IBM PowerHA SystemMirror for AIX
Standard Edition

バージョン 7.2.1

PowerHA SystemMirror のクライアント・アプリケーションのプログラミング

IBM

IBM PowerHA SystemMirror for AIX Standard Edition

バージョン **7.2.1**

PowerHA SystemMirror のクライアント・アプリケーションのプログラミング

IBM

お願い

本書および本書で紹介する製品をご使用になる前に、 113 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 PowerHA SystemMirror 7.2.1 Standard Edition for AIX および新しい版で明記されていない限り、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。

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 PowerHA SystemMirror for AIX
Standard Edition
Version 7.2.1
Programming client applications for
PowerHA SystemMirror

発行： 日本アイ・ビー・エム株式会社

担当： トランスレーション・サービス・センター

目次

本書について

本書では、PowerHA® SystemMirror® ソフトウェアで提供されるクラスター情報プログラム (Clinfo) クライアント・アプリケーション・プログラミング・インターフェース (API) および管理情報ベース (MIB) について説明します。

強調表示

本書では、以下の強調表示規則を使用します。

太字	システムによって名前が事前に定義されているコマンド、サブルーチン、キーワード、ファイル、構造、ディレクトリー、およびその他の項目を示します。また、ユーザーが選択するボタン、ラベル、アイコンなどのグラフィカル・オブジェクトも示します。
イタリック	実際の名前または値をユーザーが指定する必要があるパラメーターを示します。
モノスペース	特定のデータ値の例、画面に表示されるものと同様のテキスト例、プログラマーが作成するものと同様のプログラム・コード部分の例、システムからのメッセージ、実際に入力する必要がある情報などを示します。

AIX での大/小文字の区別

AIX® オペレーティング・システムは、すべてケース・センシティブとなっています。これは、英大文字と小文字が区別されるということです。例えば、**ls** コマンドを使用するとファイルをリスト表示できます。LS と入力した場合、そのようなコマンドは**ない**という応答がシステムから返ってきます。同様に、**FILEA**、**FiLea**、および **filea** は、同じディレクトリーにある場合でも、3 つの異なるファイル名です。予期しない処理が実行されないように、常に正しい大/小文字を使用するようにしてください。

ISO 9000

当製品の開発および製造には、ISO 9000 登録品質システムが使用されました。

関連情報

- PowerHA SystemMirror バージョン 7.2.1 の PDF 資料は、『PowerHA SystemMirror 7.2.1 PDFs』トピックで入手可能です。
- PowerHA SystemMirror バージョン 7.2.1 のリリース・ノートは、『PowerHA SystemMirror 7.2.1 release notes』トピックで入手可能です。

PowerHA SystemMirror のクライアント・アプリケーションのプログラミング

この資料は、PowerHA SystemMirror ソフトウェアで提供されるクラスター情報プログラム (Clinfo) クライアントのアプリケーション・プログラミング・インターフェース (API) および管理情報ベース (MIB) について説明します。

アプリケーションは、PowerHA SystemMirror for AIX MIB に保管されている PowerHA SystemMirror クラスターに関する情報にアクセスできます。 アプリケーションによるアクセスは、Simple Network Management Protocol (SNMP) 要求を行って直接的に、または Clinfo の C または C++ API を使用して間接的に行われます。

クラスター情報プログラム

ここに含まれるトピックでは、クラスター情報プログラム (Clinfo) の概要を紹介し、PowerHA SystemMirror for AIX クラスターに関して Clinfo が受け取り、維持する状況情報について説明します。

クラスター情報プログラムの概要

PowerHA SystemMirror クラスターは、時間とともにさまざまな状態に変化することがあります。 例えば、ノードがクラスターに参加またはクラスターから離脱したり、アプリケーションがバックアップ・ノードにフォールオーバーしたりすることがあります。 これらの変化はそれぞれにクラスターの状態に影響します。クラスターは動的であるため、アプリケーションは、変化が起こったときにそれに対応できるよう、クラスターに関する最新かつ正確な情報を取得できる必要があります。 Clinfo はこのサービスを提供します。

PowerHA SystemMirror クラスター・ソフトウェアは、業界標準のネットワーク・プロトコルである SNMP を通じて状態情報とクラスター・イベントをエクスポートします。 SNMP API 用のプログラムを作成することは容易でない場合があり、ノードやリソース・グループなどの単一クラスター・エンティティーに関連するすべての情報を取得するために複数のトランザクションが必要になることがあります。

Clinfo API および関連のコンポーネントには、簡単なプログラミング・モデルを使用して同じクラスター情報を提供するアクセス・ルーチンのライブラリーがあります。 PowerHA SystemMirror から入手できる SNMP 情報の詳細、または Clinfo API の実装に関する詳細については、『実装の詳細』を参照してください。

関連資料:

108 ページの『実装の詳細』

Clinfo には、**clinfo** デーモンと API ライブラリーの 2 つのキー・コンポーネントがあります。

始めに

PowerHA SystemMirror をインストールすると、Clinfo API とその関連のコンポーネントがインストールおよび構成されます。

Clinfo の使用を開始する前に、いくつか考慮すべき事項があります。

1. Clinfo を使用するには、PowerHA SystemMirror クラスター・サービスの開始時に Clinfo エージェントを開始することを指定する必要があります。 これには、`smitty clstart` の「クラスター情報デーモンを始動する **(Startup cluster information Daemon)**」オプションを使用します。
2. Clinfo は、public 以外の SNMP コミュニティー名を認識し、使用します。 ご使用のインストール済み環境で public 以外の名前が使用されている場合、または特定の名前を指定する必要がある場合は、『実装の詳細』を参照してください。
3. Clinfo は、PowerHA SystemMirror をインストールした各ノードで実行されます。 また、いずれかの PowerHA SystemMirror ノードとの TCP/IP 接続を持つその他のマシンでも実行できます。 この機能の追加情報については、『実装の詳細』を参照してください。

関連資料:

108 ページの『実装の詳細』

Clinfo には、**clinfo** デーモンと API ライブラリーの 2 つのキー・コンポーネントがあります。

Clinfo API

アプリケーションは、Clinfo API 関数を通じてクラスター情報にアクセスします。 開発者は Clinfo C API または Clinfo C++ API を使用してクラスターの状況情報にアクセスできます。それにより、Clinfo プログラムを実行するクライアントがこの情報をローカルで使用できます。

PowerHA SystemMirror for AIX には 2 つのバージョンの Clinfo C および C++ API ライブラリーが含まれています。1 つは単一スレッド (非スレッド) アプリケーション用 (**libcl.a** および **libclpp.a**)、もう 1 つはマルチスレッド・アプリケーション用 (**libcl_r.a** および **libclpp_r.a**) です。 必ずご使用のアプリケーションに適したバージョンとリンクしてください。 **libcl_r.a** は **libcl.a** のスレッド・セーフ・バージョンで、**libclpp_r.a** は **libclpp.a** のスレッド・セーフ・バージョンです。

これらの各ライブラリーには 32 ビットと 64 ビットのオブジェクトが含まれ、これらのオブジェクトは AIX 稼働環境に応じて実行時にロードされます。

Clinfo C API および Clinfo C++ API のルーチンの詳しい説明については、これらの API を参照してください。

関連概念:

10 ページの『Clinfo C API』

Clinfo C アプリケーション・プログラミング・インターフェース (API) は、PowerHA SystemMirror クラスターに関する状況情報を取得するためにアプリケーションで使用できる高水準インターフェースです。ここに含まれるトピックでは、Clinfo C API で使用可能なそれぞれの C 言語のルーチンおよびユーティリティーについて説明します。

69 ページの『Clinfo C++ API』

Clinfo C++ API は、PowerHA SystemMirror for AIX クラスターに関する状況情報を取得するために C++ アプリケーションで使用できるオブジェクト指向インターフェースです。 ここに含まれるトピックでは、Clinfo C++ API で使用可能なそれぞれの C++ 言語オブジェクトおよびメソッドについて説明します。

Clinfo によってトラッキングされるイベント

Clinfo は、クラスター・マネージャーからクラスター・イベントに関する状況情報を受け取ります。 この情報には、API 内のルーチンによりアクセスできます。 Clinfo は、クラスターがさまざまな状態を推移するときに、トポロジー・イベントをトラッキングします。

Clinfo がトラッキングする状態には、以下が含まれます。

- クラスターが稼働状態またはダウン状態である
- クラスターの副状態が安定または不安定になった
- アプリケーションがオンラインになった、またはバックアップ・ノードにフェイルオーバーした
- ネットワークで障害が起こった
- ノードがクラスターに参加している途中である
- ノードがクラスターに参加した
- ノードがクラスターを離脱中である (すなわち、ノードで障害が起こった)
- ノードがクラスターを離脱した
- 新しい 1 次クラスター・マネージャーが選出された (オプション・イベント)
- クラスター・サイト (構成されている場合) の状態の変化

イベントの詳細リストは本書の後のセクションに記載されています。または、アプリケーション内にコンパイルされる **clinfo.h** インクルード・ファイルにもあります。

Clinfo は動的再構成イベントを受け取りますが、そのイベントをトラッキングしません。つまり、アプリケーションは動的再構成イベントの通知を受け取るための登録ができません。 Clinfo は動的再構成イベントを受け取ると、クラスターの副状態を CLSS_RECONFIG に設定します。 アプリケーションは **cl_getcluster** ルーチンを使用してこの情報を取得できます。 ノードの稼働状態またはダウン状態を示すイベントなど、動的再構成によってトリガーされるイベントは、アプリケーションから見ることができます。

Clinfo によってトラッキングされるクラスター情報

クラスター情報の一部は Clinfo によって維持されます。

クラスター

PowerHA SystemMirror クラスターは、協働して高可用性環境を実現するプロセッサーのグループのことです。

使用可能なクラスター情報

Clinfo は、構成済みクラスターに関して以下の情報を維持します。

クラスター名

クラスター名はクラスターを一意的に識別します。 管理者はクラスターの構成時にその名前を指定します。

クラスター ID

クラスター ID は各クラスターを識別します。 クラスター ID は、クラスターの構成時に PowerHA SystemMirror によって割り当てられる (つまり、ユーザー定義でない) 数値です。

クラスターの状態

クラスターは、以下の定義済み状態のいずれかに該当します。

項目	説明
CLS_UP	クラスター内の少なくとも 1 つのノードが稼働状態であり、1 次ノードが定義されています。
CLS_DOWN	稼働状態のノードはありません。
CLS_UNKNOWN	Clinfo が通信できないか、またはどのアクティブ・クラスター上の SNMP プロセスともまだ通信していません。
CLS_ NOTCONFIGURED	クラスターはまだ構成されていません。

クラスターの副状態

クラスターは、いくつかの定義済み副状態のいずれかに該当します。

項目	説明
CLSS_ERROR	エラーが発生して、手操作による介入が必要です。
CLSS_RECONFIG	クラスターの動的再構成が進行中です。
CLSS_STABLE	クラスターは安定 (再構成が行われていない) 状態です。
CLSS_UNSTABLE	クラスターは不安定状態です。 イベント・ヒストリーにどのイベントが起こっているかが示されます。
CLSS_UNKNOWN	Clinfo は、クラスター・ノード上の SNMP プロセスと通信できません。
CLSS_ NOTCONFIGURED	クラスターはまだ構成されていません。
CLSS_NOTSYNCHED	一部の基本クラスター構成情報は存在しますが、そのクラスターの検証と同期はまだ行われていません。

1 次ノード名

1 次ノードの指定は、前のリリースにあった非推奨機能での指定がそのまま残ったものです。 現在は何も機能がなく、動作への影響はありません。 バイナリー互換のためだけに維持されています。

ノードのクラスター数

クラスター内で定義されているノードの数。

ネットワークのクラスター数

クラスター内のネットワークの総数。

リソース・グループのクラスター数

構成済みのリソース・グループの総数。

サイトのクラスター数

サイトが使用されている場合は、構成済みのサイトの数。

ノード

ノードとは、クラスターを構成するそれぞれのプロセッサーのことです。 クラスター内の各ノードは **clstrmgr**、**clinfo**、および **clsmuxpd** デーモンを実行します。

使用可能なノード情報

Clinfo は、ノードに関して以下の情報を維持します。

クラスター ID

このノードが属するクラスターの ID。

ノード名

ノード名はユーザーが割り当てるストリングです。ノード名には最大 32 文字を指定でき、先頭は数値であってはなりません。

ノードの状態

ノードは、以下の定義済み状態のいずれかに該当します。

項目	説明
CLS_UP	ノードは稼働中です。
CLS_DOWN	ノードはダウン状態です。
CLS_JOINING	ノードはクラスターに参加している途中です。
CLS_LEAVING	ノードはクラスターを離脱している途中です。

ネットワーク・インターフェース

ノードに接続されているサービス・インターフェースの番号およびアドレスです。

ネットワーク・インターフェース

ネットワーク・インターフェースとは、ノードとネットワークとの間の物理接続のことです。

PowerHA SystemMirror クラスターは複数のネットワーク接続および Point-to-Point 接続をサポートし、さらに RS232 シリアル・ライン Point-to-Point 接続もサポートします。 各ネットワークには、各クラスター・ノード上に 1 つ以上のネットワーク・インターフェースがあります。

使用可能なネットワーク・インターフェース情報

Clinfo は、ネットワーク・インターフェースに関して以下の情報を維持します。

クラスター ID

このインターフェースが属するクラスターの ID。

ノード名

このインターフェースが接続されているノードの名前。

アクティブ・ノード ID

そのアドレスが現在アクティブであるノードの ID。

インターフェース名

インターフェースの名前はそのインターフェースの **/etc/hosts** ファイルにある名前と同じです (つまり、ホストの IP アドレスと関連付けられている名前)。

インターフェース ID

このインターフェースが接続されているネットワークのネットワーク ID。

インターフェース・アドレス

/etc/hosts ファイルで定義されているインターフェースの IP アドレス。

インターフェースの状態

インターフェースは、いくつかの定義済み状態のいずれかに該当します。 以下の値は、ネットワーク・インターフェースの状態を示します。

項目	説明
CLS_UP	インターフェースは稼働中です。
CLS_DOWN	ネットワーク・インターフェースまたはネットワークがダウン状態です。
CLS_INVALID	このインターフェースはこのノードに定義されていません。

インターフェースのロール

インターフェースは、いくつかの定義済みロールのいずれかに該当します。 以下の値は、ネットワーク・インターフェースのロールを示します。

項目	説明
CL_INT_ROLE_INVALID	ネットワーク・インターフェースは無効です。
CL_INT_ROLE_SERVICE	ネットワーク・インターフェースはサービス・インターフェースとして定義されています。
CL_INT_ROLE_STANDBY	そのネットワーク・インターフェースのロールは非推奨のものです。
CL_INT_ROLE_BOOT	ネットワーク・インターフェースはサービス・インターフェースとして定義されています。
CL_INT_ROLE_SH_SERVICE	そのネットワーク・インターフェースのロールは非推奨のものです。

リソース・グループ

リソース・グループには、PowerHA SystemMirror が高可用性を確保しているアプリケーションのインスタンスに関連付けられているすべてのリソースが含まれます。

リソース・グループは、ノードがクラスターに参加するときにオンラインになります。リソース・グループは、障害が発生するとクラスター・ノード間で移動されます。グループの状態とロケーションは Clinfo を通じて入手できます。

使用可能なリソース・グループ情報

Clinfo は、リソース・グループに関して以下の情報を維持します。

クラスター ID

このリソース・グループが属するクラスターの ID。

グループ名

これは、リソース・グループが最初に定義されるときに付けられる名前です。

グループ ID

そのグループに関連付けられる数値の ID。

グループの始動ポリシー

このリソース・グループに使用される始動ポリシー:

- ホーム・ノードのみでオンライン
- 最初に使用可能なノードでオンライン
- すべての使用可能ノードでオンライン
- 配布ポリシーを使用してオンライン

グループのフォールオーバー・ポリシー

このリソース・グループのフォールオーバー・ポリシー:

- リスト内の次の優先順位のノードにフォールオーバー
- 動的ノード優先順位を使用してフォールオーバー
- オフラインにする (エラー・ノード上でのみ)

グループのフォールバック・ポリシー

このリソース・グループのフォールバック・ポリシー:

- リスト内で優先順位のより高いノードにフォールバック
- フォールバックしない

グループのサイト・ポリシー

そのグループに複製リソースが含まれる場合、2 つのサイト間でリソース・グループの始動、フォールオーバー、およびフォールバックが行われる際に、以下のポリシーが使用されます。

- 1 次サイトを優先
- 一方のサイトでオンライン
- 両方のサイトでオンライン

ノードの数

リソース・グループに参加しているノードの数。

グループのノード ID

リソース・グループに参加しているすべてのノードのノード ID。

グループのノード状態

各ノード上のリソース・グループの状態。

リソース・グループの状態

リソース・グループは、1 つ以上のクラスター・ノードで以下のいずれかの状態に該当します。

項目	説明
CL_RGNS_INVALID	そのノードはこのリソース・グループに含まれません。
CL_RGNS_ONLINE	このグループおよびそのすべてのリソースは稼働状態であり、ノードで使用できます。
CL_RGNS_OFFLINE	このグループは現在そのノードでアクティブではありません。
CL_RGNS_ACQUIRING	このグループは、このノードで獲得されている途中です。
CL_RGNS_RELEASING	このグループは、このノードから解放されている途中です。
CL_RGNS_ERROR	グループをオンラインまたはオフラインに切り替えようとして、エラーが発生しました。 このグループのリソースはアクティブではありません。 手操作による介入が必要です。

注**:** このリストにはリソース・グループについて考えられるすべての状態が含まれているわけではありません。サイトが定義されている場合、リソース・グループの 1 次および 2 次インスタンスはオンライン、オフライン、エラー状態、または管理対象外である可能性があります。 さらに、リソース・グループのインスタンスは、獲得または解放の途中である場合も考えられます。 これに対応するリソース・グループの状態はここにリストされていませんが、どのアクションを実行するかを説明する描写的な名前が付いています。

クラスター・ネットワーク

PowerHA SystemMirror クラスターには、TCP/IP ネットワークと非 IP ベース・ネットワークの両方が含まれる場合があります。通常、非 IP ベース・ネットワークはディスク・ハートビート・トラフィックに使用されます。

使用可能なネットワーク情報

Clinfo デーモンは、クラスター内のネットワークに関する以下の情報を提供します。

ネットワーク名

ネットワークに関連付けられた名前。この名前はユーザーが指定できます。または PowerHA SystemMirror によって生成することもできます。

ネットワーク ID

PowerHA SystemMirror によって生成される数値のネットワーク ID。

ネットワーク・タイプ

イーサネットなどの、物理タイプのネットワーク。

ネットワーク属性

IP ベース・ネットワークのネットワーク属性は、「パブリック」または「プライベート」に設定することができます。この属性を「プライベート」に設定すると、このネットワークは Oracle によってプライベート・ネットワークとして使用されることを示します。 デフォルトは「パブリック」です。

ネットワーク・ノード情報

そのネットワークに接続されているクラスター・ノードごとに、以下の情報が提供されます。

項目	説明
ノード **ID**	各ノードのノード ID。
ノードの状態	ノード上のネットワークの状態。 この値は、グローバル・ネットワーク状態と異なる場合があります。

ネットワークの状態

ネットワーク状態にはグローバル状態とノードごとの状態があり、いくつかの事前定義値のいずれかになります。

項目	説明
CLS_UP	ネットワークは稼働状態です。 ネットワークがノード上で起動している場合、グローバル状態もセットアップされています。
CLS_DOWN	ネットワークは稼働状態ではありません。 ネットワークは 1 つ以上のノードでダウン状態の可能性があり (つまり、ノードごとの状態は **CLS_DOWN** です)、グローバル状態は引き続き **CLS_UP** です。

クラスター・サイト

サイトが構成されると、クラスター内のノードはクラスター・サイトにグループ化されます。 クラスター・サイトの状態はトラッキングされ、クラスター・イベントが生成されます。

使用可能なクラスター・サイト情報

Clinfo は、クラスター・サイトに関して以下の情報を維持します。

サイト ID

PowerHA SystemMirror が生成したサイトの ID。

サイト名

サイトの作成時に指定されたサイト名。

サイトの優先順位

サイトの優先順位は、サイト間でのデータのミラーリングの優先度と方向を決定します。 クラスター・サイトには、以下のいずれかの優先度を指定できます。

項目	説明
CL_SITE_PRIMARY	これは、アプリケーションが実行されるサイトです。 データはこのサイトからバックアップ・サイトにミラーリングされます。
CL_SITE_SECONDARY	このサイトはバックアップとして機能します。
CL_SITE_TERTIARY	このサイトは 2 次サイトから送られたデータをミラーリングします。 この値は、将来使用するために予約されています。

サイトのバックアップ方式

これは、そのサイトのバックアップ通信方式です。 バックアップ方式を構成する場合、以下のいずれかの方式になります。

項目	説明
CL_SITE_BACKUP_DBFS	ダイヤル・バック・フェイルセーフ
CL_SITE_BACKUP_SGN	シリアル Global Network
CL_SITE_BACKUP_NONE	バックアップ通信は構成されていません。

サイトの状態

サイトのグローバル状態:

項目	説明
CLS_UP	サイト内の 1 つ以上のノードが稼働状態です。
CLS_DOWN	サイト内のすべてのノードがダウン状態です。

サイトのノード数

このサイト内のノードの数。

サイトのノード ID

このサイトに参加しているノード ID のリスト。

Clinfo C API

Clinfo C アプリケーション・プログラミング・インターフェース (API) は、PowerHA SystemMirror クラスターに関する状況情報を取得するためにアプリケーションで使用できる高水準インターフェースです。ここに含まれるトピックでは、Clinfo C API で使用可能なそれぞれの C 言語のルーチンおよびユーティリティーについて説明します。

注**: cl_registerwithclsmuxpd()** ルーチンの代わりにアプリケーション・モニターを使用してください。「プランニング・ガイド」のクラスターの初期プランニングに関するセクションを参照してください。

このトピックを読む前に、『クラスター情報プログラム』をお読みください。ここには、PowerHA SystemMirror クラスターに関して Clinfo が維持する情報のタイプが説明されています。

関連概念:

1 ページの『クラスター情報プログラム』
ここに含まれるトピックでは、クラスター情報プログラム (Clinfo) の概要を紹介し、PowerHA SystemMirror for AIX クラスターに関して Clinfo が受け取り、維持する状況情報について説明します。

関連情報:

クラスターの初期プランニング

アプリケーションでの Clinfo C API の使用

ここに含まれるトピックでは、アプリケーションでの Clinfo C API の使用方法について説明します。

PowerHA SystemMirror for AIX には、マルチスレッド・アプリケーション用と単一スレッド・アプリケーション用に別々のライブラリーが含まれています。 必ずご使用のアプリケーションに適したライブラリーとリンクしてください。

注**:** Clinfo **cluster.es.client.lib** ライブラリーには、32 ビットと 64 ビットの両方のオブジェクトがある **libcl.a** が含まれています。 Clinfo API を使用する 64 ビット・アプリケーションを作成するには、アプリケーションを 64 ビット環境で再コンパイル/再リンクする必要があります。

ヘッダー・ファイル

Clinfo C API を使用する各ソース・モジュールに必要な **include** ディレクティブを指定する必要があります。

このディレクティブには以下が含まれます。

```
#include <sys/types.h>
#include <netinet/in.h>
#include <cluster/clinfo.h>
```

このリストにある **include** ディレクティブに加え、**cl_registereventnotify** ルーチンを使用する各ソース・モジュールに以下の **include** ディレクティブを指定してください。

```
#include <signal.h>
```

クライアント **API** を使用するアプリケーション用のコンパイラー・プリプロセッサー・ディレクティブ

クライアント API を使用するアプリケーションをコンパイルするときは、コンパイラー・フラグ **—D__HAES__** を使用します。

これは、ヘッダー・ファイルで使用される **#ifdef** プリプロセッサー・ディレクティブのために必要です。この原因は、前に別のバージョンの PowerHA SystemMirror (HAS) がサポートされていたことです。

libcl.a および **libcl_r.a** ライブラリーのリンク

Clinfo C API を使用する単一スレッド・アプリケーションのオブジェクト・ロード・コマンドに必要なディレクティブを追加する必要があります。

このディレクティブには以下が含まれます。

```
-lcl
```

Clinfo C API を使用するマルチスレッド・アプリケーションのオブジェクト・ロード・コマンドに以下のディレクティブを追加する必要があります。

```
-lcl_r
```

libcl.a および **libcl_r.a** ライブラリーには、Clinfo C API をサポートするルーチンが含まれています。

定数

Clinfo C API ルーチンは、**clinfo.h** ファイルで定義された定数を使用します。

この定数には以下が含まれます。

項目	説明
CL_MAXNAMELEN	クラスター、ノード、またはインターフェースの名前を指定する文字ストリングの最大長 (256)。 **CL_MAXNAMELEN** の値は、**sys/param.h** ファイルで定義されている **MAXHOSTNAMELEN** と同じです。
CL_MAX_EN_REQS	許可されるイベント通知要求の最大数 (10)。
CL_ERRMSG_LEN	エラー・メッセージの最大長 (128 文字)。
CL_MAXCLUSTERS	クラスターの数に関する情報を保持している配列のデータ構造内の現行スペース・サイズを指定します。
CL_MAXNODES	クラスター内のノードの数に関する情報を保持している配列のデータ構造内の現行スペース・サイズを指定します。
CL_MAXNETIFS	ノードに接続されているインターフェースの数に関する情報を保持している配列のデータ構造内の現行スペース・サイズを指定します。
CL_MAXGROUPS	リソース・グループの数に関する情報を保持している配列のデータ構造内の現行スペース・サイズを指定します。

データ型およびデータ構造

Clinfo C API は、**clinfo.h** ファイルで定義されている、さまざまなデータ型およびデータ構造を使用します。

状態情報を含む列挙型**:**

この列挙データ型は、クラスター、ノード、インターフェース、またはイベント通知の状態を記述します。

```
enum cls_state {
    CLS_INVALID,
    CLS_VALID,
    CLS_UP,
    CLS_DOWN,
    CLS_UNKNOWN,
    CLS_GRACE,
    CLS_JOINING,
    CLS_LEAVING,
    CLS_IN_USE,
    CLS_PRIMARY
};
```

副状態情報を含む列挙型**:**

この列挙データ型は、クラスターの副状態を記述します。

```
enum cls_substate {
    CLSS_UNKNOWN,
    CLSS_STABLE,
    CLSS_UNSTABLE,
    CLSS_ERROR,
    CLSS_RECONFIG
    CLSS_NOT_CONFIGURED
};
```

