不通のノードの障害を確認できない場合です。これを、論理区画 (LPAR) 環境の区画と混同しないでください。

活動記録ログ (QHST) または QCSTCTL ジョブ・ログにエラー・メッセージ CPFBB20 を受け取った場合、クラスターが区画に分割されているので、その回復方法を知る必要があります。以下の例は、4 つのノード (A、B、C、および D) から成るクラスターを含むクラスター区画を示しています。この例では、クラスター・ノード B および C の間で通信が停止したので、そのクラスターが 2 つのクラスター区画に分割された場合を示しています。クラスターが区画に分割される前に、4 つのクラスター・リソース・グループが存在していました。それらは任意のタイプであることが可能で、CRG A、CRG B、CRG C、および CRG D と呼ばれます。この例は、クラスター・リソース・グループごとのリカバリー・ドメインを示しています。

表 22. クラスター区画におけるリカバリー・ドメインの例

ノード A	ノード B	x	ノード C	ノード D
CRG A (バックアップ 1)	CRG A (プライマリー)			
	CRG B (プライマリー)		CRG B (バックアップ 1)	
	CRG C (プライマリー)		CRG C (バックアップ 1)	CRG C (バックアップ 2)
CRG D (バックアップ 2)	CRG D (プライマリー)		CRG D (バックアップ 1)	
区画 1			**区画 2**	

通信パス内のいずれかの場所で最大転送単位 (MTU) がクラスター通信の調整可能パラメーター、メッセージ・フラグメント・サイズよりも小さくなるとき、クラスターは区画に分割されることがあります。クラスター IP アドレスの MTU を検証するには、対象ノードで TCP/IP ネットワーク状況の処理 (WRKTCPSTS) コマンドを使用します。通信パス全体の各ステップでも、MTU を検査しなければなりません。 MTU がメッセージ・フラグメント・サイズよりも小さい場合、パスの MTU を増加させるか、またはメッセージ・フラグメント・サイズを小さくします。クラスター・リソース・サービス情報検索 (QcstRetrieveCRSInfo) API を使用して調整パラメーターの現行設定値を表示し、クラスター・リソース・サービス変更 (QcstChgClusterResourceServices) API を使用して設定値を変更することができます。

クラスターが区画に分割された状態にある原因が修正された後、クラスターは再確立された通信リンクを検出して、メッセージ CPFBB21 を活動記録ログ (QHST) または QCSTCTL ジョブ・ログに出します。これにより、オペレーターはクラスターがクラスター区画の状態から回復したことを知ります。クラスターが区画に分割された状態が修正されると、数分後にクラスターが互いにマージして元に戻る可能性があることに注意してください。

関連概念

33 ページの『クラスター区画』
クラスター区画は、通信障害の結果として生じる、アクティブなクラスター・ノードのサブセットです。 区画のメンバーは、相互の接続を維持します。

94 ページの『クラスター区画の回避』
典型的なネットワーク関連クラスター区画が発生しないようにする最も有効な手段としては、クラスター内のすべてのノードを結び付ける冗長通信パスを構成します。

27 ページの『マージ』
マージ操作は、区画に分割されたノードが再び通信を開始するときに発生することを除けば、再結合操作と似ています。

21 ページの『例: 障害』
通常、フェイルオーバーはノード障害によって起こりますが、その他の理由でも起こることがあります。

プライマリー・クラスター区画と 2 次クラスター区画の判別

クラスター区画内で実行可能なクラスター・リソース・グループ・アクションのタイプを判別するには、その区画が 1 次または 2 次のどちらのクラスター区画かが分からなければなりません。 区画が検出された場合には、クラスターで定義されている各クラスター・リソース・グループごとに、各区画は 1 次または 2 次のどちらかの区画に指定されます。

プライマリー・バックアップ・モデルの場合、現在のノード役割がプライマリーであるノードが 1 次区画に収容されます。 他の区画はすべて 2 次になります。 1 次区画は、すべてのクラスター・リソース・グループで同じとは限りません。

対等モデルには、次のような区画規則があります。

- リカバリー・ドメイン・ノードがすべて 1 つの区画に含まれる場合、その区画が 1 次区画になります。
- リカバリー・ドメイン・ノードが区画を超える場合、1 次区画なしになります。 どちらの区画も 2 次区画になります。
- クラスター・リソース・グループがアクティブになっていて、特定の区画内に対等ノードがない場合、そのクラスター・リソース・グループは、その区画内で終端します。
- 2 次区画内で操作上の変更が許可されるのは、そのような操作上の変更に対する制約事項が満足された場合だけです。
- 2 次区画内では、構成上の変更は許可されません。

クラスター・リソース・グループ API ごとに、以下の制限があります。

表 23. クラスター・リソース・グループ API 区画の制約事項

クラスター・リソース・グループ API	1 次区画で許可される	2 次区画で許可される
リカバリー・ドメインへのノードの追加	X	
CRG 装置項目の追加		
クラスター・リソース・グループの変更	X	
CRG 装置項目の変更	X	X
クラスター・リソース・グループの作成		
クラスター・リソース・グループの削除	X	X
情報の配布	X	X
クラスター・リソース・グループの終了[1]	X	
切り替えの開始	X	
クラスター・リソース・グループのリスト	X	X
クラスター・リソース・グループ情報の一覧表示	X	X
リカバリー・ドメインからのノードの除去	X	
CRG 装置項目の除去	X	
クラスター・リソース・グループの開始[1]	X	

注:

1. 対等クラスター・リソース・グループの場合はすべての区画で許可されていますが、その影響を受けるのは API を実行中の区画だけです。

これらの制限の適用によって、クラスターの区画化が解消されたときに、クラスター・リソース・グループを同期することができます。区画化された状態からノードがクラスターに再結合するとき、1 次区画内のクラスター・リソース・グループのバージョンが 2 次区画のノードにコピーされます。

