Oracle® Solaris Cluster リファレンスマニュアル

Part No: E51742
2014 年 7 月、E51742-01

ORACLE

目次

はじめに

この『*Oracle Solaris Cluster リファレンスマニュアル*』では、Oracle Solaris Cluster ソフトウェアのコマンド、機能、その他の公開インタフェースに関する参照情報を提供します。このマニュアルは、Oracle ソフトウェアおよびハードウェアに詳しいシステム管理者を対象にしています。このマニュアルは、企画やプレセールスに使用する目的で作成したものではありません。このマニュアルの内容は、Oracle Solaris オペレーティングシステムに関する知識と、Oracle Solaris Cluster ソフトウェアとともに使用するボリュームマネージャーソフトウェアに関する専門知識を前提にしています。

Oracle Solaris オペレーティングシステムの使用経験の深さとは関係なく、オンラインのマニュアルページはSPARCベースのシステムや x86 ベースのシステムとその機能に関する情報の習得に役立ちます。

マニュアルページは、コマンドの機能を一定の書式で簡潔に説明するための資料です。マニュアルページは、リファレンス (参照用) マニュアルです。チュートリアル（独習用）として作成したものではありません。

注記 - Oracle Solaris Cluster ソフトウェアは SPARC と x86 の 2 つのプラットフォームで動作します。このマニュアルの内容は、章、セクション、コメント、黒丸印の項目、図、表、または例などで特に断らないかぎり、両方のプラットフォームに関連するものです。

概要

このマニュアルページの各セクションの内容と、そこで参照できる情報の要約を次に示します。

- セクション 1: オペレーティングシステムで使用できるコマンドをアルファベット順に説明しています。
- セクション 1CL: Oracle Solaris Cluster の保守と管理に使用するコマンドをアルファベット順に説明しています。
- セクション 1HA: Oracle Solaris Cluster 高可用性 (HA) コマンドをアルファベット順に説明しています。

- セクション 1M: 主にシステムの保守と管理に使用されるコマンドをアルファベット順に説明しています。
- セクション 3HA: Oracle Solaris Cluster HA およびデータサービス機能 (関数) をアルファベット順に説明しています。
- セクション 4 では、各種ファイル形式の概要について説明します。該当する場合は、ファイル書式についての C 言語の構造体宣言を記載してあります。
- セクション 5: リソースタイプの説明など、その他の Oracle Solaris Cluster ドキュメントが含まれます。
- セクション 5CL: Oracle Solaris Cluster の標準、環境、およびマクロを説明しています。
- セクション 7: Oracle Solaris Cluster のデバイスとネットワークインタフェースを説明しています。
- セクション 7P: Oracle Solaris Cluster のプロトコルについて説明しています。

マニュアルページの一般形式は、次のようになります。各マニュアルセクションのマニュアルページの一期な書式と見出し項目の順序を示します。ただし、実際には必要な見出し項目の記事だけが記載されます。たとえば、バグが報告されていない場合は、「バグ」のセクションは省かれます。詳しくは `intro` のページおよび各ページセクションの説明を参照してください。また、マニュアルページに関する一般的な情報については、Unresolved link to " man1" を参照してください。

名前　このセクションには、文書化されたコマンドや機能の名称が記載され、それに続いてそれらのものの機能が説明されています。

形式　このセクションには、コマンドまたは機能の構文が示されています。コマンドやファイルが標準パス内に存在しない場合は、そのフルパス名が示されています。オプションと引数は、アルファベット順に、1 文字の引数を最初に、引数付きのオプションを次に配列してあります。ただし、必要な場合はこの順序を変更してあります。

このセクションでは、次の特殊文字が使用されます。

[ ]	大かっこ。大かっこで囲まれたオプションや引数は、任意指定です。大かっこで囲まれていない引数は、必ず指定しなければなりません。
…	省略記号。前にある引数にいくつかの値を指定できるか、前にある引数を繰り返し指定できることを表します (例: " `filename` …")。
|	区切り文字。この文字で分けられている引数のの内、どちらか 1 つのみを 1 時点で指定することができます。

{ }	中かっこ。中かっこで囲まれたオプションや引数は、互いに依存しています。中かっこに囲まれたすべての文字をそれぞれ 1 つのユニットとして扱う必要があります。
protocol	このセクションは、サブセクション 3R のみにあり、プロトコル記述ファイルを示しています。
説明	このセクションには、サービスの機能と動作が定義されています。それにより、そのコマンドが何をするのかが簡潔に説明されます。「説明」では、「オプション」や引用「例」は扱いません。対話型コマンド、サブコマンド、リクエスト、マクロ、および関数は、「用途」で説明されます。
IOCTL	このセクションは、セクション 7 のページだけに現れます。該当するパラメータを Unresolved link to " ioctl2" システムコールに提供するデバイスクラスのみが `ioctl` と呼ばれ、独自の見出しで記載されます。特定のデバイスに対する `ioctl` コールが、その特定のデバイスのマニュアルページにアルファベット順で一覧表示されます。`ioctl` コールは、特定のデバイスクラスに対して使用されます。この種のすべてのコールの名前の末尾は `io` で終わっています。例: Unresolved link to " mtio7I"。
オプション	このセクションにはコマンドオプションが一覧表示され、それぞれのオプションの働きについて簡潔なサマリーが示されます。オプションは文字通りに一覧表示され、「形式」セクションに記載された順序で示されます。オプションに指定できる引数については、このオプションの項とで説明され、該当する値がある場合は、デフォルト値が示されます。
オペランド	このセクションでは、コマンドのオペランドの一覧を示し、各オペランドがコマンドのアクションにどのように影響を及ぼすかを説明します。
出力	このセクションでは、標準出力、標準エラー、またはコマンドが生成する出力ファイルなどの出力が説明されます。
戻り値	マニュアルページの関数が戻り値を返す場合は、このセクションにそれらの値がリストされ、それが返される条件が示されます。関数が 0 や –1 などの定数しか返せない場合は、それらの値がタグ付きの段落に示されます。さもなければ、1 つの段落に各関数の戻り値が示されます。空 (void) として宣言された関数は、値を返しません。したがって、「戻り値」では説明されません。
エラー	実行に失敗すると、ほとんどの関数は広域変数 `errno` に失敗の原因を示すエラーコードをセットします。このセクションでは、関数が発生するすべてのエラーコードをリストし、それらのエラーの原因になる条件が説明されます。複数の条件が同じエラーを起こす場合は、それぞれの条件が段落を分けエラーコードとともに示されます。

用途	このセクションには、詳細な説明を必要とする特別な規則、機能、およびコマンドが記載されます。ここに掲載されるサブセクションでは、次の組み込み機能が説明されます。 コマンド 修飾子 変数 式 入力文法
例	このセクションでは、使用例やコマンドや関数の使用法が説明されます。可能な場合は、コマンド行入力とそれに対するマシンの応答出力を示す完全な例が示されます。例が示されている場合は常に、プロンプトが `example%` として示されるか、またはユーザーが `root` の役割である必要がある場合は `example#` として示されます。例のあとには、説明、変数の置換規則、または戻り値が示されます。ほとんどの例では、「形式」、「説明」、「オプション」、および「用途」セクションのコンセプトが例証されます。
環境変数	このセクションには、コマンドや関数が影響を与える環境変数は一覧表示され、そのあとにその効果が要約して示されます。
終了ステータス	このセクションには、コマンドからそれを呼び出したプログラムやシェルに返される値がリストされ、そのような値が返される原因になる条件が示されます。通常は、実行に成功するとゼロが返され、さまざまなエラー条件に対してはゼロ以外の値が返されます。
ファイル	このセクションには、マニュアルページで参照されているすべてのファイル名、関連ファイル、およびコマンドが作成するか必要とするファイルがリストされます。それぞれのファイル名のあとに、サマリーまたは説明が付きます。
属性	このセクションでは、コマンドの特性、ユーティリティー、およびデバイスドライバがリストされ、属性タイプとその値の定義が示されます。詳細は、Unresolved link to " attributes5" を参照してください。
関連項目	このセクションには、他のマニュアルページ、社内ドキュメント、および外部出版物への参照先が示されます。
診断	このセクションには、診断メッセージとその発生原因の要約が示されます。
警告	このセクションには、重大な支障をもたらす恐れのある特別な条件に対する警告メッセージが示されます。「警告」は、診断レベルのメッセージとは異なります。

注	このセクションには、ページ上のどのセクションにも属さない、その他の情報が示されます。「注」は、知っていると役に立つことがらの要点をユーザーに示します。ここでは、重大な情報は扱われません。
バグ	このセクションには、既知のバグと、可能な場合はその対策が示されます。

概要

名前

Intro, intro — Oracle Solaris Cluster の保守コマンドの紹介

このセクションでは、Oracle Solaris Cluster のオブジェクト指向のコマンドセットについて説明します。従来の Oracle Solaris Cluster コマンドセットもまだ使用できますが、オブジェクト指向のコマンドを使用すれば、クラスタをより直観的に構成できます。なお、従来のコマンドセットでは、今後の新機能を利用できない可能性があります。

オブジェクト指向のコマンドセットでは、共通の接頭辞として、`cl` を使用します。従来のコマンドセットでは、接頭辞に `sc` を使用していました。`sc` コマンドと `cl` コマンドは、どちらも `/usr/cluster/bin` にあります。

このコマンドセットの多くのコマンドには、長い形式と短い形式の両方があります。たとえば、273 ページの`clresource`(1CL) と 397 ページの`clrs`(1CL) は同一です。

オブジェクト指向のコマンドは、それぞれ 1 種類のクラスタオブジェクトを管理するように設計されています。コマンド名は、そのコマンドで管理するオブジェクト名を示します。たとえば、`clresource` コマンドは、Oracle Solaris Cluster データサービスのリソースを管理します。コマンド内で、サブコマンドが特定のクラスタオブジェクト上で許可される操作を定義します。

オブジェクト指向のコマンドセットのコマンドの一般的な形式は次のとおりです。

cmdname [*subcommand*] [*option…*] [*operand* …]

オブジェクト指向のコマンドとともに使用するオプションにも長い形式と短い形式があります。短い形式のオプションは、1 つのダッシュ (-) とそのあとに続く 1 つの文字で指定します。長い形式のオプションは、2 つのダッシュ (--) とそのあとに続く 1 つのオプションワードで指定します。たとえば、`-p` は短い形式のプロパティーオプションです。`--property` は、長い形式です。

オプション引数を受け入れるオプションもあれば、受け入れないオプションもあります。オプションがオプション引数を受け入れる場合、オプション引数が必要です。`-?` オプションには引数は不要です。ただし、`--property` オプションには操作するプロパティーを特定するオプション引数が必要です。

1 つのダッシュ (-) のあとに引数を付けないで、短い形式のオプションをグループ化することができます。例: `-eM`。オプションのあとのオプション引数のグループをコンマ、タブまたは空白文字で区切る必要があります。タブまたは空白を使用する場合、オプション引数を引用符で囲みます (-o xxx,z,yy または -o "xxx z yy")。

長いオプション名でオプション引数を指定するには、`--input=`*configurationfile* 形式または `--input` *configurationfile* 形式のいずれかを使用します。

このコマンドセットのコマンドはすべて `-?` または `--help` オプションを受け入れます。サブコマンドなしでこれらのオプションを指定すると、コマンドのサマリーのヘルプが表示されます。サブコマンドを指定すると、そのサブコマンドだけのヘルプが表示されます。

一部のコマンドは、構成ファイルと組み合わせて機能します。このファイルの必要な形式については、1447 ページの`clconfiguration`(5CL) のマニュアルページを参照してください。

このコマンドセットの多くのサブコマンドは、適用できるすべてのオブジェクトを示すオペランドとして、`+` を受け入れます。

コマンドのリスト

このセクションでは、Oracle Solaris Cluster 製品で使用できるオブジェクト指向のコマンドをアルファベット順に説明します。

37 ページの`claccess`(1CL)
: ノードを追加するための Oracle Solaris Cluster アクセスポリシーを管理する

61 ページの`cldevice`(1CL)、43 ページの`cldev`(1CL)
: Oracle Solaris Cluster デバイスを管理します

79 ページの`cldevicegroup`(1CL)、99 ページの`cldg`(1CL)
: Oracle Solaris Cluster デバイスグループを管理します

119 ページの`clinterconnect`(1CL)、131 ページの`clintr`(1CL)
: Oracle Solaris Cluster インターコネクトを管理する

169 ページの`clnasdevice`(1CL)、153 ページの`clnas`(1CL)
: Oracle Solaris Cluster の NAS デバイスへのアクセスを管理する

185 ページの`clnode`(1CL)
: Oracle Solaris Cluster ノードを管理します

217 ページの`clpstring`(1CL)
: Oracle Solaris Cluster プライベート文字列の管理

239 ページの`clquorum`(1CL)、227 ページの`clq`(1CL)
: Oracle Solaris Cluster 定足数を管理する

251 ページの clreslogicalhostname(1CL), 429 ページの clrslh(1CL)
論理ホスト名の Oracle Solaris Cluster リソースを管理します

273 ページの clresource(1CL)、397 ページの clrs(1CL)
Oracle Solaris Cluster データサービスのリソースを管理します

305 ページの clresourcegroup(1CL)、369 ページの clrg(1CL)
Oracle Solaris Cluster データサービスのリソースグループの管理

333 ページの clresourcetype(1CL)、473 ページの clrt(1CL)
Oracle Solaris Cluster データサービスのリソースタイプを管理します

347 ページの clressharedaddress(1CL)、451 ページの clrssa(1CL)
共有アドレスの Oracle Solaris Cluster リソースを管理します

487 ページの clsetup(1CL)
Oracle Solaris Cluster を対話により構成します

491 ページの clsnmphost(1CL)
Oracle Solaris Cluster SNMP ホストを管理します

499 ページの clsnmpmib(1CL)、143 ページの clmib(1CL)
Oracle Solaris Cluster SNMP MIB を管理します

509 ページの clsnmpuser(1CL)
Oracle Solaris Cluster SNMP ユーザを管理します

535 ページの cltelemetryattribute(1CL)
システムリソースモニタリングの構成

551 ページの cluster(1CL)
クラスタのグローバル構成とグローバルステータスの管理

613 ページの clzonecluster(1CL)、583 ページの clzc(1CL)
Oracle Solaris Cluster のゾーンクラスタを管理します

従来の Oracle Solaris Cluster コマンドとオブジェクト指向コマンドの対応

新しいコマンドセットはオブジェクト指向なので、従来のコマンドセットとの明確な 1 対 1 の対応はありません。次のリストは、従来のコマンドセットの一般的な Oracle Solaris Cluster コマンドとそれに対応するオブジェクト指向のコマンドセットをいくつか示しています。