リソース・グループの状態を含む列挙型**:**

この列挙データ型には、ノード上のリソース・グループについて考えられるすべての状態が含まれます。

```
enum cl_resource_states{
    CL_RGNS_INVALID=1,
    CL_RGNS_ONLINE=2,
    CL_RGNS_OFFLINE=4,
```

```
    CL_RGNS_ACQUIRING=16,
    CL_RGNS_RELEASING=32,
    CL_RGNS_ERROR=64
};
```

リソース・グループのポリシーを含む列挙型:

この列挙データ型には、リソース・グループのすべてのポリシー (ノードとサイトの両方) が含まれます。

```
enum cl_rg_policies {
    CL_RGP_INVALID = 0,
    CL_RGP_ONLINE_ON_HOME_NODE = 1,
    CL_RGP_ONLINE_ONFIRST_AVAILBLE_NODE = 2,
    CL_RGP_ONLINE_USING_DISTRIBUTION_POLICY = 3,
    CL_RGP_ONLINE_ALL_NODES = 4,
    CL_RGP_FALLOVER_TO_PRIORITY_NODE = 5,
    CL_RGP_FALLOVER_USING_DNP = 6,
    CL_RGP_BRING_OFFLINE = 7,
    CL_RGP_FALLBACK_TO_HIGHER_PRIORITY_NODE = 8,
    CL_RGP_NEVER_FALLBACK = 9,
    CL_RGP_PREFER_PRIMARY_SITE = 10,
    CL_RGP_ONLINE_ON_EITHER_SITE = 11,
    CL_RGP_ONLINE_ON_BOTH_SITES = 12,
    CL_RGP_IGNORE_SITES = 13
};
```

インターフェースのロールを含む列挙型:

この列挙型にはインターフェースのロールが含まれます。

```
enum cl_interface_role {
    CL_INT_ROLE_INVALID = 0,
    CL_INT_ROLE_SERVICE = 16,
    CL_INT_ROLE_STANDBY = 32,      /* deprecated */
    CL_INT_ROLE_BOOT = 64,
    CL_INT_ROLE_SH_SERVICE = 128,  /* deprecated */
};
```

ネットワーク属性を含む列挙型:

この列挙データ型は、ネットワークの属性を指定します。

```
/*
* Enumeration of Network attributes. Note that the public/private
attribute is for use by Oracle only - PowerHA SystemMirror does not use this attribute.
*/

enum cl_network_attribute
{
    CL_NET_ATTR_INVALID = 0,   /* IP networks can be public or private */
    CL_NET_TYPE_PUBLIC = 1,
    CL_NET_TYPE_PRIVATE = 2,   /* non-IP (serial) networks are always */
    CL_NET_TYPE_SERIAL = 4
};
```

サイトの優先度を含む列挙型:

この列挙データ型は、サイトの優先度を記述します。 サイトはリソースの処理時に優先度を指定して定義されます。

```
/*
* Enumeration of Site Priorities
*/

enum cl_site_priority
{
```

```
    CL_SITE_PRI_NONE = 0,
    CL_SITE_PRI_PRIMARY = 1,
    CL_SITE_PRI_SECONDARY = 2,
    CL_SITE_PRI_TERTIARY = 4
};
```

サイトのバックアップ通信方式を含む列挙型:

この列挙データ型には、サイトに対して構成できるオプションのバックアップ方式が含まれます。

```
/*
* Enumeration of Site Backup Communication Options
*/

enum cl_site_backup
{
    CL_SITE_BACKUP_NONE = 0,
    CL_SITE_BACKUP_DBFS = 1,
    CL_SITE_BACKUP_SGN = 2
};
```

リソース・グループを表すデータ構造:

この構造には、各リソース・グループに関して使用可能なすべての情報が含まれています。

```
struct cl_group {
    int  clg_clusterid;
    int  clg_group_id;
    char clg_name[CL_MAXNAMELEN];
    enum cl_rg_policies clg_policy;   /* deprecated */
    enum cl_rg_policies clg_startup_policy;
    enum cl_rg_policies clg_fallover_policy;
    enum cl_rg_policies clg_fallback_policy;
    enum cl_rg_policies clg_site_policy;
    char clg_user_policy_name[CL_MAXNAMELEN];
    int  clg_num_nodes;
    int  clg_node_ids[MAXNODES];
        /* list of nodes (ids) in this group */
    enum cl_resource_states clg_node_states[MAXNODES];
        /* state on each */
    int  clg_num_resources;
    int  clg_resource_id[MAXRESOURCES];
        /* list of resources (id) group */
    enum cl_resource_states clg_res_state[MAXRESOURCES]; /* state
*/ int clg_vrmf;                        /* version of this client */
};
```

ネットワーク・インターフェースを表すデータ構造:

このデータ構造はネットワーク・インターフェースを表します。

```
struct cl_netif {
    int cli_clusterid;                  /* Cluster Id */
    int cli_nodeid;                     /* Cluster node Id - used internally only */
    char cli_nodename[CL_MAXNAMELEN];   /* Cluster node name */
    int cli_interfaceid;                /* Cluster Node Interface Id */
    enum cls_state cli_state;           /* Cluster Node Interface State */
    char cli_name[CL_MAXNAMELEN];       /* Cluster Node Interface Name */
    int cli_active_nodeid;              /* Cluster node Id where addr is up */
    enum cl_interface_role cli_role;    /* Role of interface (boot/service)*/
    int cli_networkid;                  /* Cluster Network ID for this Interface */
    int cli_vrmf;
    struct sockaddr_storage cli_addr_v6; /*Cluster Node Interface IP Address */
};
#define cli_addr (*((struct sockaddr_in*)&cli_addr_v6)) /* For backward compatibility*/
```

ノードを表すデータ構造:

このデータ構造はクラスター・ノードを表します。

```
struct cl_node {
    int cln_clusterid;                  /* Cluster Id */
    int cln_nodeid;                     /* Cluster node Id */
    char cln_nodename[CL_MAXNAMELEN];   /* Cluster node name */
    enum cls_state cln_state;           /* node state */
    int cln_nif;                        /* number of interfaces */
    struct cl_netif *cln_if;            /* interfaces */
    int cln_glidle;                     /* CPU.glidle */
    int cln_real_mem_free;              /* Paging space utilitization */
    int cln_disk_busy;                  /* disk busy */
    int cln_vrmf;                       /* version of this client */
};
```

クラスターを表すデータ構造:

このデータ構造はクラスターを表します。

```
struct cl_cluster{
    int  clc_clusterid;               /* cluster id*/
    enum cls_state clc_state;         /* Cluster State */
    enum cls_substate clc_substate;   /* Cluster Substate */
    char clc_primary[CL_MAXNAMELEN];  /* Cluster Primary Node */
    char clc_name[CL_MAXNAMELEN];     /* Cluster Name */
    int  clc_number_of_nodes;         /* number of cluster nodes */
    int  clc_number_of_groups;        /* number of resource groups */
    int  clc_number_of_networks;      /* number of networks */
    int  clc_number_of_sites;         /* number of sites */
    int clc_vrmf;                     /* version of this client */
};
```

イベント通知登録要求を表すデータ構造:

このデータ構造はイベント通知登録要求を表します。

```
struct cli_enr_req_t {
    int event_id;                   /* event id */
    int cluster_id;                 /* cluster id */
    int node_id;                    /* node id(internal use only)*/
    char node_name[CL_MAXNAMELEN];  /* node name */
    int net_id;                     /* network id */
    int signal_id;                  /* signal id */
    int vrmf;
};
```

イベント通知メッセージを表すデータ構造:

このデータ構造はイベント通知メッセージを表します。

```
struct cli_en_msg_t {
    int event_id;                 /* event id */
    int cluster_id;               /* cluster id */
    int node_id;                  /* node id(internal use only)*/
    char node_name[CL_MAXNAMELEN]; /* node name */
    int net_id;                   /* network id */
    int vrmf;<
};
```

クラスター・ネットワークを表すデータ構造:

このデータ構造には、各クラスター・ネットワークの情報が含まれています。

```
/*
* Structure containing information relating to a network.
*/
struct cl_net {
    int clnet_clusterid;                  /* Cluster Id */
    char clnet_name[CL_MAXNAMELEN];       /* Cluster network name */
    int clnet_id;                         /* Cluster Network Id */
    char clnet_type[CL_MAXNAMELEN];       /* ether, token, etc */
    enum cl_network_attribute clnet_attr; /* public/serial */
    enum cls_state clnet_state;           /* Cluster Network State */
      /* Note that this is the cluster wide or "global" network state */
      /* which may be different than the state of the network on any */
      /* particular node */
    int clnet_numnodes;
      /* Number of nodes connected to this Network */
    int clnet_node_ids[MAXNODES];
      /* Node ids connected to this Network */
    enum cls_state clnet_node_states[MAXNODES];
      /* Network State per Node */
    int clnet_vrmf;                        /* version of this client */
    enum cl_net_family clnet_type;        /* v4/v6 */
};
enum cl_net_family {
  CL_INET_INVALID = 0,
  CL_INET4 = 1,
  CL_INET6 = 2,
};
```

クラスター・サイトを表すデータ構造:

このデータ構造には、PowerHA SystemMirror クラスター内で構成されている各サイトの情報が含まれています。サイトの構成はオプションである点を覚えておいてください。

```
/*
* Structure containing information relating to a site
*/
struct cl_site {
    int clsite_clusterid;             /* Cluster ID */
    int clsite_id;
    char clsite_name[CL_MAXNAMELEN];
    enum cl_site_priority clsite_priority;
    enum cl_site_backup clsite_backup;
    enum cls_state clsite_state;      /* Cluster Site State */
    int clsite_numnodes;
    int clsite_nodeids[MAXNODES];     /* List of nodes (IDs) in this group */
    int clsite_vrmf;                  /* version of this client */
};
```

前のリリースの Clinfo からのアプリケーションのアップグレード

前のリリースの PowerHA SystemMirror ではクラスター ID を手動で構成していましたが、現在は自動的に作成されるようになりました。

クラスター名またはその他のパラメーターが指定された場合にクラスター ID を返す呼び出しは、いくつか存在します。

- **cl_getclusterid** — クラスター名が指定されると、クラスター ID を返します。
- **cl_getclusteridbyifaddr** — ネットワーク・インターフェース・アドレスが指定されると、クラスター ID を返します。
- **cl_getclusteridbyifname** — ネットワーク・インターフェース名が指定されると、クラスター ID を返します。

アプリケーションがクラスター ID を必要とする API 呼び出しを使用する場合、クラスター ID を返すこれらのルーチンのいずれかに呼び出しを追加する必要があります。 前のリリースの Clinfo では、文字ストリング (ノード名) を使用する代わりに、整数を使用してクラスター・ノード (ノード ID) を示していました。 以下のセクションでは、ご使用のアプリケーション内の呼び出しを、以前には *nodeid* パラメーターを使用していた各種 Clinfo C API ルーチンに変換する方法の例をいくつか紹介します。各ルーチンごとに、ノード ID を使用する場合の使用例を示し、その後に、ノード名を使用する例を示します。

関連資料:

34 ページの『cl_getclusterid ルーチン』
cl_getclusterid ルーチンは、指定された名前のクラスターのクラスター ID を返します。

35 ページの『cl_getclusteridbyifaddr ルーチン』
指定されたネットワーク・インターフェース・アドレスのクラスターのクラスター ID を返します。 このルーチンは IPv4 アドレスのみ処理可能です。

36 ページの『cl_getclusteridbyifname ルーチン』
指定されたネットワーク・インターフェース名のクラスターのクラスター ID を返します。

cl_getlocalid

以下は、**cl_getlocalid** ルーチンの、前のリリースの例と新規バージョンの例です。

以下は、ノード ID を使用する、前のリリースの **cl_getlocalid** ルーチンの例です。

```
int clusterid, nodeid, status;
status = cl_getlocalid (&clusterid, &nodeid);
if (status != CLE_OK) {
    if (status == CLE_IVNODE) {
 printf ("This node is not a cluster member");
    } else {
 cl_perror (status, "Can't get local cluster ID");
 }
} else {
    printf ("member of cluster %d node %d",clusterid, nodeid);
}
```

C API **cl_getlocalid** ルーチンの新規バージョンの例では、宣言ステートメントが変わっていることに注目してください。 以前はすべての変数が整数として宣言されていましたが、現在は、*nodename* を文字ストリングとして宣言し、それに伴う変更を **printf** ステートメントに行う必要があります。

```
int clusterid, status;
char nodename[CL_MAXNAMELEN];
status = cl_getlocalid (&clusterid, nodename);
if (status != CLE_OK) {
    if (status == CLE_IVNODE) {
 printf ("This node is not a cluster member");
    } else {
 cl_perror (status, "Can't get local cluster ID");
    }
} else {
    printf ("member of cluster %d node %s",clusterid, nodename);
}
```

cl_getnodeidbyifname

以下は、**cl_getnodeidbyifname** ルーチンの、前のリリースの例と新規バージョンの例です。

以下は、ノード ID を使用する、前のリリースの **cl_getnodeidbyifname** ルーチンの例です。

```
int clusterid, nodeid;
char *interfacename;
strcpy (interfacename,"editserver");
```

```
clusterid = 1;
nodeid = cl_getnodeidbyifname (clusterid, interfacename);
if (nodeid < 0) {
    cl_perror(nodeid,"Can't get node ID");
} else {
    printf("ID of %s on cluster %d is %d", interfacename, clusterid, nodeid);
}
```

C API **cl_getnodeidbyifname** ルーチンの新規バージョンの例では、ルーチンの名前そのものが **cl_getnodenamebyifname** に変更されていることに注目してください。この宣言を、整数の *nodeid* ではなくストリングの *nodename* を組み込むように変更してから、それに伴う変更を **printf** ステートメントに行う必要があります。

```
int clusterid, status;
char nodename[MAXNAMELEN];
char *interfacename[MAXNAMELEN];
strcpy (interfacename,"editserver");
clusterid = 1;
status = cl_getnodenamebyifname (clusterid, interfacename, nodename);
if (status != CLE_OK)
    cl_perror(nodename,"Can't get node name");
} else {
    printf("name of %s on cluster %d is %s", interfacename, clusterid, nodename);
}
```

cl_getprimary

以下は、**cl_getprimary** ルーチンの、前のリリースの例と新規バージョンの例です。

以下は、ノード ID を使用する、前のリリースの **cl_getprimary** ルーチンの例です。

```
int clusterid, primary; /* clusterid is arbitrary.*/
clusterid = 1;
primary = cl_getprimary (clusterid);
if (primary < 0) {
    cl_perror (primary, "Can't get cluster primary");
} else {
    printf ("Primary node for cluster %d is %d", clusterid, primary);

}
```

C API **cl_getprimary** ルーチンの新規バージョンの例では、宣言の中で、整数の *nodeid* がストリングの *nodename* に変わっていること、およびそれに伴う変更が **printf** ステートメントに行われていることに注目してください。さらに、前のバージョンは必要な情報 (*nodeid*) が整数であったのに対し、現在は必要な情報がストリング (*nodename*) になっているため、if 文の変更も必要になります。

```
int clusterid, status;
char nodename[CL_MAXNAMELEN];
/* clusterid is arbitrary. */
clusterid = 1;
status = cl_getprimary (clusterid, nodename);
if (status != CLE_OK) {
    cl_perror (status, "Can't get cluster primary");
} else {
    printf ("Primary node for cluster %d is %s", clusterid, nodename);
}
```

cl_isaddravail

以下は、**cl_isaddravail** ルーチンの、前のリリースの例と新規バージョンの例です。

以下は、ノード ID を使用する、前のリリースの **cl_isaddravail** ルーチンの例です。

```
int clusterid, nodeid, status;
struct sockaddr_in addr;      /* clusterid, nodeid, and addr are arbitrary.*/
clusterid = nodeid = 1;
addr.sin_family = AF_INET;
addr.sin_addr.s_addr = inet_addr ("1.1.1.1");
status = cl_isaddravail (clusterid, nodeid, &addr);
if (status != CLE_OK) {
    cl_perror (status, "Interface not available");
} else {
    printf ("Interface address for %s is available", inet_ntoa
    (addr.sin_addr.s_addr));
}
```

C API **cl_isaddravail** ルーチンの新規バージョンの例では、宣言ステートメントが *nodeid* ではなく *nodename* を使用するように変更されています。 それに伴う変更が **printf** ステートメントに行われていることに注目してください。さらに、割り当てステートメントが *nodename* に対して strcpy を使用するように変更されています。

```
int clusterid, status;
char nodename[CL_MAXNAMELEN];
struct sockaddr_in addr;    /* clusterid, nodename, and addr are arbitrary. */
clusterid = 1;
strcpy (nodename, "node1");
addr.sin_family = AF_INET;
addr.sin_addr.s_addr = inet_addr ("1.1.1.1");
status = cl_isaddravail (clusterid, nodename, &addr);
if (status != CLE_OK) {
    cl_perror (status, "Interface not available");
} else {
    printf ("Interface address for %s is available", inet_ntoa
    (addr.sin_addr.s_addr));
}
```

メモリー割り当てルーチン

これらのルーチンは、クラスターまたはノードのマップ情報を保管するためのメモリーの割り当てまたは解放を行います。

該当するメモリー割り当てルーチンは、対応する取得ルーチンより前に呼び出す必要があり、その後、処理完了後に解放ルーチンを呼び出してストレージを解放します。

項目	説明
cl_alloc_clustermap	それぞれに 32 のインターフェースを持つ 32 のノードの 4 つのクラスターのリストに、ストレージを割り当てます。
cl_alloc_groupmap	64 のリソース・グループのリストにストレージを割り当てます。
cl_alloc_netmap	ネットワークのリストにストレージを割り当てます。
cl_alloc_nodemap	それぞれに 32 のインターフェースを持つ 32 のノードのリストにストレージを割り当てます。
cl_alloc_sitemap	サイトのリストにストレージを割り当てます。
cl_free_clustermap	**cl_alloc_clustermap** を呼び出すことによって割り当てられた、クラスターのリスト用のストレージを解放します。
cl_free_groupmap	**cl_alloc_groupmap** によって割り当てられた、リソース・グループのリスト用のストレージを解放します。
cl_free_netmap	**cl_alloc_netmap** を呼び出すことによって割り当てられた、ネットワークのリスト用のストレージを解放します。
cl_free_nodemap	**cl_alloc_nodemap** を呼び出すことによって割り当てられた、ノードのリスト用のストレージを解放します。

項目	説明
cl_free_sitemap	**cl_alloc_sitemap** を呼び出すことによって割り当てられた、サイトのリスト用のストレージを解放します。

以下の例は、これらのルーチンの使用方法を示しています。クラスター内のノードに関して返される情報を入れるストレージを割り当てるために、CL_MAXNODES は使用されなくなったことに注意してください。 追加の例については、**cl_getclusters** および **cl_getnodemap** ルーチンに関する参照ページを参照してください。

```
int status;
struct cl_cluster *clustermap;

cl_alloc_clustermap (&clustermap);
status = cl_getclusters(clustermap);
if (status < 0) {
    cl_perror(status, "Can't get cluster information");
} else {
    printf("There are currently %d running clusters", status);
}
...
cl_free_clustermap (clustermap);
```

新たに API **cl_node_free()** が追加されています。これは単一の **cl_node** 構造体に関連付けられているストレージを解放します。以下の例を検討してください。

```
struct cl_node nodebuf;
    cl_getnode(clusterid, "ppstest5", &nodebuf);
    printf("Node %s is id %d¥n", nodebuf.cln_nodename, nodebuf.cln_nodeid);
    cl_node_free(&nodebuf);
);
```

cl_getnode() を呼び出した後、cln_if フィールドには、ノードに関連付けられているネットワーク・インターフェース構造体のリストが記入されます。 このリストは **cl_getnode()** によって動的に割り当てられます。長時間実行プログラムでのメモリー・リークを防ぐために、このリストは解放する必要があります。

cl_node_free() は、**cl_node** 構造体のネットワーク・インターフェース・ストレージ・フィールドを解放します。 詳しくは、後出の **cl_node_free()** の API の説明を参照してください。

要求

Clinfo C API には、いくつかのタイプの要求があります。

クラスター情報要求

クラスター情報要求では、クラスターに関する情報が返されます。

項目	説明
cl_getcluster	指定されたクラスター ID のクラスターに関するすべての既知情報を返します。
cl_getclusterid	指定されたクラスター名のクラスターのクラスター ID を返します。
cl_getclusteridbyifaddr	指定されたネットワーク・インターフェース・アドレスのクラスターのクラスター ID を返します。 このルーチンは IPv4 アドレスのみ処理可能です。
cl_getclusteridbyifaddr6	指定されたネットワーク・インターフェース・アドレスのクラスターのクラスター ID を返します。 このルーチンは IPv4 アドレスおよび IPv6 アドレスの両方を処理可能です。
cl_getclusteridbyifname	指定されたネットワーク・インターフェース名のクラスターのクラスター ID を返します。
cl_getclusters	すべての作動可能クラスターに関する情報を返します。
cl_isclusteravail	指定されたクラスター ID のクラスターの状況を返します。

ノード情報要求

ノード情報要求では、クラスター内のノードに関する情報が返されます。

項目	説明
cl_bestroute	ローカル・マシンから指定されたノードへの最も直接的な経路を示すローカルまたはリモートの IP アドレスのペアを返します。
cl_bestroute6	ローカル・マシンから指定されたノードへの最も直接的な経路を示すローカルまたはリモートの IP アドレスのペアを返します。 このルーチンは IPv4 アドレスおよび IPv6 アドレスの両方を処理可能です。
cl_getlocalid	要求を発行しているホストのクラスター ID とノード名を返します。
cl_getnode	クラスター ID またはノード名のペアによって指定されたノードに関する情報を返します。
cl_getnodeaddr	指定されたクラスター ID またはノード名のペアに関連付けられている IPv4 アドレスを返します。
cl_getnodeaddr6	指定されたクラスター ID/ノード名のペアに関連付けられている IPv4 アドレスまたは IPv6 アドレスを返します。
cl_getnodenamebyifaddr	指定されたクラスター ID またはインターフェース・アドレスのペアを持つノードの名前を返します。 このルーチンは IPv4 アドレスのみを返します。
cl_getnodenamebyifaddr6	指定されたクラスター ID またはインターフェース・アドレスのペアを持つノードの名前を返します。 このルーチンは IPv4 アドレスおよび IPv6 アドレスの両方を処理可能です。
cl_getnodenamebyifname	指定されたクラスター ID/インターフェース名のペアを持つノードの名前を返します。
cl_getnodemap	指定されたクラスター内のすべてのノードに関する既知情報をすべて返します。
cl_getprimary	指定されたクラスターの 1 次クラスター・マネージャーのノード名を返します。
cl_isnodeavail	指定されたノードの状況を返します。

ネットワーク・インターフェース情報要求

ネットワーク・インターフェース情報要求では、ノードに接続されているインターフェースに関する情報が返されます。

項目	説明
cl_getifaddr	指定されたクラスター ID および名前を持つインターフェースの IPv4 インターフェース・アドレスを返します。
cl_getifaddr6	指定されたクラスター ID および名前を持つインターフェースの IPv4 または IPv6 インターフェース・アドレスを返します。
cl_getifname	指定されたクラスター ID およびアドレスを持つ IPv4 インターフェースのインターフェース名を返します。
cl_getifname6	指定されたクラスター ID およびアドレスを持つ IPv4 または IPv6 インターフェースのインターフェース名を返します。
cl_isaddravail	指定したネットワーク・インターフェースの状況を戻します。 このルーチンは IPv4 アドレスのみを返します。
cl_isaddravail6	指定したネットワーク・インターフェースの状況を戻します。 このルーチンは IPv4 アドレスおよび IPv6 アドレスの両方を処理可能です。

ネットワーク情報要求

ネットワーク情報要求では、クラスターに含まれるネットワークに関する情報が返されます。

項目	説明
cl_getnetmap	クラスター内のすべてのネットワークに関する既知情報をすべて返します。
cl_getnet	特定のネットワークに関する情報を返します。
cl_getnetbyname	特定の名前付きネットワークに関する情報を返します。
cl_getnetsbytype	指定されたタイプを使用して構成されたネットワークに関する情報を返します。
cl_getnetsbyattr	指定された属性を使用して構成されたネットワークに関する情報を返します。
cl_getnetstatebynode	指定されたノード上の指定されたネットワークの状態を返します。

イベント通知要求

イベント通知ルーチンでは、クラスター、ノード、またはネットワークのイベントに関する情報が返されます。

項目	説明
cl_getevent	イベント・シグナルの受信時に情報を取得し、Clinfo から受け取ったイベント通知メッセージを返します。
cl_registereventnotify	イベント通知要求のリストを Clinfo に登録し、登録済みイベントの発生時に呼び出しプロセスにシグナルを返します。
cl_unregistereventnotify	イベント通知要求のリストの Clinfo への登録を抹消します。

リソース・グループ情報要求

リソース・グループ情報要求では、クラスター・リソース・グループに関する情報が返されます。

項目	説明
cl_getgroupmap	指定されたクラスター内のすべてのリソース・グループに関する既知情報をすべて返します。
cl_getgroup	指定されたクラスター内の特定のリソース・グループに関するすべての情報を返します。
cl_getgroupsbynode	指定されたノードがメンバーになっているリソース・グループをすべて返します。
cl_getgroupnodestate	指定されたノード上の指定されたグループの状態を返します。

サイト情報要求

サイト情報要求では、PowerHA SystemMirror クラスター内で構成されているサイトに関する情報が返されます。

項目	説明
cl_getsitemap	クラスター内のすべてのサイトに関する既知情報をすべて返します。
cl_getsite	特定のサイトに関する情報を返します。
cl_getsitebyname	特定の名前付きサイトに関する情報を返します。
cl_getsitebypriority	指定された優先度を使用して構成されたサイトに関する情報を返します。