対等モデルの 2 つの 2 次区画をマージすると、アクティブの状況のクラスター・リソース・グループを収容している区画が優先されることが宣言されます。クラスター・リソース・グループの両方の区画が同じ状況である場合、クラスター・リソース・グループのリカバリー・ドメイン内に最初にリストされているノードを収容している区画が優先されることが宣言されます。優先区画内のクラスター・リソース・グループのバージョンが、もう一方の区画内のノードにコピーされます。

区画が検出されると、クラスター・ノード項目追加、クラスター・バージョンの調整、およびクラスター作成 API はどの区画内でも実行できません。デバイス・ドメイン項目追加 API を実行できるのは、デバイ

ス・ドメイン内のどのノードも区画に分割されていない場合だけです。他のすべてのクラスター制御 API は、区画内で実行できます。 しかし、API が実行するアクションの影響はその API を実行している区画内に限定されます。

区画化ノードを障害ノードに変更する

時々、「区画」状況が報告されているときに、実際にはノード障害が生じていることがあります。 このことは、クラスター・リソース・サービスが 1 つ以上のノードとの通信を失ったものの、 ノードが引き続き作動可能かどうかを検出できない場合に生じます。 この状況が生じた場合に、ノードに障害が起きたことを知らせるための簡単なメカニズムがあります。

重要: ノードに障害が起きたことをクラスター・リソース・サービスに知らせると、区画状況からの回復を行うのがより簡単になります。ただし、実際にはそのノードが引き続きアクティブであり、本当の区画が発生している場合には、ノード状況を障害に変更してはなりません。そのように変更すると、複数の区画に属するノードが、クラスター・リソース・グループのためのプライマリーの役割をするものと見なされてしまいます。 2 つのノードがいずれも自分をプライマリー・ノードと認識してしまうと、複数のノードがそれぞれ独自にファイルのコピーに変更を加えた場合、ファイルやデータベースなどのデータがばらばらになったり破壊されたりすることがあります。さらに、各区画内のそれぞれ 1 つのノードがプライマリーの役割を割り当てられていた場合、この 2 つの区画をマージすることはできません。

ノードの状況が「失敗」に変更されると、区画内のそれぞれのクラスター・リソース・グループのリカバリー・ドメイン内のノードの役割が、配列し直されることがあります。「失敗」に設定されるノードは、最後のバックアップとして割り当てられます。複数のノードに障害が起き、それらの状況を変更する必要がある場合に、ノードを変更する順序が、リカバリー・ドメインのバックアップ・ノードの最終的な順序に影響を与えます。障害が起きたノードが CRG のプライマリー・ノードであった場合には、最初のアクティブのバックアップが、新規プライマリー・ノードとして再割り当てされます。

関連概念

27 ページの『マージ』
マージ操作は、区画に分割されたノードが再び通信を開始するときに発生することを除けば、再結合操作と似ています。

25 ページの『再結合』
再結合とは、一度メンバーから外れた後に、再びクラスターの活動メンバーになることです。

関連タスク

158 ページの『ヒント: クラスター区画』
クラスター区画に対しては、以下のヒントを参考にしてください。

関連資料

クラスター・ノード項目の変更 (CHGCLUNODE)

クラスター・ノード項目の変更 API (QcstChangeClusterNodeEntry)

クラスター・ノードの開始 (STRCLUNOD)

クラスター・ノードの開始 (QcstStartClusterNode) API

iSeries ナビゲーターを使用する場合:

これを実行するには、オプション 41 (i5/OS- HA 切り替え可能リソース) がインストールされていて、ライセンス交付を受けている必要があります。

クラスター・リソース・サービスが、ノードとの通信が途切れたときに、そのノードがまだ作動可能かどうかを検出できない場合には、iSeries ナビゲーター内の「ノード」コンテナーに示されるクラスター・ノードの状況は、**通信なし**になります。ノードの状況を**通信なし**から**失敗**に変更する必要があるかもしれません。その後、ノードを再始動することができます。

ノードの状況を**通信なし**から**失敗**に変更するには、以下のようにします。

1. iSeries ナビゲーターで、「**マネージメント・セントラル**」を展開します。
2. 「**クラスター**」を展開します。
3. 状況を変更したいノードが属するクラスターを展開します。
4. 「**ノード**」をクリックします。
5. 状況を変更したいノードを右マウス・ボタンでクリックし、「**クラスター**」 → 「**状況の変更**」を選択します。

次に、ClusterChange 状況を選択します。

以下の手順で、ノードを再始動します。

1. ノードを右マウス・ボタンでクリックし、「**クラスター**」 → 「**開始**」を選択します。

CL コマンドおよび API を使用する場合:

ノードの状況を**通信なし**から**失敗**に変更するには、以下のようにします。

1. CHGCLUNODE コマンドまたは クラスター・ノード項目変更 (QcstChangeClusterNodeEntry) API を使用して、ノードの状況を「区画」から「失敗」に変更する。このことは、実際に障害が起きたすべてのノードに対して行う必要があります。
2. STRCLUNOD コマンド または クラスター・ノードの開始 (QcstStartClusterNode) API を使用して、クラスター・ノードを開始することで、ノードがクラスターを再結合できるようにすることができます。

ヒント: クラスター区画

クラスター区画に対しては、以下のヒントを参考にしてください。

1. 区画のマージを可能にするために、区画内の操作を制限するための規則が設けられています。このような制限がなければ、クラスターの再構成には、かなりの量の作業が必要となります。
2. 1 次区画内のノードが破棄された場合、2 次区画内で特殊な処理が必要となることがあります。そのような事態が発生する原因となる最も一般的なケースは、1 次区画を構成するサイトの消失です。『区画エラーからの回復』の中の例を使用する一方で、区画 1 が破棄されたと仮定します。この場合、クラスター・リソース・グループ B、C、および D のプライマリー・ノードは、 区画 2 になければなりません。最も簡単な回復は、「クラスター・ノード項目の変更」を使用してノード A とノード B の両方を障害に設定することです。その方法の詳細は、『区画化ノードを障害ノードに変更する』を参照してください。回復は、手動で行うこともできます。そのためには、以下の操作を実行します。
   a. ノード A および B を、区画 2 のクラスターから除去します。区画 2 が現在のクラスターです。
   b. 新しいクラスターに必要な任意の論理複製環境を確立します。つまり、クラスター・リソース・グループ開始 API/CL コマンドなどです。