`scstat`	`cluster status` さらに、多くのオブジェクト指向のコマンドで `status` サブコマンドを使用することもできます。
`scinstall`	`cluster create` を使用して、XML 構成ファイルからクラスタを作成します。 対話形式でクラスタを作成するには、`scinstall` を使用します。
`scrgadm`	■ `clresource` ■ `clresourcetype` ■ `clresourcegroup` これらの特定のリソースタイプを操作するときに、`clressharedaddress` および `clreslogicalhostname` を使用すると、さらに便利です。
`scswitch`	■ `clresource` ■ `clresourcetype` ■ `clresourcegroup` ■ `clreslogicalhostname` ■ `clressharedaddress` ■ `clnode evacuate` (ノードからすべてのリソースグループおよびデバイスグループを退避させるために使用)
`scconf`	■ `cldevicegroup` ■ `clinterconnect` ■ `clquorum` ■ `clnode` ■ `claccess` `cluster show` を `scconf -p` の代わりに使用します。
`sccheck`	`cluster check`
`scdidadm`	`cldevice`
`scgdevs`	`cldevice populate`

scdpm	cldevice
scnas、scnasdir	clnasdevice
scsetup	clsetup

オブジェクト指向の Oracle Solaris Cluster コマンドは、指定されたすべてのオペランドで成功すると、ゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

これらの終了コードは、このセットのコマンド全体で共有されます。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

5 CL_ERECONF

クラスタは再構成されます

クラスタを再構成しています。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

9 CL_ESTATE

オブジェクトの状態が不正です

その時点で変更できない、または常時変更できないプロパティー、リソースグループ、またはその他のオブジェクトを変更しようとしました。

10 CL_EMETHOD

リソースのメソッドが失敗しました

リソースのメソッドが失敗しました。次のいずれかの理由のために、メソッドは失敗しました。

- validate メソッドは、リソースを作成しようとしたときに、あるいは、リソースのプロパティーを変更しようとしたときに失敗しました。
- validate 以外のメソッドは、リソースを有効、無効、または削除しようとしたときに失敗しました。

15 CL_EPROP

無効なプロパティーです

-p、-y、または -x オプションで指定したプロパティーまたは値が存在しないか、許可されていません。

18 CL_EINTERNAL

内部エラーが発生しました

内部エラーは、ソフトウェアの欠陥またはその他の欠陥を示しています。

35 CL_EIO

I/O エラー

物理的な入出力エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

37 CL_EOP

操作が許可されていません

サポートされていない構成に対する操作を実行しようとしたか、サポートされていない操作を実行しました。

38 CL_EBUSY

オブジェクトはビジーです

アクティブなクラスタノードへの最後のクラスタインターコネクトパスからケーブルを取り外そうとしました。または、参照を削除していないクラスタ構成からノードを削除しようとしました。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

41 CL_ETYPE

無効なタイプです

-t または -p オプションで指定したタイプは存在しません。

50 CL_ECLMODE

ノードはクラスタモードです

クラスタモードでブートされているノードで操作を実行しようとしました。しかし、この操作は、非クラスタモードでブートされているノードだけで実行できます。

51 CL_ENOTCLMODE

ノードはクラスタモードではありません

非クラスタモードでブートされているノードで操作を実行しようとしました。しかし、この操作は、クラスタモードでブートされているノードだけで実行できます。

Unresolved link to " getopt1"

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

OSC4 1

名前

libschost.so.1 — 物理ホスト名の代わりに論理ホスト名を提供する共有オブジェクト物理ホスト名の代わりに論理ホスト名を提供する共有オブジェクト

```
libschost.so.1
```

libschost.so.1 共有オブジェクトが提供する機能を使用すると、起動対象プロセスおよびその子孫に対して物理ホスト名を選択的に構成できます。

Oracle Solaris Cluster 環境では、アプリケーションはフェイルオーバーまたはスイッチオーバー後、同一ホスト名へのアクセスを試みることがあります。このフェイルオーバーまたはスイッチオーバーは、フェイルオーバーまたはスイッチオーバー後、物理ホスト名が変更されているため失敗します。そのような場合、アプリケーションデータサービスは libschost.so.1 共有オブジェクトを使用すると、物理ホスト名の代わりに論理ホスト名をアプリケーションに提供できます。

libschost.so.1 を有効にするには、 SC_LHOSTNAME 環境変数のほかに次の 2 つの環境変数も設定する必要があります。

```
LD_PRELOAD_32=$LD_PRELOAD_32:/usr/cluster/lib/libschost.so.1
LD_PRELOAD_64=$LD_PRELOAD_64:/usr/cluster/lib/64/libschost.so.1
```

LD_PRELOAD_32 と LD_PRELOAD_64 の両方の環境変数を設定すると、libschost.so.1 共有オブジェクトは 32 ビットおよび 64 ビットアプリケーションの両方で動作します。

実行時リンカーがアクセスするデフォルトの信頼できるディレクトリは、32 ビットオブジェクトの場合は /usr/lib/secure、64 ビットオブジェクトの場合は /usr/lib/secure/64 です。セキュリティ保護されたアプリケーションで libschost.so.1 共有オブジェクトを使用する場合は、libschost.so.1 共有オブジェクトに信頼できるディレクトリからアクセスする必要があります。

これを行うには、32 ビットアプリケーションの場合は /usr/cluster/lib/libschost.so.1 から /usr/lib/secure/libschost.so.1、64 ビットアプリケーションの場合は /usr/cluster/lib/64/libschost.so.1 から /usr/lib/secure/64/libschost.so.1 へのシンボリックリンクをそれぞれ作成します。

これらのシンボリックリンクの作成後、LD_PRELOAD_32 および LD_PRELOAD_64 環境変数で libschost.so.1 共有オブジェクトを、信頼できるディレクトリから使用します。

また、crle コマンドを使用して、セキュリティ保護されたアプリケーションの信頼できるディレクトリを追加したり変更したりすることもできます。Unresolved link to " crle1" のマニュアルページを参照してください。

プリロードが終わったら、libschost.so.1 共有オブジェクトは次の環境変数を読み、それをホスト名として返します。

SC_LHOSTNAME=*hostname*

SC_LHOSTNAMEでは、論理ホスト名を指定します。指定したホスト名は、起動対象および子孫のすべてのプロセスで利用できます。

hostname 値は、 MAXHOSTNAMELEN の文字数以下とすることができます。定数 MAXHOSTNAMELEN は、netdb.h ヘッダーファイル内で 256 文字として定義されています。

例 1 C の実行時における論理ホスト名の構成

次の例の C コードでは、論理ホスト名を使用してホスト名を構成しています。この例には、Oracle Solaris Cluster 関数 scds_get_rs_hostnames() に対する呼び出しと Oracle Solaris Cluster データ構造 scds_handle_t および scds_net_resource_list_t に対する参照が含まれています。

scds_get_rs_hostnames() 関数は、リソースによって使用されるホスト名のリストを提供します。このコードは、このリスト内の先頭のホスト名の値を環境変数 SC_LHOSTNAME に割り当てます。

次のコードの実行後に起動されるアプリケーションは、物理ホスト名ではなく論理ホスト名を取得します。

```
/* 13 bytes to hold "SC_LHOSTNAME=" string */
#define HOSTLENGTH (MAXHOSTNAMELEN + 13)

/* 14 bytes to hold "LD_PRELOAD_XX=" string */
#define PATHLENGTH (MAXPATHLEN + 14)

char lhostname[HOSTLENGTH], ld_32[PATHLENGTH], \
     ld_64[PATHLENGTH];

scds_get_rs_hostnames(scds_handle, &snrlp);
if (snrlp != NULL && snrlp->num_netresources != 0) {
    snprintf(lhostname, HOSTLENGTH, "SC_LHOSTNAME=%s", \
        snrlp->netresources[0].hostnames[0]);
    putenv(lhostname);
}

/* Setting LD_PRELOAD_32 environment variable */
if (getenv("LD_PRELOAD_32") == NULL)
    snprintf(ld_32, PATHLENGTH, "LD_PRELOAD_32="
```

```
        "/usr/cluster/lib/libschost.so.1");
else
    snprintf(ld_32, PATHLENGTH, "LD_PRELOAD_32=%s:"
        "/usr/cluster/lib/libschost.so.1", \
         getenv("LD_PRELOAD_32"));

putenv(ld_32);

/* Setting LD_PRELOAD_64 environment variable */
if (getenv("LD_PRELOAD_64") == NULL)
    snprintf(ld_64, PATHLENGTH, "LD_PRELOAD_64="
        "/usr/cluster/lib/64/libschost.so.1");
else
    snprintf(ld_64, PATHLENGTH,
        "LD_PRELOAD_64=%s:/usr/cluster/lib/"
        "64/libschost.so.1", getenv("LD_PRELOAD_64"));

putenv(ld_64);
```

例 2 シェルコマンドによる実行時の論理ホスト名の構成

次の例のシェルコマンドでは、gethostnames コマンドを使用し論理ホスト名によってホスト名をアプリケーションデータサービスで構成する方法を示しています。gethostnames コマンドは次の引数を取ります。

- -R *resource-name*
- -G *resourcegroup-name*
- -T *resourcetype-name*

gethostnames コマンドは、該当するリソースと関連付けられているすべての論理ホスト名をセミコロン (;) で区切って返します。このコマンドは、このリスト内の先頭のホスト名の値を環境変数 SC_LHOSTNAME に割り当てます。

```
phys-schost-1$ LD_PRELOAD_32=$LD_PRELOAD_32:/usr/cluster/lib/libschost.so.1
phys-schost-1$ LD_PRELOAD_64=$LD_PRELOAD_64:/usr/cluster/lib/64/libschost.so.1
phys-schost-1$ SC_LHOSTNAME=`/usr/cluster/lib/scdsbuilder/src/scripts/gethostnames \
               -R nfs-r -G nfs-rg -T SUNW.nfs:3.1 |cut -f1 -d","`
phys-schost-1$ export LD_PRELOAD_32 LD_PRELOAD_64 SC_LHOSTNAME
```

例 3 シェルコマンドによるセキュリティ保護されたアプリケーションの論理ホスト名の構成

次の例のシェルコマンドでは、論理ホスト名を構成しています。次のシェルコマンドの実行後に起動されるセキュリティ保護されたアプリケーションはすべて、物理ホスト名ではなく環境変数 SC_LHOSTNAME の値 (つまり、論理ホスト名) を取得します。

```
phys-schost-1$ cd /usr/lib/secure
```

```
phys-schost-1$ ln -s /usr/cluster/lib/libschost.so.1 .
phys-schost-1$ cd /usr/lib/secure/64
phys-schost-1$ ln -s /usr/cluster/lib/64/libschost.so.1 .
phys-schost-1$ LD_PRELOAD_32=$LD_PRELOAD_32:/usr/lib/secure/libschost.so.1
phys-schost-1$ LD_PRELOAD_64=$LD_PRELOAD_64:/usr/lib/secure/64/libschost.so.1
phys-schost-1$ SC_LHOSTNAME=test
phys-schost-1$ export LD_PRELOAD_32 LD_PRELOAD_64 SC_LHOSTNAME
```

`/usr/cluster/lib/libschost.so.1`

32 ビットアプリケーション用共有オブジェクトのデフォルトの格納先

`/usr/cluster/lib/64/libschost.so.1`

64 ビットアプリケーション用共有オブジェクトのデフォルトの格納先

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

Unresolved link to " crle1", Unresolved link to " cut1", Unresolved link to " hostname1", Unresolved link to " ld1", Unresolved link to " ld.so.11", Unresolved link to " proc1", Unresolved link to " uname1", Unresolved link to " exec2", Unresolved link to " sysinfo2", Unresolved link to " uname2", Unresolved link to " gethostname3C", Unresolved link to " putenv3C", Unresolved link to " snprintf3C", Unresolved link to " system3C", Unresolved link to " proc4"

論理ホスト名が継承されます。

次のコマンドまたは関数を呼び出してホスト名を取得するユーザープログラムは、物理ホスト名ではなく論理ホスト名を取得できます。

- `hostname` コマンド
- `uname` command
- `uname()` function
- `sysinfo()` 関数
- `gethostname()` 関数

他のコマンドまたは関数によってホスト名を取得するユーザープログラムは、論理ホスト名を取得することができません。

OSC4 1cl

名前

claccess — ノード用の Oracle Solaris Cluster アクセスポリシーの管理

```
/usr/cluster/bin/claccess -V

/usr/cluster/bin/claccess [subcommand] -?

/usr/cluster/bin/claccess subcommand [options] -v [hostname[,…]]

/usr/cluster/bin/claccess allow -h hostname[,…]

/usr/cluster/bin/claccess allow-all

/usr/cluster/bin/claccess deny -h hostname[,…]

/usr/cluster/bin/claccess deny-all

/usr/cluster/bin/claccess list

/usr/cluster/bin/claccess set -p protocol=authprotocol

/usr/cluster/bin/claccess show
```

`claccess` コマンドは、クラスタ構成にアクセスしようとするマシンのネットワークアクセスポリシーを制御します。`claccess` コマンドに短い形式はありません。

クラスタは、クラスタ構成にアクセスできるマシンのリストを管理します。クラスタはまた、これらのノードがクラスタ構成にアクセスするのに使用する承認プロトコルの名前を格納します。

マシンをクラスタ構成に追加することを要求する場合など (185 ページの`clnode`(1CL) を参照)、マシンがクラスタ構成にアクセスしようとすると、クラスタはこのリストをチェックして、ノードにアクセス権限があるかどうかを判定します。そのノードがアクセス権を持っている場合、そのノードは、クラスタ構成にアクセスすることが承認および許可されます。

`claccess` コマンドは、次のタスクに使用できます。

- 任意の新しいマシンが自分自身をクラスタ構成に追加したり、自分自身をクラスタ構成から削除したりすることを許可する
- 任意のノードが自分自身をクラスタ構成に追加したり、自分自身をクラスタ構成から削除したりすることを禁止する
- チェックする承認タイプを制御する

このコマンドは、大域ゾーンだけで使用できます。

`claccess` コマンドの一般的な形式は次のとおりです。

```
claccess [subcommand] [options]
```

subcommand を省略できるのは、*options* で `-?` オプションまたは `-V` オプションが指定されている場合のみです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式は、このマニュアルページの「オプション」セクションのオプションの説明で紹介されています。

サブコマンド

サポートされるサブコマンドには次のものがあります。

`allow`

指定されたマシン (1 つまたは複数) がクラスタ構成にアクセスすることを許可します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" を参照してください。

`deny` および `allow-all` サブコマンドの説明も参照してください。

`allow-all`

すべてのマシンが自分自身をクラスタ構成に追加して、クラスタ構成にアクセスすることを許可します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

`deny-all` および `allow` サブコマンドの説明も参照してください。

`deny`

指定されたマシン (1 つまたは複数) がクラスタ構成にアクセスすることを禁止します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