ユーティリティー

Clinfo C API にはいくつかのユーティリティー・ルーチンがあります。

cl_initialize ルーチン

cl_initialize() ルーチンは、前のリリースで共有メモリーに接続するために使用されていました。

cl_initialize() ルーチンには機能は何もなくなっており、常に **CLE_OK** を返します。 **cl_initialize()** は後方互換性の確保のみを目的に提供されています。 このルーチンを新しいプログラムで使用しないでください。

cl_errmsg ルーチン

cl_errmsg ルーチンは Clinfo によって返される状況コードを取得して、そのエラー・コードに対応するテキストを返します。

構文

```
char *cl_errmsg (int status)
```

パラメーター

項目	説明
status	クラスター情報エラーの状況。

状況コード

ヌル終了エラー・ストリング。

例えば、次のストリングがあります。

```
"Invalid status"
```

このストリングは、status パラメーターが有効なクラスター・エラー・コードを示していない場合のものです。

例

```
char *msg;
msg = cl_errmsg(CLE_BADARGS);
if (strcmp(msg, "Invalid status") != 0) {
    printf("CLE_BADARGS means %s", msg);
} else {
    printf("Can't lookup CLE_BADARGS");
}
```

cl_errmsg_r ルーチン

cl_errmsg_r ルーチンは Clinfo によって返される状況コードを取得して、そのエラー・コードに対応するテキストを返します。

これは、**cl_errmsg** ルーチンのスレッド・セーフ・バージョンです。マルチスレッド・アプリケーションを使用する場合は、このルーチンを使用する必要があります。

構文

```
char *cl_errmsg_r (int status, char cbuf)
```

パラメーター

項目	説明
status	クラスター情報エラーの状況。
cbuf	返されるメッセージ用のストレージは、長さが少なくとも **CL_ERRMSG_LEN** でなければなりません (128 文字に十分な長さ)。

状況コード

ヌル終了エラー・ストリング。

例えば、次のストリングがあります。

```
"Invalid status"
```

このストリングは、status パラメーターが有効なクラスター・エラー・コードを示していない場合のものです。

例

```
char *msg;
char cbuf[CL_ERRMSG_LEN];
msg = cl_errmsg_r(CLE_BADARGS, cbuf);
if (strcmp(msg, "Invalid status") != 0) {
printf("CLE_BADARGS means %s", msg);
} else {
printf("Can't lookup CLE_BADARGS");
}
```

cl_perror ルーチン

cl_perror ルーチンは、指定されたエラー・コードを記述するメッセージを標準エラーに書き込みます。**cl_perror** は、提供されたエラー・ストリングをエラー・メッセージの前に配置し、エラー・メッセージの後にコロンを付けます。

構文

```
void cl_perror (int status, char *message)
```

例えば、以下を指定すると、

```
cl_perror(CLE_IVNODENAME, "Can't service this request");
```

以下の出力が生成されます。

```
Can't service this request:Illeagle Node Name.
```

cl_perror ルーチンは、Clinfo 要求によって返された状況コードからエラー・メッセージを生成する場合に役立ちます。有効なクラスター・エラー・コードでない状況が指定されると、**cl_perror** ルーチンは以下のストリングを書き込みます。

```
Error n
```

ここで、n は指定された状況コードの値です。

パラメーター

項目	説明
status	クラスター・エラー・コード。
message	状況説明を行うメッセージ。

例

```
struct cl_node nodebuf;
  int clusterid = 99999999;      /* invalid cluster id */
  int status;
  char nodename[CL_MAXNAMELEN];

  if ( (status = cl_getnode (clusterid, nodename, &nodebuf)) < 0 ) {
      cl_perror(status, "can't get node information");
  }
```

cl_alloc_clustermap ルーチン

cl_alloc_clustermap ルーチンはクラスターのリストにストレージを割り当てます。 このルーチンは、**cl_getclusters** ルーチンより先に呼び出す必要があります。

cl_getclusters ルーチンを呼び出した後に、**cl_free_clustermap** ルーチンを呼び出してストレージを解放します。

構文

```
int cl_alloc_clustermap (struct cl_cluster **clustermap)
```

パラメーター

項目	説明
clustermap	クラスター・マップの基本ポインター。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、clustermap 引数に NULL ポインターが指定されたことを示します。

例

cl_getclusters ルーチンの例を参照してください。

関連資料:

33 ページの『cl_getcluster ルーチン』

cl_getcluster は、クラスター ID によって指定されたクラスターに関する情報を返します。

cl_alloc_groupmap ルーチン

cl_alloc_groupmap ルーチンはリソース・グループ・ディスクリプターのリストにストレージを割り当てます。 このルーチンは、**cl_getgroups** ルーチンより先に呼び出す必要があります。

cl_getgroups ルーチンを呼び出した後、処理が完了したら **cl_free_groupmap** ルーチンを呼び出してストレージを解放します。

構文

```
int cl_alloc_groupmap (struct cl_group **groupmap);
```

パラメーター

項目	説明
groupmap	グループ・マップの基本ポインター。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、groupmap 引数に NULL ポインターが指定されたことを示します。

例

cl_getgroupmap ルーチンの例を参照してください。

関連資料:

39 ページの『cl_getgroupmap ルーチン』

cl_getgroupmap ルーチンは、クラスター内のリソース・グループに関する情報を返します。 このルーチンを呼び出してメモリー内のストレージを予約する前に、**cl_alloc_groupmap** を呼び出す必要があります。 このルーチンを呼び出した後は、**cl_free_groupmap** を呼び出す必要があります。

cl_alloc_netmap ルーチン

一連のネットワーク情報構造にストレージを割り当てます。

構文

```
int cl_alloc_netmap (struct cl_site **netmap)
```

パラメーター

項目	説明
netmap	新しく割り当てられたストレージへのポインターが返される、構造 **cl_net** のポインターへのポインター。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、**netmap** パラメーターに NULL ポインターが指定されたことを示します。

例

cl_getnetmap ルーチンの例を参照してください。

関連資料:

48 ページの『cl_getnetmap ルーチン』
クラスター内のすべてのネットワークに関する情報を返します。

cl_alloc_netmap6 ルーチン

一連のサイト情報構造にストレージを割り当てます。

構文

```
int cl_alloc_netmap6(struct cl_net_v6** netmap)
```

パラメーター

項目	説明
netmap	新しく割り当てられたストレージへのポインターが返される、構造 **cl_net_v6** のポインターへのポインター。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_BADARGS	欠落している、または無効なパラメーター。

cl_alloc_nodemap ルーチン

cl_alloc_nodemap ルーチンは、ノードのリストおよび各ノードに関連付けられているインターフェースにストレージを割り当てます。 このルーチンは、**cl_getnodemap** ルーチンより先に呼び出す必要があります。

cl_getnodemap ルーチンを呼び出した後、処理が完了したら **cl_free_nodemap** ルーチンを呼び出してストレージを解放します。

構文

```
int cl_alloc_nodemap (struct cl_node **nodemap)
```

パラメーター

項目	説明
nodemap	ノード・マップの基本ポインター。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_BADARGS	欠落している、または無効なパラメーター。

例

cl_getnodemap ルーチンの例を参照してください。

関連資料:

54 ページの『cl_getnodemap ルーチン』
cl_getnodemap ルーチンは、クラスター内のノードに関する情報を返します。 このルーチンを呼び出してメモリー内のストレージを予約する前に、**cl_alloc_nodemap** を呼び出す必要があります。 このルーチンを呼び出した後は、**cl_free_nodemap** を呼び出す必要があります。

cl_alloc_nodemap6 ルーチン

一連のサイト情報構造にストレージを割り当てます。

構文

```
int cl_alloc_nodemap6(struct cl_node_v6** nodemap)
```

パラメーター

項目	説明
nodemap	新しく割り当てられたストレージへのポインターが返される、構造 **cl_net_v6** のポインターへのポインター。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_BADARGS	欠落している、または無効なパラメーター。

cl_alloc_sitemap ルーチン

一連のサイト情報構造にストレージを割り当てます。

構文

```
int cl_alloc_sitemap (struct cl_net **sitemap)
```

パラメーター

項目	説明
sitemap	新しく割り当てられたストレージへのポインターが返される、構造 **cl_site** のポインターへのポインター。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、**sitemap** パラメーターに NULL ポインターが指定されたことを示します。

例

cl_getsitemap ルーチンの例を参照してください。

関連資料:

60 ページの『cl_getsitemap ルーチン』
クラスター内のすべてのサイトに関する既知情報をすべて返します。

cl_bestroute ルーチン

cl_bestroute ルーチンは、指定されたノードへの最も直接的な経路を示すローカル/リモート IP アドレスのペアを返します。このルーチンは、ネットマスク値を利用してネットワーク・インターフェースのペアを突き合わせます。 このルーチンは IPv4 アドレスのみ処理可能です。

cl_bestroute ルーチンによって返される経路は、その要求を発行しているホストによって異なります。Clinfo は最初にローカル・ノード上の正常なすべてのネットワーク・インターフェースのリストを作成し、その後に、このリストを指定されたノード上の使用可能インターフェースと比較します。 このルーチンはネットワークによってインターフェースをソートし、そのネットワーク上の最初の正常なインターフェースと、そのネットワーク上のリモート・ノード上にある最初の正常なインターフェースとの突き合わせを試みます。 プライベート・ネットワークが定義されている場合、プライベート・ネットワークは非プライベート・ネットワークより先に検討されます。

同じネットワーク上にローカルとリモートのインターフェースのペアが存在する場合、これらは **laddr** および**raddr** パラメーターの中に含めて返されます。そうでない場合、指定されたノード上のインターフェースがリモート・インターフェースとして選択され、最初に見つかったローカル・インターフェースが経路のローカル・エンドとして返されます。

構文

```
int cl_bestroute (int clusterid, char *nodename,
struct sockaddr_in *laddr, struct sockaddr_in *raddr)
```

パラメーター

項目	説明
clusterid	希望するノードのクラスター ID。
nodename	希望するノードの名前。
laddr	返されるローカル・ネットワーク・インターフェース・アドレス用のストレージ。
raddr	返されるリモート・ネットワーク・インターフェース・アドレス用のストレージ。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_BADARGS	欠落している、または無効なパラメーター。
CLE_NOROUTE	使用できる経路がありません。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNODENAME	要求は無効なノード名を指定しました。

例

```
int clusterid = 1113325332;
  int status;
  char nodename[CL_MAXNAMELEN] = "node1";
  struct sockaddr_in laddr, raddr;

  status = cl_bestroute(clusterid, nodename, &laddr, &raddr);
  if (status != CLE_OK) {
      cl_perror(status, "can't get route");
  } else {
```

```
    printf("best route to node %s is from ", nodename);
    printf("%s to ", inet_ntoa(laddr.sin_addr));
    printf("%s¥n", inet_ntoa(raddr.sin_addr));
}
```

cl_bestroute6 ルーチン

cl_bestroute6 ルーチンは、指定されたノードへの最も直接的な経路を示すローカル/リモート IP アドレスのペアを返します。 このルーチンは、ネットマスク値を利用してネットワーク・インターフェースのペアを突き合わせます。

cl_bestroute6 ルーチンによって返される経路は、その要求を発行しているホストによって異なります。Clinfo は最初にローカル・ノード上の正常なすべてのネットワーク・インターフェースのリストを作成し、その後に、このリストを指定されたノード上の使用可能インターフェースと比較します。 このルーチンはネットワークによってインターフェースをソートし、そのネットワーク上の最初の正常なインターフェースと、そのネットワーク上のリモート・ノード上にある最初の正常なインターフェースとの突き合わせを試みます。 プライベート・ネットワークが定義されている場合、プライベート・ネットワークは非プライベート・ネットワークより先に検討されます。

同じネットワーク上にローカルとリモートのインターフェースのペアが存在する場合、これらは **laddr** および **raddr** パラメーターの中に含めて返されます。そうでない場合、指定されたノード上のインターフェースがリモート・インターフェースとして選択され、最初に見つかったローカル・インターフェースが経路のローカル・エンドとして返されます。

構文

```
int cl_bestroute6 (int clusterid, char *nodename,
struct sockaddr  *laddr, size_t size_of_laddr,
struct sockaddr *raddr, size_t size_of_raddr)
```

パラメーター

項目	説明
clusterid	希望するノードのクラスター ID。
nodename	希望するノードの名前。
laddr	返されるローカル・ネットワーク・インターフェース・アドレス用のストレージ。
size_of_laddr	「addr」バッファーのサイズ。少なくともサイズ「struct sockaddr_in」(IPv4 ソケットの場合) および「struct sockaddr6_in」(IPv6 ソケットの場合) であることが必要です。
raddr	返されるリモート・ネットワーク・インターフェース・アドレス用のストレージ。
size_of_raddr	「addr」バッファーのサイズ。少なくともサイズ「struct sockaddr_in」(IPv4 ソケットの場合) および「struct sockaddr6_in」(IPv6 ソケットの場合) であることが必要です。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_BADARGS	欠落している、または無効なパラメーター。
CLE_NOROUTE	使用できる経路がありません。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNODENAME	要求は無効なノード名を指定しました。

cl_free_clustermap ルーチン

cl_free_clustermap ルーチンは、前に **cl_alloc_clustermap** を呼び出してクラスターのリストに割り当てられたストレージを解放します。

構文

```
void cl_free_clustermap (struct cl_cluster *clustermap)
```

パラメーター

項目	説明
clustermap	解放されるクラスター・マップへのポインター。

例

cl_getclusters ルーチンの例を参照してください。

関連資料:

33 ページの『cl_getcluster ルーチン』

cl_getcluster は、クラスター ID によって指定されたクラスターに関する情報を返します。

cl_free_groupmap ルーチン

cl_free_groupmap ルーチンは、前に **cl_alloc_groupmap** を呼び出して割り当てられたストレージを解放します。

構文

```
void cl_free_groupmap (struct cl_group *groupmap);
```

パラメーター

項目	説明
groupmap	解放されるグループ・マップへのポインター。

例

cl_getgroupmap ルーチンの例を参照してください。

関連資料:

39 ページの『cl_getgroupmap ルーチン』

cl_getgroupmap ルーチンは、クラスター内のリソース・グループに関する情報を返します。 このルーチンを呼び出してメモリー内のストレージを予約する前に、**cl_alloc_groupmap** を呼び出す必要があります。 このルーチンを呼び出した後は、**cl_free_groupmap** を呼び出す必要があります。

cl_free_netmap ルーチン

前に **cl_alloc_netmap** を呼び出して割り当てられたストレージを解放します。

構文

```
void cl_free_netmap (struct cl_net *netmap)
```

パラメーター

項目	説明
netmap	解放されるネットワーク・マップへのポインター。

状況コード

なし。

例

cl_getnetmap ルーチンの例を参照してください。

関連資料:

48 ページの『cl_getnetmap ルーチン』
クラスター内のすべてのネットワークに関する情報を返します。

cl_free_netmap6 ルーチン

前に **cl_alloc_netmap6** を呼び出して割り当てられたストレージを解放します。

構文

```
void cl_free_netmap6 (struct cl_net *netmap)
```

パラメーター

項目	説明
netmap	解放される netmap へのポインター。

状況コード

なし。

cl_free_nodemap ルーチン

cl_free_nodemap ルーチンは、前に **cl_alloc_nodemap** ルーチンを呼び出して割り当てられたストレージを解放します。

構文

```
int cl_free_nodemap (struct cl_node *nodemap)
```

パラメーター

項目	説明
nodemap	解放されるノード・マップへのポインター。

例

cl_getnodemap ルーチンの例を参照してください。

関連資料:

54 ページの『cl_getnodemap ルーチン』
cl_getnodemap ルーチンは、クラスター内のノードに関する情報を返します。 このルーチンを呼び出してメモリー内のストレージを予約する前に、**cl_alloc_nodemap** を呼び出す必要があります。 このルーチン

を呼び出した後は、**cl_free_nodemap** を呼び出す必要があります。

cl_free_sitemap ルーチン

前に **cl_alloc_sitemap** を呼び出して割り当てられたストレージを解放します。

構文

```
void cl_free_sitemap (struct cl_site *sitemap)
```

パラメーター

項目	説明
sitemap	解放されるサイト・マップへのポインター。

状況コード

なし。

例

cl_getsitemap ルーチンの例を参照してください。

関連資料:

60 ページの『cl_getsitemap ルーチン』

クラスター内のすべてのサイトに関する既知情報をすべて返します。

cl_getcluster ルーチン

cl_getcluster は、クラスター ID によって指定されたクラスターに関する情報を返します。

構文

```
int cl_getcluster (int clusterid, struct cl_cluster *clusterbuf)
```

パラメーター

項目	説明
clusterid	希望するクラスターのクラスター ID。
clusterbuf	返されるクラスター情報用のストレージ。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。

例

```
int clusterid = 1113325332;
int status;
struct cl_cluster cluster;
```

```
  status = cl_getcluster(clusterid, &cluster);
  if (status != CLE_OK) {
     cl_perror(status, "Can't get cluster information");
  } else {
printf("cluster id:
printf("cluster name:
printf("state= %d [%s]¥n",
  cluster.clc_state,
  get_state(cluster.clc_state));
printf("substate=
printf("primary= <
printf("nodes= %d, sites= %d, groups= %d, networks= %d¥n",
  cluster.clc_number_of_nodes,
  cluster.clc_number_of_sites,
  cluster.clc_number_of_groups,
  cluster.clc_number_of_networks);
  }
```

関連資料:

25 ページの『cl_alloc_clustermap ルーチン』
cl_alloc_clustermap ルーチンはクラスターのリストにストレージを割り当てます。 このルーチンは、**cl_getclusters** ルーチンより先に呼び出す必要があります。

31 ページの『cl_free_clustermap ルーチン』
cl_free_clustermap ルーチンは、前に **cl_alloc_clustermap** を呼び出してクラスターのリストに割り当てられたストレージを解放します。

cl_getclusterid ルーチン

cl_getclusterid ルーチンは、指定された名前のクラスターのクラスター ID を返します。

構文

```
int cl_getclusterid (char *clustername)
```

パラメーター

項目	説明
clustername	目的のクラスターの名前。

状況コード

クラスター ID を表す数値が負でない場合は、正常な状況を示します。 そうでない場合、以下のいずれかのエラー状況コードになります。

項目	説明
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。
CLE_IVCLUSTERNAME	要求は無効なクラスター名を指定しました。

例

```
int clusterid = 1113325332;
  char clustername[CL_MAXNAMELEN] = "site1";

  clusterid = cl_getclusterid (clustername);
  if (clusterid < 0) {
```

```
    cl_perror (clusterid, "can't get cluster ID");
  } else {
    printf ("cluster %s has id %d¥n", clustername, clusterid);
  }
```

cl_getclusteridbyifaddr ルーチン

指定されたネットワーク・インターフェース・アドレスのクラスターのクラスター ID を返します。 このルーチンは IPv4 アドレスのみ処理可能です。

構文

```
int cl_getclusteridbyifaddr (struct sockaddr_in *addr)
```

パラメーター

項目	説明
addr	必要なクラスター ID を含むネットワーク・インターフェース・アドレス。

状況コード

クラスター ID を表す数値が負でない場合は、正常な状況を示します。 そうでない場合、以下のいずれかのエラー状況コードになります。

項目	説明
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVADDRESS	要求は無効なネットワーク・インターフェース・アドレスを指定しました。

例

```
char ifaddr[CL_MAXNAMELEN] = "9.57.28.23";
  int clusterid;
  struct sockaddr_in addr;

  /*
  * inet_addr converts addrs to
  * Internet numbers.
  */

  addr.sin_family = AF_INET;
  addr.sin_addr.s_addr = inet_addr (ifaddr);
  clusterid = cl_getclusteridbyifaddr (&addr);
  if (clusterid < 0) {
      cl_perror (clusterid,"can't get cluster ID");
  } else {
      printf("cluster id w/ interface address %s is %d¥n",
      inet_ntoa (addr.sin_addr.s_addr), clusterid);
  }
```

cl_getclusteridbyifaddr6 ルーチン

指定されたネットワーク・インターフェース・アドレスのクラスターのクラスター ID を返します。

構文

```
int cl_getclusteridbyifaddr6 (struct sockaddr *addr, size_t size_of_addr)
```

パラメーター

項目	説明
addr	必要なクラスター ID を含むネットワーク・インターフェース・アドレス。
size_of_addr	「addr」バッファーのサイズ。このサイズは、少なくとも「struct sockaddr_in」(IPv4 ソケットの場合) および「struct sockaddr6_in」(IPv6 ソケットの場合) であることが必要です。

状況コード

クラスター ID を表す数値が負でない場合は、正常な状況を示します。 そうでない場合、以下のいずれかのエラー状況コードになります。

項目	説明
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVADDRESS	要求は無効なネットワーク・インターフェース・アドレスを指定しました。

cl_getclusteridbyifname ルーチン

指定されたネットワーク・インターフェース名のクラスターのクラスター ID を返します。

構文

```
int cl_getclusteridbyifname (char *interfacename)
```

パラメーター

項目	説明
interfacename	必要なクラスター ID を含むネットワーク・インターフェース名。

状況コード

クラスター ID を表す数値が負でない場合は、正常な状況を示します。 そうでない場合、以下のいずれかのエラー状況コードになります。

項目	説明
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。
CLE_IVNETIFNAME	要求は無効なネットワーク・インターフェース名を指定しました。

例

```
int clusterid;
char interfacename[CL_MAXNAMELEN] = "geotest9";

clusterid = cl_getclusteridbyifname (interfacename);
if (clusterid < 0) {
    cl_perror (clusterid, "can't get cluster id");
} else {
    printf ("cluster id w/ interface named %s is %d¥n",
    interfacename, clusterid);
}
```

cl_getclusters ルーチン

正常なすべてのクラスターに関する情報を返します。

構文

```
int cl_getclusters (struct cl_cluster *clustermap)
```

パラメーター

項目	説明
clustermap	返されるクラスター情報。 このルーチンを呼び出す前に **cl_alloc_clustermap** を呼び出して、この情報用のストレージを割り当てる必要があります。 その後、処理が完了したら、**cl_free_clustermap** を呼び出してこのストレージ・スペースを解放します。

状況コード

このルーチンはアクティブ・クラスターの数を返します。 このルーチンが正常に実行されない場合、以下のいずれかのエラー状況コードが返されます。

項目	説明
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。

例

この例では、**cl_errmsg** ルーチンを使用して単一スレッド・アプリケーションの正しいプログラミングを示しています。ご使用のプログラムがマルチスレッドの場合は、**cl_errmsg_r** ルーチンを使用する必要があります。

```
  int i;
  int numClusters = -1;
  char cbuf[CL_ERRMSG_LEN];
  struct cl_cluster *clustermap;

  cl_alloc_clustermap (&clustermap);
  numClusters = cl_getclusters(clustermap);
  if(numClusters < 0)
  {
      printf("cl_getclusters: (failed) %s¥n",
      cl_errmsg(numClusters));
/*
** for threadsafe compilation use:
      cl_errmsg_r(numClusters,cbuf));
*/
  }
  else
      printf("there are currently %d running clusters¥n", numClusters);
      for(i=0; i < numClusters; i++)
  {
printf("¥n  cluster id:
printf("  cluster name:
printf("  state=
printf("  substate=
printf("  primary=
printf("  nodes= %d, sites= %d, groups= %d, networks= %d¥n",
  clustermap[i].clc_number_of_nodes,
  clustermap[i].clc_number_of_sites,
  clustermap[i].clc_number_of_groups,
```

```
clustermap[i].clc_number_of_networks);
}
}
cl_free_clustermap(clustermap);
```

cl_getevent ルーチン

cl_getevent ルーチンはイベント通知メッセージを返します。 呼び出し元は、前の **cl_registereventnotify** 要求で指定されたシグナルの受信後にのみ、この要求を発行することが必要です。

構文

```
int cl_getevent (struct cli_en_msg_t * en_msg)
```

パラメーター

項目	説明
en_msg	1 つのイベント通知メッセージを保持するのに十分な大きさのイベント通知メッセージ・バッファー。 これが正常に返されると、このバッファーには、受信したイベント、クラスター、および (該当する場合は) ネットワークおよびノードの ID が含まれます。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。

例

cl_registereventnotify ルーチンの例を参照してください。

関連資料:

65 ページの『cl_registereventnotify ルーチン』
cl_registereventnotify ルーチンは、イベント通知要求のリストを Clinfo に登録します。

cl_getgroup ルーチン

cl_getgroup ルーチンは、指定されたクラスター内の指定されたリソース・グループに関する情報を返します。

構文

```
int cl_getgroup (int clusterid, char *groupname,
   struct cl_group *groupbufp);
```

パラメーター

項目	説明
clusterid	希望するクラスターのクラスター ID。
groupname	リソース・グループの名前。
groupbufp	情報が返される場所である **cl_group** 構造へのポインター。

状況コード

項目	説明
CLE_OK	成功。
CLE_BADARGS	欠落している、または無効な引数。
CLE_SYSERR	システム・エラー。
CLE_NOCLINFO	クラスター情報が利用できません。
CLE_IVCLUSTERID	無効なクラスター ID。
CLE_IVNODENAME	無効なリソース・グループ名。

例

```
  int clusterid = 1113325332;
  int status, j;
  char* groupname = "rg01";
  struct cl_group group;

  status = cl_getgroup(clusterid, groupname, &group);
  if (status != CLE_OK){
      cl_perror(status, "can't get resource group information");
  } else {
      printf("resource group %s has %d nodes.¥n",
      group.clg_name,
      group.clg_num_nodes);
      for(j=0; j < group.clg_num_nodes; j++){
  printf("node w/ id %d is in state %d [%s]¥n",
  group.clg_node_ids[j],
  group.clg_node_states[j],
/* user defined function char* cvrt_rg_state(enum cl_resource_states state)
** to convert state id numbers to text
*/
 cvrt_rg_state(group.clg_node_states[j]));
      }
  }
```

cl_getgroupmap ルーチン

cl_getgroupmap ルーチンは、クラスター内のリソース・グループに関する情報を返します。 このルーチンを呼び出してメモリー内のストレージを予約する前に、**cl_alloc_groupmap** を呼び出す必要があります。 このルーチンを呼び出した後は、**cl_free_groupmap** を呼び出す必要があります。