   区画 2 内のクラスター定義からノードが除去されるので、区画 1 および区画 2 のマージ試行が失敗します。クラスター定義内のミスマッチを訂正するために、区画 1 内のそれぞれのノードに対してクラスター削除 (QcstDeleteCluster) API を実行します。その後、区画 1 からクラスターにノードを追加し、

すべてのクラスター・リソース・グループ定義、リカバリー・ドメイン、および論理複製を再設定します。このことは、大量の作業を必要とし、エラーも発生しがちです。それで、この手順は、サイトがなくなった状況でのみ行うべきです。

3. ノード開始操作の処理は、開始するノードの状況によって異なります。

   ノードに障害が起きたか、またはノード終了操作がノードを終了しました。

   a. クラスター・リソース・サービスが、追加されるノードで開始されます。
   b. クラスター定義が、クラスター内のアクティブ・ノードから、開始されるノードにコピーされます。
   c. リカバリー・ドメイン内で開始されるノードを持つクラスター・リソース・グループが、クラスター内のアクティブ・ノードから、開始されるノードにコピーされます。クラスター・リソース・グループは、開始されるノードから、クラスター内のアクティブ・ノードにコピーされません。

   ノードが区画化ノードである場合:

   a. アクティブ・ノードのクラスター定義が、開始されるノードのクラスター定義と比較されます。定義が同じであれば、マージ操作として開始が続行されます。定義が一致しない場合には、マージが停止し、ユーザーの介入が必要となります。
   b. マージが継続する場合、開始されるノードがアクティブ状況に設定されます。
   c. リカバリー・ドメイン内で開始されるノードを持つクラスター・リソース・グループが、クラスター・リソース・グループの 1 次区画から、クラスター・リソース・グループの 2 次区画にコピーされます。クラスター・リソース・グループは、開始されるノードから、クラスター内ですでにアクティブとなっているノードにコピーされることがあります。

関連タスク

157 ページの『区画化ノードを障害ノードに変更する』
時々、「区画」状況が報告されているときに、実際にはノード障害が生じていることがあります。 このことは、クラスター・リソース・サービスが 1 つ以上のノードとの通信を失ったものの、 ノードが引き続き作動可能かどうかを検出できない場合に生じます。 この状況が生じた場合に、ノードに障害が起きたことを知らせるための簡単なメカニズムがあります。

関連資料

クラスターの削除 (QcstDeleteCluster) API

クラスターの回復

生じる可能性のある他のクラスター障害から回復する方法について示します。

クラスター・ジョブ障害からの回復

クラスター・リソース・サービス・ジョブの障害は、通常、何か他の問題があることを示しています。

障害が起きたジョブに関連したジョブ・ログを調べて、なぜ障害が起きたのかを記述するメッセージを探します。 エラー状態を訂正します。

クラスター回復変更 (CHGCLURCY) コマンド を使って、ノード上でクラスタリングの終了と再始動を行う必要なしに終了したクラスター・リソース・グループのジョブを再始動することができます。

1. `CHGCLURCY CLUSTER(EXAMPLE)CRG(CRG1)NODE(NODE1)ACTION(*STRCRGJOB)` このコマンドは、ノード NODE1 上のクラスター・リソース・グループのジョブ CRG1 が投入される原因になります。 NODE1 上でクラスター・リソース・グループのジョブを開始するには、NODE1 上でクラスタリングがアクティブになっている必要があります。
2. ノードでクラスタリングを再始動します。

IBM ビジネス・パートナー・クラスター管理プロダクトを使用している場合は、プロダクトに付属している資料を参照してください。

関連概念

129 ページの『ジョブ構造とユーザー待ち行列』
クラスターを管理するには、ジョブ構造とユーザー待ち行列についての知識が必要です。

関連タスク

115 ページの『クラスター・ノードの終了』
ノードを停止または終了すると、そのノード上のクラスター・リソース・サービスが停止します。

115 ページの『クラスター・ノードの開始』
クラスター・ノードを開始すると、クラスター内のノード上のクラスター・リソース・サービスも開始されます。 クラスター・バージョン 3 から、クラスターにアクティブ・ノードがあれば、あるノードが自分自身を開始して現在アクティブなクラスターに再結合することができるようになりました。

損傷を受けたクラスター・オブジェクトの回復

オブジェクトが損傷を受けることはあまりありませんが、 クラスター・リソース・サービス・オブジェクトが損傷を受ける可能性はあります。

それがアクティブ・ノードである場合、システムは、クラスター内の別の回復を試行します。システムは次の回復ステップを実行します。

内部オブジェクトが損傷を受けた場合

1. 損傷を受けたノードが終了します。
2. クラスター内に別のアクティブ・ノードが最低 1 つある場合には、損傷を受けたノードが自動的に自身を再始動し、クラスターに再結合します。再結合のプロセスで、損傷を受けた状態が修正されます。

クラスター・リソース・グループが損傷を受けた場合

1. 損傷を受けた CRG を持つノードが、その CRG に関連した、現在処理中の操作に失敗します。システムは、別のアクティブ・ノードからの CRG の自動回復を試行します。
2. リカバリー・ドメイン内にアクティブなメンバーが最低 1 つある場合には、CRG 回復が作動します。ない場合には、CRG ジョブが終了します。