`allow` および `deny-all` サブコマンドの説明も参照してください。

`deny-all`

すべてのマシンがクラスタ構成にアクセスすることを禁止します。

クラスタを初めて構成したあと、デフォルトの設定では、どのノードにもクラスタ構成へのアクセス権はありません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

`allow-all` および `deny` サブコマンドの説明も参照してください。

`list`

クラスタ構成にアクセスする承認を持っているマシンの名前を表示します。承認プロトコルも表示するには、`show` サブコマンドを使用します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

`set`

承認プロトコルを `-p` オプションで指定した値に設定します。デフォルトでは、システムは `sys` を承認プロトコルとして使用します。「オプション」の `- p` オプションを参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

`show`

クラスタ構成にアクセスするアクセス権を持っているマシンの名前を表示します。承認プロトコルも表示します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

次のオプションがサポートされています。

`-?`
`--help`

ヘルプ情報を表示します。このオプションを使用すると、ほかの処理は実行されません。

このオプションを指定するとき、サブコマンドは指定してもしなくてもかまいません。このオプションをサブコマンドなしで指定すると、このコマンドのサブコマンドのリストが表示されます。サブコマンド付きでこのオプションを指定すると、サブコマンドの使用方法が表示されます。

`-h` *hostname*
`--host=`*hostname*
`-- host` *hostname*

アクセスを付与または拒否するノードの名前を指定します。

`-p` protocol=*authentication-protocol*
`--authprotocol=`*authentication-protocol*
`--authprotocol` *authentication-protocol*

マシンがクラスタ構成へのアクセス権を持っているかどうかをチェックするのに使用する承認プロトコルを指定します。

サポートされるプロトコルは、des と sys (または unix) です。デフォルトの認証型は sys ですが、これは最低限のセキュリティー保護しか実行しません。ノードの追加と削除の詳細は、Unresolved link to " Oracle Solaris Cluster システム管理 の第 8 章クラスタノードの管理"を参照してください。これらの認証タイプの詳細は、Unresolved link to " Managing Kerberos and Other Authentication Services in Oracle Solaris 11.2 の第 10 章Configuring Network Services Authentication"を参照してください。

-V
--version

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンドやオペランドなどのオプションは無視されます。-V オプションは、コマンドのバージョンだけを表示します。ほかの処理は実行されません。

-v
--verbose

詳細な情報を標準出力 stdout に表示します。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

次の終了コードが返されます。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link

to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

18 CL_EINTERNAL

内部エラーが発生しました

内部エラーは、ソフトウェアの欠陥またはその他の欠陥を示しています。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

例 4 新しいホストのアクセスの許可

次の `claccess` コマンドは、新しいホストがクラスタ構成にアクセスすることを許可します。

```
# claccess allow -h phys-schost-1
```

例 5 承認タイプの設定

次の `claccess` コマンドは、現在の承認タイプを `des` に設定します。

```
# claccess set -p protocol=des
```

例 6 すべてのホストのアクセスの拒否

次の `claccess` コマンドは、すべてのホストがクラスタ構成にアクセスすることを拒否します。

```
# claccess deny-all
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページの`Intro`(1CL), 185 ページの`clnode`(1CL), 551 ページの`cluster`(1CL)

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のサブコマンドとオプションを指定してこのコマンドを実行できます。

- -? オプション
- -V オプション

スーパーユーザー以外のユーザーがほかのサブコマンドを指定してこのコマンドを実行するには、RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
allow	solaris.cluster.modify
allow-all	solaris.cluster.modify
deny	solaris.cluster.modify
deny-all	solaris.cluster.modify
list	solaris.cluster.read
set	solaris.cluster.modify
show	solaris.cluster.read

名前

cldevice, cldev — Oracle Solaris Cluster デバイスの管理

```
/usr/cluster/bin/cldevice -V

/usr/cluster/bin/cldevice [subcommand] -?

/usr/cluster/bin/cldevice subcommand [options] -v [+ | device …]

/usr/cluster/bin/cldevice check [-n node[,…]] [+]

/usr/cluster/bin/cldevice clear [-n node[,…]] [+]

/usr/cluster/bin/cldevice combine -t replication-type -g
     replication-device-group -d destination-device device

/usr/cluster/bin/cldevice export [-o {- | configfile}] [-n node[,…]]
     [+ | device…]

/usr/cluster/bin/cldevice list [-n node[,…]] [+ | device …]

/usr/cluster/bin/cldevice monitor [-i {- | clconfigfile}] [-n
     node[,…]] {+ | disk-device …}

/usr/cluster/bin/cldevice populate

/usr/cluster/bin/cldevice refresh [-n node[,…]] [+]

/usr/cluster/bin/cldevice rename -d destination-device device

/usr/cluster/bin/cldevice repair [-n node[,…]]
     {+ | device …}

/usr/cluster/bin/cldevice replicate -t replication-type [-S
     source-node] -D destination-node [+]

/usr/cluster/bin/cldevice set
     -p default_fencing={global | pathcount | scsi3 | nofencing | nofencing-noscrub}
     [-n node[,…]] device …

/usr/cluster/bin/cldevice show [-n node[,…]] [+ | device …]

/usr/cluster/bin/cldevice status [-s state] [-n node[,…]] [+ |
     [disk-device ]]

/usr/cluster/bin/cldevice unmonitor [-i {- | clconfigfile}]
     [-n node[,…]] {+ | disk-device …}
```

`cldevice` コマンドは、Oracle Solaris Cluster 環境のデバイスを管理します。このコマンドは、Oracle Solaris Cluster デバイス識別子 (DID) 疑似デバイスドライバを管理し、ディスクデバイスパスをモニターするのに使用します。

- DID ドライバは、あるデバイスへの複数のパスが使用可能である場合でも一意のデバイス ID をそのデバイスに提供します。詳細は、1479 ページのdid(7) のマニュアルページを参照してください。
- ディスクパスとは、クラスタノードと物理ディスクまたは LUN ストレージデバイス間の接続のことです。ディスクパスには、Oracle Solaris カーネルドライバスタック、ホストバスアダプタ、および介在する任意のケーブル、スイッチ、またはネットワーク接続が含まれます。

`cldev` コマンドは、`cldevice` コマンドの短い形式です。どちらの形式のコマンドも使用できます。

`list` および `show` サブコマンドを除き、`cldevice` コマンドは、オンラインであり、かつクラスタモードにあるクラスタノードから実行する必要があります。

このコマンドの一般的な形式は次のとおりです。

`cldevice` [*subcommand*] [*options*] [*operands*]

subcommand を省略できるのは、*options* で `-?` オプションまたは `-V` オプションが指定されている場合のみです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式については、このマニュアルページの「オプション」セクションを参照してください。

詳細は、19 ページのIntro(1CL) のマニュアルページを参照してください。

このコマンドは、大域ゾーンだけで使用できます。

サブコマンド

サポートされるサブコマンドには次のものがあります。

`check`

デバイスの物理デバイスに対する整合性検査を、カーネル表現と比較して実行します。この整合性検査で問題が発見されると、エラーメッセージが表示されます。この処理は、すべてのデバイスが検査されるまで継続されます。

デフォルトでは、このサブコマンドは現在のノードだけに影響します。`-n` オプションは、別のノードに接続されているデバイスの検査処理を実行するのに使用します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

`clear`

現在のノードから切断された配下のデバイスへの DID 参照をすべて削除します。

デフォルトでは、このサブコマンドは現在のノードだけに影響します。-n オプションは、クリアー処理を実行する別のクラスタノードを指定するのに使用します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

combine

指定されたデバイスを指定された対象先に結合します。

combine サブコマンドは、ソースデバイスのパスを対象先デバイスのパスに結合します。このようにパスを結合すると、DID インスタンス番号が 1 つになり、対象先の DID インスタンス番号と同じになります。このサブコマンドは、SRDF を使用してレプリケートされる EMC LUN に対応する DID インスタンスを結合するために使用します。

combine サブコマンドは、ストレージベースの複製用に DID デバイスを手動で構成するのに使用できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

export

クラスタデバイスの構成情報をエクスポートします。

-o オプションでファイル名を指定する場合、構成情報はその新しいファイルに書き込まれます。-o オプションを指定しない場合、構成情報は標準出力に書き込まれます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

list

すべてのデバイスパスを表示します。

オペランドをまったく指定しない場合、あるいは、プラス記号 (+) をオペランドに指定する場合、すべてのデバイスが報告されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

monitor

指定したディスクパスのモニタリングをオンにします。

monitor サブコマンドは、ディスクデバイスだけに機能します。このサブコマンドは、テープなどのデバイスには影響しません。

monitor サブコマンドを使用して、ディスクパスモニタリングデーモン scdpmd をチューニングできます。構成ファイルの詳細は、1259 ページのscdpmd.conf(4) のマニュアルページを参照してください。

デフォルトでは、このサブコマンドはすべてのノードからのパスのモニタリングをオンにします。

-i オプションは、ディスクパスのモニタープロパティーを設定するクラスタ構成ファイルを指定するのに使用します。-i オプションは、指定されたファイルでモニタリン

グ対象のマークが付いているディスクパス上でディスクパスのモニタリングを開始します。ほかのディスクパスに変更は行われません。クラスタ構成ファイルの詳細は、1447 ページのclconfiguration(5CL) のマニュアルページを参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

`populate`

グローバルデバイス名前空間を生成します。

グローバルデバイス名前空間は /global ディレクトリの下にマウントされます。名前空間は、物理デバイスへの論理リンクの集合から構成されます。/dev/global ディレクトリはクラスタ内の各ノードから見えるので、どのノードからでも個々の物理デバイスが確認できます。したがって、クラスタ内のどのノードからでも、広域デバイスの名前空間に追加されたディスク、テープ、または CD-ROM にアクセスできます。

populate サブコマンドを使用すると、管理者は、システムをリブートせずに、新しいグローバルデバイスをグローバルデバイス名前空間に接続できます。これらのデバイスは、テープドライブ、CD-ROM ドライブ、またはディスクドライブでもかまいません。

populate サブコマンドを実行する前に、Unresolved link to " devfsadm1M" コマンドを実行する必要があります。代わりに、再構成リブートを実行して、グローバルデバイス名前空間を再構築し、新しいグローバルデバイスを接続してもかまいません。再構成リブートの詳細は、Unresolved link to " boot1M" のマニュアルページを参照してください。

populate サブコマンドは、現在のクラスタメンバーであるノードから実行してください。

populate サブコマンドはその作業をリモートノード上で非同期的に実行します。したがって、このコマンドを実行したノード上でコマンドが完了しても、すべてのクラスタノード上でこのコマンドが完了しているわけではありません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

`refresh`

クラスタノード上にある現在のデバイスツリーに対してデバイス構成情報を更新します。このコマンドにより、rdsk および rmt のデバイスツリーの完全な検索が行われます。このコマンドにより、新たに認識されたデバイス識別子ごとに、新しい DID インスタンス番号が割り当てられます。また、新たに認識されたデバイスごとに、新しいパスが追加されます。

デフォルトでは、このサブコマンドは現在のノードだけに影響します。-n オプションは、refresh サブコマンドと一緒に使用して、リフレッシュ処理を実行するクラスタノードを指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

`rename`

指定されたデバイスに新しい DID インスタンス番号をつけます。

このコマンドは、ソースデバイスの DID インスタンス番号に対応する DID デバイスパスを削除して、指定された宛先 DID インスタンス番号を使ってその DID デバイスパスを作成

し直します。このサブコマンドは、間違って変更された DID インスタンス番号を復元するのにも使用できます。

共有ストレージに接続されているすべてのクラスタノードで `rename` サブコマンドを実行したあとに、`devfsadm` および `cldevice populate` コマンドを実行し、グローバルデバイス名前空間を更新して、構成の変更を反映させます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。

`repair`

指定されたデバイスに対して修復手順を実行します。

デフォルトでは、このサブコマンドは現在のノードだけに影響します。`-n` オプションは、修復処理を実行するクラスタノードを指定するのに使用します。

オペランドをまったく指定しない場合、あるいは、プラス記号 (`+`) をオペランドに指定する場合、このコマンドは、現在のノードに接続されているすべてのデバイスについての構成情報を更新します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。

`replicate`

ストレージベースのレプリケーション機能で使用する DID デバイスを構成します。

注記 - `replicate` サブコマンドでは、DID インスタンスと EMC SRDF を結合する方法はサポートされていません。DID インスタンスと SRDF を結合するには、`cldevice combine` を使用します。

`replicate` サブコマンドは、ソースノード上の各 DID インスタンス番号を宛先ノード上の対応する DID インスタンス番号に結合します。レプリケートされたデバイスの各ペアは、単一の論理 DID デバイスにマージされます。

デフォルトでは、現在のノードがソースノードです。`-s` オプションは、別のソースノードを指定するのに使用します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。

`set`

指定されたデバイスのプロパティーを変更します。

`-p` オプションは、変更するプロパティーを指定するのに使用します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。

`show`

指定されたすべてのデバイスパスの構成レポートを表示します。

このレポートには、デバイスへのパスと、そのパスがモニター状態またはモニター解除状態のどちらにあるかが表示されます。

デフォルトでは、このサブコマンドはすべてのデバイスの構成情報を表示します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

`status`

指定されたすべてのディスクデバイスパスのステータスを表示します。

デフォルトでは、このサブコマンドはすべてのノードからのすべてのディスクパスのステータスを表示します。

`status` サブコマンドはディスクデバイスだけに機能します。このレポートには、テープなどのデバイスは報告されません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

`unmonitor`

コマンドのオペランドとして指定されたディスクパスのモニタリングをオフにします。

デフォルトでは、このサブコマンドはすべてのノードからのパスのモニタリングをオフにします。

`unmonitor` サブコマンドはディスクデバイスだけに機能します。このサブコマンドは、テープなどのデバイスには影響しません。

`-i` オプションは、ディスクパスのモニタリングをオフにするクラスタ構成ファイルを指定するのに使用します。ディスクパスのモニタリングがオフになるのは、指定されたファイルでモニタリング解除のマークが付いているディスクパスです。ほかのディスクパスに変更は行われません。詳細は、1447 ページの`clconfiguration`(5CL) のマニュアルページを参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。

次のオプションがサポートされています。

`-?`
`--help`

ヘルプ情報を表示します。

このオプションは単独でもサブコマンド付きでも使用できます。

- このオプションを単独で使用する場合、使用可能なサブコマンドのリストが出力されます。
- このオプションをサブコマンドを付けて使用する場合、そのサブコマンドの使用法オプションが出力されます。

このオプションを使用する場合、ほかの処理は実行されません。

-D *destination-node*
-destinationnode=*destination-node*
-destinationnode *destination-node*

デバイスをレプリケートする宛先ノードを指定します。ノードは、ノード名またはノード ID のどちらででも指定できます。

-D オプションは、replicate サブコマンドだけで有効です。

-d *destination-device*
--device=*destination-device*
--device *destination-device*

ストレージベースのレプリケーションを行うために、レプリケーション先デバイスの DID インスタンス番号を指定します。

DID インスタンス番号は -d オプションだけで使用します。ほかの形式の DID 名や UNIX のフルパス名は、対象先を指定するのには使用しません。

-d オプションは、rename および combine サブコマンドでのみ有効です。

-g *replication-device-group*

レプリケーションデバイスグループを指定します。このオプションは、combine サブコマンドとのみ使用できます。

-i {- | *clconfigfile*}
--input={- | *clconfigfile*}
--input {- | *clconfigfile*}

モニターするディスクパスまたはモニタリングしないディスクパスで使用される構成情報を指定します。この情報は、1447 ページのclconfiguration(5CL) のマニュアルページに定義されている形式に準拠している必要があります。この情報は、ファイルに含めることも、標準入力を介して指定することもできます。標準入力を指定するには、ファイル名の代わりに、マイナス記号 (-) を指定します。

-i オプションは、monitor および unmonitor サブコマンドでのみ有効です。

コマンドで指定するオプションは、構成ファイルで設定されている任意のオプションより優先されます。構成パラメータがクラスタ構成ファイルに存在しない場合、これらのパラメータをコマンド行で指定してください。

-n *node*[,…]
--node=*node*[,…]
--node *node*[,…]

サブコマンドが、-n オプションで指定されるノードからのディスクパスだけを含むことを指定します。ノードは、ノード名またはノード ID のどちらででも指定できます。

-o {- | *configfile*}
--output={- | *configfile*}
--output {- | *configfile*}

1447 ページの`clconfiguration`(5CL) のマニュアルページで定義されている形式で、ディスクパス構成情報を書き込みます。この情報は、ファイルまたは標準出力のどちらにでも書き込むことができます。

`-o` オプションは、`export` サブコマンドだけで有効です。

このオプションの引数としてファイル名を指定する場合、このコマンドは新しいファイルを作成して、そのファイルに構成情報を出力します。同じ名前のファイルがすでにある場合、このコマンドはエラーで終了します。既存のファイルに変更は行われません。

このオプションの引数としてマイナス記号 (-) を指定する場合、このコマンドは標準出力に構成情報を表示します。このコマンドのほかの標準出力はすべて抑制されます。

`-p default_fencing={global | pathcount| scsi3 | nofencing | nofencing-noscrub}`
`--property=default_fencing={global|pathcount|scsi3|nofencing|nofencing-noscrub}`
`--property default_fencing={global|pathcount|scsi3|nofencing|nofencing-noscrub}`

変更するプロパティーを指定します。

このオプションは `set` サブコマンドと一緒に使用して、次のプロパティーを変更します。

`default_fencing`

指定したデバイスについて、グローバルなデフォルトのフェンシングアルゴリズムを変更します。定足数デバイスとして構成されているデバイスのデフォルトのフェンシングアルゴリズムは変更できません。

デバイスのデフォルトのフェンシングアルゴリズムは、次の値のいずれかに設定できます。

`global`

グローバルなデフォルトのフェンシング設定を使用します。フェンシングのグローバルデフォルトの設定については、551 ページの`cluster`(1CL) のマニュアルページを参照してください。

`nofencing`

Persistent Group Reservation (PGR) キーをチェックし、いずれかのキーを削除したあとで、指定されたデバイスのフェンシングをオフにします。

注意 - Serial Advanced Technology Attachment (SATA) ディスクのような、SCSI をサポートしていないディスクを使用している場合は、フェンシングをオフにします。

`nofencing-noscrub`

最初に PGR キーをチェックまたは削除*せずに*、指定されたデバイスのフェンシングをオフにします。

注意 - Serial Advanced Technology Attachment (SATA) ディスクのような、SCSI をサポートしていないディスクを使用している場合は、フェンシングをオフにします。

`pathcount`

共有デバイスに接続されている DID パスの数でフェンシングプロトコルを決定します。

- 使用する DID パスが 3 未満のデバイスには、このコマンドは SCSI-2 プロトコルを設定します。
- 使用する DID パスが 3 以上のデバイスには、このコマンドは SCSI-3 プロトコルを設定します。

`scsi3`

SCSI-3 プロトコルを設定します。そのデバイスが SCSI-3 プロトコルをサポートしない場合、フェンシングプロトコルの設定は変更されません。

`-S` *source-node*
`--sourcenode=`*source-node*
`--sourcenode` *source-node*

デバイスを宛先ノードにレプリケートするソースノードを指定します。ノードは、ノード名またはノード ID のどちらででも指定できます。

`-S` オプションは、`replicate` サブコマンドでのみ有効です。

`-s` *state*[,…]
`--state=`*state*[,…]
`--state` *state*[,…]

指定したステータスのディスクパスのステータス情報を表示します。

`-s` オプションは、`status` サブコマンドだけで有効です。`-s` オプションを指定する場合、出力されるステータスは指定した *state* にあるディスクパスだけに制限されます。次に、*state* に可能な値を示します。

- `fail`
- `ok`
- `unknown`
- `unmonitored`

`-t`

レプリケーションデバイスタイプを指定します。このオプションは、`replicate` および `combine` サブコマンドとともに使用できます。

`-V`
`--version`

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンド、オペランド、またはほかのオプションは無視されます。-V オプションは、コマンドのバージョンを表示するだけです。その他の処理は行いません。

-v
--verbose

詳細な情報を標準出力に表示します。

このオプションは、このコマンドの任意の形式に指定できます。

次のオペランドがサポートされています。

device

デバイスの名前を指定します。指定できるデバイスは、ディスク、テープ、および CD-ROM ですが、これらだけに制限されるわけではありません。

サブコマンドが複数のデバイスを受け入れる場合、プラス記号 (+) を使用すると、すべてのデバイスを指定できます。

cldevice コマンドのサブコマンドはすべて、repair サブコマンドを除き、デバイスパスをオペランドとして受け付けます。repair サブコマンドは、オペランドとしてデバイス名だけを受け付けます。*device* 名には、フルグローバルパス名、デバイス名、または DID インスタンス番号のいずれかを指定できます。これらの形式のデバイス名の例は、それぞれ、/dev/did/dsk/d3、d3、および 3 です。詳細は、1479 ページのdid(7) のマニュアルページを参照してください。

デバイス名はまた、/dev/rdsk/c0t0d0s0 のような UNIX のフルパス名でもかまいません。

指定されたデバイスには、複数のノードからそのデバイスへのパスが複数存在する可能性もあります。-n オプションを使用しない場合、すべてのノードから指定されたデバイスへのパスがすべて選択されます。

monitor、unmonitor、および status サブコマンドは、ディスクデバイスをオペランドとして受け付けます。

このコマンドセットにあるすべてのコマンドの終了ステータスコードの完全なリストについては、19 ページのIntro(1CL) のマニュアルページを参照してください。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

0 CL_NOERR

エラーなし

1 CL_ENOMEM

十分なスワップ空間がありません。

3 CL_EINVAL

無効な引数

6 CL_EACCESS

アクセス権がありません

9 CL_ESTATE

オブジェクトの状態が不正です

15CL_EPROP

無効なプロパティーです

35 CL_EIO

I/O エラー

36 CL_ENOENT

そのようなオブジェクトはありません。

37 CL_EOP

操作が許可されていません

例 7 クラスタにあるすべてのディスクパスのモニタリング

次の例では、クラスタインフラストラクチャーにあるすべてのディスクパスのモニタリングを有効にする方法を示します。