構文

```
int cl_getgroupmap (int clusterid, struct cl_group *groupmap)
```

パラメーター

項目	説明
clusterid	希望するクラスターのクラスター ID。
groupmap	返されるリソース・グループ情報用のストレージ。

状況コード

負でない数値 (クラスター内のグループの数) が返された場合、要求は正常に完了しました。

項目	説明
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。

例

```
  int i,j;
  int clusterid = 1113325332;
  int groups;
  char cbuf[CL_ERRMSG_LEN];
  struct cl_group *groupmap;

  cl_alloc_groupmap(&groupmap);
  if (groupmap==NULL){
      printf("unable to allocate storage: cl_alloc_groupmap = NULL¥n");
      exit(-1);
  }
  groups = cl_getgroupmap(clusterid, groupmap);
  if(groups < 0) {
      cl_perror(groups, "can't get resource group map");
  } else {
      printf("cluster %d has %d resource groups¥n", clusterid, groups);
      for(i=0; i < groups; i++){
  printf("resource group %d [%s] has %d nodes¥n",
  groupmap[i].clg_group_id,
  groupmap[i].clg_name,
  groupmap[i].clg_num_nodes);
  printf("policies:¥n");
  printf("¥tstartup  %d ¥n",groupmap[i].clg_startup_policy);
  printf("¥tfallover %d ¥n",groupmap[i].clg_fallover_policy);
  printf("¥tfallback %d ¥n",groupmap[i].clg_fallback_policy);
  printf("¥tsite     %d ¥n",groupmap[i].clg_site_policy);
  printf("¥tuser     %d ¥n",groupmap[i].clg_user_policy_name);
  printf("node states:¥n");
  for(j=0; j < groupmap[i].clg_num_nodes; j++){
 printf("¥tnode %d is in state %d ¥n",
  groupmap[i].clg_node_ids[j],
  groupmap[i].clg_node_states[j]);
    }
  printf("resources
  for(j=0; j < groupmap[i].clg_num_resources; j++){
 printf("¥tresource %d is in state %d¥n",
  groupmap[i].clg_resource_id[j],
  groupmap[i].clg_res_state[j]);
  }
      }
  }
```

関連資料:

25 ページの『cl_alloc_groupmap ルーチン』

cl_alloc_groupmap ルーチンはリソース・グループ・ディスクリプターのリストにストレージを割り当てます。 このルーチンは、**cl_getgroups** ルーチンより先に呼び出す必要があります。

31 ページの『cl_free_groupmap ルーチン』
cl_free_groupmap ルーチンは、前に **cl_alloc_groupmap** を呼び出して割り当てられたストレージを解放します。

cl_getgroupnodestate ルーチン

cl_getgroupnodestate ルーチンは、指定されたノード上の指定されたグループの状態を返します。

構文

```
enum cl_resource_states cl_getgroupnodestate (int clusterid,
  char *groupname, int nodeid);
```

パラメーター

項目	説明
clusterid	希望するクラスターのクラスター ID。
groupname	リソース・グループの名前。
nodeid	クラスター・ノードの ID。

状況コード

cl_resource_states 内のいずれかの列挙値を返します。 この列挙は、リソース・グループの状態を含む列挙型で記述されます。

例

```
  enum cl_resource_states state;
  int clusterid = 1113325332;
  int nodeid = 1;
  char groupname[CL_MAXNAMELEN] = "rg01";

  state = cl_getgroupnodestate(clusterid, groupname, nodeid);
  if (state == CL_RGNS_INVALID){
  cl_perror(state, "can't get group node state");
  } else {
  printf("node w/ id %d is currently in state %d in group %s¥n",
  nodeid, state, groupname);
  }
```

関連資料:

12 ページの『リソース・グループの状態を含む列挙型』
この列挙データ型には、ノード上のリソース・グループについて考えられるすべての状態が含まれます。

cl_getgroupsbynode ルーチン

cl_getgroupsbynode ルーチンは、指定されたノードがメンバーになっているリソース・グループのすべてに関する情報を返します。このルーチンでは戻される情報用のストレージが割り当てられるので、呼び出し元はこのストレージを解放する必要があることを覚えておいてください。

構文

```
int cl_getgroupsbynode (int clusterid, int nodeid,
  struct cl_group **groupbufp, int *groupcountp);
```

パラメーター

項目	説明
clusterid	希望するクラスターのクラスター ID。
nodeid	クラスター・ノードの ID。
groupbufp	情報が返される場所である **cl_group** 構造へのポインター。
groupcountp	返されるグループの数が格納される整数へのポインター。

状況コード

項目	説明
CLE_OK	成功。
CLE_BADARGS	欠落している、または無効な引数。
CLE_SYSERR	システム・エラー。
CLE_NOCLINFO	クラスター情報が利用できません。
CLE_IVCLUSTERID	無効なクラスター ID。
CLE_IVNODENAME	無効なノード ID。

例

```
  int clusterid = 1113325332;
  int nodeid = 1;
  int status;
  int groupcount;
  int j;
  struct cl_group *groups;

  status = cl_getgroupsbynode(clusterid, nodeid, &groups, &groupcount);
  if (status != CLE_OK){
      cl_perror(status,"failed to get resource group information");
  } else {
      printf("node %d is a member of %d groups:¥n",nodeid, groupcount);
      for(j=0; j < groupcount; j++){
printf("node %d is in group %s¥n",
 nodeid, groups[j].clg_name);}
      if (groupcount) free (groups);
  }
```

cl_getifaddr ルーチン

指定されたクラスター ID およびインターフェース名を持つインターフェースのアドレスを返します。 このルーチンは IPv4 アドレスのみ処理可能です。

構文

```
int cl_getifaddr (int clusterid, char *interfacename,
struct sockaddr_in *addr)
```

パラメーター

項目	説明
clusterid	希望するネットワーク・インターフェースのクラスター ID。
interfacename	希望するネットワーク・インターフェースの名前。
addr	返されるネットワーク・インターフェース・アドレス用のストレージ。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNETIFNAME	要求は無効なネットワーク・インターフェース名を指定しました。

例

```
int clusterid = 1113325332;
int status;
char* interfacename = "geotest9";
struct sockaddr_in addr;

status = cl_getifaddr (clusterid, interfacename, &addr);
if (status != CLE_OK) {
    cl_perror (status, "Can't find interface address");
} else {
    printf ("interface address w/ name %s is %s¥n",interfacename,
    inet_ntoa (addr.sin_addr.s_addr));
}
```

cl_getifaddr6 ルーチン

指定されたクラスター ID およびインターフェース名を持つインターフェースのアドレスを返します。

構文

```
int cl_getifaddr6(int clusterid, char *interfacename,
struct sockaddr *addr, size_t size_of_addr)
```

パラメーター

項目	説明
clusterid	希望するネットワーク・インターフェースのクラスター ID。
interfacename	希望するネットワーク・インターフェースの名前。
addr	返されるネットワーク・インターフェース・アドレス用のストレージ。
size_of_addr	「addr」バッファーのサイズ。少なくともサイズ「struct sockaddr_in」(IPv4 ソケットの場合) および「struct sockaddr6_in」(IPv6 ソケットの場合) であることが必要です。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNETIFNAME	要求は無効なネットワーク・インターフェース名を指定しました。

cl_getifname ルーチン

指定されたクラスター ID および IP アドレスを持つインターフェースの名前を返します。 このルーチンは IPv4 アドレスのみ処理可能です。

構文

```
int cl_getifname (int clusterid, struct sockaddr_in *addr,

char *interfacename)
```

パラメーター

項目	説明
clusterid	希望するネットワーク・インターフェースのクラスター ID。
addr	目的のネットワーク・インターフェースの IP アドレス。
interfacename	返されるネットワーク・インターフェース名用のストレージ。 interfacename パラメーターの長さは CL_MAXNAMELEN 文字を超えてはなりません。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVADDRESS	要求は無効なネットワーク・インターフェース・アドレスを指定しました。

例

```
int clusterid = 1113325332;
int status;
struct sockaddr_in addr;
char interfacename[CL_MAXNAMELEN] = "9.57.28.23";

addr.sin_family = AF_INET;
addr.sin_addr.s_addr = inet_addr (interfacename);
status = cl_getifname (clusterid, &addr, interfacename);
if (status != CLE_OK) {
    cl_perror (status, "can't find interface name");
}
```

```
else {
    printf ("interface name w/ address %s is %s¥n",
inet_ntoa (addr.sin_addr.s_addr), interfacename);
}
```

cl_getifname6 ルーチン

指定されたクラスター ID および IP アドレスを持つインターフェースの名前を返します。

構文

```
int cl_getifname6 (int clusterid, struct sockaddr *addr,  size_t size_of_addr,
char *interfacename)
```

パラメーター

項目	説明
clusterid	希望するネットワーク・インターフェースのクラスター ID。
addr	目的のネットワーク・インターフェースの IP アドレス。
interfacename	返されるネットワーク・インターフェース名用のストレージ。interfacename パラメーターの長さは CL_MAXNAMELEN 文字を超えてはなりません。
size_of_addr	「addr」バッファーのサイズ。少なくともサイズ「struct sockaddr_in」(IPv4 ソケットの場合) および「struct sockaddr6_in」(IPv6 ソケットの場合) であることが必要です。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVADDRESS	要求は無効なネットワーク・インターフェース・アドレスを指定しました。

cl_getlocalid ルーチン

要求を発行しているノードのクラスター ID とノード名を返します。 この要求では、クラスター内で現在アクティブでないノードについては、エラー状況コードが返されます。

構文

```
int cl_getlocalid (int *clusterid, char *nodename)
```

パラメーター

項目	説明
clusterid	返されるクラスター ID 用のストレージ。
nodename	返されるノード名用のストレージ。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。
CLE_IVNODE	ノードがクラスター・ノードではありません。

例

```
int clusterid, status;
char nodename[CL_MAXNAMELEN] = "node1";

status = cl_getlocalid (&clusterid, nodename);
if (status != CLE_OK) {
    if (status == CLE_IVNODE) {
printf("this node is not a cluster member¥n");
    } else {
cl_perror(status, "can't get local cluster ID");
    }
} else {
    printf ("this node [%s] is a member of cluster id %d¥n",
    nodename,clusterid);
}
```

cl_getnet ルーチン

指定されたネットワークに関する情報を返します。

構文

```
int cl_getnet (int clusterid, int netid, struct cl_net *netbuf)
```

パラメーター

項目	説明
clusterid	希望するクラスターのクラスター ID。
netid	ネットワークの ID。
netbuf	情報が返される場所である **cl_net** 構造へのポインター。

状況コード

項目	説明
CLE_OK	成功。
CLE_BADARGS	欠落している、または無効な引数。
CLE_SYSERR	システム・エラー。
CLE_NOCLINFO	クラスター情報が利用できません。
CLE_IVCLUSTERID	無効なクラスター ID。
CLE_IVNETID	無効なネットワーク ID。

例

```
  int clusterid = 1113325332;
  int netid = 1;
  int status, j;
  struct cl_net netmap;

  status = cl_getnet(clusterid, netid, &netmap);
  if (status == CLE_OK)
  {
 printf("information for cluster network %s (id %d):¥n",
netmap.clnet_name, netmap.clnet_id);
 printf("network is type %s¥n", netmap.clnet_type);
 printf("network attribute is %d¥n", netmap.clnet_attr);
 printf("there are %d nodes on this network¥n",
netmap.clnet_numnodes);
 for (j=0; j<netmap.clnet_numnodes; j++)
 {
      enum cls_state node_state;
      printf("  node id = %d, state = %d,",
      netmap.clnet_node_ids[j],
      netmap.clnet_node_states[j]);
      cl_getnetstatebynode( clusterid, netmap.clnet_id,
      netmap.clnet_node_ids[j], &node_state);
      printf(" state (cl_getnetstatebynode) = %d¥n",
     node_state);
 }
  }
```

関連資料:

51 ページの『cl_getnetstatebynode ルーチン』
指定されたノード上の指定されたネットワークの状態を返します。

cl_getnetbyname ルーチン

特定の名前付きネットワークに関する情報を返します。

構文

```
int cl_getnetbyname (int clusterid, char *netname,
  struct cl_net *netbuf)
```

パラメーター

項目	説明
clusterid	希望するクラスターのクラスター ID。
netname	ネットワークの名前。
netbuf	情報が返される場所である **cl_net** 構造へのポインター。

状況コード

項目	説明
CLE_OK	成功。
CLE_BADARGS	欠落している、または無効な引数。
CLE_SYSERR	システム・エラー。
CLE_NOCLINFO	クラスター情報が利用できません。
CLE_IVCLUSTERID	無効なクラスター ID。
CLE_IVNETNAME	無効なネットワーク名。

例

```
  char netname[CL_MAXNAMELEN];
  int clusterid = 1;
  int status;
  struct cl_net netmap;

  status = cl_getnetbyname(clusterid,netname,&netmap);
  if (status != CLE_OK) {
display_error(status,cmd);
  } else {
printf("network %s is type %s: connected to %d nodes¥n",
netmap.clnet_name,
netmap.clnet_type,
netmap.clnet_numnodes);
  }
```

cl_getnetmap ルーチン

クラスター内のすべてのネットワークに関する情報を返します。

構文

```
int cl_getnetmap (int clusterid, struct cl_net *netmap)
```

パラメーター

項目	説明
clusterid	希望するクラスターのクラスター ID。
netmap	返されるネットワーク情報用のストレージ。

状況コード

負でない数値 (クラスター内のネットワークの数) が返された場合、要求は正常に完了しました。

項目	説明
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。

例

```
  int clusterid = 1113325332;
  int num_nets;
  int i,j;
  struct cl_net *nm;

  cl_alloc_netmap(&nm);
  num_nets = cl_getnetmap(clusterid, nm);
  printf("¥n¥ndumping netmap with %d nets for cluster %d¥n¥n",
      num_nets, nm->clnet_clusterid);
  for (i=0; i<num_nets; i++)
  {
      printf("info for network: %s (%d)¥n", nm[i].clnet_name,
     nm[i].clnet_id);
      printf(" type = %s¥n", nm[i].clnet_type);
      printf(" attribute = %d¥n", nm[i].clnet_attr);
      printf(" state = %d¥n", nm[i].clnet_state);
      printf(" number of nodes = %d¥n", nm[i].clnet_numnodes);
      for (j=0; j<nm[i].clnet_numnodes; j++)
      {
```

```
printf(" node id = %d , state = %d¥n",
nm[i].clnet_node_ids[j],
nm[i].clnet_node_states[j]);
}
}
cl_free_netmap(nm);
```

関連資料:

26 ページの『cl_alloc_netmap ルーチン』
一連のネットワーク情報構造にストレージを割り当てます。

31 ページの『cl_free_netmap ルーチン』
前に **cl_alloc_netmap** を呼び出して割り当てられたストレージを解放します。

cl_getnetsbyattr ルーチン

指定された属性 (パブリック、プライベート、非 IP (シリアル)) を使用して構成されたネットワークに関する情報を返します。

構文

```
int cl_getnetsbyattr (int clusterid, int *netattr,
   struct cl_net **netbuf, int *netcount)
```

パラメーター

項目	説明
clusterid	希望するクラスターのクラスター ID。
netattr	ネットワーク・リストを取得する対象となるネットワーク属性。
netbuf	情報が返される場所である **cl_net** 構造へのポインター。
netcount	指定されたタイプと一致するネットワークの数が返される整数へのポインター。

状況コード

項目	説明
CLE_OK	成功。 このコードは、返されるネットワークがない場合、すなわち、指定された属性を持つネットワークが存在しない場合でも返される可能性があるので注意してください。
CLE_BADARGS	欠落している、または無効な引数。
CLE_SYSERR	システム・エラー。
CLE_NOCLINFO	クラスター情報が利用できません。
CLE_IVCLUSTERID	無効なクラスター ID。
CLE_IVNETATTR	無効なネットワーク属性。

例

```
int status,i;
int netcount;
int clusterid = 1;
struct cl_net *netmap;
enum cl_network_attribute attr;

attr = CL_NET_TYPE_PRIVATE;
status = cl_getnetsbyattr(clusterid, attr, &netmap,&netcount);
if (status != CLE_OK) {
   cl_perror(status,"cl_getnetsbyattr() failed");
} else {
```

```
     printf("there are %d private networks in this cluster.¥n",netcount);
     for (i=0; i<netcount; i++)
     {
struct cl_net network;
 status = cl_getnetbyname(clusterid, netmap[i].clnet_name,
   &network);
 if (status != CLE_OK) {
      cl_perror(status,"cl_getnetbyname() failed");
} else {
      printf(" network %s is type %s: connected to %d nodes¥n",
  network.clnet_name,
  network.clnet_type,
  network.clnet_numnodes);
 }
     }
  }
```

cl_getnetsbytype ルーチン

指定されたネットワーク・タイプを使用して構成されたネットワークに関する情報を返します。

構文

```
int cl_getnetsbytype (int clusterid, char *nettype,
   struct cl_net **netbuf, int *netcount)
```

パラメーター

項目	説明
clusterid	必要なクラスターのクラスター ID。
nettype	ネットワーク・リストを取得する対象となるネットワーク・タイプ。 ネットワークのタイプは、現在 PowerHA SystemMirror に対して定義されているネットワーク・インターフェース・モジュール (NIM) のリストから取得されます。
netbuf	情報が返される場所である **cl_net** 構造へのポインター。
netcount	指定されたタイプに一致するネットワークの数が返される整数へのポインター。

状況コード

項目	説明
CLE_OK	成功。 このコードは、返されるネットワークがない場合、つまり指定されたタイプのネットワークが存在しない場合でも返されることがあります。
CLE_BADARGS	欠落している、または無効な引数。
CLE_SYSERR	システム・エラー。
CLE_NOCLINFO	クラスター情報が利用できません。
CLE_IVCLUSTERID	無効なクラスター ID。
CLE_IVNETTYPE	無効なネットワーク・タイプ。

例

```
char net_type[CL_MAXNAMELEN] = "ether";
  int status,i,j;
  int netcount;
  int clusterid = 1113325332;
  struct cl_net *netmap;

  status = cl_getnetsbytype(clusterid, net_type, &netmap, &netcount);
  if (status != CLE_OK) exit(status);
  printf("there are %d networks of type:
  for (i=0; i<netcount; i++)
```

```
{
  struct cl_net network;
  status = cl_getnetbyname(clusterid, netmap[i].clnet_name, &network);
  if (status != CLE_OK) exit(status);
  printf("network %s has attribute %d and is connected to %d nodes¥n",
  network.clnet_name,
  network.clnet_attr,
  network.clnet_numnodes);
}
```

cl_getnetstatebynode ルーチン

指定されたノード上の指定されたネットワークの状態を返します。

構文

```
int cl_getnetstatebynode (int clusterid, int netid,
  int nodeid, enum cls_state *state)
```

パラメーター

項目	説明
clusterid	希望するクラスターのクラスター ID。
netid	ネットワーク状態を取得する対象となるネットワーク ID。
nodeid	ネットワーク状態を取得する対象となるノード ID。
netstate	指定されたノード上の指定されたネットワークの状態が返される整数へのポインター。

状況コード

項目	説明
CLE_OK	成功。
CLE_BADARGS	欠落している、または無効な引数。
CLE_SYSERR	システム・エラー。
CLE_NOCLINFO	クラスター情報が利用できません。
CLE_IVCLUSTERID	無効なクラスター ID。
CLE_IVNETID	無効なネットワーク ID。
CLE_IVNODEID	無効なノード ID。 この戻り値は、ID が単に無効である (このクラスターに対して有効なノード ID ではない) ことを示す場合、またはノードが指定されたネットワークに接続されていない (指定されたネットワーク上にインターフェースがない) ことを示す場合があります。

例

cl_getnet ルーチンの例を参照してください。

関連資料:

46 ページの『cl_getnet ルーチン』
指定されたネットワークに関する情報を返します。

cl_getnode ルーチン

クラスター ID/ノード名のペアによって指定されたノードに関する情報を返します。

構文

```
int cl_getnode (int clusterid, char *nodename,
  struct cl_node *nodebuf);
```

パラメーター

項目	説明
clusterid	希望するノードのクラスター ID。
nodename	目的のノードのノード名。
nodebuf	返されるノード情報用のストレージ。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNODENAME	要求は無効なノード名を指定しました。

例

```
  int clusterid = 1113325332;
  int status;
  char* nodename = "node1";
  struct cl_node nodebuf;

  status = cl_getnode (clusterid, nodename, &nodebuf);
  if (status != CLE_OK) {
      cl_perror(status, "can't get node info");
  } else {
      printf("node named %s on cluster %d : id= %d, state= %d¥n",
 nodename,
 clusterid,
 nodebuf.cln_nodeid,
 nodebuf.cln_state);
  }
  cl_node_free(&nodebuf);
```

cl_getnodeaddr ルーチン

指定されたクラスター ID/ネットワーク・インターフェース名のペアに関連付けられている IP アドレスを返します。 このルーチンは IPv4 アドレスのみ処理可能です。

構文

```
int cl_getnodeaddr (int clusterid, char *interfacename,
struct sockaddr_in *addr)
```

パラメーター

項目	説明
clusterid	希望するネットワーク・インターフェースのクラスター ID。
interfacename	希望するネットワーク・インターフェースの名前。
addr	返されるネットワーク・インターフェース・アドレス用のストレージ。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNETIFNAME	要求は無効なネットワーク・インターフェース名を指定しました。

例

```
int clusterid = 1113325332;
int status;
char* interfacename = "geotest9";
struct sockaddr_in addr;

status = cl_getnodeaddr(clusterid, interfacename, &addr);
if (status != CLE_OK) {
    cl_perror(status, "can't get node addr");
} else {
    printf("address of interface %s on cluster %d is %s¥n",
    interfacename, clusterid, inet_ntoa(addr.sin_addr));
}
```

cl_getnodeaddr6 ルーチン

指定されたクラスター ID/ネットワーク・インターフェース名のペアに関連付けられている IP アドレスを返します。

構文

```
int cl_getnodeaddr6 (int clusterid, char *interfacename,
struct sockaddr *addr, size_t size_of_addr)
```

パラメーター

項目	説明
clusterid	希望するネットワーク・インターフェースのクラスター ID。
interfacename	希望するネットワーク・インターフェースの名前。
addr	返されるネットワーク・インターフェース・アドレス用のストレージ。
size_of_addr	「addr」バッファーのサイズ。少なくともサイズ「struct sockaddr_in」(IPv4 ソケットの場合) および「struct sockaddr6_in」(IPv6 ソケットの場合) であることが必要です。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNETIFNAME	要求は無効なネットワーク・インターフェース名を指定しました。

cl_getnodemap ルーチン

cl_getnodemap ルーチンは、クラスター内のノードに関する情報を返します。 このルーチンを呼び出してメモリー内のストレージを予約する前に、**cl_alloc_nodemap** を呼び出す必要があります。 このルーチンを呼び出した後は、**cl_free_nodemap** を呼び出す必要があります。

構文

```
int cl_getnodemap (int clusterid, struct cl_node *nodemap)
```

パラメーター

項目	説明
clusterid	希望するクラスターのクラスター ID。
nodemap	返されるノード情報用のストレージ。

状況コード

負でない数値 (クラスター内のノードの数) が返された場合、要求は正常に完了しました。

項目	説明
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。

例

```
int clusterid = 1113325332;
int i;
int nodes;
struct cl_node *nodemap;

cl_alloc_nodemap (&nodemap);
if (nodemap==NULL){
    printf("unable to allocate storage: cl_alloc_nodemap = NULL¥n");
    exit(-1);
}
nodes = cl_getnodemap(clusterid, nodemap);
if(nodes < 0)
{
    cl_perror(nodes,"can't get node map");
} else {
  printf("cluster %d has %d nodes:¥n", clusterid, nodes);
  for(i=0; i < nodes; i++){
    printf("node %s in state %d has %d interfaces¥n",
    nodemap[i].cln_nodename,
    nodemap[i].cln_state,
    nodemap[i].cln_nif);
```

```
    if(clusterid != nodemap[i].cln_clusterid){
  printf("structure has invalid cluster ID: %d",
  nodemap[i].cln_clusterid);
    }
   }
  }
  cl_free_nodemap(nodemap);
```

関連資料:

27 ページの『cl_alloc_nodemap ルーチン』

cl_alloc_nodemap ルーチンは、ノードのリストおよび各ノードに関連付けられているインターフェースにストレージを割り当てます。 このルーチンは、**cl_getnodemap** ルーチンより先に呼び出す必要があります。

32 ページの『cl_free_nodemap ルーチン』

cl_free_nodemap ルーチンは、前に **cl_alloc_nodemap** ルーチンを呼び出して割り当てられたストレージを解放します。

cl_getnodenamebyifaddr ルーチン

指定されたインターフェース・アドレスを持つノードの名前を返します。 Clinfo はクラスター内の各ノードでネットワーク・インターフェースをスキャンします。一致が見つかった場合、そのインターフェース・アドレスに関連付けられているノードのノード名が返されます。

構文

```
int cl_getnodenamebyifaddr (int clusterid, struct sockaddr_in *addrp,
   char *nodename)
```

パラメーター

項目	説明
clusterid	希望するノードのクラスター ID。
addr	目的のノードのネットワーク・インターフェース・アドレス。
nodename	希望するノードの名前。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVADDRESS	要求は無効なネットワーク・インターフェース・アドレスを指定しました。

例

```
int clusterid = 1113325332;
int status;
char nodename[CL_MAXNAMELEN];
struct sockaddr_in addr;

addr.sin_family = AF_INET;
addr.sin_addr.s_addr = inet_addr ("9.57.28.23");
status = cl_getnodenamebyifaddr (clusterid, &addr, nodename);
if (status !=CLE_OK) {
     cl_perror(status,"can't get node name");
```

```
} else {
     printf("node name of interface w/ address %s on
             cluster %d is %s¥n",
inet_ntoa(addr.sin_addr), clusterid, nodename);
}
```

cl_getnodenamebyifaddr6 ルーチン

指定されたインターフェース・アドレスを持つノードの名前を返します。 Clinfo はクラスター内の各ノードでネットワーク・インターフェースをスキャンします。一致が見つかった場合、そのインターフェース・アドレスに関連付けられているノードのノード名が返されます。

構文

```
int cl_getnodenamebyifaddr6 (int clusterid, struct sockaddr *addrp, size_t size_of_addr, char *nodename)
```

パラメーター

項目	説明
clusterid	希望するノードのクラスター ID。
addr	目的のノードのネットワーク・インターフェース・アドレス。
size_of_addr	「addr」バッファーのサイズ。少なくともサイズ「struct sockaddr_in」(IPv4 ソケットの場合) および「struct sockaddr6_in」(IPv6 ソケットの場合) であることが必要です。
nodename	希望するノードの名前。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVADDRESS	要求は無効なネットワーク・インターフェース・アドレスを指定しました。

cl_getnodenamebyifname ルーチン

指定されたインターフェース名を持つノードのノード名を返します。 Clinfo はクラスター内の各ノードでネットワーク・インターフェースをスキャンします。一致が見つかった場合、そのノードのノード名が返されます。

構文

```
int cl_getnodenamebyifname (int clusterid, char *interfacename,
  char *nodename)
```