システムが他のアクティブ・ノードを識別したり、それに到達したりできない場合には、以下の回復ステップを実行する必要があります。

内部オブジェクトが損傷を受けた場合

内部クラスタリング・エラーを受け取ります (CPFBB46、CPFBB47、または CPFBB48)。

1. 損傷を含むノードのクラスタリングを終了します。
2. 損傷を含むノードのクラスタリングを再始動します。このことを、クラスター内の別のアクティブ・ノードから実行します。
3. ステップ 1 および 2 で問題が解決しない場合には、損傷を受けたノードをクラスターから除去します。
4. クラスターおよび適切なクラスター・リソース・グループのリカバリー・ドメインに、システムを戻します。

クラスター・リソース・グループが損傷を受けた場合

オブジェクトが損傷を受けたことを知らせるエラーを受け取ります (CPF9804)。

1. 損傷を受けたクラスター・リソース・グループを含むノードで、クラスタリングを終了します。
2. CRG を削除します (DLTCRG コマンドを使用)。
3. CRG オブジェクトを含むクラスター内にアクティブなノードが他にない場合には、メディアから復元します。
4. 損傷を受けたクラスター・リソース・グループを含むノードで、クラスタリングを開始します。これは、どのアクティブ・ノードからでも実行できます。
5. クラスタリングを開始すると、システムはすべてのクラスター・リソース・グループを再同期します。クラスター内の他のノードに CRG が含まれていない場合には、CRG を再作成する必要があるかもしれません。

完全なシステム消失後のクラスターの回復

サーバーでの想定外の電力喪失が原因で完全なシステム消失が起きた場合、ここに記載されている情報と、「バックアップおよび回復の手引き」の中の該当するチェックリストを一緒に使用して、システム全体を回復してください。

シナリオ 1: 同じシステムへの復元

1. ライセンス内部コードと i5/OS との間で、デバイス・ドメイン情報に不整合が生じるのを防ぐため、オプション 3 (ライセンス内部コードの導入および構成の回復を使用して、ライセンス内部コードをインストールするようお勧めします。

   注: 「ライセンス内部コードの導入および構成の回復」操作が成功するためには、同じディスク装置を持っている必要があります。ただし、ロード・ソース・ディスク装置に障害が起きた場合は例外です。また、同じリリースを回復しなければなりません。
2. ライセンス内部コードをインストールした後、「*バックアップおよび回復の手引き*」の中の第 5 章『*ディスク構成を復元する方法*』の手順に従ってください。このステップによって、ASP を再構成しなくて済むようになります。
3. システム情報を回復し、回復したばかりのノードでクラスタリングを開始する用意ができたら、アクティブ・ノードからクラスタリングを開始する必要があります。このことにより、最新の構成情報が、回復したノードに伝搬されます。

シナリオ 2: 別のシステムへの復元

システム情報を回復し、ジョブ・ログをチェックして、すべてのオブジェクトが復元されたことを確認した後、次のステップを実行して、正確なクラスター・デバイス・ドメイン構成を取得する必要があります。

1. 復元したばかりのノードから、クラスターを削除します。
2. アクティブ・ノードから、以下の手順を実行します。
   a. 回復されたノードをクラスターから除去します。
   b. 回復されたノードをクラスターに戻します。
   c. 回復されたノードをデバイス・ドメインに戻します。
   d. クラスター・リソース・グループを作成するか、またはリカバリー・ドメインにノードを追加します。

関連タスク

131 ページの『クラスターのバックアップおよび回復』
システムにクラスタリングを使用する場合でも、データを保護するためのバックアップ戦略および回復戦略を計画することは依然として重要です。

関連情報

バックアップおよび回復

災害後のクラスターの回復

災害が起きてすべてのノードが失われた場合、クラスターを再構成する必要があります。

そのようなシナリオに備えるために、クラスター構成情報を保管して、その情報のハードコピー印刷出力を取っておくことをお勧めします。

関連タスク

131 ページの『クラスターのバックアップおよび回復』
システムにクラスタリングを使用する場合でも、データを保護するためのバックアップ戦略および回復戦略を計画することは依然として重要です。

バックアップ・テープからのクラスターの復元

通常の操作時には、バックアップ・テープから復元する必要が生じることはありません。

これが必要なのは、災害が起きてクラスター内のすべてのノードが消失した場合のみです。災害が発生した場合には、バックアップ戦略および回復戦略の作成後に定めた一般回復手順に従って回復を図ります。

関連タスク

131 ページの『クラスターのバックアップおよび回復』
システムにクラスタリングを使用する場合でも、データを保護するためのバックアップ戦略および回復戦略を計画することは依然として重要です。

関連情報

バックアップおよび回復

iSeries ナビゲーター・クラスター管理についてよく尋ねられる質問

クラスターの作成および管理を行う iSeries ナビゲーターのグラフィカル・ユーザー・インターフェースに関する質問と答え。

クラスター作成および管理用の IBM グラフィカル・ユーザー・インターフェースは、iSeries ナビゲーターで利用可能で、オプション 41 (HA 切り替え可能リソース) を通してアクセスできます。このインターフェースの詳細については、iSeries ナビゲーター・クラスター管理を参照してください。

以下に、iSeries ナビゲーター・クラスター管理の質問とその回答をリストします。

一般

1. クラスターを作成するための前提条件をまとめたチェックリストはありますか?

iSeries ナビゲーター・クラスター管理

1. クラスター機能は、iSeries ナビゲーター・インターフェースのどこにありますか？
2. クラスターを作成する方法は?
3. 「クラスター」フォルダーと「マネージメント・セントラル」システム・グループとの関係は？
4. ネットワーク上のいくつかの iSeries システムで、 既にクラスターが定義されています。 iSeries ナビゲーターを使ってこれらを表示および管理するために追加するには、どうすればよいですか?