```
# cldevice monitor +
```

例 8 単一のディスクパスのモニタリング

次の例では、ディスク `/dev/did/dsk/d3` へのパスが有効であるすべてのノードで、このパスのモニタリングを有効にする方法を示します。

```
# cldevice monitor /dev/did/dsk/d3
```

例 9 単一ノードのディスクパスをモニタリング

次の例では、ノード `phys-schost-2` でディスク `/dev/did/dsk/d4` と `/dev/did/dsk/d5` へのパスのモニタリングを有効にする方法を示します。

最初の例では、-n オプションを使用して、モニタリングをノード phys-schost-2 に接続されているディスクパスに制限し、さらに、モニタリングを指定されたデバイス d4 と d5 に制限しています。

```
# cldevice monitor -n phys-schost-2 d4 d5
```

2 番目の例では、モニターするディスクパスを *node*:*device* の名前である phys-schost-2:d4 と phys-schost-2:d5 で指定しています。

```
# cldevice monitor phys-schost-2:d4 phys-schost-2:d5
```

例 10 すべてのディスクパスとそのステータスの出力

次の例では、クラスタのすべてのディスクパスとそのステータスを出力する方法を示します。

```
# cldevice status
Device Instance             Node               Status
---------------             ----               ------
/dev/did/rdsk/d1            phys-schost-2      Unmonitored

/dev/did/rdsk/d2            phys-schost-2      Unmonitored

/dev/did/rdsk/d3            phys-schost-1      Ok
                            phys-schost-2      Ok

/dev/did/rdsk/d4            phys-schost-1      Ok
                            phys-schost-2      Ok

/dev/did/rdsk/d5            phys-schost-1      Unmonitored
```

例 11 ステータスが fail であるすべてのディスクパスの出力

次の例では、ノード phys-schost-2 上でモニターされており、ステータスが fail であるすべてのディスクパスを出力する方法を示します。

```
# cldevice status -s fail -n phys-schost-1
Device Instance             Node               Status
---------------             ----               ------
/dev/did/rdsk/d3            phys-schost-1      Fail

/dev/did/rdsk/d4            phys-schost-1      Fail
```

例 12 単一ノードからのすべてのディスクパスのステータスの出力

次の例では、ノード phys-schost-2 上でオンラインであるすべてのディスクパスのパスとステータスを出力する方法を示します。

```
# cldevice status -n phys-schost-1
```

```
Device Instance             Node                Status
---------------             ----                ------
/dev/did/rdsk/d3            phys-schost-1       Ok

/dev/did/rdsk/d4            phys-schost-1       Ok

/dev/did/rdsk/d5            phys-schost-1       Unmonitored
```

例 13 新しいデバイスのデバイス構成データベースへの追加

次の例では、このコマンドを実行したノード phys-schost-2 の現在のデバイス構成で、CCR データベースを更新する方法を示します。このコマンドは、クラスタのほかのノードに接続されているデバイスのデータベースは更新しません。

```
phys-schost-2# cldevice refresh
```

例 14 単一 DID でのデバイスの結合

次の例では、あるデバイスのパスを別のデバイスのパスと結合する方法を示します。このようにパスを結合すると、DID インスタンス番号が 1 つになり、対象先の DID インスタンス番号と同じになります。

```
# cldevice combine -t srdf -g devgrp1 -d 20 30
```

例 15 デバイスインスタンスのデバイスパスのリストの表示

次の例では、DID ドライバのインスタンス 3 に対応するすべてのデバイスのパスのリストを表示する方法を示します。

```
# cldevice list 3
d3
```

例 16 クラスタのすべてのデバイスパスのリストの表示

次の例では、任意のクラスタノードに接続されているすべてのデバイスのすべてのデバイスパスのリストを表示する方法を示します。

```
# cldevice list -v
DID Device          Full Device Path
----------          ----------------
d1                  phys-schost-1:/dev/rdsk/c0t0d0
d2                  phys-schost-1:/dev/rdsk/c0t1d0
d3                  phys-schost-1:/dev/rdsk/c1t8d0
d3                  phys-schost-2:/dev/rdsk/c1t8d0
d4                  phys-schost-1:/dev/rdsk/c1t9d0
d4                  phys-schost-2:/dev/rdsk/c1t9d0
```

```
d5                  phys-schost-1:/dev/rdsk/c1t10d0
d5                  phys-schost-2:/dev/rdsk/c1t10d0
d6                  phys-schost-1:/dev/rdsk/c1t11d0
d6                  phys-schost-2:/dev/rdsk/c1t11d0
d7                  phys-schost-2:/dev/rdsk/c0t0d0
d8                  phys-schost-2:/dev/rdsk/c0t1d0
```

例 17 デバイスに関する構成情報の表示

次の例では、デバイス `c4t8d0` に関する構成情報を表示する方法を示します。

```
# cldevice show /dev/rdsk/c4t8d0

=== DID Device Instances ===

DID Device Name:                                /dev/did/rdsk/d3
  Full Device Path:                               phys-schost1:/dev/rdsk/c4t8d0
  Full Device Path:                               phys-schost2:/dev/rdsk/c4t8d0
  Replication:                                    none
  default_fencing:                                nofencing
```

例 18 単一デバイスの SCSI プロトコルの設定

次の例では、デバイス `11` (インスタンス番号で指定) を SCSI-3 プロトコルに設定します。このデバイスは、構成された定足数デバイスではありません。

```
# cldevice set -p default_fencing=scsi3 11
```

例 19 PGR キーの最初のチェックを実行せずにデバイスのフェンシングをオフにする

次の例では、デバイス上のディスク `/dev/did/dsk/d5` のフェンシングをオフにします。このコマンドは、Persistent Group Reservation (PGR) キーの最初のチェックを*実行せず*、すべての PGR キーを削除して、フェンシングをオフにします。

```
# cldevice set -p default_fencing=nofencing-noscrub d5
```

Serial Advanced Technology Attachment (SATA) ディスクのような、SCSI をサポートしていないディスクを使用している場合は、フェンシングをオフにします。

例 20 2 ノードクラスタ `phys-schost` ですべてのデバイスのフェンシングをオフにする

次の例では、`phys-schost` という名前の 2 ノードクラスタですべてのデバイスのフェンシングをオフにします。

```
# cluster set -p global_fencing=nofencing
# cldevice set -p default_fencing=global -n phys-schost-1,phys-schost-2 d5
```

cluster コマンドおよび global_fencing プロパティーの詳細は、551 ページのcluster(1CL) のマニュアルページを参照してください。

Serial Advanced Technology Attachment (SATA) ディスクのような、SCSI をサポートしていないディスクを使用している場合は、フェンシングをオフにします。

例 21 デバイス名による修復手順の実行

次の例では、デバイス /dev/dsk/c1t4d0 に関連付けられていたデバイス識別子で修復手順を実行する方法を示します。このデバイスは新しいデバイスに置き換えられ、今では新しいデバイス識別子が関連付けられています。データベース中では、repair サブコマンドは、このインスタンス番号が現在新しいデバイス識別子に対応していることを記録しています。

```
# cldevice repair c1t4d0
```

例 22 インスタンス番号による修復手順の実行

次の例では、デバイス識別子で修復手順を実行する代替方法を示します。この例では、置き換えられるデバイスへのデバイスパスに関連付けられているインスタンス番号を指定しています。置き換えられたデバイスのインスタンス番号は 2 です。