パラメーター

項目	説明
clusterid	希望するノードのクラスター ID。
interfacename	目的のノードのネットワーク・インターフェース名。
nodename	希望するノードの名前。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNETIFNAME	要求は無効なネットワーク・インターフェース名を指定しました。

例

```
  int clusterid = 1113325332;
  int status;
  char nodename[CL_MAXNAMELEN];
  char interfacename[CL_MAXNAMELEN] = "geotest9";<

  status = cl_getnodenamebyifname (clusterid, interfacename, nodename);
  if (status != CLE_OK) {
      cl_perror(status,"can't get node name");
  } else {
      printf("interface=
      printf("node name of interface w/ name %s on cluster %d is %s¥n",
interfacename,
clusterid,
nodename);
  }
```

cl_getprimary ルーチン

指定されたクラスターのユーザー指定の 1 次クラスター・マネージャーのノード名を返します。

構文

```
int cl_getprimary (int clusterid, char *nodename)
```

パラメーター

項目	説明
clusterid	目的の 1 次ノード名を含むクラスターの ID。
nodename	この引数では、1 次クラスター・マネージャー・ノードとして指定されているノードの名前が返されます。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_NOPRIMARY	1 次クラスター・マネージャーに関する情報がありません。
CLE_IVCLUSTER	そのクラスターは使用できません。

例

```
int clusterid = 1113325332;
int status;
char nodename[CL_MAXNAMELEN] = "node1";

status = cl_getprimary (clusterid, nodename);
if (status != CLE_OK) {
    cl_perror(status,"can't get cluster primary");
} else {
    printf ("primary node for cluster %d is %s¥n",
    clusterid, nodename);
}
```

cl_getsite ルーチン

特定のサイトに関する情報を返します。

構文

```
int cl_getsite(int clusterid, int siteid, struct cl_site *sitebuf)
```

パラメーター

項目	説明
clusterid	目的のクラスター。
siteid	クラスターの ID。
sitebuf	返される **cl_site** 構造を保持します。

状況コード

項目	説明
CLE_OK	成功。
CLE_BADARGS	欠落している、または無効な引数。
CLE_SYSERR	システム・エラー。
CLE_NOCLINFO	クラスター情報が利用できません。
CLE_IVCLUSTERID	無効なクラスター ID。
CLE_IVSITEID	無効なサイト ID。

例

```
int clusterid = 1113325332;
int status;
int siteid = 1;
struct cl_site site;

status = cl_getsite(clusterid, siteid, &site);
if (status == CLE_OK) {
    printf("site %s (%d) has %d nodes and is priority %d¥n",
site.clsite_name,
```

```
  site.clsite_id,
  site.clsite_numnodes,
  site.clsite_priority);
  } else {
      cl_perror(status,"can't get site information");
  }
```

cl_getsitebyname ルーチン

特定の名前付きサイトに関する情報を返します。

構文

```
int cl_getsitebyname (int clusterid, char *sitename, struct cl_site *sitebuf)
```

パラメーター

項目	説明
clusterid	目的のクラスター。
sitename	返されるサイトの名前。
sitebuf	返される **cl_site** 構造を保持します。

状況コード

項目	説明
CLE_OK	成功。
CLE_BADARGS	欠落している、または無効な引数。
CLE_SYSERR	システム・エラー。
CLE_NOCLINFO	クラスター情報が利用できません。
CLE_IVCLUSTERID	無効なクラスター ID。
CLE_IVSITENAME	無効なサイト ID。

例

```
  int clusterid = 1113325332;
  int status;
  char sitename[CL_MAXNAMELEN] = "geo9";
  struct cl_site site;

  status = cl_getsitebyname(clusterid, sitename, &site);
  if (status == CLE_OK)
      printf("site %s (%d) has %d nodes and is priority %d¥n",
site.clsite_name,
site.clsite_id,
site.clsite_numnodes,
site.clsite_priority);
```

cl_getsitebypriority ルーチン

指定された優先度を使用して構成されたサイトに関する情報を返します。

構文

```
int cl_getsitebypriority (int clusterid, enum cl_site_priority,
    priority, struct cl_site *sitebuf)
```

パラメーター

項目	説明
clusterid	目的のクラスター。
priority	列挙されたサイトの優先度の値のいずれか。
site_buf	返されるサイト情報を保持します。

状況コード

項目	説明
CLE_OK	成功。
CLE_BADARGS	欠落している、または無効な引数。
CLE_SYSERR	システム・エラー。
CLE_NOCLINFO	クラスター情報が利用できません。
CLE_IVCLUSTERID	無効なクラスター ID。
CLE_IVSITEPRIORITY	無効なサイト優先度。

例

```
  enum cl_site_priority priority = CL_SITE_PRI_PRIMARY;
  int clusterid = 1113325332;
  int status;
  struct cl_site site;

  status = cl_getsitebypriority(clusterid, priority, &site);
  if (status == CLE_OK)
      printf("site %s (%d) has %d nodes and is priority %d¥n",
site.clsite_name,
site.clsite_id,
site.clsite_numnodes,
site.clsite_priority);
```

cl_getsitemap ルーチン

クラスター内のすべてのサイトに関する既知情報をすべて返します。

構文

```
int cl_getsitemap (int clusterid, struct cl_site *sitemap)
```

パラメーター

項目	説明
clusterid	目的のクラスター。
sitemap	**cl_getsitemap** を呼び出す前に、**cl_alloc_sitemap(&sitemap)** を使用してストレージを割り当てる必要があります。 そのストレージは、必ず **cl_free_sitemap(sitemap)** を使用して解放します。

状況コード

負でない数値 (クラスター内のサイトの数) が返された場合、要求は正常に完了しました。

項目	説明
CLE_BADARGS	欠落している、または無効な引数。
CLE_SYSERR	システム・エラー。
CLE_NOCLINFO	クラスター情報が利用できません。
CLE_IVCLUSTERID	無効なクラスター ID。

例

```
int clusterid = 1113325332;
int status;
int i,j;
int nbr_sites;
struct cl_site *sitemap;

cl_alloc_sitemap(&sitemap);
nbr_sites = cl_getsitemap(clusterid, sitemap);
printf("dumping sitemap with %d sites for cluster %d¥n¥n",
    nbr_sites, sitemap->clsite_clusterid);
for (i=0; i<nbr_sites; i++)
{
    printf("info for site %s (%d)¥n",
    sitemap[i].clsite_name, sitemap[i].clsite_id);
    printf(" priority = %d ¥n", sitemap[i].clsite_priority);
    printf(" backup = %d ¥n", sitemap[i].clsite_backup);
    printf(" state = %d ¥n", sitemap[i].clsite_state);
    printf(" number of nodes = %d¥n", sitemap[i].clsite_numnodes);
    for (j=0; j<sitemap[i].clsite_numnodes; j++)
    {
printf(" node id = %d¥n",
sitemap[i].clsite_nodeids[j]);
    }
}
cl_free_sitemap(sitemap);
```

関連資料:

28 ページの『cl_alloc_sitemap ルーチン』
一連のサイト情報構造にストレージを割り当てます。

33 ページの『cl_free_sitemap ルーチン』
前に **cl_alloc_sitemap** を呼び出して割り当てられたストレージを解放します。

cl_isaddravail ルーチン

指定したネットワーク・インターフェースの状況を戻します。このルーチンは IPv4 アドレスのみ処理可能です。

構文

```
int cl_isaddravail (int clusterid, char *nodename,
struct sockaddr_in *addr)
```

パラメーター

項目	説明
clusterid	希望するネットワーク・インターフェースのクラスター ID。
nodename	希望するネットワーク・インターフェースのノード名。
addr	希望するネットワーク・インターフェースのアドレス。

状況コード

項目	説明
CLE_OK	指定されたアドレスは所定のノードから使用可能です。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNODENAME	要求は無効なノード名を指定しました。
CLE_IVADDRESS	要求は無効なインターフェース・アドレスを指定しました。
CLE_IVNETIF	ネットワーク・インターフェースは使用可能ではありません。

例

```
  int clusterid = 1113325332;
  int status;
  char ifaddr[CL_MAXNAMELEN] = "9.57.28.23";
  char nodename[CL_MAXNAMELEN] = "node1";
  struct sockaddr_in addr;

  addr.sin_family = AF_INET;
  addr.sin_addr.s_addr = inet_addr (ifaddr);
  status = cl_isaddravail (clusterid, nodename, &addr);
  if (status != CLE_OK) {
      cl_perror(status,"interface not available");
  } else {
      printf ("interface address for %s is available¥n",
inet_ntoa (addr.sin_addr.s_addr));
  }
```

cl_isaddravail6 ルーチン

指定したネットワーク・インターフェースの状況を戻します。

構文

```
int cl_isaddravail6 (int clusterid, char *nodename,
struct sockaddr *addr, size_t size_of_addr)
```

パラメーター

項目	説明
clusterid	希望するネットワーク・インターフェースのクラスター ID。
nodename	希望するネットワーク・インターフェースのノード名。
addr	希望するネットワーク・インターフェースのアドレス。
size_of_addr	「addr」バッファーのサイズ。少なくともサイズ「struct sockaddr_in」(IPv4 ソケットの場合) および「struct sockaddr6_in」(IPv6 ソケットの場合) であることが必要です。

状況コード

項目	説明
CLE_OK	指定されたアドレスは所定のノードから使用可能です。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNODENAME	要求は無効なノード名を指定しました。
CLE_IVADDRESS	要求は無効なインターフェース・アドレスを指定しました。
CLE_IVNETIF	ネットワーク・インターフェースは使用可能ではありません。

cl_isclusteravail ルーチン

指定されたクラスター ID のクラスターの状況を返します。

構文

```
int cl_isclusteravail (int clusterid)
```

パラメーター

項目	説明
clusterid	希望するクラスターのクラスター ID。

状況コード

項目	説明
CLE_OK	このクラスターは使用可能です。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_IVCLUSTER	指定されたクラスターは使用できません。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。

例

```
int clusterid = 1113325332;
int status;

status = cl_isclusteravail (clusterid);
if (status != CLE_OK) {
    cl_perror (status, "cluster is not available");
} else {
    printf ("cluster %d is available¥n", clusterid);
}
```

cl_isnodeavail ルーチン

指定されたノードが稼働中であるかどうかを示します。

構文

```
int cl_isnodeavail (int clusterid, char *nodename)
```

パラメーター

項目	説明
clusterid	希望するノードのクラスター ID。
nodename	目的のノードのノード名。

状況コード

項目	説明
CLE_OK	このノードは指定されたクラスターのものです。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNODENAME	要求は無効なノード名を指定しました。
CLE_IVNODE	このノードは使用できません。

例

```
int clusterid = 1113325332;
int status;
char nodename[CL_MAXNAMELEN] = "node1";

status = cl_isnodeavail (clusterid, nodename);
if (status != CLE_OK) {
    cl_perror(status,"node is unavailable");
} else {
    printf ("node %s on cluster %d is available¥n",
   nodename, clusterid);
}
```

cl_model_release ルーチン

cl_model_release() ルーチンは、前のリリースで共有メモリーを切り離すために使用されていました。

cl_model_release() ルーチンには機能は何もなくなっており、常に **CLE_OK** を返します。
cl_model_release() は後方互換性の確保のみを目的に提供されています。 このルーチンを新しいプログラムで使用しないでください。

cl_node_free ルーチン

前に cl_getnode ルーチンを呼び出して返されたノードに関連付けられている、ネットワーク・インターフェースに割り当て済みのストレージを削除します。 メンバー **cln_if** はヌルに設定されます。

構文

```
int cl_node_free
```

パラメーター

項目	説明
nodebifp	ノード・バッファー

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。

ソースの例

```
int clusterid;
struct cl_node nodebuf;
struct sockaddr_in addr;
addr.sin_family = AF_INET;
addr.sin_addr.s_addr = inet_addr("9.57.28.8");

clusterid = cl_getclusteridbyifaddr (&addr);

cl_getnode(clusterid, "ppstest5", &nodebuf);
printf ("node id: %d has %d interfaces ¥n",
     nodebuf.cln_nodeid, nodebuf.cln_nif);

/* call cl_node_free to release storage for the network */
/* interfaces (part of the cl_node struct) which was    */
/* allocated by cl_getnode */
cl_node_free(&nodebuf);

printf ("after cl_node_free, node id: %d has %d interfaces ¥n",
    nodebuf.cln_nodeid, nodebuf.cln_nif);
```

出力例

```
node id: 1 has 5 interfaces
after cl_node_free, node id: 1 has 0 interfaces
```

cl_registereventnotify ルーチン

cl_registereventnotify ルーチンは、イベント通知要求のリストを Clinfo に登録します。

各要求では、イベント ID、クラスター ID、シグナル ID、および、(該当する場合は) ノード名かネットワーク ID (またはこの両方) を指定します。 このルーチンが正常に実行されると、指定された記述に合致するイベントが発生した時点で Clinfo は呼び出しプロセスにシグナル通知します。 このシグナルが受信されると、直前に発生したイベントに関する情報を取得するために **cl_getevent** ルーチンが呼び出される可能性があります。

イベント ID として -1 を指定すると、すべてのネットワーク・イベントおよびクラスター・イベントの通知を受け取る登録を行えます。 これらすべてのイベントの通知を受け取る登録をした場合、特定のイベントの通知を登録抹消することはできません。その場合は、すべてのイベントを登録抹消する必要があります。

ノード名 ID を含むイベントについては、-1 のイベント ID を使用できない点に注意してください。 すべてのノード名を登録する場合は、NULL ストリングを指定します。 現在は、ネットワーク ID を個々に指定できません。 イベント ID として -1 を使用してください。

このルーチンはクラスター ID が 0 の場合は機能しません。クラスター ID として別の番号を割り当ててください。

構文

```
#include <signal.h>
int cl_registereventnotify (int num_reqs, struct cli_enr_req_t*enr_list)
```

パラメーター

項目	説明
num_reqs	発行されているイベント通知登録要求の数 (限度は 10)。
enr_list	イベント通知登録要求のリスト。 登録が要求される可能性のあるイベントには、以下が含まれます。 **CL_SWAPPING_ADAPTER**. 指定されたアダプターはスワップされている途中です。 このイベントには、クラスター、ノード、およびネットワークの ID が必要です。 クラスターおよびネットワークの ID の値が -1 の場合、これらすべてのコンポーネントを示します。 すべてのノード名を示すには NULL ストリングを指定します。 これは、**hacmp.out** の **swap_adapter** イベントに相当します。 **CL_SWAPPED_ADAPTER**。 指定されたアダプターはスワップされました。 このイベントには、クラスター、ノード、およびネットワークの ID が必要です。 クラスターおよびネットワークの ID の値が -1 の場合、これらすべてのコンポーネントを示します。 すべてのノード名を示すには NULL ストリングを指定します。 これは、**hacmp.out** の **swap_adapter_complete** イベントに相当します。 **CL_JOINING_NETWORK**。 指定されたネットワークは参加している途中です。 このイベントには、クラスターおよびネットワークの ID が必要です。 これらの ID のいずれかの値が -1 の場合、そのすべてのコンポーネントを示します。 これは、**hacmp.out** の **network_up** イベントに相当します。 **CL_JOINED_NETWORK**。 指定されたネットワークは稼働状態です。 このイベントには、クラスターおよびネットワークの ID が必要です。これらの ID のいずれかの値が -1 の場合、そのすべてのコンポーネントを示します。 これは、**hacmp.out** の **network_up_complete** イベントに相当します。 **CL_FAILING_NETWORK**。 指定されたネットワークはダウン状態になっている途中です。このイベントには、クラスターおよびネットワークの ID が必要です。これらの ID のいずれかの値が -1 の場合、そのすべてのコンポーネントを示します。 これは、**hacmp.out** の **network_down** イベントに相当します。 **CL_FAILED_NETWORK**。指定されたネットワークはダウン状態です。 このイベントには、クラスターおよびネットワークの ID が必要です。 これらの ID のいずれかの値が -1 の場合、そのすべてのコンポーネントを示します。 これは、**hacmp.out** の **network_down_complete** イベントに相当します。

項目	説明
	CL_JOINING_NODE。 指定されたノードは稼働状態になっている途中です。このイベントには、クラスターおよびノードの ID が必要です。 これは、**hacmp.out** の **node_up** イベントに相当します。 CL_JOINED_NODE。指定されたノードは稼働状態です。 このイベントには、クラスターおよびノードの ID が必要です。これは、**hacmp.out** の **node_up_complete** イベントに相当します。 **CL_FAILING_NODE**。指定されたノードはダウン状態になっている途中です。 このイベントには、クラスターおよびノードの ID が必要です。 これは、**hacmp.out** の **node_down** イベントに相当します。 **CL_FAILED_NODE**。 指定されたノードはダウン状態です。 このイベントには、クラスターおよびノードの ID が必要です。 これは、**hacmp.out** の **node_down_complete** イベントに相当します。 **CL_NEW_PRIMARY**。指定されたクラスターに新しい 1 次ノードがあります。 このイベントにはクラスター ID が必要です。 1 次ノードの概念は前のリリースで使用されていた属性であり、アプリケーションに関してノードのロールに対応していない可能性があります。 **CL_STABLE**。 指定されたクラスターは安定状態になりました。 このイベントにはクラスター ID が必要です。値が -1 の場合、このすべてのコンポーネントを示します。 **CL_UNSTABLE**。指定されたクラスターは不安定状態になりました。 このイベントにはクラスター ID が必要です。 値が -1 の場合、このすべてのコンポーネントを示します。
SIGNALS	任意の適切な UNIX シグナルを指定できます。この場合、シグナルについて理解していることが前提となります。 SIGUSR1 および SIGUSR2 が推奨されます。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNODENAME	要求は無効なノード名を指定しました。
CLE_IVREQNUM	無効な数 (10 を超える数) のイベント通知登録要求が指定されました。
CLE_IVNETID	無効なネットワーク ID が指定されました。
CLE_IVEVENTID	無効なイベント ID が指定されました。

例

以下は、単純なテスト・プログラムでの **cl_registereventnotify**、**cl_unregistereventnotify**、および **cl_getevent** ルーチンの使用例を示しています。

この例では、クラスター 83 の node2 に接続されているネットワークで、FAILING_NETWORK イベントを引き起こすほどの重大な問題が発生したときに SIGUSR1 シグナルで通知を受け取るよう、アプリケーションを登録します。 登録後、アプリケーションはイベントを待ちます。 イベントが発生すると、Clinfo は SIGUSR1 シグナルをアプリケーションに送信します。この後、アプリケーションは **cl_getevent** を使用してそのイベントに関する情報を取り寄せ、受け取った情報を出力します。 最終的に、アプリケーションはこのイベントの通知を登録抹消します。

```
#include <stdio.h>
#include <sys/types.h>
#include <signal.h>
```

```
#include <cluster/clinfo.h>
volatile int no_signal = 1;

/* Signal handler for catching event notification signal */
void
catch_sig(int sig)
{
no_signal = 0;
}

int
main(int argc, char *argv[])
{
  int ret_code, i;
  struct cli_enr_req_t en_req;  /*  Event notification request  */
  struct cli_en_msg_t  en_msg;  /*  Event notification message */
  en_req.event_id = CL_FAILING_NETWORK; /* specify a failing
    network  event  */
  en_req.cluster_id = 83;
  strcpy (en_req.node_name,"node2");
  en_req.net_id = 1;      /*  no net id necessary
    for this event  */
  en_req.signal_id = SIGUSR1;/*  Request to register for event
    notification */

  if(ret_code = cl_registereventnotify((int) 1, &en_req)!=CLE_OK )
  {
    printf("cl_en_test: cl_registereventnotify failed
    with error
  exit(1);
  }
/* Set up a signal handler to catch the event notification
   signal from Clinfo   */

  ret_code = signal(SIGUSR1, catch_sig);
  if ( ret_code < 0 ){
    perror("cl_en_test");
    exit(1);
  }
    /*  Wait for signal  */

  printf("cl_en_test: waiting for signal from Clinfo.");
  while (no_signal){
  pause();}
    /*  Execution will start here after catch_sig executes when
the signal is received.  Get the event notification message.  */

  if ( ret_code = cl_getevent(&en_msg) != CLE_OK )
      {
  printf("cl_en_test: cl_getevent failed with error %d.",
     ret_code);
  exit(1);
       }
/* Print out the event notification information received */
  printf("cl_en_test: Event notification message received
    from Clinfo:");
  printf("cl_en_test: Event id = %d", en_msg.event_id);
  printf("cl_en_test: Cluster id = %d", en_msg.cluster_id);
  printf("cl_en_test: Node name = %s", en_msg.node_name);
  printf("cl_en_test: Net id = %d", en_msg.net_id);
/*  Request to unregister the event notification  */
    if ( (ret_code = cl_unregistereventnotify((int) 1,&en_req))!=CLE_OK)
  {
  printf("cl_en_test: cl_unregistereventnotify failed with
```

```
  error %d.", ret_code);
  exit(1);
  }
}
```

関連資料:

38 ページの『cl_getevent ルーチン』
cl_getevent ルーチンはイベント通知メッセージを返します。 呼び出し元は、前の **cl_registereventnotify** 要求で指定されたシグナルの受信後にのみ、この要求を発行することが必要です。

『cl_unregistereventnotify ルーチン』
cl_unregistereventnotify ルーチンは、Clinfo へのイベント通知要求のリストの登録を抹消します。 各要求では、イベント ID、クラスター ID、シグナル ID、および、(該当する場合は) ノード ID かネットワーク ID (またはこの両方) を指定します。 このルーチンが正常に実行されると、Clinfo は指定された記述に合致するすべてのイベントの登録を削除します。

cl_unregistereventnotify ルーチン

cl_unregistereventnotify ルーチンは、Clinfo へのイベント通知要求のリストの登録を抹消します。 各要求では、イベント ID、クラスター ID、シグナル ID、および、(該当する場合は) ノード ID かネットワーク ID (またはこの両方) を指定します。 このルーチンが正常に実行されると、Clinfo は指定された記述に合致するすべてのイベントの登録を削除します。

イベント ID として -1 を指定してすべてのネットワーク・イベントまたはクラスター・イベントの通知を受け取る登録を行った場合は、すべてを登録抹消する必要があります。 すべてのイベントを登録した後に、個々のイベントの登録を抹消することはできません。特定のイベントを登録した場合は、それらのイベントを個別に登録抹消してください。

構文

```
int cl_unregistereventnotify (int num_reqs,
  struct cli_enr_req_t *enr_list)
```

パラメーター

cl_registereventnotify ルーチンのパラメーターを参照してください。

状況コード

cl_registereventnotify ルーチンの状況コードを参照してください。

例

cl_registereventnotify ルーチンの例を参照してください。

関連資料:

65 ページの『cl_registereventnotify ルーチン』
cl_registereventnotify ルーチンは、イベント通知要求のリストを Clinfo に登録します。

Clinfo C++ API

Clinfo C++ API は、PowerHA SystemMirror for AIX クラスターに関する状況情報を取得するために C++ アプリケーションで使用できるオブジェクト指向インターフェースです。 ここに含まれるトピックでは、Clinfo C++ API で使用可能なそれぞれの C++ 言語オブジェクトおよびメソッドについて説明します。

このセクションに加え、C API について説明する『Clinfo C API』もお読みください。 C++ API ルーチンは C API ルーチンを呼び出します。

C++ および AIX XL C++ コンパイラーについて詳しくは、「*C for AIX C/C++ ランゲージ・リファレンス バージョン 6*」を参照してください。

注**:**

- Clinfo API は、リリース間のバージョン互換性をサポートします。前のリリースの Clinfo を使用して Clinfo プログラムをコンパイルした場合、そのプログラムを新しいリリースの Clinfo で使用するために再コンパイルする必要はありません。 ただし、このリリースの新しい機能を使用する場合は、再コンパイルが必要です。
- **cl_registerwithclsmuxpd()** API は非推奨となっています。 この API を使用してコンパイルされたプリスクリプトおよびポストスクリプトはロードできません。 **cl_registerwithclsmuxpd()** ルーチンの代わりにアプリケーション・モニターを使用してください。「プランニング・ガイド」のクラスターの初期プランニングに関するセクションを参照してください。

関連概念:

10 ページの『Clinfo C API』

Clinfo C アプリケーション・プログラミング・インターフェース (API) は、PowerHA SystemMirror クラスターに関する状況情報を取得するためにアプリケーションで使用できる高水準インターフェースです。ここに含まれるトピックでは、Clinfo C API で使用可能なそれぞれの C 言語のルーチンおよびユーティリティーについて説明します。

関連情報:

クラスターの初期プランニング

Clinfo C++ オブジェクト・クラス

Clinfo C++ API には、オブジェクト・クラスとして、クラスター、ノード、ネットワーク・インターフェース、およびリソース・グループがあります。

各ネットワーク・インターフェースは単一のノードに属し、各ノードと各リソース・グループは単一のクラスターに属し、これがクラス構造を形成します。 クラスター・クラスが最も一般性が高く、インターフェース・クラスが最も特定度の高いクラスです。

クラスへの関数のグループ化は関数が使用するデータに基づいて行われ、関数は許容される最も一般的なクラスに入れられます。 ネットワーク・インターフェース・クラスとクラスター・クラスの両方に、**CL_getclusterid** という名前の関数があることに注目してください。 名前は同じでもクラスによって区別される、別個の関数が存在します。 この命名規則は C++ で標準的なものです。

すべての値は、参照によって受け渡しされる関数仮パラメーターではなく、通常の関数からの戻り値を使用して呼び出し元に渡されます。 これらの記述が渡されるデータについて述べている場合、このデータはその関数がメンバーになっているオブジェクトから取得されたものであることを暗黙に示します。

状況は常に **CL_status** 引数で返されます。この引数は参照によって受け渡しされ、この引数に対してエラー条件があるかどうかを検査する必要があります。 戻り値は常にデータです。 これについての例外は **CL_isavail()** 関数です。この関数の基本戻りデータは状況であるため、この関数は **CL_status** 引数を使用する代わりに状況を返します。

Clinfo C++ API には、上記で述べたどのクラスにも属さない関数が 2 つあります。 **CL_getallinfo** は、クラスター・オブジェクトではなくクラスターの配列で作用します。 この関数はクラスターの配列を返

し、配列ポインターは参照によって受け渡しされます。 **CL_getlocalid** は、ノード・オブジェクトではなくローカル・ノードで作用します。 ローカル・ホストは、その独自の名前を返します。

アプリケーションでの Clinfo C++ API の使用

このセクションでは、アプリケーションでの Clinfo C++ API の使用方法について説明します。

PowerHA SystemMirror for AIX には、マルチスレッド・アプリケーション用と単一スレッド・アプリケーション用に別々のライブラリーが含まれています。 必ずご使用のアプリケーションに適したライブラリーとリンクしてください。