5. クラスター内のノードで状況が「開始済み」になっているものが 1 つもありません。 どのノードをまず開始するべきですか?
6. なぜ、どのノードを最初に開始させるべきかに注意する必要がありますか？
7. 「切り替え可能装置」、および「切り替え可能アプリケーション」フォルダーの中の「現行プライマリー・ノード」列は、何を意味しますか?
8. iSeries ナビゲーターで、装置クラスター・リソース・グループ (CRG) を探し出す方法は？
9. iSeries ナビゲーターで、アプリケーション・クラスター・リソース・グループ (CRG) を探し出す方法は？
10. iSeries ナビゲーターで、データ・クラスター・リソース・グループ (CRG) を探し出す方法は？
11. 「切り替え可能ハードウェア」フォルダーまで戻らずに、切り替え可能装置 (装置 CRG) の状況を表示できるようにしたいと考えています。 どのようにすればできますか?

通信

1. iSeries ナビゲーターのクラスター機能は、 クラスター内のノードと通信を行うために、どの IP アドレスを使用しますか？ノード名の IP アドレスを使用するのではないのですか？

セキュリティー

1. iSeries ナビゲーターの「クラスター」フォルダーの コンテキスト・メニュー内の多くが、使用不可または表示されないのはなぜですか？
2. iSeries ナビゲーターのクラスター機能は、 アプリケーション管理値を使用しますか？
3. iSeries ナビゲーターのクラスター機能が、 クラスター内の自分のノードへのサインオン・ウィンドウを表示するのはなぜですか？

トラブルシューティング

1. 「クラスター」フォルダーがマネージメント・セントラル内で表示されないのはなぜですか？
2. 既にクラスターがあるのに、「クラスター」フォルダーに表示されません。なぜですか？
3. 「クラスター」フォルダーに最新の状況が表示されないのはなぜですか？
4. 私の場合、切り替え可能ハードウェア・グループまたは切り替え可能アプリケーションのフェイルオーバーが行われません。なぜでしょうか？
5. オブジェクトに損傷があるというメッセージを受け取りました。どのように対応できますか？
6. IP アドレスを参照するため、ノードのウィザードで 「参照」ボタンを使っています。表示されるはずの TCP/IP アドレスのすべてが参照ウィンドウに表示されないのはなぜですか？
7. iSeries ナビゲーターの「クラスター」フォルダーの コンテキスト・メニュー内の多くが、使用不可または表示されないのはなぜですか？
8. 「新規クラスター」ウィザードを使用すると、「新規クラスター - 切り替え可能ソフトウェアが見つかりません」というタイトルのパネルが表示されました。 何か問題がありますか？
9. ノードの 1 つの状況が 「通信なし」となっています。 どのように訂正できますか？

一般

クラスターを作成するための前提条件をまとめた、チェックリストはありますか？

はい。クラスター構成チェックリストを使用して、ご使用の環境でクラスターを構成する準備が整っていることを確認してください。

質問に戻る

iSeries ナビゲーターのクラスターの管理: クラスター機能は、iSeries ナビゲーター・インターフェースのどこに置かれていますか?

iSeries ナビゲーター・クラスター管理インターフェースは、ソフトウェア・パッケージ IBM iSeries Access の一部として使用可能です。クラスター機能は、iSeries ナビゲーターの「マネージメント・セントラル」フォルダー内にあります。詳細は、 iSeries ナビゲーター・クラスター管理を参照してください。

質問に戻る

クラスターを作成する方法は？

iSeries ナビゲーターの「新規クラスター」ウィザードを使用して単純なクラスターを作成するには、以下の手順で行います。

1. iSeries ナビゲーターで、「**マネージメント・セントラル**」を展開します。
2. 「**クラスター**」を右マウス・ボタン・クリックしてから、「**新規クラスター**」を選択します。
3. ウィザードの指示に従って、クラスターを作成します。

クラスターの作成および構成の完全な詳細は、クラスターの構成を参照してください。

質問に戻る

「クラスター」フォルダーと「マネージメント・セントラル」システム・グループとの関係は？

iSeries ナビゲーターを使用してクラスターを作成すると、マネージメント・セントラル・サーバーにシステム・グループも作成されます。このシステム・グループはクラスター名と同じ名前を持ち、このシステム・グループのエンドポイント・システムはクラスターのノードになります。また、このシステム・グループは、そのシステム・グループがクラスターを表す特殊システム・グループであることを iSeries ナビゲーターが判断できるよう、それ自身の特殊タイプを持っています。

重要: マネージメント・セントラル・システムは、システム・グループを含んでいます。 iSeries ナビゲーターの現行のマネージメント・セントラル・システムを変更することを選択した場合は、新しいマネージメント・セントラル・システムは特殊クラスター・システム・グループを持たず、その結果、そのクラスターが「クラスター」フォルダーに表示されなくなります。

質問に戻る

ネットワーク上のいくつかの iSeries システムで、既にクラスターが定義されています。 iSeries ナビゲーターを使ってこれらを表示および管理するために追加するには、どうすればよいですか?

既存のクラスターを iSeries ナビゲーターで表示されるように追加するには、以下の手順で行います。

1. iSeries ナビゲーターで、「**マネージメント・セントラル**」を展開します。
2. 「**クラスター**」を右マウス・ボタン・クリックしてから、「**既存のクラスターの追加**」を選択します。
3. 「**既存のクラスターの追加**」ウィンドウで、クラスターのサーバーの 1 つを指定します。
4. 「OK」をクリックします。

質問に戻る

クラスター内のノードで状況が「開始済み」になっているものが 1 つもありません。どのノードをまず開始するべきですか?

最近まで、状況が「開始済み」だったノードを開始します。たとえば、クラスター内に A と B の 2 つのノードがあるとします。 ノード A は現在開始されておらず、ノード B も現在開始されていません。 しかし、ノード B は状況が「開始済み」として最後まで実行されていたノードです。ノート B はクラスターに関する最新の情報を保持しているため、ノード B を最初に開始するべきです。

質問に戻る

なぜ、どのノードを最初に開始させるべきかに注意する必要がありますか?