```
# cldevice repair 2
```

例 23 グローバルデバイス名前空間の生成

次の例では、新しいグローバルデバイスを追加するか、または DID デバイスを新しいインスタンス番号に移動したあと、グローバルデバイス名前空間を設定する方法を示します。

```
# devfsadm
# cldevice populate
```

例 24 DID デバイスの移動

次の例では、移動元インスタンス 15 の DID インスタンスを新しい DID インスタンス 10 に移動し、グローバルデバイス名前空間を更新して、構成の変更を反映させます。

```
# cldevice rename 15:10
# devfsadm
# cldevice populate
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページの`Intro`(1CL), 551 ページの`cluster`(1CL), Unresolved link to " boot1M", Unresolved link to " devfsadm1M", 1447 ページの`clconfiguration`(5CL), Unresolved link to " rbac5", 1479 ページの`did`(7)

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のオプションを指定してこのコマンドを実行できます。

- `-?` (ヘルプ) オプション
- `-V` (バージョン) オプション

スーパーユーザー以外のユーザーがほかのサブコマンドを指定してこのコマンドを実行するには、RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
`check`	`solaris.cluster.read`
`clear`	`solaris.cluster.modify`
`combine`	`solaris.cluster.modify`
`export`	`solaris.cluster.read`
`list`	`solaris.cluster.read`
`monitor`	`solaris.cluster.modify`
`populate`	`solaris.cluster.modify`
`refresh`	`solaris.cluster.modify`
`rename`	`solaris.cluster.modify`
`repair`	`solaris.cluster.modify`
`replicate`	`solaris.cluster.modify`
`set`	`solaris.cluster.modify`
`show`	`solaris.cluster.read`
`status`	`solaris.cluster.read`
`unmonitor`	`solaris.cluster.modify`

ディスクパスのステータスの変化をログに記録するには、`syslogd` コマンドを使用します。

マルチポート式のテープドライブまたは CD-ROM ドライブはそれぞれ、個々の物理接続をするごとにネームスペースに表示されます。

名前

cldevice, cldev — Oracle Solaris Cluster デバイスの管理

```
/usr/cluster/bin/cldevice -V

/usr/cluster/bin/cldevice [subcommand] -?

/usr/cluster/bin/cldevice subcommand [options] -v [+ | device …]

/usr/cluster/bin/cldevice check [-n node[,…]] [+]

/usr/cluster/bin/cldevice clear [-n node[,…]] [+]

/usr/cluster/bin/cldevice combine -t replication-type -g
     replication-device-group -d destination-device device

/usr/cluster/bin/cldevice export [-o {- | configfile}] [-n node[,…]]
     [+ | device…]

/usr/cluster/bin/cldevice list [-n node[,…]] [+ | device …]

/usr/cluster/bin/cldevice monitor [-i {- | clconfigfile}] [-n
     node[,…]] {+ | disk-device …}

/usr/cluster/bin/cldevice populate

/usr/cluster/bin/cldevice refresh [-n node[,…]] [+]

/usr/cluster/bin/cldevice rename -d destination-device device

/usr/cluster/bin/cldevice repair [-n node[,…]]
     {+ | device …}

/usr/cluster/bin/cldevice replicate -t replication-type [-S
     source-node] -D destination-node [+]

/usr/cluster/bin/cldevice set
     -p default_fencing={global | pathcount | scsi3 | nofencing | nofencing-noscrub}
     [-n node[,…]] device …

/usr/cluster/bin/cldevice show [-n node[,…]] [+ | device …]

/usr/cluster/bin/cldevice status [-s state] [-n node[,…]] [+ |
     [disk-device ]]

/usr/cluster/bin/cldevice unmonitor [-i {- | clconfigfile}]
     [-n node[,…]] {+ | disk-device …}
```

`cldevice` コマンドは、Oracle Solaris Cluster 環境のデバイスを管理します。このコマンドは、Oracle Solaris Cluster デバイス識別子 (DID) 疑似デバイスドライバを管理し、ディスクデバイスパスをモニターするのに使用します。

- DID ドライバは、あるデバイスへの複数のパスが使用可能である場合でも一意のデバイス ID をそのデバイスに提供します。詳細は、1479 ページのdid(7) のマニュアルページを参照してください。
- ディスクパスとは、クラスタノードと物理ディスクまたは LUN ストレージデバイス間の接続のことです。ディスクパスには、Oracle Solaris カーネルドライバスタック、ホストバスアダプタ、および介在する任意のケーブル、スイッチ、またはネットワーク接続が含まれます。

`cldev` コマンドは、`cldevice` コマンドの短い形式です。どちらの形式のコマンドも使用できます。

`list` および `show` サブコマンドを除き、`cldevice` コマンドは、オンラインであり、かつクラスタモードにあるクラスタノードから実行する必要があります。

このコマンドの一般的な形式は次のとおりです。

`cldevice` [*subcommand*] [*options*] [*operands*]

subcommand を省略できるのは、*options* で `-?` オプションまたは `-V` オプションが指定されている場合のみです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式については、このマニュアルページの「オプション」セクションを参照してください。

詳細は、19 ページのIntro(1CL) のマニュアルページを参照してください。

このコマンドは、大域ゾーンだけで使用できます。

サブコマンド

サポートされるサブコマンドには次のものがあります。

`check`

デバイスの物理デバイスに対する整合性検査を、カーネル表現と比較して実行します。この整合性検査で問題が発見されると、エラーメッセージが表示されます。この処理は、すべてのデバイスが検査されるまで継続されます。

デフォルトでは、このサブコマンドは現在のノードだけに影響します。`-n` オプションは、別のノードに接続されているデバイスの検査処理を実行するのに使用します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

`clear`

現在のノードから切断された配下のデバイスへの DID 参照をすべて削除します。

デフォルトでは、このサブコマンドは現在のノードだけに影響します。-n オプションは、クリアー処理を実行する別のクラスタノードを指定するのに使用します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

combine

指定されたデバイスを指定された対象先に結合します。

combine サブコマンドは、ソースデバイスのパスを対象先デバイスのパスに結合します。このようにパスを結合すると、DID インスタンス番号が 1 つになり、対象先の DID インスタンス番号と同じになります。このサブコマンドは、SRDF を使用してレプリケートされる EMC LUN に対応する DID インスタンスを結合するために使用します。

combine サブコマンドは、ストレージベースの複製用に DID デバイスを手動で構成するのに使用できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

export

クラスタデバイスの構成情報をエクスポートします。

-o オプションでファイル名を指定する場合、構成情報はその新しいファイルに書き込まれます。-o オプションを指定しない場合、構成情報は標準出力に書き込まれます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

list

すべてのデバイスパスを表示します。

オペランドをまったく指定しない場合、あるいは、プラス記号 (+) をオペランドに指定する場合、すべてのデバイスが報告されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

monitor

指定したディスクパスのモニタリングをオンにします。

monitor サブコマンドは、ディスクデバイスだけに機能します。このサブコマンドは、テープなどのデバイスには影響しません。

monitor サブコマンドを使用して、ディスクパスモニタリングデーモン scdpmd をチューニングできます。構成ファイルの詳細は、1259 ページのscdpmd.conf(4) のマニュアルページを参照してください。

デフォルトでは、このサブコマンドはすべてのノードからのパスのモニタリングをオンにします。

-i オプションは、ディスクパスのモニタープロパティーを設定するクラスタ構成ファイルを指定するのに使用します。-i オプションは、指定されたファイルでモニタリン

グ対象のマークが付いているディスクパス上でディスクパスのモニタリングを開始します。ほかのディスクパスに変更は行われません。クラスタ構成ファイルの詳細は、1447 ページのclconfiguration(5CL) のマニュアルページを参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

`populate`

グローバルデバイス名前空間を生成します。

グローバルデバイス名前空間は /global ディレクトリの下にマウントされます。名前空間は、物理デバイスへの論理リンクの集合から構成されます。/dev/global ディレクトリはクラスタ内の各ノードから見えるので、どのノードからでも個々の物理デバイスが確認できます。したがって、クラスタ内のどのノードからでも、広域デバイスの名前空間に追加されたディスク、テープ、または CD-ROM にアクセスできます。

populate サブコマンドを使用すると、管理者は、システムをリブートせずに、新しいグローバルデバイスをグローバルデバイス名前空間に接続できます。これらのデバイスは、テープドライブ、CD-ROM ドライブ、またはディスクドライブでもかまいません。

populate サブコマンドを実行する前に、Unresolved link to " devfsadm1M" コマンドを実行する必要があります。代わりに、再構成リブートを実行して、グローバルデバイス名前空間を再構築し、新しいグローバルデバイスを接続してもかまいません。再構成リブートの詳細は、Unresolved link to " boot1M" のマニュアルページを参照してください。

populate サブコマンドは、現在のクラスタメンバーであるノードから実行してください。

populate サブコマンドはその作業をリモートノード上で非同期的に実行します。したがって、このコマンドを実行したノード上でコマンドが完了しても、すべてのクラスタノード上でこのコマンドが完了しているわけではありません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

`refresh`

クラスタノード上にある現在のデバイスツリーに対してデバイス構成情報を更新します。このコマンドにより、rdsk および rmt のデバイスツリーの完全な検索が行われます。このコマンドにより、新たに認識されたデバイス識別子ごとに、新しい DID インスタンス番号が割り当てられます。また、新たに認識されたデバイスごとに、新しいパスが追加されます。

デフォルトでは、このサブコマンドは現在のノードだけに影響します。-n オプションは、refresh サブコマンドと一緒に使用して、リフレッシュ処理を実行するクラスタノードを指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

`rename`

指定されたデバイスに新しい DID インスタンス番号をつけます。

このコマンドは、ソースデバイスの DID インスタンス番号に対応する DID デバイスパスを削除して、指定された宛先 DID インスタンス番号を使ってその DID デバイスパスを作成

し直します。このサブコマンドは、間違って変更された DID インスタンス番号を復元するのにも使用できます。

共有ストレージに接続されているすべてのクラスタノードで rename サブコマンドを実行したあとに、devfsadm および cldevice populate コマンドを実行し、グローバルデバイス名前空間を更新して、構成の変更を反映させます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

repair

指定されたデバイスに対して修復手順を実行します。

デフォルトでは、このサブコマンドは現在のノードだけに影響します。-n オプションは、修復処理を実行するクラスタノードを指定するのに使用します。

オペランドをまったく指定しない場合、あるいは、プラス記号 (+) をオペランドに指定する場合、このコマンドは、現在のノードに接続されているすべてのデバイスについての構成情報を更新します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

replicate

ストレージベースのレプリケーション機能で使用する DID デバイスを構成します。

注記 - replicate サブコマンドでは、DID インスタンスと EMC SRDF を結合する方法はサポートされていません。DID インスタンスと SRDF を結合するには、cldevice combine を使用します。

---

replicate サブコマンドは、ソースノード上の各 DID インスタンス番号を宛先ノード上の対応する DID インスタンス番号に結合します。レプリケートされたデバイスの各ペアは、単一の論理 DID デバイスにマージされます。

デフォルトでは、現在のノードがソースノードです。-S オプションは、別のソースノードを指定するのに使用します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

set

指定されたデバイスのプロパティーを変更します。

-p オプションは、変更するプロパティーを指定するのに使用します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

show

指定されたすべてのデバイスパスの構成レポートを表示します。

このレポートには、デバイスへのパスと、そのパスがモニター状態またはモニター解除状態のどちらにあるかが表示されます。

デフォルトでは、このサブコマンドはすべてのデバイスの構成情報を表示します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

`status`

指定されたすべてのディスクデバイスパスのステータスを表示します。

デフォルトでは、このサブコマンドはすべてのノードからのすべてのディスクパスのステータスを表示します。

`status` サブコマンドはディスクデバイスだけに機能します。このレポートには、テープなどのデバイスは報告されません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

`unmonitor`

コマンドのオペランドとして指定されたディスクパスのモニタリングをオフにします。

デフォルトでは、このサブコマンドはすべてのノードからのパスのモニタリングをオフにします。

`unmonitor` サブコマンドはディスクデバイスだけに機能します。このサブコマンドは、テープなどのデバイスには影響しません。

`-i` オプションは、ディスクパスのモニタリングをオフにするクラスタ構成ファイルを指定するのに使用します。ディスクパスのモニタリングがオフになるのは、指定されたファイルでモニタリング解除のマークが付いているディスクパスです。ほかのディスクパスに変更は行われません。詳細は、1447 ページの`clconfiguration`(5CL) のマニュアルページを参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。

次のオプションがサポートされています。

`-?`
`--help`

ヘルプ情報を表示します。

このオプションは単独でもサブコマンド付きでも使用できます。

- このオプションを単独で使用する場合、使用可能なサブコマンドのリストが出力されます。
- このオプションをサブコマンドを付けて使用する場合、そのサブコマンドの使用法オプションが出力されます。

このオプションを使用する場合、ほかの処理は実行されません。

-D *destination-node*
-destinationnode=*destination-node*
-destinationnode *destination-node*

デバイスをレプリケートする宛先ノードを指定します。ノードは、ノード名またはノード ID のどちらででも指定できます。

-D オプションは、replicate サブコマンドだけで有効です。

-d *destination-device*
--device=*destination-device*
--device *destination-device*

ストレージベースのレプリケーションを行うために、レプリケーション先デバイスの DID インスタンス番号を指定します。

DID インスタンス番号は -d オプションだけで使用します。ほかの形式の DID 名や UNIX のフルパス名は、対象先を指定するのには使用しません。

-d オプションは、rename および combine サブコマンドでのみ有効です。

-g *replication-device-group*

レプリケーションデバイスグループを指定します。このオプションは、combine サブコマンドとのみ使用できます。

-i {- | *clconfigfile*}
--input={- | *clconfigfile*}
--input {- | *clconfigfile*}

モニターするディスクパスまたはモニタリングしないディスクパスで使用される構成情報を指定します。この情報は、1447 ページのclconfiguration(5CL) のマニュアルページに定義されている形式に準拠している必要があります。この情報は、ファイルに含めることも、標準入力を介して指定することもできます。標準入力を指定するには、ファイル名の代わりに、マイナス記号 (-) を指定します。

-i オプションは、monitor および unmonitor サブコマンドでのみ有効です。

コマンドで指定するオプションは、構成ファイルで設定されている任意のオプションより優先されます。構成パラメータがクラスタ構成ファイルに存在しない場合、これらのパラメータをコマンド行で指定してください。

-n *node*[,…]
--node=*node*[,…]
--node *node*[,…]

サブコマンドが、-n オプションで指定されるノードからのディスクパスだけを含むことを指定します。ノードは、ノード名またはノード ID のどちらででも指定できます。