Clinfo C API へのリンク

C++ プログラムは、適切なリンケージ・ディレクティブを使用して Clinfo C API を呼び出すことができます。

例:

```
extern "C" int cl_getclusterid (char *);
```

詳しくは、「*AIX XL C++/6000 V1.1.1 Language Reference*」を参照してください。

C++ プログラムが Clinfo C API を呼び出せるようにするリンケージ・ディレクティブは **clinfo.h** に含まれています。ただし、よりオブジェクト指向性の高い C++ API が必要な場合は、Clinfo C++ API を使用できます。

Clinfo C++ API は、クラスター、ノード、およびネットワーク・インターフェースのオブジェクト・クラスを定義します。 これらのオブジェクトからデータを取得する Clinfo C++ 関数はすべて、それらのクラスのメンバー関数です (メソッドと呼ばれる場合もあります)。

ヘッダー・ファイル

Clinfo C++ API を使用する各ソース・モジュールに include ディレクティブを指定する必要があります。

この include ディレクティブは次のとおりです。

```
#include <cluster/clinfo.H>
```

libclpp.a および libclpp_r.a ライブラリー

Clinfo C API を使用する単一スレッド・アプリケーションのオブジェクト・ロード・コマンドにディレクティブを追加する必要があります。

このディレクティブは次のとおりです。

```
-lclpp -lcl -lclstr
```

Clinfo C API を使用するマルチスレッド・アプリケーションのオブジェクト・ロード・コマンドに以下のディレクティブを追加する必要があります。

```
-lclpp_r -lcl_r  -lclstr_r
```

libclpp.a および **libclpp_r.a** ライブラリーには、Clinfo C++ API をサポートするルーチンが保持されています。**libcl.a** および **libcl_r.a** ライブラリーには、Clinfo C API をサポートするルーチンが含まれています。

AIX のもとで C++ プログラムをコンパイルするには、xlC コンパイラーを使用します。

注**:** Clinfo の **cluster.es.client.lib** ライブラリーには、32 ビットと 64 ビットの両方のオブジェクトがある **libcl.a** が含まれるようになりました。 Clinfo API を使用する 64 ビット・アプリケーションを作成するには、アプリケーションを 64 ビット環境で再コンパイル/再リンクする必要があります。

定数

Clinfo C++ API ルーチンは、**clinfo.h** ファイルで定義された定数を使用します。

項目	説明
CL_MAXNAMELEN	クラスター、ノード、またはインターフェースの名前を指定する文字ストリングの最大長 (256)。 **CL_MAXNAMELEN** の値は、**sys/param.h** ファイルで定義されている **MAXHOSTNAMELEN** と同じです。
CL_MAX_EN_REQS	許可されるイベント通知要求の最大数 (10)。
CL_ERRMSG_LEN	エラー・メッセージの最大長 (128 文字)。
CLSMUXPD_SVC_PORT	このソフトウェアの前のバージョンで使用されていた定数 **CL_MAXCLUSTERS**、**CL_MAXNODES**、および **CL_MAXNETIFS** は、データ構造内のさまざまな配列の現在のサイズを指定するマクロによって置き換えられます。 これらのマクロには、それぞれが置き換えた定数と同じ名前が付いています。 これらのマクロは宣言ステートメント内で配列境界として使用できません。定数が使用されていたためです。
CL_MAXCLUSTERS	クラスターの数に関する情報を保持している配列の共有メモリー内での現行スペース・サイズを指定します。
CL_MAXNODES	クラスター内のノードの数に関する情報を保持している配列の共有メモリー内での現行スペース・サイズを指定します。
CL_MAXNETIFS	ノードに接続されているインターフェースの数に関する情報を保持している配列の共有メモリー内での現行スペース・サイズを指定します。
CL_MAXGROUPS	リソース・グループの数に関する情報を保持している配列の共有メモリー内での現行スペース・サイズを指定します。

例として、以下のコード・フラグメントでは、クラスター・オブジェクトの配列のサイズを指定するための定数 **CL_MAXNODES** の使用方法を示しています。 その後の例では、これを、同じ名前のルーチンの呼び出しに置き換える方法を示しています。

```
CL_cluster clusters[CL_MAXNODES];
CL_cluster *ret = &clusters[0];
ret = CL_getallinfo(ret, s);
if (s < 0)
cl_errmsg(s);
for (int i=0; i<CL_MAXNODES; i++) {
printf("[%d] cl %d", i, ret[i].clc_clusterid);
printf(" st %d", ret[i].clc_state);
printf(" su %d", ret[i].clc_substate);
printf(" pr %d", ret[i].clc_primary);
printf(" na %s", ret[i].clc_name.name);
}
```

以下は、定数 **CL_MAXNODES** を使用しなくなった例です。

```
CL_cluster clusters[8];
CL_cluster *ret = &clusters[0];
int numclus;
numclus = CL_getallinfo(ret, s);
if (s < 0)
cl_perror(s, progname);
printf("number of clusters found: %d", numclus);
for (int i=0; i<numclus; i++) {
printf("[%d] cl %d", i, ret[i].clc_clusterid);
printf(" st %d", ret[i].clc_state);
```

```
printf(" su %d", ret[i].clc_substate);
printf(" pr %s", ret[i].clc_primary);
printf(" na %s", ret[i].clc_name.name);
}
```

データ型およびデータ構造

Clinfo C++ API は、**clinfo.H** ファイルで定義されているデータ型およびデータ構造を使用します。

clinfo.H ファイルには、**sys/types.h**、**netinet/in.h**、および **clinfo.h** の各ファイルも含まれています。

基本データ型およびクラス定義**:**

このデータ型は、**clinfo.h** ファイルで定義されている C データ型を C++ に変換します。

```
typedef int CL_clusterid;
typedef int CL_nodeid;
typedef int CL_ifid;
typedef struct sockaddr_in CL_ifaddr;
typedef enum cls_state CL_state;
typedef enum clss_substate CL_substate;
typedef int CL_status;
typedef int CL_groupid;
typedef enum cl_rg_policies CL_rg_policies;
typedef enum cl_resource_states CL_resource_states;
class CL_clustername {public: char name[CL_MAXNAMELEN]; };
class CL_nodename {public: char name[CL_MAXNAMELEN]; };
class CL_ifname {public: char name[CL_MAXNAMELEN]; };
class CL_route {
public:
CL_ifaddr localaddr;
CL_ifaddr remoteaddr;
};
class CL_groupname {public: char name[CL_MAXNAMELEN]; };
class CL_user_policy_name {public: char name[CL_MAXNAMELEN]; };
```

クラスター・オブジェクト・クラス**:**

クラスター・クラスのデータおよびメンバー関数:

```
class CL_cluster  {
public:
CL_clusterid clc_clusterid;// Cluster Id
CL_state clc_state;// Cluster State
CL_substate clc_substate;// Cluster Substate
CL_nodename clc_primary; // Cluster Primary Node
CL_clustername clc_name; // Cluster Name
CL_node *clc_node; // Pointer to child node array
CL_group *clc_group; // pointer to child resource group array

int CL_getallinfo(CL_node*, CL_status&);
int CL_getgroupinfo(CL_group*, CL_status&);
CL_clusterid CL_getclusterid(CL_status&);
CL_cluster CL_getinfo(CL_status&);
CL_status CL_getprimary(CL_status&, CL_nodename);
CL_status CL_isavail();
CL_cluster& operator=(const struct cl_cluster&);
};
```

ネットワーク・インターフェース・オブジェクト・クラス**:**

ネットワーク・インターフェース・クラスのデータおよびメンバー関数。

```
class CL_netif  {
 public:
 CL_clusterid cli_clusterid; // Cluster Id
```

```
 CL_nodeid cli_nodeid;  // Cluster node Id
 CL_nodename cli_nodename; // Cluster node name
 CL_ifid cli_interfaceid; // Cluster Node Interface Id
 CL_state cli_state;  // Cluster Node Interface State
 CL_ifname cli_name;  // Cluster Node Interface Name
 CL_ifaddr cli_addr;  // Cluster Node Interface IP Address
 CL_node *cli_pnode;  // pointer to parent Node object
 CL_ifaddr6 cli_addr6;  // Cluster Node Interface IP Address

 CL_clusterid CL_getclusterid(CL_status&);
 CL_clusterid CL_getclusterid6(CL_status&);
 CL_ifaddr CL_getifaddr(CL_status&);
 CL_ifaddr6 CL_getifaddr6(CL_status &s);
  CL_ifname CL_getifname(CL_status&);
 CL_ifname CL_getifname6(CL_status &);
 CL_ifaddr CL_getnodeaddr(CL_status&);
 CL_ifaddr6 CL_getnodeaddr6(CL_status&);
 CL_nodename CL_getnodenamebyif(CL_status&);
 CL_nodename CL_getnodenamebyif6(CL_status &);
 CL_status CL_isavail();
 CL_status CL_isavail6();
 CL_netif& operator=(const struct cl_netif&);
};
```

ノード・オブジェクト・クラス:

ノード・クラスのデータおよびメンバー関数。

```
class CL_node  {
 public:
 CL_clusterid cln_clusterid; // Cluster Id
 CL_nodeid cln_nodeid;         // Cluster node id - used internally
 CL_nodename cln_nodename; // Cluster node name
 CL_state cln_state;  // Cluster Node State
 int cln_nif;   // Cluster Node Number of Interfaces
 CL_netif *cln_if;         // Cluster Node interfaces
 CL_cluster *cln_pcluster; // pointer to parent cluster object

 CL_route CL_bestroute(CL_status&);
 CL_route6 CL_bestroute6(CL_status&);
 CL_node CL_getinfo(CL_status&);
 CL_status CL_isavail();
 CL_node& operator=(const struct cl_node&);
};
```

注: オブジェクト・クラスのデータに含まれる親オブジェクトへのポインターは、オブジェクトのツリー構造をセットアップする場合に提供されます。 これらのポインターは、Clinfo C++ API では埋め込まれません。

リソース・グループ・オブジェクト・クラス:

リソース・グループのデータおよびメンバー関数

```
class CL_group  {
    public:
    CL_clusterid clg_clusterid;  // Cluster Id
    CL_groupid clg_group_id   // Resource Group Id
    CL_groupname clg_name;    // Resource group name

    /* The following field is deprecated in PowerHA SystemMirror 5.2 and will not be
     * used. The data field itself is not removed from the data
     * structures to maintain the backward compatibility */

    CL_rg_policies clg_policy;// Resource Group Policy
    CL_rg_policies clg_startup_policy;
```

```
    CL_rg_policies clg_fallover_policy;
    CL_rg_policies clg_fallback_policy;
    CL_rg_policies clg_site_policy;     // Resource Group site policy
    CL_user_policy_name clg_user_policy_name;
 // User defined policy

    int clg_num_nodes;
    int clg_node_ids[MAXNODES];  // Node ids in this group

    CL_resource_states clg_node_states[MAXNODES];
//and their state
    CL_cluster *cln_pcluster;
// pointer to parent cluster object
    CL_group CL_getinfo(CL_status&);
    CL_group& operator=(const struct cl_group&);
};
```

注: オブジェクト・クラスのデータに含まれる親オブジェクトへのポインターは、オブジェクトのツリー構造をセットアップする場合に提供されます。 これらのポインターは、Clinfo C++ API では埋め込まれません。

どのオブジェクト・クラスにも含まれない関数:

どのオブジェクト・クラスにも含まれない関数があります。

```
int CL_getallinfo (CL_cluster *, CL_status&);
```

この関数は、クラスターの数を返すとともに、特定のクラスター・オブジェクトでなくすべてのクラスターに関する情報も返すため、**CL_cluster** クラスのメンバー関数ではありません。

```
CL_node CL_getlocalid(CL_status&);
```

この関数はローカル・ホストに関する情報を返すため、**CL_node** クラスのメンバー関数ではありません。

クラス CL_netif データの割り当て: cli_addr および cli_name

このコード例は、ネットワーク名およびアドレスを割り当てる方法を示します。

これらの割り当ては、この章の参照ページに含まれている例で使用されています。

cli_addr を割り当てるには、char *addr = "1.2.3.4" では次のようになります。

```
netif.cli_addr.sin_family = AF_INET;
netif.cli_addr.sin_addr.s_addr = inet_addr(addr);
```

cli_name を割り当てる場合は、次のようになります。

```
strcpy(netif.cli_name.name, "node_name");
```

代入演算子の多重定義

クラス **CL_cluster**、**CL_netif**、**CL_group**、および **CL_node** の = 代入演算子は、C 構造 **cl_cluster**、**cl_netif**、**cl_group**、および **cl_node** をそれらに対応する C++ CL_ クラスに割り当てるために多重定義されます。

Clinfo C API を使用して呼び出される関数

以下の関数は、C から C++ に変換されていません。

これらの関数は Clinfo C API を使用して呼び出す必要があります。

イベント関数

```
int cl_registereventnotify(int, struct cli_enr_req_t *);
int cl_unregistereventnotify(int, struct cli_enr_req_t *);
int cl_getevent(cli_en_msg_t *);
```

ユーティリティー関数

```
void cl_perror(int, char *);
char *cl_errmsg (int status);
    /*for single-threaded applications*/
char *cl_errmsg_r(int status, char cbuf);
    /*for multi-threaded applications*/
```

前のリリースの Clinfo からのアプリケーションのアップグレード

前のリリースの Clinfo では、文字ストリング (ノード名) を使用する代わりに、整数を使用してクラスター・ノード (ノード ID) を示していました。

以下のセクションでは、ご使用のアプリケーション内の呼び出しを、以前には *nodeid* パラメーターを使用していた各種 Clinfo C++ API ルーチンに変換する方法の例をいくつか紹介します。 各ルーチンごとに、ノード ID を使用する場合の使用例を示し、その後に、ノード名を使用する例を示します。

CL_getlocalid:

以下は、**CL_getlocalid** ルーチンの、前のリリースの例と新規バージョンの例です。

以下は、ノード ID を使用する、前のリリースの **CL_getlocalid** ルーチンの例です。

```
CL_status s;
CL_node lnode;lnode = node.CL_getlocalid(s);
if (s < 0)
cl_errmsg(s);
printf("cluster id = %d, node id = %d", lnode.cln_clusterid,
lnode.cln_nodeid);
```

C++ API **CL_getlocalid** ルーチンの新規バージョンの例では、**printf** ステートメントが変わっていることに注目してください。

```
// This function is not a member of a class.

CL_status s;
CL_node lnode;
char cbuf[CL_ERRMSG_LEN];
lnode = CL_getlocalid(s);
if (s < 0)
cl_errmsg_r(s, cbuf);
printf("cluster id = %d, node name = %s", lnode.cln_clusterid,
lnode.cln_nodename.name);
```

CL_isavail:

以下は、**CL_isavail** ルーチンの、前のリリースの例と新規バージョンの例です。

以下は、ノード ID を使用する、前のリリースの **CL_isavail** ルーチンの例です。

```
CL_status s;
CL_netif netif;
netif.cli_clusterid = 2;
netif.cli_nodeid = 2;
netif.cli_addr.sin_family = AF_INET;
netif.cli_addr.sin_addr.s_addr = inet_addr(addr);
s = netif.CL_isavail();

printf("status = %d", s);
```

C++ API **CL_isavail** ルーチンの新規バージョンの例では割り当てステートメントが変更されています。前のバージョンでは *cli_nodeid* が使用されていましたが、*cln_name.name* が使用されるようになったためです。

```
CL_status s;
CL_netif netif;
netif.cli_clusterid = 2;
strcpy(netif.cln_name.name, "moby");
netif.cli_addr.sin_family = AF_INET;
netif.cli_addr.sin_addr.s_addr = inet_addr(addr);
s = netif.CL_isavail();

printf("status = %d", s);
```

CL_getprimary

以下は、**CL_getprimary** ルーチンの、前のリリースの例と新規バージョンの例です。

以下は、ノード ID を使用する、前のリリースの **CL_getprimary** ルーチンの例です。

```
CL_cluster clus;
CL_status s;
CL_nodeid nid;
clus.clc_clusterid = 2;
nid = clus.CL_getprimary(s);
if (s < 0)
cl_errmsg(s);
printf("nodeid = %d", nid);
```

C++ API **CL_getprimary** ルーチンの新規バージョンの例では、1 次ノードが整数ではなくその名前 (ストリング) によって認識されるようになったために、変更が行われています。

```
CL_clusterid clusterid;
CL_cluster clus;
CL_status status;
CL_nodename name;
CL_node node;
char cbuf[CL_ERRMSG_LEN];
clus.clc_clusterid = 2;
status = clus.CL_getprimary(cluster.clc_clusterid);
if (status < 0)
cl_errmsg_r(status, cbuf);
printf( "Cluster %d's primary node is %s",
cluster.clc_clusterid, cluster.clc_primary.name);
```

要求

Clinfo C++ API にはさまざまなタイプの要求があります。

クラスター情報要求

以下のクラスター情報要求では、クラスターに関する情報が返されます。

項目	説明
CL_getallinfo	C 関数 **cl_getnodemap** を呼び出します。 この要求は、**cl_cluster** クラスに関連付けられるメンバー関数です。 この要求は、クラスター ID が指定されると、クラスター内のすべてのノードに関する情報を返します。 ノード・オブジェクトの配列へのポインターが、参照引数としてこの関数に渡されます。
CL_getallinfo	C 関数 **cl_getclusters** を呼び出します。 検出されたクラスターの数を返します。 クラスター・オブジェクトの配列へのポインターが、参照引数としてこの関数に渡されます。 (すべてのクラスターに関する情報を取得するために **CL_getallinfo** が呼び出される場合、この関数はクラス **CL_cluster** のメンバー関数ではありません。)
CL_getclusterid	C 関数 **cl_getclusterid** を呼び出します。 クラスター名が指定されると、クラスター ID を返します。
CL_getinfo	C 関数 **cl_getcluster** を呼び出します。 指定されたクラスター ID のクラスターに関する情報を返します。
CL_getprimary	C 関数 **cl_getprimary** を呼び出します。 指定されたクラスター内のユーザー定義の 1 次クラスター・マネージャーのノード名を返します。
CL_isavail	C 関数 **cl_isclusteravail** を呼び出します。 指定されたクラスター ID のクラスターの状況を返します。
CL_getgroupinfo	C 関数 **cl_getgroupmap** を呼び出します。 検出されたリソース・グループの数を返します。 リソース・グループ・オブジェクトの配列へのポインターが、参照オブジェクトとして渡されます。

ノード情報要求

以下のノード情報要求では、クラスター内のノードに関する情報が返されます。

項目	説明
CL_bestroute	C 関数 **cl_bestroute** を呼び出します。 指定されたノードへの最も直接的な経路を示すローカルまたはリモートの IPv4 アドレスのペアを返します。
CL_bestroute6	C 関数 **cl_bestroute6** を呼び出します。 指定されたノードへの最も直接的な経路を示すローカル/リモートの IPv4 または IPv6 アドレスのペアを返します。
CL_getinfo	C 関数 **cl_getnode** を呼び出します。 クラスター ID またはノード名のペアによって指定されたノードに関する情報を返します。
CL_getlocalid	C 関数 **cl_getlocalid** を呼び出します。 要求を発行しているホストのクラスター ID とノード名を含む **CL_node** オブジェクトを返します。 (この関数は、**CL_node** クラスのメンバー関数ではありません。)
CL_isavail	C 関数 **cl_isnodeavail** を呼び出します。 対応するクラスター ID およびノード名が示されると、指定されたノードの状況を返します。

ネットワーク・インターフェース情報要求

以下のネットワーク・インターフェース情報要求では、ノードに接続されているインターフェースに関する情報が返されます。

項目	説明
CL_getclusterid	C 関数 **cl_getclusteridbyifaddr** または **cl_getclusteridbyifname** を呼び出します。 IPv4 アドレスが指定されている場合はクラスターのネットワーク・インターフェース名を返します。名前が指定されている場合は、クラスターの IPv4 ネットワーク・インターフェース・アドレスを返します。
CL_getclusterid6	C 関数 **cl_getclusteridbyifaddr6** または **cl_getclusteridbyifname** を返します。 IPv4 または IPv6 アドレスが指定されている場合はクラスターのネットワーク・インターフェース名を返します。名前が指定されている場合は、クラスターの IPv4 または IPv6 ネットワーク・インターフェース・アドレスを返します。

項目	説明
CL_getifaddr	C 関数 **cl_getifaddr** を呼び出します。 指定されたクラスター ID およびネットワーク・インターフェース名を持つインターフェースの IPv4 ネットワーク・インターフェース・アドレスを返します。
CL_getifaddr6	C 関数 **cl_getifaddr6** を呼び出します。 指定されたクラスター ID およびネットワーク・インターフェース名を持つインターフェースの IPv4 または IPv6 ネットワーク・インターフェース・アドレスを返します。
CL_getifname	C 関数 **cl_getifname** を呼び出します。 指定されたクラスター ID および IPv4 ネットワーク・インターフェース・アドレスを持つインターフェースのインターフェース名を返します。
CL_getifname6	C 関数 **cl_getifname6** を呼び出します。 指定されたクラスター ID および IPv4 または IPv6 ネットワーク・インターフェース・アドレスを持つインターフェースのインターフェース名を返します。
CL_getnodeaddr	C 関数 **cl_getnodeaddr** を呼び出します。 指定されたクラスター ID およびネットワーク・インターフェース名に関連付けられている IPv4 ネットワーク・インターフェース・アドレスを返します。
CL_getnodeaddr6	C 関数 **cl_getnodeaddr6** を呼び出します。指定されたクラスター ID およびネットワーク・インターフェース名に関連付けられている IPv4 および IPv6 ネットワーク・インターフェース・アドレスを返します。
CL_getnodenamebyif	C 関数 **cl_getnodenamebyifaddr** または **cl_getnodenamebyifname** を呼び出します。 IPv4 インターフェース・アドレスがわかっている場合は **cl_getnodenamebyifaddr** を呼び出し、それ以外の場合は **cl_getnodenamebyifname** を呼び出して、ノード名を返します。
CL_getnodenamebyif6	C 関数 **cl_getnodenamebyifaddr6** または **cl_getnodenamebyifname** を呼び出します。 IPv4 または IPv6 インターフェース・アドレスがわかっている場合は **cl_getnodenamebyifaddr6** を呼び出し、それ以外の場合は **cl_getnodenamebyifname** を呼び出して、ノード名を返します。
CL_isavail	C 関数 **cl_isaddravail** を呼び出します。 対応するクラスター ID、ノード名、および IPv4 ネットワーク・インターフェース・アドレスが示されると、指定されたネットワーク・インターフェースの状況を返します。
CL_isavail6	C 関数 **cl_isaddravail6** を呼び出します。対応するクラスター ID、ノード名、および IPv4 または IPv6 ネットワーク・インターフェース・アドレスが示されると、指定されたネットワーク・インターフェースの状況を返します。

リソース・グループ情報要求

以下のリソース・グループ情報要求では、クラスター・リソース・グループに関する情報が返されます。

項目	説明
CL_getinfo	指定されたクラスター内の特定のリソース・グループに関するすべての情報を返します。

イベント通知要求

以下のイベント通知ルーチンでは、クラスター、ノード、またはネットワークのイベントに関する情報が返されます。 これらのルーチンは Clinfo C API から呼び出す必要があります。

項目	説明
cl_getevent	イベント・シグナルの受信時に情報を取得し、Clinfo から受け取ったイベント通知メッセージを返します。
cl_registereventnotify	イベント通知要求のリストを Clinfo に登録し、登録済みイベントの発生時に呼び出しプロセスにシグナルを返します。
cl_unregistereventnotify	イベント通知要求のリストの Clinfo への登録を抹消します。

CL_cluster::CL_getallinfo ルーチン

クラスター内のすべてのノードに関する情報を返します。

構文

```
int *CL_cluster::CL_getallinfo (CL_node *nodes, CL_status s)
```

必須入力オブジェクト・データ

項目	説明
CL_cluster::clc_clusterid	希望するクラスターのクラスター ID。

戻り値

項目	説明
nodes	ノード・オブジェクトの配列へのポインターが、参照引数としてこの関数に渡されます。
int	クラスター内のノードの数。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。

例

```
CL_cluster cluster;
CL_status status;
CL_node nodes[8];
CL_node *ret = &nodes[0];
int numnodes;

cluster.clc_clusterid = 1113325332;
numnodes = cluster.CL_getallinfo(ret, status);
if (status < 0) {
  cl_perror(status, progname);
} else {
  printf("number of nodes in cluster: %d¥n", numnodes);
  for (int i=0; i<numnodes; i++) {
      printf("[%d] clusterid %d", i, ret[i].cln_clusterid);
      printf(" nodeid %d", ret[i].cln_nodeid);
```

```
      printf(" state %d", ret[i].cln_state);
      printf(" interfaces %d¥n", ret[i].cln_nif);
  }
}
```

CL_getlocalid ルーチン

要求を発行しているホストのクラスター ID とノード名を含む **CL_node** オブジェクトを返します。 この要求では、クラスター内で現在アクティブでないノードについては、エラー状況が返されます。

構文

```
CL_node CL_getlocalid (CL_status s)
```

必須入力オブジェクト・データ

なし。

戻り値

項目	説明
CL_node	ローカル・クラスターおよびノード名を持つノード・オブジェクト。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVNODE	ノードがクラスター・ノードではありません。

例

この例では、**cl_errmsg** ルーチンを使用して単一スレッド・アプリケーションの正しいプログラミングを示しています。ご使用のプログラムがマルチスレッドの場合は、**cl_errmsg_r** ルーチンを使用する必要があります。

```
  CL_status status;
  CL_node lnode;

  lnode = CL_getlocalid(status);
  if (status < 0) {
      cl_errmsg(status);
  } else {
      printf("cluster id = %d, node name = %s¥n", lnode.cln_clusterid,
lnode.cln_nodename.name);
  }
```

CL_cluster::CL_getallinfo ルーチン

クラスター内のすべてのノードに関する情報を返します。

構文

```
int *CL_cluster::CL_getallinfo (CL_node *nodes, CL_status s)
```

必須入力オブジェクト・データ

項目	説明
CL_cluster::clc_clusterid	希望するクラスターのクラスター ID。

戻り値

項目	説明
nodes	ノード・オブジェクトの配列へのポインターが、参照引数としてこの関数に渡されます。
int	クラスター内のノードの数。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。