最近まで状況が「開始済み」となっていたノードが、クラスターに関する最新の情報を含んでいるノードであるためです。もし最も長い時間ダウンしていた他のノードから開始した場合、そのノードが持っているクラスターに関する情報が古くなっている可能性があるため、これは重要なことです。これらの他のノードから開始した場合、クラスター内の他のノードに対して、古い情報が伝搬される危険があります。たとえば、ノード A と B を持つ 2 ノード・クラスターがあったとします。 ノード B が最近まで「開始済み」の状況だったアクティブ・ノードの場合、最新のクラスター情報はノード B に含まれています。 ノード A を最初に開始することを選択すると、ノード A にはいくつかの古い情報が含まれている可能性がありますが、それでも開始されます。その後、ノード B を開始させると、ノード B はクラスター内の現在のアクティブ・ノードと結合します (ノード A と結合します)。 ノード A にある古いクラスター情報がノード B に伝搬され、その結果、両方のノードがクラスターに関する古い情報を持つことになります。これが、ノード B を最初に開始することが重要である理由です。古いクラスター情報は、切り替え可能装置の構成に影響を与える可能性があります。切り替え可能ハードウェア・グループが別の現行ノードを表示しているときに、バックアップ・ノード上でディスク装置が報告を行ったことが原因で、切り替え可能装置の開始でなんらか問題が生じた場合は、リカバリー・ドメイン内のノードの役割を変更して、ディスク装置のノードを所有するノードをプライマリー・ノードにする必要があります。

質問に戻る

「切り替え可能ハードウェア」、「切り替え可能ソフトウェア」、および「切り替え可能データ」フォルダーの中にある「現行プライマリー・ノード」列は何を意味していますか？

「現行プライマリー・ノード」列には、切り替え可能装置または切り替え可能ソフトウェア・プロダクトのプライマリー・ノードとして現在動作しているノードが示されます。または、クラスター API 用語では、プライマリー・リカバリー・ドメイン内で現行の役割を持つノードを意味しています。

質問に戻る

iSeries ナビゲーターで、装置クラスター・リソース・グループ (CRG) を探し出す方法は？

装置 CRG (クラスター・リソース・グループ) は、切り替え可能ハードウェア・グループとして参照され、「クラスター」フォルダー内の「**切り替え可能ハードウェア**」フォルダーにあります。

質問に戻る

iSeries ナビゲーターで、アプリケーション・クラスター・リソース・グループ (CRG) を探し出す方法は？

アプリケーション CRG (クラスター・リソース・グループ) は、切り替え可能ソフトウェア・プロダクトとして参照され、「クラスター」フォルダー内の「**切り替え可能ソフトウェア**」フォルダーにあります。

質問に戻る

iSeries ナビゲーターで、データ・クラスター・リソース・グループ (CRG) を探し出す方法は？

データ CRG (クラスター・リソース・グループ) は、切り替え可能データ・グループとして参照され、「クラスター」フォルダー内の「**切り替え可能データ**」フォルダーにあります。

質問に戻る

「切り替え可能ハードウェア」フォルダーまで戻らずに、切り替え可能ハードウェア・グループ (装置 CRG) の状況を見たいと思います。どのようにすればできますか?

状況を見たいときに毎回「切り替え可能ハードウェア」フォルダーまでナビゲートする代わりに、「**切り替え可能ハードウェア**」フォルダー上で右マウス・ボタン・クリックし、「**オープン**」を選択して、切り替え可能ハードウェア・ビューの新しいウィンドウを開くこともできます。独立したこのウィンドウは、切り替え可能ハードウェア・グループ (デバイス CRG) と、それに関連した状況情報を表示します。これは、「**切り替え可能ソフトウェア**」および「**切り替え可能データ**」にも適用されます。

質問に戻る

通信: iSeries ナビゲーターのクラスター機能は、クラスター内のノードとの通信で、どの IP アドレスを使用しますか？ ノード名の IP アドレスを使用するのではないのですか？

メインの「クラスター」フォルダーの中にある「サーバー」列は、構成されたクラスターに関する情報を表示するものです。各クラスターのプロパティー・パネルには、サーバー名があります。「サーバー」列にリストされているサーバーは、iSeries ナビゲーター・インターフェースがクラスターと通信するために使用するクラスター内のノードです。これはこのサーバー上のクラスター・オブジェクトと iSeries ナビゲーターが通信する方法に適用され、クラスター内のノードが他と通信する方法には適用されません。 iSeries ナビゲーター・クラスター管理によって使用されるサーバーは、現行のマネージメント・セントラル・サーバーとは何も行いません。

iSeries ナビゲーターがクラスターと通信するために使用するノードがダウンしている場合、クラスターの処理を実行するため、通信手段をクラスター内の他のノードに変更することができます。

iSeriesナビゲーター・インターフェースがクラスターと通信するために使用するサーバーを変更するには、以下の手順で行います。

1. iSeries ナビゲーターで、「**マネージメント・セントラル**」を展開します。
2. 「**クラスター**」を展開します。
3. クラスターを右マウス・ボタン・クリックし、「**サーバーの変更**」を選択します。

質問に戻る

セキュリティー: iSeries iSeries ナビゲーターの「クラスター」フォルダーのコンテキスト・メニュー内の多くが、使用不可または表示されないのはなぜですか？

いくつかの操作は、クラスターの現行の構成の状態によってのみ、使用可能になります。たとえば、既に停止しているノードを停止することはできず、ノードの最大数として構成されている 4 つのノードが既に存在しているクラスターにさらにノードを追加することはできません。それぞれのタスクのオンライン・ヘルプには、なぜそれらの項目のいくつかが利用不能または無効になるのかについて説明されています。

いくつかの操作は、十分な権限がない場合に利用不能になります。 iSeries ナビゲーターを使用しており、*SECOFR ユーザー・クラス権限を持つ場合、すべてのクラスターの操作、および管理にアクセスできます。 iSeries ナビゲーターは現行のマネージメント・セントラル・システムのアプリケーション管理者権限を使用して、 iSeries ナビゲーター・クラスター管理のさまざまな操作のためのアプリケーション管理者権限を持っているかどうかを判断します。