-o {- | *configfile*}
--output={- | *configfile*}
--output {- | *configfile*}

1447 ページの`clconfiguration`(5CL) のマニュアルページで定義されている形式で、ディスクパス構成情報を書き込みます。この情報は、ファイルまたは標準出力のどちらにでも書き込むことができます。

`-o` オプションは、`export` サブコマンドだけで有効です。

このオプションの引数としてファイル名を指定する場合、このコマンドは新しいファイルを作成して、そのファイルに構成情報を出力します。同じ名前のファイルがすでにある場合、このコマンドはエラーで終了します。既存のファイルに変更は行われません。

このオプションの引数としてマイナス記号 (-) を指定する場合、このコマンドは標準出力に構成情報を表示します。このコマンドのほかの標準出力はすべて抑制されます。

`-p default_fencing={global | pathcount| scsi3 | nofencing | nofencing-noscrub}`
`--property=default_fencing={global|pathcount|scsi3|nofencing|nofencing-noscrub}`
`--property default_fencing={global|pathcount|scsi3|nofencing|nofencing-noscrub}`

変更するプロパティーを指定します。

このオプションは `set` サブコマンドと一緒に使用して、次のプロパティーを変更します。

`default_fencing`

指定したデバイスについて、グローバルなデフォルトのフェンシングアルゴリズムを変更します。定足数デバイスとして構成されているデバイスのデフォルトのフェンシングアルゴリズムは変更できません。

デバイスのデフォルトのフェンシングアルゴリズムは、次の値のいずれかに設定できます。

`global`

グローバルなデフォルトのフェンシング設定を使用します。フェンシングのグローバルデフォルトの設定については、551 ページの`cluster`(1CL) のマニュアルページを参照してください。

`nofencing`

Persistent Group Reservation (PGR) キーをチェックし、いずれかのキーを削除したあとで、指定されたデバイスのフェンシングをオフにします。

注意 - Serial Advanced Technology Attachment (SATA) ディスクのような、SCSI をサポートしていないディスクを使用している場合は、フェンシングをオフにします。

`nofencing-noscrub`

最初に PGR キーをチェックまたは削除*せずに*、指定されたデバイスのフェンシングをオフにします。

注意 - Serial Advanced Technology Attachment (SATA) ディスクのような、SCSI をサポートしていないディスクを使用している場合は、フェンシングをオフにします。

`pathcount`

共有デバイスに接続されている DID パスの数でフェンシングプロトコルを決定します。

- 使用する DID パスが 3 未満のデバイスには、このコマンドは SCSI-2 プロトコルを設定します。
- 使用する DID パスが 3 以上のデバイスには、このコマンドは SCSI-3 プロトコルを設定します。

`scsi3`

SCSI-3 プロトコルを設定します。そのデバイスが SCSI-3 プロトコルをサポートしない場合、フェンシングプロトコルの設定は変更されません。

`-S` *source-node*
`--sourcenode=`*source-node*
`--sourcenode` *source-node*

デバイスを宛先ノードにレプリケートするソースノードを指定します。ノードは、ノード名またはノード ID のどちらででも指定できます。

`-S` オプションは、`replicate` サブコマンドでのみ有効です。

`-s` *state*[,…]
`--state=`*state*[,…]
`--state` *state*[,…]

指定したステータスのディスクパスのステータス情報を表示します。

`-s` オプションは、`status` サブコマンドだけで有効です。`-s` オプションを指定する場合、出力されるステータスは指定した *state* にあるディスクパスだけに制限されます。次に、*state* に可能な値を示します。

- `fail`
- `ok`
- `unknown`
- `unmonitored`

`-t`

レプリケーションデバイスタイプを指定します。このオプションは、`replicate` および `combine` サブコマンドとともに使用できます。

`-V`
`--version`

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンド、オペランド、またはほかのオプションは無視されます。-V オプションは、コマンドのバージョンを表示するだけです。その他の処理は行いません。

-v
--verbose

詳細な情報を標準出力に表示します。

このオプションは、このコマンドの任意の形式に指定できます。

次のオペランドがサポートされています。

device

デバイスの名前を指定します。指定できるデバイスは、ディスク、テープ、および CD-ROM ですが、これらだけに制限されるわけではありません。

サブコマンドが複数のデバイスを受け入れる場合、プラス記号 (+) を使用すると、すべてのデバイスを指定できます。

cldevice コマンドのサブコマンドはすべて、repair サブコマンドを除き、デバイスパスをオペランドとして受け付けます。repair サブコマンドは、オペランドとしてデバイス名だけを受け付けます。*device* 名には、フルグローバルパス名、デバイス名、または DID インスタンス番号のいずれかを指定できます。これらの形式のデバイス名の例は、それぞれ、/dev/did/dsk/d3、d3、および 3 です。詳細は、1479 ページのdid(7) のマニュアルページを参照してください。

デバイス名はまた、/dev/rdsk/c0t0d0s0 のような UNIX のフルパス名でもかまいません。

指定されたデバイスには、複数のノードからそのデバイスへのパスが複数存在する可能性もあります。-n オプションを使用しない場合、すべてのノードから指定されたデバイスへのパスがすべて選択されます。

monitor、unmonitor、および status サブコマンドは、ディスクデバイスをオペランドとして受け付けます。

このコマンドセットにあるすべてのコマンドの終了ステータスコードの完全なリストについては、19 ページのIntro(1CL) のマニュアルページを参照してください。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

0 CL_NOERR

エラーなし

1 CL_ENOMEM

十分なスワップ空間がありません。

3 CL_EINVAL

無効な引数

6 CL_EACCESS

アクセス権がありません

9 CL_ESTATE

オブジェクトの状態が不正です

15CL_EPROP

無効なプロパティーです

35 CL_EIO

I/O エラー

36 CL_ENOENT

そのようなオブジェクトはありません。

37 CL_EOP

操作が許可されていません

例 25 クラスタにあるすべてのディスクパスのモニタリング

次の例では、クラスタインフラストラクチャーにあるすべてのディスクパスのモニタリングを有効にする方法を示します。

```
# cldevice monitor +
```

例 26 単一のディスクパスのモニタリング

次の例では、ディスク `/dev/did/dsk/d3` へのパスが有効であるすべてのノードで、このパスのモニタリングを有効にする方法を示します。

```
# cldevice monitor /dev/did/dsk/d3
```

例 27 単一ノードのディスクパスをモニタリング

次の例では、ノード `phys-schost-2` でディスク `/dev/did/dsk/d4` と `/dev/did/dsk/d5` へのパスのモニタリングを有効にする方法を示します。

最初の例では、-n オプションを使用して、モニタリングをノード phys-schost-2 に接続されているディスクパスに制限し、さらに、モニタリングを指定されたデバイス d4 と d5 に制限しています。

```
# cldevice monitor -n phys-schost-2 d4 d5
```

2 番目の例では、モニターするディスクパスを *node*:*device* の名前である phys-schost-2:d4 と phys-schost-2:d5 で指定しています。

```
# cldevice monitor phys-schost-2:d4 phys-schost-2:d5
```

例 28 すべてのディスクパスとそのステータスの出力

次の例では、クラスタのすべてのディスクパスとそのステータスを出力する方法を示します。

```
# cldevice status
Device Instance             Node               Status
---------------             ----               ------
/dev/did/rdsk/d1            phys-schost-2      Unmonitored

/dev/did/rdsk/d2            phys-schost-2      Unmonitored

/dev/did/rdsk/d3            phys-schost-1      Ok
                            phys-schost-2      Ok

/dev/did/rdsk/d4            phys-schost-1      Ok
                            phys-schost-2      Ok

/dev/did/rdsk/d5            phys-schost-1      Unmonitored
```

例 29 ステータスが fail であるすべてのディスクパスの出力

次の例では、ノード phys-schost-2 上でモニターされており、ステータスが fail であるすべてのディスクパスを出力する方法を示します。

```
# cldevice status -s fail -n phys-schost-1
Device Instance             Node               Status
---------------             ----               ------
/dev/did/rdsk/d3            phys-schost-1      Fail

/dev/did/rdsk/d4            phys-schost-1      Fail
```

例 30 単一ノードからのすべてのディスクパスのステータスの出力

次の例では、ノード phys-schost-2 上でオンラインであるすべてのディスクパスのパスとステータスを出力する方法を示します。

```
# cldevice status -n phys-schost-1
```

```
Device Instance             Node                Status
---------------             ----                ------
/dev/did/rdsk/d3            phys-schost-1       Ok

/dev/did/rdsk/d4            phys-schost-1       Ok

/dev/did/rdsk/d5            phys-schost-1       Unmonitored
```

例 31 新しいデバイスのデバイス構成データベースへの追加

次の例では、このコマンドを実行したノード phys-schost-2 の現在のデバイス構成で、CCR データベースを更新する方法を示します。このコマンドは、クラスタのほかのノードに接続されているデバイスのデータベースは更新しません。

```
phys-schost-2# cldevice refresh
```

例 32 単一 DID でのデバイスの結合

次の例では、あるデバイスのパスを別のデバイスのパスと結合する方法を示します。このようにパスを結合すると、DID インスタンス番号が 1 つになり、対象先の DID インスタンス番号と同じになります。

```
# cldevice combine -t srdf -g devgrp1 -d 20 30
```

例 33 デバイスインスタンスのデバイスパスのリストの表示

次の例では、DID ドライバのインスタンス 3 に対応するすべてのデバイスのパスのリストを表示する方法を示します。

```
# cldevice list 3
d3
```

例 34 クラスタのすべてのデバイスパスのリストの表示

次の例では、任意のクラスタノードに接続されているすべてのデバイスのすべてのデバイスパスのリストを表示する方法を示します。

```
# cldevice list -v
DID Device          Full Device Path
----------          ----------------
d1                  phys-schost-1:/dev/rdsk/c0t0d0
d2                  phys-schost-1:/dev/rdsk/c0t1d0
d3                  phys-schost-1:/dev/rdsk/c1t8d0
d3                  phys-schost-2:/dev/rdsk/c1t8d0
d4                  phys-schost-1:/dev/rdsk/c1t9d0
d4                  phys-schost-2:/dev/rdsk/c1t9d0
```

```
d5                  phys-schost-1:/dev/rdsk/c1t10d0
d5                  phys-schost-2:/dev/rdsk/c1t10d0
d6                  phys-schost-1:/dev/rdsk/c1t11d0
d6                  phys-schost-2:/dev/rdsk/c1t11d0
d7                  phys-schost-2:/dev/rdsk/c0t0d0
d8                  phys-schost-2:/dev/rdsk/c0t1d0
```

例 35 デバイスに関する構成情報の表示

次の例では、デバイス `c4t8d0` に関する構成情報を表示する方法を示します。

```
# cldevice show /dev/rdsk/c4t8d0

=== DID Device Instances ===

DID Device Name:                                /dev/did/rdsk/d3
  Full Device Path:                               phys-schost1:/dev/rdsk/c4t8d0
  Full Device Path:                               phys-schost2:/dev/rdsk/c4t8d0
  Replication:                                    none
  default_fencing:                                nofencing
```

例 36 単一デバイスの SCSI プロトコルの設定

次の例では、デバイス `11` (インスタンス番号で指定) を SCSI-3 プロトコルに設定します。このデバイスは、構成された定足数デバイスではありません。

```
# cldevice set -p default_fencing=scsi3 11
```

例 37 PGR キーの最初のチェックを実行せずにデバイスのフェンシングをオフにする

次の例では、デバイス上のディスク `/dev/did/dsk/d5` のフェンシングをオフにします。このコマンドは、Persistent Group Reservation (PGR) キーの最初のチェックを*実行せず*、すべての PGR キーを削除して、フェンシングをオフにします。

```
# cldevice set -p default_fencing=nofencing-noscrub d5
```

Serial Advanced Technology Attachment (SATA) ディスクのような、SCSI をサポートしていないディスクを使用している場合は、フェンシングをオフにします。

例 38 2 ノードクラスタ `phys-schost` ですべてのデバイスのフェンシングをオフにする

次の例では、`phys-schost` という名前の 2 ノードクラスタですべてのデバイスのフェンシングをオフにします。

```
# cluster set -p global_fencing=nofencing
# cldevice set -p default_fencing=global -n phys-schost-1,phys-schost-2 d5
```

cluster コマンドおよび global_fencing プロパティーの詳細は、551 ページのcluster(1CL) のマニュアルページを参照してください。

Serial Advanced Technology Attachment (SATA) ディスクのような、SCSI をサポートしていないディスクを使用している場合は、フェンシングをオフにします。

例 39 デバイス名による修復手順の実行

次の例では、デバイス /dev/dsk/c1t4d0 に関連付けられていたデバイス識別子で修復手順を実行する方法を示します。このデバイスは新しいデバイスに置き換えられ、今では新しいデバイス識別子が関連付けられています。データベース中では、repair サブコマンドは、このインスタンス番号が現在新しいデバイス識別子に対応していることを記録しています。

```
# cldevice repair c1t4d0
```

例 40 インスタンス番号による修復手順の実行

次の例では、デバイス識別子で修復手順を実行する代替方法を示します。この例では、置き換えられるデバイスへのデバイスパスに関連付けられているインスタンス番号を指定しています。置き換えられたデバイスのインスタンス番号は 2 です。

```
# cldevice repair 2
```

例 41 グローバルデバイス名前空間の生成

次の例では、新しいグローバルデバイスを追加するか、または DID デバイスを新しいインスタンス番号に移動したあと、グローバルデバイス名前空間を設定する方法を示します。

```
# devfsadm
# cldevice populate
```

例 42 DID デバイスの移動

次の例では、移動元インスタンス 15 の DID インスタンスを新しい DID インスタンス 10 に移動し、グローバルデバイス名前空間を更新して、構成の変更を反映させます。

```
# cldevice rename 15:10
# devfsadm
# cldevice populate
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページの`Intro`(1CL), 551 ページの`cluster`(1CL), Unresolved link to " boot1M", Unresolved link to " devfsadm1M", 1447 ページの`clconfiguration`(5CL), Unresolved link to " rbac5", 1479 ページの`did`(7)

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のオプションを指定してこのコマンドを実行できます。

- `-?` (ヘルプ) オプション
- `-V` (バージョン) オプション

スーパーユーザー以外のユーザーがほかのサブコマンドを指定してこのコマンドを実行するには、RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
`check`	`solaris.cluster.read`
`clear`	`solaris.cluster.modify`
`combine`	`solaris.cluster.modify`
`export`	`solaris.cluster.read`
`list`	`solaris.cluster.read`
`monitor`	`solaris.cluster.modify`
`populate`	`solaris.cluster.modify`
`refresh`	`solaris.cluster.modify`
`rename`	`solaris.cluster.modify`
`repair`	`solaris.cluster.modify`
`replicate`	`solaris.cluster.modify`
`set`	`solaris.cluster.modify`
`show`	`solaris.cluster.read`
`status`	`solaris.cluster.read`
`unmonitor`	`solaris.cluster.modify`

ディスクパスのステータスの変化をログに記録するには、`syslogd` コマンドを使用します。

マルチポート式のテープドライブまたは CD-ROM ドライブはそれぞれ、個々の物理接続をするごとにネームスペースに表示されます。

名前

cldevicegroup, cldg — Oracle Solaris Cluster デバイスグループの管理

```
/usr/cluster/bin/cldevicegroup -V

/usr/cluster/bin/cldevicegroup [subcommand] -?