例

```
CL_cluster cluster;
CL_status status;
CL_node nodes[8];
CL_node *ret = &nodes[0];
int numnodes;

cluster.clc_clusterid = 1113325332;
numnodes = cluster.CL_getallinfo(ret, status);
if (status < 0) {
  cl_perror(status, progname);
} else {
  printf("number of nodes in cluster: %d¥n", numnodes);
  for (int i=0; i<numnodes; i++) {
      printf("[%d] clusterid %d", i, ret[i].cln_clusterid);
      printf(" nodeid %d", ret[i].cln_nodeid);
      printf(" state %d", ret[i].cln_state);
      printf(" interfaces %d¥n", ret[i].cln_nif);
  }
}
```

CL_cluster::CL_getclusterid ルーチン

CL_getclusterid ルーチンは、指定された名前のクラスターのクラスター ID を返します。

構文

```
CL_clusterid CL_cluster::CL_getclusterid(CL_status s)
```

必須入力オブジェクト・データ

項目	説明
CL_cluster::clc_name	ターゲット・クラスターの名前。

戻り値

項目	説明
CL_clusterid	目的のクラスター ID。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERNAME	要求は無効なクラスター名を指定しました。

例

```
    CL_cluster cluster;
    CL_status status;
    CL_clusterid clusterid;
    char cbuf[CL_ERRMSG_LEN];

    strcpy(cluster.clc_name.name, "site1");
    clusterid = cluster.CL_getclusterid(status);
    if (status < 0) {
  cl_errmsg(status);
    } else {
  printf("clusterid = %d¥n", clusterid);
    }
}
```

CL_group::CL_getinfo routine

クラスター ID およびグループ名を持つグループ・オブジェクトがある場合、そのグループに関する情報を含むグループ・オブジェクトを返します。

構文

```
CL_Group::CL_getinfo (CL_status s);
```

必須入力オブジェクト・データ

項目	説明
CL_group::clg_clusterid	目的のリソース・グループのクラスター ID。
CL_group::clg_name.name	リソース・グループの名前。

戻り値

項目	説明
CL_group	目的のリソース・グループおよびその完全な情報。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNODENAME	要求は無効なノード名を指定しました。

例

```
CL_status status;
CL_group group;
CL_group ret;

group.clg_clusterid = 1113325332;
strcpy(group.clg_name.name, "rg01");
ret = group.CL_getinfo(status);
if (status < 0) {
  cl_perror(status, progname);
} else {
  printf("There are %d nodes in group %s¥n",
  ret.clg_num_nodes, ret.clg_name.name);
}
```

CL_cluster::CL_getprimary ルーチン

指定されたクラスターのユーザー指定の 1 次クラスター・マネージャーのノード名を返します。

構文

```
CL_nodename CL_cluster::CL_getprimary (CL_status s)
```

必須入力オブジェクト・データ

項目	説明
CL_cluster::clc_clusterid	1 次 ID が要求されているクラスター ID。

戻り値

項目	説明
CL_nodename	1 次クラスター・マネージャーのノード名。

状況値

項目	説明
CL_status status	参照による受け渡しの状況。 戻りコードの出力パラメーター。
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_NOPRIMARY	1 次に関する情報がありません。 この状況は通常、ユーザーが 1 次クラスター・マネージャーを指定していないことを示します。
CLE_IVCLUSTER	そのクラスターは使用できません。

例

```
CL_cluster cluster;
CL_status status;
CL_nodename name;

cluster.clc_clusterid = 1;
name = cluster.CL_getprimary(status);
if (status < 0) {
    cl_errmsg(status);
} else {
    printf( "cluster %d's primary node is %s¥n",
cluster.clc_clusterid, cluster.clc_primary.name);
}
```

CL_cluster::CL_isavail ルーチン

指定されたクラスターが使用可能な場合に状況コード CLE_OK を返します。

構文

```
CL_status CL_cluster::CL_isavail()
```

必須入力オブジェクト・データ

項目	説明
CL_cluster::clc_clusterid	希望するクラスターのクラスター ID。

戻り値

項目	説明
CL_status	指定されたクラスターの状況。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_IVCLUSTER	要求は無効なクラスターを指定しました。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。

例

```
CL_status status;
CL_cluster cluster;

cluster.clc_clusterid = 1113325332;
status = cluster.CL_isavail();
printf("status = %d¥n", status);
```

CL_cluster::CL_getgroupinfo ルーチン

クラスター内のすべてのリソース・グループに関する情報を返します。

構文

```
int CL_cluster::CL_getgroupinfo (CL_group *groups, CL_status&);
```

必須入力オブジェクト・データ

項目	説明
CL_cluster::clc_clusterid	希望するクラスターのクラスター ID。

戻り値

項目	説明
groups	リソース・グループ・オブジェクトのグループへのポインターが、参照引数としてこの関数に渡されます。
int	クラスター内のリソース・グループの数。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。

例

```
CL_status status;
CL_cluster cluster;
CL_group groups[MAXGROUPS];
int numgroups;

cluster.clc_clusterid = 1113325332;
numgroups = cluster.CL_getgroupinfo(&groups[0], status);
if (status < 0) {
  cl_perror(status, progname);
} else {
  printf("There are %d groups in cluster %d¥n", numgroups,
  cluster.clc_clusterid);
  for (int i=0; i<numgroups; i++) {
    printf("Group %d is id %d¥n", i, groups[i].clg_group_id);
    printf("Group %d is %s and has %d nodes¥n",
    i, groups[i].clg_name.name, groups[i].clg_num_nodes);
  }
}
```

CL_group::CL_getinfo routine

クラスター ID およびグループ名を持つグループ・オブジェクトがある場合、そのグループに関する情報を含むグループ・オブジェクトを返します。

構文

```
CL_Group::CL_getinfo (CL_status s);
```

必須入力オブジェクト・データ

項目	説明
CL_group::clg_clusterid	目的のリソース・グループのクラスター ID。
CL_group::clg_name.name	リソース・グループの名前。

戻り値

項目	説明
CL_group	目的のリソース・グループおよびその完全な情報。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNODENAME	要求は無効なノード名を指定しました。

例

```
CL_status status;
CL_group group;
CL_group ret;

group.clg_clusterid = 1113325332;
strcpy(group.clg_name.name, "rg01");
ret = group.CL_getinfo(status);
if (status < 0) {
  cl_perror(status, progname);
} else {
  printf("There are %d nodes in group %s¥n",
  ret.clg_num_nodes, ret.clg_name.name);
}
```

CL_netif::CL_getclusterid ルーチン

指定されたネットワーク・インターフェース・アドレスを持つクラスターの名前を返します。 または、対応するネットワーク・インターフェース名が示されると、クラスターのネットワーク・インターフェース・アドレスを返します。

構文

```
CL_clusterid CL_netif::CL_getclusterid (CL_status s)
```

必須入力オブジェクト・データ

項目	説明
CL_netif::cli_addr	必要なクラスター ID を含むネットワーク・インターフェース・アドレス。

または

項目	説明
CL_netif::cli_name	必要なクラスター ID を含むネットワーク・インターフェース名。

戻り値

項目	説明
CL_clusterid	クラスター ID (負でない整数)。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_SYSERR	システム・エラー。 AIX グローバル **errno** 変数で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVADDRESS	要求は無効なネットワーク・インターフェース・アドレスを指定しました。
CLE_IVNETIFNAME	要求は無効なネットワーク・インターフェース名を指定しました。

例

```
// example using interface name

CL_status status;
CL_clusterid clid;
CL_netif netif;

strcpy(netif.cli_name.name, "geotest9");
netif.cli_addr.sin_addr.s_addr = NULL;
clid = netif.CL_getclusterid(status);
if (status < 0) {
  cl_errmsg(status);
} else {
  printf("clusterid = %d¥n", clid);
}

// example using interface address

  CL_status status;
  CL_clusterid clusterid;
  CL_netif netif;
  char *addr = "1.1.1.7";

  netif.cli_addr.sin_family = AF_INET;
  netif.cli_addr.sin_addr.s_addr = inet_addr(addr);
  netif.cli_name.name[0] = NULL;
  clusterid = netif.CL_getclusterid(status);
  if (status < 0) {
```

```
cl_errmsg(status);
  } else {
printf("clusterid = %d¥n", clusterid);
  }
```

CL_netif::CL_getclusterid6 ルーチン

指定されたネットワーク・インターフェース・アドレスを持つクラスターの名前を返します。 または、対応するネットワーク・インターフェース名が示されると、クラスターのネットワーク・インターフェース・アドレスを返します。 このルーチンは IPv4 アドレスおよび IPv6 アドレスの両方を処理可能です。

構文

```
CL_clusterid CL_netif::CL_getclusterid6(CL_status s)
```

必須入力オブジェクト・データ

項目	説明
CL_netif::cli_addr6	必要なクラスター ID を含むネットワーク・インターフェース・アドレス。

または

項目	説明
CL_netif::cli_name	必要なクラスター ID を含むネットワーク・インターフェース名。

戻り値

項目	説明
CL_clusterid	クラスター ID (負でない整数)。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_SYSERR	システム・エラー。 AIX グローバル **errno** 変数で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVADDRESS	要求は無効なネットワーク・インターフェース・アドレスを指定しました。
CLE_IVNETIFNAME	要求は無効なネットワーク・インターフェース名を指定しました。

例

```
CL_status status;
CL_clusterid clid;
CL_netif netif;

strcpy(netif.cli_name.name, "geotest9");
netif.cli_addr6.sin6_addr.s6_addr = NULL;
clid = netif.CL_getclusterid6(status);
if (status < 0)
{
cl_errmsg(status);
}
else
{
printf("clusterid = %d¥n", clid);
```

```
}

// example using interface address

CL_status status;
CL_clusterid clusterid;
CL_netif netif;
char *addr = "fe80::1";

((struct sockaddr_in6)netif.cli_addr6).sin6_family = AF_INET6;
inet_pton (AF_INET6, addr,
 &(((struct sockaddr_in6 *)&netif.cli_addr6)->sin6_addr));
netif.cli_name.name[0] = NULL; clusterid = netif.CL_getclusterid6(status);
if (status < 0)
{
cl_errmsg(status);
}
else
{
printf("clusterid = %d¥n", clusterid);
}
```

CL_netif::CL_getifaddr ルーチン

指定されたクラスター ID およびネットワーク・インターフェース名を持つインターフェースのネットワーク・インターフェース・アドレスを返します。 このルーチンは IPv4 アドレスのみ処理可能です。

構文

```
CL_ifaddr CL_netif::CL_getifaddr (CL_status s)
```

必須入力オブジェクト・データ

項目	説明
CL_netif::cli_clusterid	希望するネットワーク・インターフェースのクラスター ID。
CL_netif::cli_name	希望するネットワーク・インターフェースの名前。

戻り値

項目	説明
CL_ifaddr	目的のネットワーク・インターフェース・アドレス。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル **errno** 変数で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNETIFNAME	要求は無効なネットワーク・インターフェース名を指定しました。

例

```
CL_status status;
CL_ifaddr ifaddr;
CL_netif netif;
char cbuf[CL_ERRMSG_LEN];
netif.cli_clusterid = 1113325332;
strcpy(netif.cli_name.name, "geotest9");
ifaddr = netif.CL_getifaddr(status);
if (status < 0) {
  cl_errmsg(status);
} else {
  printf("ifaddr = %s¥n", inet_ntoa(ifaddr.sin_addr));
}
```

CL_netif::CL_getifaddr6 ルーチン

指定されたクラスター ID およびネットワーク・インターフェース名を持つインターフェースのネットワーク・インターフェース・アドレスを返します。 このルーチンは IPv4 アドレスおよび IPv6 アドレスの両方を処理可能です。

構文

```
CL_ifaddr6 CL_netif::CL_getifaddr6 (CL_status s)
```

必須入力オブジェクト・データ

項目	説明
CL_netif::cli_clusterid	希望するネットワーク・インターフェースのクラスター ID。
CL_netif::cli_name	希望するネットワーク・インターフェースの名前。

戻り値

項目	説明
CL_ifaddr6	目的のネットワーク・インターフェース・アドレス。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル **errno** 変数で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNETIFNAME	要求は無効なネットワーク・インターフェース名を指定しました。

例

```
CL_status status;
CL_ifaddr6 ifaddr;
CL_netif netif;
char cbuf[CL_ERRMSG_LEN];
char *addr;
netif.cli_clusterid = 1113325332;
strcpy(netif.cli_name.name, "geotest9");
ifaddr = netif.CL_getifaddr6(status);
```

```
if (status < 0)
{
cl_errmsg(status);
}
else
{
printf("ifaddr = %s¥n", inet_ntop(AF_INET6,
&((struct sockaddr_in6 *)&ifaddr->sin6_addr), addr, INET6_ADDRSTRLEN);
}
```

CL_netif::CL_getifname ルーチン

クラスター ID およびネットワーク・インターフェース・アドレスが示されると、ネットワーク・インターフェース名を返します。または、クラスター ID およびノード名が示されると、ネットワーク・インターフェース名を返します。 このルーチンは IPv4 アドレスのみ処理可能です。

要求で **cli_addr** パラメーターが指定されると、 Clinfo はアドレスのネットワーク部分を調べ、同じネットワーク上でインターフェースを探します。 一致が見つかると、**CL_getifname** ルーチンはそのインターフェースに関連付けられている名前を返します。

cli_addr パラメーターが NULL の場合、Clinfo は指定されたノード上で、ローカル・ホストから最も簡単にアクセスできるインターフェースを選択し、**CL_getifname** ルーチンはそのインターフェースと関連付けられている名前を返します。どちらのインターフェースでもローカル・ノードからのアクセスのしやすさが同じ場合、Clinfo は一方を選び、そのインターフェースに関連付けられている名前を返します。

どのケースでも、**CL_getifname** ルーチンは名前をヌル終了文字列として返します。

構文

```
CL_ifname CL_netif::CL_getifname (CL_status s)
```

必須入力オブジェクト・データ

項目	説明
CL_netif::cli_clusterid, cli_addr	目的のネットワーク・インターフェースのクラスター ID およびネットワーク・インターフェース・アドレス。

または

項目	説明
CL_netif::cli_clusterid, cli_nodename	目的のネットワーク・インターフェースのクラスター ID およびノード名。

戻り値

項目	説明
CL_ifname	ネットワーク・インターフェース名。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル **errno** 変数で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVADDRESS	要求は無効なネットワーク・インターフェース・アドレスを指定しました。
CLE_IVNODENAME	要求は無効なノード名を指定しました。

例

```
// CL_netif::CL_getifname get interfacename given clusterid and nodename

  CL_status status;
  char cbuf[CL_ERRMSG_LEN];
  CL_ifname ifname;
  CL_netif netif;

  netif.cli_addr.sin_addr.s_addr = NULL;
  netif.cli_clusterid = 1113325332;
  strcpy (netif.cli_nodename.name, "node1");
  ifname = netif.CL_getifname(status);
  if (status < 0) {
     cl_errmsg(status);
  } else {
     printf("ifname = %s¥n", ifname.name);
  }
```

CL_netif::CL_getifname6 ルーチン

クラスター ID およびネットワーク・インターフェース・アドレスが示されると、ネットワーク・インターフェース名を返します。または、クラスター ID およびノード名が示されると、ネットワーク・インターフェース名を返します。 このルーチンは IPv4 アドレスおよび IPv6 アドレスの両方を処理可能です。

要求で **cli_addr6** パラメーターが指定されると、 Clinfo はアドレスのネットワーク部分を調べ、同じネットワーク上でインターフェースを探します。 一致が見つかると、**CL_getifname6** ルーチンはそのインターフェースに関連付けられている名前を返します。

cli_addr6 パラメーターが NULL の場合、Clinfo は指定されたノード上で、ローカル・ホストから最も簡単にアクセスできるインターフェースを選択し、**CL_getifname6** ルーチンはそのインターフェースと関連付けられている名前を返します。どちらのインターフェースでもローカル・ノードからのアクセスのしやすさが同じ場合、Clinfo は一方を選び、そのインターフェースに関連付けられている名前を返します。

どのケースでも、**CL_getifname6** ルーチンは名前をヌル終了文字列として返します。

構文

```
CL_ifname CL_netif::CL_getifname6 (CL_status s)
```

必須入力オブジェクト・データ

項目	説明
CL_netif::cli_clusterid, cli_addr6	目的のネットワーク・インターフェースのクラスター ID およびネットワーク・インターフェース・アドレス。

または

項目	説明
CL_netif::cli_clusterid, cli_nodename	目的のネットワーク・インターフェースのクラスター ID およびノード名。

戻り値

項目	説明
CL_ifname	ネットワーク・インターフェース名。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル **errno** 変数で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVADDRESS	要求は無効なネットワーク・インターフェース・アドレスを指定しました。
CLE_IVNODENAME	要求は無効なノード名を指定しました。

例

```
// CL_netif::CL_getifname get interfacename given clusterid and nodename

CL_status status;
char cbuf[CL_ERRMSG_LEN];
CL_ifname ifname;
CL_netif netif;

netif.cli_addr6.sin6_addr.s6_addr = NULL;
netif.cli_clusterid = 1113325332;
strcpy (netif.cli_nodename.name, "node1");
ifname = netif.CL_getifname6(status);
if (status < 0)
{
cl_errmsg(status);
}
else
{
printf("ifname = %s¥n", ifname.name);
}
```

CL_netif::CL_getnodeaddr ルーチン

指定されたクラスター ID およびネットワーク・インターフェース名に関連付けられている IP アドレスを返します。 このルーチンは IPv4 アドレスのみ処理可能です。

構文

```
CL_ifaddr CL_netif::CL_getnodeaddr(CL_status s)
```

必須入力オブジェクト・データ

項目	説明
CL_netif::cli_clusterid	希望するネットワーク・インターフェースのクラスター ID。
CL_netif::cli_name	希望するネットワーク・インターフェースの名前。

戻り値

項目	説明
CL_ifaddr	ネットワーク・インターフェース・アドレス。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル **errno** 変数で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNETIFNAME	要求は無効なネットワーク・インターフェース名を指定しました。

例

```
CL_status status;
CL_netif netif;
CL_ifaddr ifaddr;
char cbuf[CL_ERRMSG_LEN];

netif.cli_clusterid = 1113325332;
strcpy(netif.cli_name.name, "geotest9");
ifaddr = netif.CL_getnodeaddr(status);
if (status < 0) {
  cl_errmsg(status);
} else {
  printf("ifaddr = %s¥n", inet_ntoa(ifaddr.sin_addr));
}
```

CL_netif::CL_getnodeaddr6 ルーチン

指定されたクラスター ID およびネットワーク・インターフェース名に関連付けられている IP アドレスを返します。 このルーチンは IPv4 アドレスおよび IPv6 アドレスの両方を処理可能です。

構文

```
CL_ifaddr6 CL_netif::CL_getnodeaddr6(CL_status s)
```

必須入力オブジェクト・データ

項目	説明
CL_netif::cli_clusterid	希望するネットワーク・インターフェースのクラスター ID。
CL_netif::cli_name	希望するネットワーク・インターフェースの名前。

戻り値

項目	説明
CL_ifaddr6	ネットワーク・インターフェース・アドレス。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル **errno** 変数で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNETIFNAME	要求は無効なネットワーク・インターフェース名を指定しました。

例

```
CL_status status;
CL_netif netif;
CL_ifaddr6 ifaddr;
char cbuf[CL_ERRMSG_LEN];
char *addr;

netif.cli_clusterid = 1113325332;
strcpy(netif.cli_name.name, "geotest9");
ifaddr = netif.CL_getnodeaddr6(status);
if (status < 0)
{
cl_errmsg(status);
}
else
{
printf("ifaddr = %s¥n", inet_ntop(AF_INET6, &((struct sockaddr_in6 *)
&ifaddr->sin6_addr), addr, INET6_ADDRSTRLEN);
}
```

CL_netif::CL_getnodenamebyif ルーチン

クラスター ID とネットワーク・インターフェース・アドレス、またはクラスター ID とネットワーク・インターフェース名が示されると、ノード名を返します。

ネットワーク・インターフェース・アドレスが指定されていて、**cli_name** が NULL の場合、**cli_name** が返されます。 逆に、**cli_name** が指定されていて、**cli_addr** が NULL の場合は、**cli_addr** が返されます。 **cli_name** と **cli_addr** が両方とも NULL 以外の場合は、**cli_addr** が優先されます。 両方とも NULL の場合は、エラー・コード CLE_ BADARGS が返されます。

注**:** このルーチンは、前のリリースで提供されていた **CL_getnodename** ルーチンを置き換えます。

構文

```
CL_nodename CL_netif::CL_getnodenamebyif (CL_status s)
```

必須入力オブジェクト・データ

項目	説明
CL_netif::cli_clusterid, cli_addr	目的のノードのクラスター ID、およびノードのネットワーク・インターフェース・アドレス。

または

項目	説明
CL_netif::cli_clusterid, cli_name	目的のノードのクラスター ID、およびネットワーク・インターフェースの名前。

戻り値

項目	説明
CL_nodename	ノード名。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル **errno** 変数で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVADDRESS	要求は無効なネットワーク・インターフェース・アドレスを指定しました。
CLE_IVNETIFNAME	要求は無効なネットワーク・インターフェース名を指定しました。

例

```
CL_status status;
char cbuf[CL_ERRMSG_LEN];
CL_nodename nname;
CL_netif netif;
char *addr = "9.57.28.23";

netif.cli_clusterid = 1113325332;
netif.cli_addr.sin_family = AF_INET;
netif.cli_addr.sin_addr.s_addr = inet_addr(addr);
netif.cli_name.name[0] = NULL;
nname = netif.CL_getnodenamebyif(status);
if (status < 0) {
   cl_errmsg(status);
} else {
   printf("node name = %s¥n", nname.name);
}
```

CL_netif::CL_getnodenamebyif6 ルーチン

クラスター ID とネットワーク・インターフェース・アドレス、またはクラスター ID とネットワーク・インターフェース名が示されると、ノード名を返します。 このルーチンは IPv4 アドレスおよび IPv6 アドレスの両方を処理可能です。

ネットワーク・インターフェース・アドレスが指定されていて、**cli_name** が NULL の場合、**cli_name** が返されます。 逆に、**cli_name** が指定されていて、**cli_addr6** が NULL の場合は、**cli_addr6** が返されます。 **cli_name** と **cli_addr6** が両方とも NULL 以外の場合は、**cli_addr6** が優先されます。 両方とも NULL の場合は、エラー・コード CLE_ BADARGS が返されます。

注**:** このルーチンは、前のリリースで提供されていた **CL_getnodename** ルーチンを置き換えます。

構文

```
CL_nodename CL_netif::CL_getnodenamebyif6 (CL_status s)
```

必須入力オブジェクト・データ

項目	説明
CL_netif::cli_clusterid, cli_addr6	目的のノードのクラスター ID、およびノードのネットワーク・インターフェース・アドレス。

または

項目	説明
CL_netif::cli_clusterid, cli_name	目的のノードのクラスター ID、およびネットワーク・インターフェースの名前。

戻り値

項目	説明
CL_nodename	ノード名。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル **errno** 変数で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVADDRESS	要求は無効なネットワーク・インターフェース・アドレスを指定しました。
CLE_IVNETIFNAME	要求は無効なネットワーク・インターフェース名を指定しました。

例

```
CL_status status;
char cbuf[CL_ERRMSG_LEN];
CL_nodename nname;
CL_netif netif;
char *addr = "fe80::1";

netif.cli_clusterid = 1113325332;
netif.cli_addr6.sin6_family = AF_INET6;
inet_pton (AF_INET6, addr,
 &(((struct sockaddr_in6 *)&netif.cli_addr6)->sin6_addr));
netif.cli_name.name[0] = NULL;
```

```
nname = netif.CL_getnodenamebyif6(status);
if (status < 0)
{
cl_errmsg(status);
}
else
{
printf("node name = %s¥n", nname.name);
}
```

CL_netif::CL_isavail ルーチン

指定されたネットワーク・インターフェースが使用可能な場合に状況コード CLE_OK を返します。 このルーチンは IPv4 アドレスのみ処理可能です。

構文

```
CL_status CL_netif::CL_isavail()
```

必須入力オブジェクト・データ

項目	説明
CL_netif::cli_clusterid	希望するネットワーク・インターフェースのクラスター ID。
CL_netif::cli_nodename	希望するネットワーク・インターフェースのノード名。
CL_netif::cli_addr	希望するネットワーク・インターフェースのアドレス。

戻り値

項目	説明
CL_status	指定されたネットワーク・インターフェースの状況。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル **errno** 変数で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNODENAME	要求は無効なノード名を指定しました。
CLE_IVADDRESS	要求は無効なインターフェース・アドレスを指定しました。
CLE_IVNETIF	ネットワーク・インターフェースは使用可能ではありません。

例

```
CL_status status;
CL_netif netif;
char *addr = "9.57.28.23";

netif.cli_clusterid = 1113325332;
strcpy(netif.cli_name.name, "geotest9");
netif.cli_addr.sin_family = AF_INET;
netif.cli_addr.sin_addr.s_addr = inet_addr(addr);
strcpy(netif.cli_nodename.name, "node1");
status = netif.CL_isavail();
```

```
if (status < 0) {
     cl_perror(status,"netif.CL_isavail failed");
}
printf("status = %d¥n", status);
```

CL_netif::CL_isavail6 ルーチン

指定されたネットワーク・インターフェースが使用可能な場合に状況コード CLE_OK を返します。 このルーチンは IPv4 アドレスおよび IPv6 アドレスの両方を処理可能です。

構文

```
CL_status CL_netif::CL_isavail6()
```

必須入力オブジェクト・データ

項目	説明
CL_netif::cli_clusterid	希望するネットワーク・インターフェースのクラスター ID。
CL_netif::cli_nodename	希望するネットワーク・インターフェースのノード名。
CL_netif::cli_addr6	希望するネットワーク・インターフェースのアドレス。

戻り値

項目	説明
CL_status	指定されたネットワーク・インターフェースの状況。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル **errno** 変数で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNODENAME	要求は無効なノード名を指定しました。
CLE_IVADDRESS	要求は無効なインターフェース・アドレスを指定しました。
CLE_IVNETIF	ネットワーク・インターフェースは使用可能ではありません。

例

```
CL_status status;
CL_netif netif;
char *addr = "fe80::1";

netif.cli_clusterid = 1113325332;
strcpy(netif.cli_name.name, "geotest9");
netif.cli_addr6.sin6_family = AF_INET6;
inet_pton (AF_INET6, addr,
 &(((struct sockaddr_in6 *)&netif.cli_addr6)->sin6_addr));
strcpy(netif.cli_nodename.name, "node1");
status = netif.CL_isavail6();
if (status < 0)
{
cl_perror(status,"netif.CL_isavail6 failed");
}
printf("status = %d¥n", status);
```