アプリケーション管理の処理の詳細については、 アプリケーション管理を参照してください。

質問に戻る

iSeries ナビゲーターのクラスター機能は、アプリケーション管理値を使用しますか？

はい。iSeries ナビゲーター・クラスター管理は現行のマネージメント・セントラル・システムのアプリケーション管理者権限値を使用して、さまざまなクラスター操作のためのアプリケーション管理者権限を持っているかどうかを判断します。

iSeries ナビゲーターは、アクセスするための 2 つのタイプの権限設定、「**クラスター操作**」および「**クラスター管理**」を持っています。

「**クラスター操作**」権限では、次のことが可能です。
- クラスターの状況の表示
- ノードの開始および停止
- 切り替え可能ハードウェアおよび切り替え可能ソフトウェアの開始および停止
- 切り替え可能ハードウェアおよび切り替え可能ソフトウェアの手動切り替えの実行

「**クラスター管理**」権限では、次のことが可能です。
- クラスターの作成/削除
- ノードの追加および削除
- 切り替え可能ハードウェア、切り替え可能ソフトウェア、およびディスク・プールの追加および削除
- 切り替え可能ハードウェアおよび切り替え可能ソフトウェアのプロパティーの変更

質問に戻る

iSeries ナビゲーターのクラスター機能が、クラスター内の自分のノードへのサインオン・ウィンドウを表示するのはなぜですか？

いくつかのケースで、iSeries ナビゲーターはクラスター内のすべてのノードと通信しようとします。これはクラスターの状態に依存します。 iSeries ナビゲーターがノードと通信する必要があるときは、まず既存のオープンした接続を検索しようとして、iSeries ナビゲーターの既存のサインオン・キャッシュを検索します。既存の接続が見つからなかった場合、ユーザーにサインオンを求めます。サインオン・ウィンドウでキャンセルした場合、 iSeries ナビゲーターはユーザーにクラスター操作を許可しようとします。 iSeries ナビゲーターがノードと通信できない場合、いくつかの操作が不可能になります。

質問に戻る

トラブルシューティング: 「クラスター」フォルダーがマネージメント・セントラルの下に表示されないのはなぜですか？

PC 上に、iSeries Access for Windows® のフルインストールが行われていない可能性があります。基本インストールを行ったか、いくつかのカスタム・オプションを選択したのかもしれません。インストールの詳細は、iSeries Access を参照してください。

質問に戻る

既にクラスターがあるのに、「クラスター」フォルダーに表示されません。なぜですか？

端的に言えば、マネージメント・セントラル・システム上にクラスターを表すシステム・グループがないため、表示されません。クラスターを表すシステム・グループは、クラスターが作成されるか、または「既存のクラスターの追加」アクションによって「クラスター」フォルダーにクラスターが追加されるときに、iSeries ナビゲーターによって作成されます。システム・グループを表示するには、「マネージメント・セントラル」内の「**システム・グループ**」フォルダーを展開します。クラスター・システム・グループが「サード・パーティー・プラグイン」システム・グループとして表示されますが、すべての「サード・パーティー・プラグイン」システム・グループがクラスターというわけではありません。

質問に戻る

「クラスター」フォルダーに最新の状況が表示されないのはなぜですか？

iSeries ナビゲーターは構成されたクラスターに関する情報を、クラスター・ノードに行き、そのクラスターに関する最新の情報を取得してから、スナップショットとして iSeries ナビゲーター・ウィンドウにその情報を表示します。この情報の定期更新は自動的には行われません。情報の最新のスナップショットを得る最善の方法は、手動で最新表示することです。 iSeries ナビゲーターの「**表示**」メニューを使って、「**最新表示**」オプションを選択します。別の方法は、自動的に最新表示が行われるように iSeries ナビゲーターをセットアップすることです。

質問に戻る

切り替え可能装置、切り替え可能アプリケーション、または切り替え可能データ・グループのフェイルオーバーが行われないのはなぜですか？

この場合、最もありそうなシナリオは、クラスター内で開始された切り替え可能リソース (クラスター・リソース・グループ) がない、というケースです。言い換えると、自動フェイルオーバーを実行する前に、切り替え可能リソースの状況が「開始済み」になっていなかった、ということです。フェイルオーバーを実行させるには、切り替え可能リソースが開始されていなければなりません。

質問に戻る

オブジェクトに損傷があるというメッセージを受け取りました。どのように対応できますか？

次のようなメッセージを受け取ることがあります。 CPF811C QCLUMGT のユーザーの待ち行列 QUGCLUSRQ に損傷がある

オプション 1: 1 つのオプションは、そのオブジェクトを削除し、復元するというものです。これは、そのオブジェクトを以前に保存してある場合にのみ可能です。

オプション 2:損傷があるオブジェクトを削除します。たとえば、ライブラリー QCLUMGT 内の QUGCLUSRQ が損傷を受けた場合は、このオブジェクトを削除します。その後、iSeries ナビゲーター内に既存のクラスターを追加します。クラスターを追加することにより、クラスター GUI はそのオブジェクトが存在しているかどうかをチェックし、存在していなければ再作成します。既存のクラスターの追加の詳細については、iSeries ナビゲーターを使ってこれらを表示および管理するために追加するには、どうすればよいですか？ を参照してください。

質問に戻る

IP アドレスを参照するため、ノードのウィザードで「参照」ボタンを使っています。表示されるはずの TCP/IP アドレスのすべてが参照ウィンドウに表示されないのはなぜですか？