/usr/cluster/bin/cldevicegroup subcommand [options] -v
     [devicegroup …]

/usr/cluster/bin/cldevicegroup add-device -d device
     [,…] devicegroup

/usr/cluster/bin/cldevicegroup add-node -n node[,…] [-t
     devicegroup-type[,…]] {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup create -n node[,…] -t
     devicegroup-type [-d device[,…]] [-p name=value]
     devicegroup ...

/usr/cluster/bin/cldevicegroup create -i {- | clconfigfile} [-d
     device[,…]] [-n node[,…]] [-p name=value] [-t
     devicegroup-type[,…]] {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup delete [-t devicegroup-type[,…]]
     {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup disable [-t devicegroup-type[,…]]
     {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup enable [-t devicegroup-type[,…]]
     {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup export [-n node[,…]] [-o
     {- | clconfigfile}] [-t devicegroup-type[,…]]
     {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup list [-n node[,…]] [-t
     devicegroup-type[,…]] [+ | devicegroup ...]

/usr/cluster/bin/cldevicegroup offline [-t devicegroup-type[,…]]
     {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup online [-e] [-n node] [-t
     devicegroup-type[,…]] {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup remove-device -d device
     [,…] devicegroup

/usr/cluster/bin/cldevicegroup remove-node -n node[,…]
     [-t devicegroup-type[,…]] {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup set -p name=value [-p name=value]…
     [-d device[,…]] [-n node[,…]] [-t devicegroup-type[,…]]
```

```
     {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup show [-n node[,…]] [-t
     devicegroup-type[,…]] [+ | devicegroup ...]

/usr/cluster/bin/cldevicegroup status [-n node[,…]] [-t
     devicegroup-type[,…]] [+ | devicegroup ...]