CL_node::CL_bestroute ルーチン

CL_bestroute ルーチンは、オブジェクトで指定されたノードへの最も直接的な経路を示すローカル/リモートの IP アドレスのペアを返します。 このルーチンは IPv4 アドレスのみ処理可能です。

CL_bestroute ルーチンによって返される経路は、その要求を発行しているノードによって異なります。Clinfo は最初にローカル・ノード上の正常なすべてのネットワーク・インターフェースのリストを作成し、その後に、このリストを指定されたノード上の使用可能インターフェースと比較します。 このルーチンは最初に、PowerHA SystemMirror によって定義されたプライベート・インターフェース (光シリアル・チャネルなど) をローカル・インターフェースと比較します。 一致が見つからない場合、このルーチンは PowerHA SystemMirror によって定義されたパブリック・インターフェースをローカル・インターフェースと比較します。 それでも一致がない場合、このルーチンはローカル・ノード上の最初の定義済みインターフェースと、リモート・ノード上の最初の定義済みインターフェースを選択します。

同じネットワーク上にローカルとリモートのインターフェースのペアが存在する場合、これらは **CL_route** に含めて返されます。 そうでない場合、指定されたノード上のインターフェースがリモート・インターフェースとして選択され、1 次ローカル・インターフェースが経路のローカル・エンドとして返されます。

構文

```
CL_route CL_node::CL_bestroute(CL_status s)
```

必須入力オブジェクト・データ

項目	説明
CL_node::cln_clusterid, cln_nodename	ターゲット・ノードのクラスター ID およびノード名。

戻り値

項目	説明
CL_route	指定されたノードへの最も直接的な経路を示すローカル/リモートの IP アドレスのペア。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_OK	要求は正常に完了しました。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_NOROUTE	使用できる経路がありません。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNODENAME	要求は無効なノード名を指定しました。

例

```
CL_status status;
CL_node node;
CL_route route;
char cbuf[CL_ERRMSG_LEN];

node.cln_clusterid = 1113325332;
strcpy(node.cln_nodename.name, "node1");
route = node.CL_bestroute(status);
```

```
if (status < 0) {
  cl_errmsg(status);
} else {
  // don't call inet_ntoa twice in one printf!
  printf("local = %s  ", inet_ntoa(route.localaddr.sin_addr));
  printf("remote = %s¥n", inet_ntoa(route.remoteaddr.sin_addr));
}
```

CL_node::CL_bestroute6 ルーチン

CL_bestroute6 ルーチンは、オブジェクトで指定されたノードへの最も直接的な経路を示すローカルまたはリモートの IP アドレスのペアを返します。 このルーチンは IPv4 アドレスおよび IPv6 アドレスの両方を処理可能です。

CL_bestroute6 ルーチンによって返される経路は、その要求を発行しているノードによって異なります。Clinfo は最初にローカル・ノード上の正常なすべてのネットワーク・インターフェースのリストを作成し、その後に、このリストを指定されたノード上の使用可能インターフェースと比較します。 このルーチンは最初に、PowerHA SystemMirror によって定義されたプライベート・インターフェース (光シリアル・チャネルなど) をローカル・インターフェースと比較します。 一致が見つからない場合、このルーチンは PowerHA SystemMirror によって定義されたパブリック・インターフェースをローカル・インターフェースと比較します。 それでも一致がない場合、このルーチンはローカル・ノード上の最初の定義済みインターフェースと、リモート・ノード上の最初の定義済みインターフェースを選択します。

同じネットワーク上にローカルとリモートのインターフェースのペアが存在する場合、これらは **CL_route6** に含めて返されます。 そうでない場合、指定されたノード上のインターフェースがリモート・インターフェースとして選択され、1 次ローカル・インターフェースが経路のローカル・エンドとして返されます。

構文

```
CL_route CL_node::CL_bestroute6(CL_status s)
```

必須入力オブジェクト・データ

項目	説明
CL_node::cln_clusterid, cln_nodename	ターゲット・ノードのクラスター ID およびノード名。

戻り値

項目	説明
CL_route6	指定されたノードへの最も直接的な経路を示すローカルまたはリモートの IP アドレスのペア。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_OK	要求は正常に完了しました。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_NOROUTE	使用できる経路がありません。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNODENAME	要求は無効なノード名を指定しました。

例

```
CL_status status;
CL_node node;
CL_route6 route;
char cbuf[CL_ERRMSG_LEN];
char *addr;

node.cln_clusterid = 1113325332;
strcpy(node.cln_nodename.name, "node1");
route = node.CL_bestroute6(status);
if (status < 0)
{
 cl_errmsg(status);
}
else
{
 // don't call inet_ntop twice in one
 printf("local = %s ", inet_ntop(AF_INET6,
 &(((struct sockaddr_in6 *)&(route.localaddr))->sin6_addr), addr, INET6_ADDRSTRLEN);
 printf("remote = %s¥n", inet_ntop(AF_INET6,
 &(((struct sockaddr_in6 *)&(route.remoteaddr))->sin6_addr), addr,  INET6_ADDRSTRLEN);
}
```

CL_node::CL_getinfo ルーチン

クラスター ID およびノード名を持つノード・オブジェクトがある場合、そのノードに関する情報を含むノード・オブジェクトを返します。

構文

```
CL_node CL_node::CL_getinfo(CL_status s)
```

必須入力オブジェクト・データ

項目	説明
CL_node::cln_clusterid	ターゲット・ノードのクラスター ID。
CL_node::cln_nodename	ターゲット・ノードのノード名。

戻り値

項目	説明
CL_node	目的のノードとその情報。

状況値

項目	説明
CL_status s	参照による受け渡しの状況。 戻りコードを保持する出力パラメーター。
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_BADARGS	欠落している、または無効なパラメーター。 この状況は通常、出力パラメーター・アドレスに NULL ポインターが指定されたことを示します。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNODENAME	要求は無効なノード名を指定しました。

例

```
CL_status status;
CL_node node;
CL_node ret;
char cbuf[CL_ERRMSG_LEN];

node.cln_clusterid = 1113325332;
strcpy(node.cln_nodename.name, "node1");
ret = node.CL_getinfo(status);
if (status < 0) {
    cl_errmsg(status);
} else {
    printf("clusterid %d  ", ret.cln_clusterid);
    printf("nodename %s  ", ret.cln_nodename.name);
    printf("state %d  ", ret.cln_state);
    printf("nif %d¥n", ret.cln_nif);
}
```

CL_node::CL_isavail ルーチン

指定されたノードが使用可能な場合に状況コード CLE_OK を返します。

構文

```
CL_status CL_node::CL_isavail()
```

必須入力オブジェクト・データ

項目	説明
CL_node::cln_clusterid	希望するノードのクラスター ID。
CL_node::cln_nodename	目的のノードのノード名。

戻り値

項目	説明
CL_status	指定されたノードの状況。

状況コード

項目	説明
CLE_OK	要求は正常に完了しました。
CLE_SYSERR	システム・エラー。 AIX グローバル変数 **errno** で追加情報を確認してください。
CLE_IVCLUSTERID	要求は無効なクラスター ID を指定しました。
CLE_IVNODENAME	要求は無効なノード名を指定しました。
CLE_IVNODE	このノードは使用できません。

例

```
CL_status status;
CL_node node;

node.cln_clusterid = 1113325332;
strcpy(node.cln_nodename.name, "node1");
status = node.CL_isavail();
printf("status = %d¥n", status);
```

Clinfo クライアント・プログラムの例

このセクションでは、**clinfo.rc** スクリプトの例と、そのスクリプトから呼び出される C プログラムのソース・コードをリストします。 このプログラムは、所定のクラスター・ノード上の各サービス・ネットワーク・インターフェースの状況と、ノード自体の状況を報告します。

カスタマイズされた clinfo.rc スクリプトの例

ここでは、典型的なカスタマイズ済み **clinfo.rc** スクリプトのコンテキスト内での Clinfo クライアント・アプリケーション・プログラム、 **cl_status** の例を紹介します。 **clinfo.rc** は、クラスターのトポロジー変更後に Clinfo によって実行される PowerHA SystemMirror for AIX スクリプトです。 このスクリプトとプログラムは、その使用法を説明するためにコメント化されています。

```
#!/bin/ksh
################################################################################
# Filename: /usr/sbin/cluster/etc/clinfo.rc
#
# Description: clinfo.rc is run by clinfo on clients following cluster
#     topology changes. This particular example demonstrates
#     user process management for a highly available database in a
#     two-node primary/standby configuration. Most database
#     client programs are state-dependent, and require
      #restart following a node failure. This example provides
#     user notification and application shutdown during
#     appropriate topology changes.
#
################################################################################
################################################################################
# Grab Parameters Passed
################################################################################
EVENT=$1 # action, one of {join, fail, swap}
INTERFACE=$2 # target address label
CLUSTERNAME="cluster1" # cluster name
NODENAME="victor"# primary node name
WATCHIF="svc_en0"# interface to monitor

################################################################################
#   Name:  _arp_flush
#   This function flushes the entire arp cache.
#   Arguments:  none
#   Return value:  none
################################################################################
_arp_flush()
{
  for IPADDR in $(/etc/arp -a |/bin/sed -e 's/^.*(.*).*$//' -e /incomplete/d)
    do
 /etc/arp -d $IPADDR
    done
}

################################################################################
#
#   Name:  _kill_user_procs
#
#   This function kills user processes associated with the specified
#   interface.
#
#   Arguments:  interface
#   Return value:  none
################################################################################
_kill_user_procs()
{
    print _kill_user_procs
  # place commands appropriate to the database in use here
```

```
}
# The main if statement disregards status changes for all interfaces except
# WATCHIF, which in this example is svc_en0.
if [[ "$INTERFACE" = "WATCHIF" ]]
then
  case "$EVENT" in
    "join") # interface label $INTERFACE has joined the cluster
# perform necessary activity here, such as user notification, restoration of
# user access, and arp cache flushing.
    exit 0
    ;;

    "fail")# Use api calls in cl_status to determine if interface
    # failure is a result of node failure.

  CLSTAT_MSG=$(cl_status $CLUSTERNAME $NODENAME)
  CLSTAT_RETURN=$?  # return code from cl_status

  case "$CLSTAT_RETURN" in
    0)      # Node UP
 # Notify users of application availability
      wall "Primary database is now available."
  # flush arp cache
      _arp_flush
;;

    1)      # Node DOWN
      # Notify users of topology change and restart requirement
      touch /etc/nologin    # prevent new logins
      wall "Primary database node failure. Please login again
  2 minutes"
  sleep 10
      # Kill all processes attached to WATCHIF interface
      _kill_user_procs $WATCHIF
      # flush arp cache
_ arp_flush
  rm -f /etc/nologin     # enable logins
  ;;

    *)  # Indeterminate node state
 # flush arp cache
      _arp_flush
exit 1
;;

  esac  # case $CLSTAT_RETURN
  ;;
    "swap")# interface has been swapped
    # flush arp cache.
      _arp_flush
    ;;

  esac      # case $EVENT
else
    # event handling for other interfaces here, if desired
    /bin/true
fi
```

cl_status.c サンプル・プログラム

これはサンプルの c プログラムです。

```
/*
 * Program:  cl_status.c
 *
 * Purpose:  For systems running the clinfo daemon as a client, cl_status
```

```
 * will determine if the node for the network interface passed
 * to it is active in the cluster.
 *
 * Usage:    [path/]cl_status clustername nodename
 *
 * Returns:  0 = Node up
 * 1 = Node down
 * 2 = ERROR - Status Unavailable
 *
 */
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <cluster/clinfo.h>
#include <strings.h>
void usage()
{
  printf("usage: cl_status clustername nodename");
  printf("Returns status of node in PowerHA SystemMirror cluster.");
}
int main(int argc, char *argv[])
{
  int clusterid, node_status;
  char *clustername, *nodename;

  if(argc < 3)
  {
    /* incorrect syntax to cl_status call */
    usage();
    exit(2);
  }
  clustername = strdup(argv[1]);
  if (strlen(clustername) > CL_MAXNAMELEN)
  {
    printf("error: clustername exceeds maximum length of %i characters",
     CL_MAXNAMELEN);
    exit(2);
  }
  nodename = strdup(argv[2]);
  if (strlen(nodename) > CL_MAXNAMELEN)
  {
    printf("error: nodename exceeds maximum length of %i characters",
     CL_MAXNAMELEN);
    exit(2);
  }
  /* convert clustername (string) to clusterid (non-negative integer) */
  clusterid = cl_getclusterid(clustername);
  switch(clusterid)
  {
    case CLE_SYSERR: perror("system error");
      exit(5);
      break;

    case CLE_NOCLINFO: cl_perror(clusterid, "error");
      exit(5);
      break;

    case CLE_BADARGS:
    case CLE_IVCLUSTERNAME: /* typically a usage error */
 cl_perror(clusterid, "error");
 usage();
  exit(2);

    default:    /* valid clusterid returned */
     ;
  }
```

```
  node_status = cl_isnodeavail(clusterid, nodename);
  switch (node_status)
  {
    case CLE_OK: /* Node up */

    printf("node %s up", nodename);
exit(0);
break;

case CLE_IVNODENAME:  /* "Illegal node name" */
      cl_perror(node_status, "node unavailable");
      exit(2);
      break;

    default:
      cl_perror(node_status, "node unavailable");
      exit(1);
  }
}
```

実装の詳細

Clinfo には、**clinfo** デーモンと API ライブラリーの 2 つのキー・コンポーネントがあります。

clinfo デーモンは SNMP ベースのモニターです。 SNMP は、TCP/IP ベースのネットワークのモニターおよび管理に関する業界全体の標準セットです。 SNMP には、プロトコルとデータベース仕様書が 1 つずつ、および複数のデータ・オブジェクト・セットが含まれています。

これらのデータ・オブジェクト・セットが管理情報ベース (MIB) を形成します。 SNMP は、IP アドレスやアクティブ TCP 接続数などの情報を含む標準 MIB を提供します。実際の MIB 定義は、システム上で稼働するエージェント内にエンコードされます。 AIX 内の標準 SNMP エージェントは SNMP デーモン、**snmpd** です。

プログラマーは SNMP 演算を使用してネットワークをモニターおよび管理するプログラムを実装します。これらのプログラムは **snmpd** からネットワークの状態に関する情報を受け取り、その情報をクライアントとアプリケーションに受け渡します。

SNMP は、SNMP 多重化 (SMUX) プロトコルを使用して、個別の環境またはアプリケーションに関連する情報を含むエンタープライズ固有 の MIB を含めるように拡張できます。管理エージェント (SMUX ピア・デーモン) はその MIB 内で定義されているオブジェクトに関する情報を取得および維持し、この情報を専門化されたネットワーク・モニター・ステーションまたはネットワーク管理ステーションに提供します。

PowerHA SystemMirror ソフトウェアは、クラスター・マネージャー・デーモンを通じてこの SMUX ピア機能を提供します。 **clinfo** デーモンは、クラスター・マネージャーを介して PowerHA SystemMirror MIB から (間接的に) この情報を取得します。

(このソフトウェアのさまざまな種類のライブラリーの中でも) Clinfo API ライブラリーは、**clinfo** デーモンと対話してクラスター情報へのアクセスを可能にします。 同じ情報は SNMP を通じて直接入手できますが、Clinfo ライブラリーは大幅に簡略化されたプログラミング・モデルを提供して、クライアント・プログラムが複雑な SNMP API を使用しなくて済むようにします。 Clinfo API は、ノードやリソース・グループなどのクラスター・エンティティーに関連するすべての情報を取得するためのルーチン (SNMP ではこれらの項目を一度に 1 つずつフェッチする必要があります) を提供し、さらに、特定のクラスター・イベントを登録するためのルーチン (SNMP では、同様の機能を実装するためにトラップが必要です)

を提供します。 さまざまな種類のライブラリー (C、C++、スレッド・セーフなど) があることで、各種のランタイム環境に一貫性のあるモデルを提供します。

clinfo デーモンおよびライブラリーは、PowerHA SystemMirror クラスター・ノード上で実行できます。またはそのデーモンが TCP/IP を介してクラスター・ノード上の SNMP にアクセスできる場合は非クラスター・ノード上でも実行できます。

Cluster Manager および Clinfo

クラスター・マネージャー・デーモン **(clstrmgr)** は、クラスターをモニターし、必要な場合にリカバリー・アクションを開始する PowerHA SystemMirror サブシステムです。 クラスター・マネージャーはクラスターの動作について報告し、他のプログラムはそれによってクラスター内で変更が行われたかどうかを判別し、必要な場合はその変更に対応します。

クラスター・マネージャーは、クラスター情報を取得した後、イベントとイベントの結果のクラスターの状態をトラッキングしながら、PowerHA SystemMirror for AIX の MIB 内のクラスターのトポロジーを更新し、そのトポロジーを維持します。 Clinfo は、クライアント・マシンまたはクラスター・ノード上で実行され、**MIB** を照会して更新されたクラスター情報があるか確認し、アプリケーションがアプリケーション・プログラミング・インターフェースを通じて PowerHA SystemMirror for AIX の MIB 情報にアクセスできるようにします。

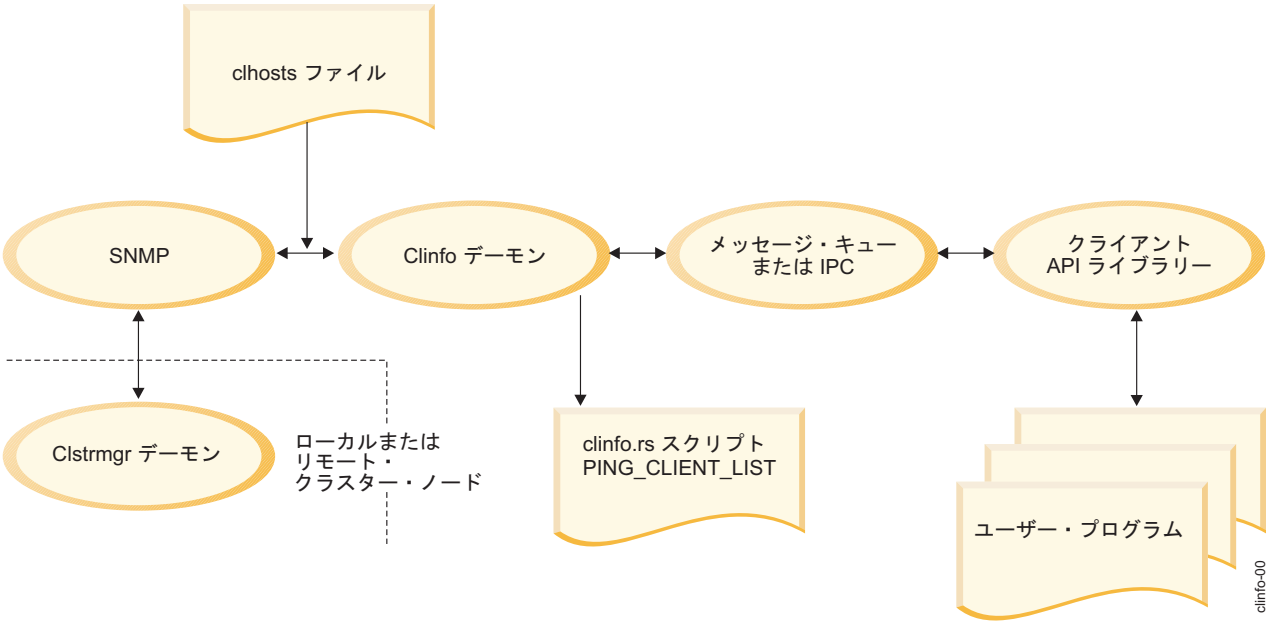
デフォルトでは、Clinfo は定期的 (15 秒ごと) に **SNMP** プロセスをポーリングして、イベントで更新された情報があるか確認します。 Clinfo はオプション **(-a)** を使用して開始できます。このオプションにより、Clinfo はイベントが発生してすぐにこの情報を受け取ることができます。 この場合、クラスター・マネージャーはイベント情報を受け取るとトラップ・メッセージを送信します。 この後、Clinfo は直ちに **MIB** を照会してイベント情報を確認し、次のポーリングまでの間待機することはありません。

注**: -a** オプションを指定して Clinfo を開始した場合、NetView® for AIX、または、SNMP トラップ・メッセージを受け取る予定のその他のアプリケーションは実行できません。

Clinfo は、開始されると **/usr/sbin/cluster/etc/clhosts** ファイルを読み取ります。このファイルには、関係する各クラスター内にあるすべての使用可能ノードのサービス・ネットワーク・インターフェースの IP アドレスまたは IP ラベルがリストされています。 Clinfo は、**clhosts** ファイル内の最初の IP アドレスから開始して、このファイル内全体を検索し、ノード上にアクティブな **SNMP** プロセスがあるか確認します。 Clinfo は **SNMP** プロセスを見つけると、その **SNMP** プロセスからクラスターのトポロジーと状態に関する情報を受け取ります。

この接続が切断されると (例えば、ノードが停止した場合など)、Clinfo は別のノードの SNMP プロセスとの接続を確立しようとします。 Clinfo は、最初に通信を確立した **SNMP** プロセスからクラスター情報を受け取ると、クラスター・トポロジー情報を、ローカル・ノード上の動的に割り当てられたデータ構造の中に内部的に保持します。 そのため、Clinfo はクラスター内の他のノードについて把握しています。

次の図は、クラスター・マネージャー、Clinfo、およびクラスター・ノードの相互関係を示しています。



Clinfo が想定どおりに機能するには、**clhosts** ファイルに、Clinfo が通信できるすべての PowerHA SystemMirror サーバー・ノードとクライアント・ノードの IP アドレスが含まれている必要があります。Clinfo デーモンはその情報を SNMP を通じて、PowerHA SystemMirror サーバー・ノードから、すなわち、クラスター・マネージャー・デーモン (**clstrmgr**) が実行されているノードから取得します。 起動中、**clinfo** デーモンは次のように **clhosts** ファイルを読み取り、どのノードが SNMP を通じて通信可能であるかを判別します。

- **clstrmgr** デーモンと同じサーバーで実行されている **clinfo** デーモンの場合、ローカル・サーバー・ベースの **/usr/es/sbin/cluster/etc/clhosts** ファイルを読み取ります。このファイルには、ループバック・アドレスに関連付けられた IP アドレスのみが含まれています。
- クライアント・ノード、すなわち **clstrmgr** デーモンが実行されていないノード上で実行されている **clinfo** デーモンの場合、最高の可用性を達成するために、クライアント・ベースの **/usr/es/sbin/cluster/etc/clhosts** ファイルにすべての PowerHA SystemMirror サーバー・ノードの IP アドレスが含まれている必要があります。 この方法により、特定の PowerHA SystemMirror サーバー・ノードが使用できなくなった (例えば、電源遮断など) 場合、クライアント・ノード上の **clinfo** デーモンが SNMP を通じて別の PowerHA SystemMirror サーバー・ノードへの接続を試みることができます。

Clinfo は、起動時にローカル **SNMP** プロセスとの通信を正常に行えない場合クラスター・マップを取得できないため、別の **SNMP** プロセスとの接続を試行できません。

関連情報:

概念および機能のガイド

SNMP コミュニティー名および Clinfo

/etc/snmpd.conf ファイルのバージョンは、使用している AIX のバージョンに応じて異なります。AIX の場合、 PowerHA SystemMirror で使用されるデフォルト・バージョンは **snmpdv3.conf** ファイルです。

PowerHA SystemMirror で使用する Simple Network Management Protocol (SNMP) のコミュニティー名は、ご使用のシステムで実行している SNMP のバージョンによって異なります。SNMP コミュニティー名は次のように決定されます。

- ご使用のシステムで SNMP V1 を実行している場合、コミュニティー名は **lssrc -ls snmpd** コマンドの出力で最初に検出された、`private` または `system` 以外の名前になります。
- ご使用のシステムで SNMP V3 を実行している場合は、コミュニティー名は **/etc/snmpdv3.conf** ファイルの `VACM_GROUP` エントリーで検出された名前になります。

Clinfo サービスでは、SNMP コミュニティー名を指定するための **-c** オプションが引き続きサポートされますが、その使用は必須ではありません。 **-c** オプションを使用すると、**ps** コマンドを実行して SNMP コミュニティー名が検出される可能性があるため、セキュリティー・リスクと見なされます。

注**:** Clinfo で SNMP コミュニティー名の保護が重要な場合は、**/tmp/hacmp.out**、**/etc/snmpd.conf**、**/smit.log**、および **/usr/tmp/snmpd.log** の許可を変更して、全ユーザーが読み取り可能にならないようにします (例えば、600)。

関連情報:

snmpd.conf ファイル

ネットワーク管理用の SNMP

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。

〒*103-8510*
東京都中央区日本橋箱崎町*19*番*21*号
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

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 Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

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、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。

記載されている性能データとお客様事例は、例として示す目的でのみ提供されています。実際の結果は特定の構成や稼働条件によって異なります。

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。これらのサンプル・プログラムは特定物として現存するままの状態で提供されるものであり、いかなる保証も提供されません。 IBM は、お客様の当該サンプル・プログラムの使用から生ずるいかなる損害に対しても一切の責任を負いません。

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した創作物には、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_.

プライバシー・ポリシーに関する考慮事項

サービス・ソリューションとしてのソフトウェアも含めた IBM® ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。 IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。ご使用の「ソフトウェア・オファリング」が、これらの Cookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項を確認ください。

この「ソフトウェア・オファリング」は、Cookie もしくはその他のテクノロジーを使用して個人情報を収集することはありません。

この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。

このような目的での Cookie を含む様々なテクノロジーの使用の詳細については、IBM の『IBM オンラインでのプライバシー・ステートメント』(http://www.ibm.com/privacy/details/jp/ja/) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』および『IBM Software Products and Software-as-a-Service Privacy Statement』(http://www.ibm.com/software/info/product-privacy) を参照してください。

商標

IBM、IBM ロゴおよび ibm.com は、世界の多くの国で登録された International Business Machines Corp. の商標です。 他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。 現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtml をご覧ください。

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。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[タ行]

[ナ行]

[ハ行]

[マ行]

[ラ行]

A

C

L

S

Printed in Japan

日本アイ・ビー・エム株式会社
〒103-8510 東京都中央区日本橋箱崎町19-21