このリストは、可能性のある IP アドレスの候補をリストしているだけです。このウィンドウに表示される、可能性のあるアドレスのリストに限定されているわけではありません。任意のクラスター・インターフェース・アドレスを入力することができます。しかし、ユーザーがプライマリー IP アドレスとして指定した IP アドレスを使って iSeries ナビゲーターが接続できない場合、後でエラーを受け取ることになるため、注意してください。 iSeries ナビゲーターはクラスター内のノードに接続するために、プライマリー IP アドレスを使用します。

質問に戻る

「新規クラスター」ウィザードを使用すると、「新規クラスター - 切り替え可能ソフトウェアが見つかりません」というタイトルのパネルが表示されました。何か問題がありますか？

いいえ、これは問題ではなく、エラーでもありません。これは正確には、iSeries ナビゲーター・インターフェースが、ウィザードを使って自動的にインストールされる切り替え可能ソフトウェアを見つけることができなかったことを示しています。 iSeries ナビゲーターでは、クラスター使用可能アプリケーション用の i5/OS アーキテクチャーに適合した、自動インストール可能な切り替え可能ソフトウェアが必要です。さらに、iSeries ナビゲーターはこのアーキテクチャーのサブセットのみをサポートしています。全体ではありません。

質問に戻る

ノードの 1 つの状況が「通信なし」となっています。どのように訂正できますか？

クラスター内の 1 つ以上のノードとの連絡が途絶え、さらにそのノードの障害を確認できない場合は、クラスター区画が発生します。詳細については、区画エラーを参照してください。

実際にノード障害があるときに、区画状態が報告されることがあります。このことは、クラスター・リソース・サービスが 1 つ以上のノードとの通信を失ったものの、ノードが引き続き作動可能かどうかを検出できない場合に生じます。この状況が生じた場合に、ノードに障害が起きたことを知らせるための簡単なメカニズムがあります。詳細については、区画化ノードを障害ノードに変更するを参照してください。

質問に戻る

クラスター・サポートについての問い合わせ先

クラスターの問題に関して IBM に問い合わせる必要がある場合は、このトピックを参照してください。

クラスターが業務上の利益を生じるかどうかを判断する場合や、クラスターをインプリメントした後で何らかの問題が発生した場合、または製品サービスなどの詳細については、弊社もしくは IBM ビジネス・パートナーの営業担当員にご相談ください。

- マーケティングでの技術的なおたずねや、IBM の相談サービスをご利用になりたい場合は、電子メール (rchclst@us.ibm.com) にて iSeries Technology Center 内の Continuous Availability Center (米国) にお問い合わせください (米国のみ)。
- それ以外の問題については、クラスタリング・ソフトウェア・パッケージを購入されたビジネス・パートナーにお問い合わせいただくか、または 1-800-IBM-4YOU (1-800-426-4968) (米国) へお電話ください。(米国のみ)

関連タスク

110 ページの『クラスターの構成』
クラスターの構成方法について理解できます。

クラスターの関連情報

クラスターに関する関連情報を検索します。

Redbooks

- Data Resilience Solutions for IBM i5/OS High Availability Clusters
- Clustering and IASPs for Higher Availability
- High Availability on the AS/400® System: A System Manager's Guide
- IBM eServer iSeries Independent ASPs: A Guide to Moving Applications to IASPs
- The System Administrator's Companion to AS/400 Availability and Recovery

Web サイト

- High Availability and Clusters (www.ibm.com/servers/eserver/iseries/ha)

  高可用性およびクラスターに関する IBM サイト

PDF ファイルの保存

表示または印刷のために PDF をワークステーションに保存するには、以下のようにします。

1. ブラウザーで PDF を右マウス・ボタンでクリックする (上部のリンクを右マウス・ボタン・クリック)。
2. Internet Explorer を使用している場合は、「**対象をファイルに保存...**」をクリックする。 Netscape Communicator を使用している場合は、「**リンクを名前を付けて保存...**」をクリックする。
3. PDF の保存先のディレクトリーに移動する。
4. 「**保存**」をクリックする。

Adobe Acrobat Reader のダウンロード

これらの PDF を表示または印刷するには、Adobe Acrobat Reader が必要です。このアプリケーションは、Adobe Web サイト (www.adobe.com/products/acrobat/readstep.html) からダウンロードできます。

コードに関するライセンス情報および特記事項

IBM は、お客様に、すべてのプログラム・コードのサンプルを使用することができる非独占的な著作使用権を許諾します。お客様は、このサンプル・コードから、お客様独自の特別のニーズに合わせた類似のプログラムを作成することができます。

強行法規で除外を禁止されている場合を除き、IBM、そのプログラム開発者、および供給者は「プログラム」および「プログラム」に対する技術的サポートがある場合にはその技術的サポートについて、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。

IBM、そのプログラム開発者、または供給者は、いかなる場合においてもその予見の有無を問わず、以下に対する責任を負いません。

1. データの喪失、または損傷。
2. 直接損害、特別損害、付随的損害、間接損害、または経済上の結果的損害
3. 逸失した利益、ビジネス上の収益、あるいは節約すべかりし費用

国または地域によっては、法律の強行規定により、上記の責任の制限が適用されない場合があります。

付録. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、IBM 機械コードのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

表示されている IBM の価格は IBM が小売り価格として提示しているもので、現行価格であり、通知なしに変更されるものです。卸価格は、異なる場合があります。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _年を入れる_. All rights reserved.

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

本書「クラスター」には、プログラムを作成するユーザーが IBM i5/OS のサービスを使用するためのプログラミング・インターフェースが記述されています。

商標

以下は、IBM Corporation の商標です。

| 400
| i5/OS
| IBM
| iSeries
| OS/400
| Redbooks

| Intel、Intel Inside (ロゴ)、および Pentium は、Intel Corporation の米国およびその他の国における商標で
| す。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

| Linux は、Linus Torvalds の米国およびその他の国における商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

使用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

個人使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することは できません。

商業的使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。ただし、 IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、第三者の権利の不侵害の保証、商品性の保証、特定目的適合性の保証および法律上

の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

Printed in Japan

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12