/usr/cluster/bin/cldevicegroup switch -n node [-t
     devicegroup-type[,…]] {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup sync [-t devicegroup-type[,…]]
     {+ | devicegroup ...}
```

`cldevicegroup` コマンドは、Oracle Solaris Cluster デバイスグループを管理します。`cldg` コマンドは、`cldevicegroup` コマンドの短縮形式です。次の 2 つのコマンドは同一です。どちらの形式のコマンドも使用できます。

このコマンドの一般的な形式は次のとおりです。

`cldevicegroup` [*subcommand*] [*options*] [*operands*]

subcommand を省略できるのは、*options* で `-?` オプションまたは `-V` オプションが指定されている場合のみです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式については、このマニュアルページの「オプション」セクションを参照してください。

`list`、`show`、および `status` を除いて、ほとんどのサブコマンドには少なくとも 1 つのオペランドが必要です。多くのサブコマンドは、適用できる**すべての**オブジェクトを示すオペランドとしてプラス記号 (+) を受け入れます。詳細は、このマニュアルページの SYNOPSIS およびほかのセクションを参照してください。

それぞれのサブコマンドはすべてのデバイスグループタイプに使用できますが、次のサブコマンドを除きます。

- `add-device` および `remove-device` サブコマンドは、`rawdisk` タイプでのみ有効です。
- `add-node`、`create`、`delete`、および `remove-node` サブコマンドは、`rawdisk` タイプでのみ有効です。

このコマンドは、大域ゾーンだけで使用できます。

サブコマンド

サポートされるサブコマンドには次のものがあります。

add-device

新しいメンバーディスクデバイスを既存の raw ディスクデバイスグループに追加します。

add-device サブコマンドは、rawdisk タイプの既存のデバイスグループだけで使用できます。デバイスグループタイプについての詳細は、-t オプションの説明を参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

ディスクデバイスを raw ディスクデバイスグループから削除する方法については、remove-device サブコマンドの説明を参照してください。

add-node

新しいノードを既存のデバイスグループに追加します。

このサブコマンドは、rawdisk デバイスグループタイプだけをサポートします。Oracle Solaris Cluster コマンドを使用して、svm または sds デバイスグループにノードを追加することはできません。代わりに、Solaris Volume Manager コマンドを使用して、Solaris Volume Manager ディスクセットにノードを追加してください。ディスクセットは、svm または sds デバイスグループとして、Oracle Solaris Cluster ソフトウェアに自動的に登録されます。デバイスグループタイプについての詳細は、-t オプションの説明を参照してください。

デバイスグループの preferenced プロパティーが true に設定されている場合、このサブコマンドはそのデバイスグループに使用できません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

ノードをデバイスグループから削除する方法については、remove-node サブコマンドの説明を参照してください。

create

新しいデバイスグループを作成します。

このサブコマンドは、rawdisk デバイスグループタイプだけをサポートします。Oracle Solaris Cluser コマンドでは、svm または sds デバイスグループは作成できません。その代わりに、Solaris Volume Manager コマンドを使用して、Solaris Volume Manager ディスクセットを作成します。ディスクセットは、svm または sds デバイスグループとして、Oracle Solaris Cluster ソフトウェアに自動的に登録されます。デバイスグループタイプについての詳細は、-t オプションの説明を参照してください。

-i オプションで構成ファイルを指定する場合、プラス記号 (+) をオペランドとして指定できます。このオペランドを使用するとき、このコマンドは、構成ファイルで指定された、まだ存在していないすべてのデバイスグループを作成します。

rawdisk デバイスグループタイプの場合、-d オプションを create サブコマンドと一緒に使用して、1 つまたは複数のデバイスをデバイスグループに指定します。デバイスを指定するとき、コマンドの呼び出しごとに 1 つの -d オプションを使用します。-i オプションを使用しないかぎり、1 つのコマンド呼び出しで複数の raw ディスクデバイスグループは作成できません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

デバイスグループを削除する方法については、delete サブコマンドの説明を参照してください。

delete

デバイスグループを削除します。

このサブコマンドは、rawdisk デバイスグループタイプだけをサポートします。

Oracle Solaris Cluster コマンドでは、svm または sds デバイスグループは削除できません。その代わりに、svm または sds デバイスグループを削除するには、Solaris Volume Manager コマンドを使用して、配下の Solaris Volume Manager ディスクセットを削除します。

削除する前に、デバイスグループはオフラインになっている必要があります。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

デバイスグループを作成する方法については、create サブコマンドの説明を参照してください。

disable

オフラインのデバイスグループを無効にします。

デバイスグループの無効な状態は、リブートしても変わりません。

オンラインのデバイスグループを無効にするには、まず offline サブコマンドを使用して、そのデバイスグループをオフラインにする必要があります。

デバイスグループがオンラインである場合、disable アクションは失敗して、指定されたデバイスグループを無効にできません。

switch サブコマンドまたは online サブコマンドを使用して、無効になっているデバイスグループをオンラインにすることはできません。まず、enable サブコマンドを使用して、デバイスグループの無効な状態をクリアしてください。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

デバイスグループを有効にする方法については、enable サブコマンドの説明を参照してください。

enable

デバイスグループを有効にします。

デバイスグループの無効な状態は、リブートしても変わりません。

無効なデバイスグループをオンラインにする前には、まず、enable サブコマンドを使用して、デバイスグループの無効な状態をクリアしてください。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

デバイスグループを無効にする方法については、disable サブコマンドの説明を参照してください。

export

デバイスグループ構成情報をエクスポートします。

-o オプションでファイル名を指定する場合、構成情報はその新しいファイルに書き込まれます。-o オプションを指定しない場合、出力は標準出力に書き込まれます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

list

デバイスグループのリストを表示します。

デフォルトでは、このサブコマンドは、autogen プロパティーが false に設定されているクラスタにあるすべてのデバイスグループのリストを表示します。クラスタ内のすべてのデバイスグループを表示するには、-v オプションも指定します。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

offline

デバイスグループをオフラインにします。

デバイスグループがオンラインである場合は、disable サブコマンドを実行する前に、offline サブコマンドを実行することによって、そのデバイスグループをオフラインにする必要があります。

オフラインのデバイスグループを起動するには、次のアクションのいずれかを実行します。

- 明示的な online サブコマンドまたは switch サブコマンドを発行します。

- そのデバイスグループ内のデバイスにアクセスします。
- そのデバイスグループに依存するファイルシステムをマウントします。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

デバイスグループをオンラインにする方法については、online サブコマンドの説明を参照してください。

online

あらかじめ指定したノードのデバイスグループをオンラインにします。

デバイスグループが無効な場合、そのデバイスグループをオンラインにする前に、次のいずれかの方法により、そのデバイスグループを有効にしてください。

- -e オプションを online サブコマンドと一緒に使用します。
- online サブコマンドを実行する前に、enable サブコマンドを実行します。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

デバイスグループをオフラインにする方法については、offline サブコマンドの説明を参照してください。

remove-device

メンバーディスクデバイスを raw ディスクデバイスグループから削除します。

remove-device サブコマンドは、rawdisk デバイスグループタイプだけに有効です。このサブコマンドは、svm または sds デバイスグループタイプに対しては有効ではありません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

ディスクデバイスを raw ディスクデバイスグループに追加する方法については、add-device サブコマンドの説明を参照してください。

remove-node

既存のデバイスグループからノードを削除します。

このサブコマンドは、rawdisk デバイスグループタイプだけをサポートします。Oracle Solaris Cluster コマンドを使用して、svm または sds デバイスグループからノードを削除す

ることはできません。代わりに、Solaris Volume Manager コマンドを使用して、Solaris Volume Manager ディスクセットからノードを削除してください。ディスクセットは、svm または sds デバイスグループとして、Oracle Solaris Cluster ソフトウェアに自動的に登録されます。デバイスグループタイプについての詳細は、-t オプションの説明を参照してください。

preferenced プロパティーが true に設定されている場合、remove-node サブコマンドはそのデバイスグループに使用できません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

ノードをデバイスグループに追加する方法については、add-node サブコマンドの説明を参照してください。

set

デバイスグループに関連付けられている属性を変更します。

rawdisk デバイスグループタイプの場合、-d オプションを set サブコマンドと一緒に使用して、指定したデバイスグループのメンバーディスクデバイスの新しいリストを指定します。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

show

デバイスグループの構成レポートを作成します。

デフォルトでは、このサブコマンドは、autogen プロパティーが false に設定されているクラスタにあるすべてのデバイスグループについて報告します。クラスタ内のすべてのデバイスグループを表示するには、-v オプションも指定します。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

status

デバイスグループのステータスレポートを作成します。

デフォルトでは、このサブコマンドは、autogen プロパティーが false に設定されているクラスタにあるすべてのデバイスグループについて報告します。クラスタ内のすべてのデバイスグループを表示するには、-v オプションも指定します。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

switch

Oracle Solaris Cluster 構成内のプライマリノードから別のノードにデバイスグループを転送します。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

sync

クラスタリングソフトウェアとデバイスグループ情報の同期を取ります。

このサブコマンドは、所有者、グループ、またはアクセス権などのボリューム属性を変更する場合に常に使用します。

また、sync サブコマンドを使用して、デバイスグループ構成をレプリケーション構成または非レプリケーション構成に変更します。

レプリケーション用に構成されたディスクを含む Solaris Volume Manager ディスクセットを作成したあとは、対応する svm または sds デバイスグループに sync サブコマンドを実行してください。Solaris Volume Manager ディスクセットは、svm または sds デバイスグループとして Oracle Solaris Cluster ソフトウェアに自動的に登録されますが、その時点でレプリケーション情報の同期はとられません。

新たに作成した rawdisk デバイスグループタイプの場合、そのディスクのレプリケーション情報を手動で同期する必要はありません。raw ディスクデバイスグループを Oracle Solaris Cluster ソフトウェアに登録すると、ディスク上のすべてのレプリケーション情報が自動的に検出されます。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

次のオプションがサポートされています。

-?
--help

ヘルプ情報を表示します。

このオプションは、単独でもサブコマンド付きでも使用できます。

- このオプションを単独で使用する場合、使用可能なサブコマンドのリストが出力されます。
- このオプションをサブコマンドを付けて使用する場合、そのサブコマンドの使用法オプションが出力されます。

このオプションを使用すると、ほかの処理は実行されません。

-d *device*[,…]
--device=*device*[,…]
--device *device*[,…]

指定されている raw ディスクデバイスグループのメンバーになるディスクデバイスのリストを指定します。

-d オプションは、タイプ rawdisk のデバイスグループに対する create および set サブコマンドでのみ有効です。常に、ノードリスト全体を指定してください。このオプションを使用して、メンバーディスクリストに個々のディスクを追加したり、メンバーディスクリストから個々のディスクを削除したりすることはできません。

ディスクは、DID グローバルデバイス名 (たとえば、d3) のみで指定します。詳細は、1479 ページのdid(7) のマニュアルページを参照してください。

-e
--enable

デバイスグループを有効にします。このオプションは、online サブコマンドと一緒に使用するときだけに有効です。

指定したデバイスグループがすでに有効である場合、-e オプションは無視され、このコマンドはデバイスグループをオンラインにする処理に進みます。

-i {- | *clconfigfile*}
--input={- | *clconfigfile*}
--input {- | *clconfigfile*}

デバイスグループの作成に使用する構成情報を指定します。この情報は、1447 ページのclconfiguration(5CL) のマニュアルページに定義されている形式に準拠している必要があります。この情報は、ファイルに含めることも、標準入力を介して指定することもできます。標準入力を指定するには、ファイル名の代わりにマイナス記号 (-) を指定します。

-i オプションは、完全修飾したデバイスグループリストに含まれるデバイスグループだけに影響します。

コマンドで指定するオプションは、構成ファイルで設定されている任意のオプションより優先されます。構成パラメータがクラスタ構成ファイルに存在しない場合、これらのパラメータをコマンド行で指定してください。

-n *node*[,…]
--node=*node*[,…]
--node *node*[,…]

ノードまたはノードリストを指定します。

デフォルトでは、ノードリストの順番は、デバイスグループのプライマリノードとして引き継がれるべき優先順位を示します。例外は、Oracle Solaris Cluster の制御範囲の外にあるローカル専用ディスクグループの場合で、プライマリノードとセカンダリノードの概念は当てはまりません。

デバイスグループの `preferenced` プロパティーが `false` に設定されている場合、ノードリストの順番は無視されます。その代わりに、グループ内で最初にデバイスにアクセスしたノードが自動的に、そのグループのプライマリノードになります。デバイスグループノードリストの `preferenced` プロパティーの設定については、`-p` オプションを参照してください。

`-n` オプションは、`svm` または `sds` デバイスグループのノードリストを指定するのには使用できません。その代わりに、Solaris ボリュームマネージャーのコマンドまたはユーティリティーを使用して、配下のディスクセットのノードリストを指定してください。

`create` および `set` サブコマンドは `-n` オプションを使用して、デバイスグループタイプ `rawdisk` 専用の潜在的なプライマリノードのリストを指定します。デバイスグループの完全なノードリストを指定してください。`-n` オプションを使用して、ノードリストに個別ノードを追加したり、ノードリストから個別ノードを削除したりすることはできません。

`switch` サブコマンドは `-n` オプションを使用して、新しいデバイスグループのプライマリノードとして単一のノードを指定します。

`export`、`list`、`show`、および `status` サブコマンドは `-n` オプションを使用して、指定されたノード上でオンラインでないデバイスグループを出力から除外します。

プライマリノードとセカンダリノードの概念は、Oracle Solaris Cluster の制御範囲の外にある `localonly` ディスクグループには適用されません。

`-o` {- | *clconfigfile*}
`--output`={- | *clconfigfile*}
`--output` {- | *clconfigfile*}

1447 ページの`clconfiguration`(5CL) のマニュアルページで規定されている形式で、デバイスグループ構成を表示します。この情報は、ファイルまたは標準出力のどちらにでも書き込むことができます。

このオプションの引数としてファイル名を指定する場合、このコマンドは新しいファイルを作成して、そのファイルに構成情報を出力します。同じ名前のファイルがすでにある場合、このコマンドはエラーで終了します。既存のファイルに変更は行われません。

このオプションの引数としてマイナス記号 (`-`) を指定すると、このコマンドは標準出力に構成情報を表示します。このコマンドのほかの標準出力はすべて抑制されます。

`-o` オプションは、`export` サブコマンドだけで有効です。

-p *name= value*
--property=*name =value*
--property *name=value*

デバイスグループプロパティーに値を設定します。

-p オプションは、create および set サブコマンドでのみ有効です。-p*name-=value* の複数のインスタンスが許可されます。

次のプロパティーがサポートされています。

autogen

autogen プロパティーは true または false の値を持つことができます。手動で作成したデバイスグループの場合、デフォルトは false です。システム作成デバイスグループの場合、デフォルトは true です。

autogen プロパティーは、list、show、および status サブコマンドのインジケータです。これらのサブコマンドは、-v オプションを使用しないかぎり、autogen プロパティーが true に設定されているデバイスをリストに載せません。

このプロパティーは、デバイスグループタイプ rawdisk だけに有効です。デバイスグループタイプについての詳細は、-t オプションを参照してください。

failback

failback プロパティーは true または false の値を持つことができます。デフォルトは false です。

failback プロパティーは、デバイスグループのプライマリノードがクラスタメンバーシップから切り離され、あとで再結合された場合のシステムの動作を指定します。

デバイスグループのプライマリノードがクラスタメンバーシップから切り離された段階で、デバイスグループはセカンダリノードに処理を継続します。そして障害の発生したノードがクラスタメンバーシップに再結合すると、デバイスグループは、そのままセカンダリノードにマスターされ続けるか、あるいは、オリジナルのプライマリノードにフェイルバックするか、のいずれかの動作を取ります。

- failback プロパティーが true に設定されている場合、デバイスグループは、オリジナルのプライマリノードにマスターされるようになります。
- failback プロパティーが false に設定されている場合、デバイスグループは、セカンダリノードにマスターされ続けます。

デバイスグループ作成中、failback プロパティーはデフォルトでは無効にされます。set 処理中、failback プロパティーは変更されません。

localonly

localonly プロパティーは true または false の値を持つことができます。デフォルトは false です。

localonly プロパティーは、rawdisk タイプのディスクグループに対してのみ有効です。

ディスクグループが特定のノードだけでマスターされるようにする場合、localonly=true というプロパティーの設定で、ディスクグループを構成します。ローカル専用ディスクグループは、Oracle Solaris Cluster ソフトウェアの制御範囲の外にあります。ローカル専用ディスクグループのノードリストには、1 つのノードだけ指定できます。ディスクグループの localonly プロパティーを true に設定するとき、そのディスクグループのノードリストに指定できるのは 1 つのノードだけです。

numsecondaries

numsecondaries プロパティーには、0 より大きく、ノードリスト内のノードの合計数より小さい整数値を指定してください。デフォルトは 1 です。

このプロパティーの設定は、デバイスグループのセカンダリノードの希望数を動的に変更する場合に使用できます。デバイスグループのセカンダリノードは、現在のプライマリノードに問題が生じた場合に、新たなプライマリノードとしての機能を引き継ぐことができます。

numsecondaries プロパティーを使用することで、可用性のレベルを維持しながら、デバイスグループのセカンダリノードの指定数を変更できます。デバイスグループのセカンダリノードリストからノードを削除すると、そのノードはプライマリノードとして機能を引き継ぐことはできなくなります。

numsecondaries プロパティーは、現在クラスタモードであるデバイスグループのノードだけに適用されます。このノードは、デバイスグループの preferenced プロパティーと一緒に使用できる必要があります。デバイスグループの preferenced プロパティーが true に設定されている場合、まず、優先順位のもっとも低いノードがセカンダリノードリストから削除されます。優先フラグがついているノードがデバイスグループの中に 1 つもないと、クラスタはノードをランダムに選択し、削除します。

デバイスグループにおけるセカンダリノードの実際の数が目標のレベル未満になると、セカンダリノードリストから削除された資格のある各ノードがリストに戻されます。セカンダリノードリストに戻される資格のある各ノードは、次の条件のすべてに適合する必要があります。

- そのノードは現在クラスタにあります。
- そのノードはそのデバイスグループに属しています。
- そのノードは現在、プライマリノードまたはセカンダリノードではありません。

二次ノードリストに戻されるのは、デバイスグループの優先順位がもっとも高いノードからです。セカンダリノードの目標数に到達するまで、優先順位が高いノードから順にセカンダリノードリストに戻されます。

あるノードがクラスタに参加したとき、そのノードの優先順位がそのデバイスグループの既存のセカンダリノードよりも高い場合、優先順位が低いノードがセカンダリノードリストから削除されます。削除されるノードは新たに追加されたノードにより置換されます。この置換は、クラスタ内に存在するセカンダリノードの実際の数が目標数を超えたときだけ発生します。

デバイスグループのノードリストの preferenced プロパティーを設定する方法についての詳細は、preferenced プロパティーを参照してください。

`preferenced`

`preferenced` プロパティーは `true` または `false` の値を持つことができます。デフォルトは `true` です。

デバイスグループの作成中、`preferenced` プロパティーが `true` に設定されている場合、ノードリストはノードの優先順位も示します。ノードの優先順位は、各ノードが、デバイスグループのプライマリノードとしてテイクオーバーしようとする順序を決定します。

デバイスグループの作成中、このプロパティーが `false` に設定されている場合、デバイスグループ内のデバイスに最初にアクセスしたノードが自動的にプライマリノードになります。指定されたノードリスト内のノードの順番は意味を持ちません。ノードリストを再指定せずにこのプロパティーに `true` を再設定しても、ノードの順番は再度有効にはなりません。

`preferenced=true` プロパティーを指定し、かつ `-n` オプションを使用してデバイスグループのノードリスト全体を優先順位どおりに指定しないかぎり、`set` 操作中にノードの優先順位は変更されません。

`-t` *devicegroup-type*[,…]
`--type=`*devicegroup-type*[,…]
`--type` *devicegroup-type*[,…]

デバイスグループタイプまたデバイスグループタイプのリストを指定します。

`create` サブコマンドの場合、1 つのデバイスグループタイプだけを指定できます。これにより、このオプションで指定したタイプのデバイスグループが作成されます。

これ以外で `-t` オプションを受け付けることができるサブコマンドの場合、このオプションにより、コマンドに指定したデバイスグループのリストが指定したデバイスグループタイプだけに限定されます。

必ずしもすべてのサブコマンドとオプションがすべてのデバイスグループタイプに対して有効であるわけではありません。たとえば、`create` サブコマンドは `rawdisk` デバイスグループタイプに対してのみ有効であり、`svm` または `sds` デバイスグループタイプには有効ではありません。

`-t` オプションでサポートされるデバイスグループタイプは次のとおりです。

`rawdisk`

raw ディスクデバイスグループを指定します。

raw ディスクとは、ボリュームマネージャーのボリュームやメタデバイスの一部ではないディスクのことです。raw ディスクデバイスグループを使用すると、デバイスグループ内にディスクセットを定義できます。デフォルトでは、システムブート時に、raw ディスクデバイスグループが、構成内のデバイス ID (DID) 擬似ドライバデバイスごとに作成されます。慣例により、初期化時には、raw ディスクデバイスグループの名前が割り当てられます。これらの名前は、DID デバイス名から派生されます。raw ディスクデバイスグループに追加されるノードごとに、`cldevicegroup` コマンドは、グループ内のすべてのデバイスがそのノードに物理的に接続されていることを確認します。

`create` サブコマンドは、raw ディスクデバイスグループを作成して、複数のディスクデバイスをデバイスグループに追加します。新しい raw ディスクデバイスグループを作成

する前に、新しいデバイスグループに追加する各デバイスを、ブート時にそのデバイスが作成されたデバイスグループから削除してください。これにより、これらのデバイスが含まれる新しい raw ディスクデバイスグループを作成できます。-n オプションで潜在的なプライマリノード優先順位リストを指定することに加えて、これらのデバイスのリストを -d オプションで指定します。

指定した単一のノード上でデバイスグループをマスターするには、-p オプションを使用して、localonly=true というプロパティーの設定で、デバイスグループを構成します。ローカル専用デバイスグループを作成するとき、ノードリストには 1 つのノードだけを指定できます。

delete サブコマンドは、デバイスグループ名をクラスタデバイスグループ構成から削除します。

set サブコマンドは、次の変更を raw ディスクデバイスグループに行います。

- 潜在的なプライマリノードの優先順位を変更する
- 新しいノードリストを指定する
- フェイルバックを有効または無効にする
- 二次ノードの目標数を設定する
- 複数のグローバルデバイスをデバイスグループに追加する

raw ディスクデバイス名が raw ディスクデバイスグループに登録されている場合、その raw ディスクデバイス名は Solaris Volume Manager デバイスグループには登録できません。

sds

もともと Solstice DiskSuite™ ソフトウェアで作成されていたデバイスグループを指定します。複数所有者ディスクセットを除いて、このデバイスグループタイプは Solaris Volume Manager デバイスグループタイプ svm と同等です。詳細は、svm デバイスグループタイプの説明を参照してください。

svm

Solaris ボリュームマネージャーデバイスグループを指定します。

Solaris ボリュームマネージャーデバイスグループは、次のコンポーネントで定義されます。

- 名前
- グループにアクセスできるノード
- ディスクセット内のデバイスのグローバルなリスト
- 潜在的なプライマリノードの優先順位やフェイルバックの動作などのアクションを制御するプロパティーの集合

Solaris Volume Manager には、多重ホストまたは共有ディスクセットの概念があります。共有ディスクセットとは、2 つ以上のホストとディスクドライブからなるグループのことです。このディスクドライブはすべてのホストからアクセス可能であり、すべての

ホスト上でデバイス名が同じです。こうしたデバイス名の統一は、raw ディスクデバイスを用いてディスクセットを構築することにより行います。デバイス ID 疑似ドライバ (DID) を使用することで、多重ホストに使用するディスクの名前はクラスタ内で整合性が保たれます。Solaris Volume Manager デバイスグループのノードリストに構成できるのは、すでにディスクセットの一部として構成されているホストだけです。共有ディスクセットにドライブを追加する場合、そのドライブが、ほかの何らかの共有ディスクセットに属していてはいけません。

Solaris Volume Manager の `metaset` コマンドはディスクセットを作成して、そのディスクセットを Solaris Volume Manager デバイスグループとして、Oracle Solaris Cluster ソフトウェアに自動的に登録します。デバイスグループを作成したあと、`cldevicegroup` コマンドの `set` サブコマンドを使用して、ノード優先順位リストと `preferenced`、`failback`、および `numsecondaries` プロパティーを設定する必要があります。

1 つのデバイスグループには、1 つの Solaris Volume Manager ディスクセットだけを割り当てることができます。デバイスグループ名は常に、ディスクセットの名前と一致する必要があります。

Solaris Volume Manager デバイスグループにノードを追加または削除するのに、`add-node` または `remove-node` サブコマンドは使用できません。その代わりに、Solaris Volume Manager の `metaset` コマンドを使用して、配下の Solaris Volume Manager ディスクセットで、ノードを追加または削除します。

Solaris Volume Manager のデバイスグループをクラスタ構成から削除するのに、`delete` サブコマンドは使用できません。その代わりに、Solaris Volume Manager の `metaset` コマンドを使用して、配下の Solaris Volume Manager ディスクセットを削除します。

`export`、`list`、`show`、`status`、および `sync` サブコマンドだけが、Solaris Volume Manager 複数所有者ディスクセットで機能します。Solaris Volume Manager デバイスグループの配下のディスクセットを追加または削除するには、Solaris Volume Manager のコマンドまたはユーティリティーを使用してください。

`-V`
`--version`

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンドやオペランドなどのオプションは無視されます。`-V` オプションは、コマンドのバージョンを表示するだけです。その他の処理は行いません。

`-v`
`--verbose`

詳細なメッセージを標準出力に表示します。

このオプションは、このコマンドの任意の形式に使用できます。

次のオペランドを指定できます。

devicegroup　　デバイスグループを指定します。

`cldevicegroup` コマンドは、Oracle Solaris Cluster デバイスグループ名だけをオペランドとして受け付けます。複数のデバイスグループ名を受け付けるほとんどの形式のコマンドの場合、プラス記号 (+) を使用するとすべての可能なデバイスグループを指定できます。

注記 - + オペランドは、手動で作成されたデバイスグループだけを含み、`autogen` プロパティーに `true` が設定されている自動的に作成されたデバイスグループをすべて無視します。Oracle Solaris Cluster ソフトウェアは、システムがブートされるたびに、このようなデバイスグループを自動的に作成します。これらの「非表示の」デバイスグループにコマンドを適用するには、各デバイスグループを明示的に指定する必要があります。

このコマンドセットにあるすべてのコマンドの終了ステータスコードの完全なリストについては、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (`CL_NOERR`) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

`0 CL_NOERR`
　エラーなし

`1 CL_ENOMEM`
　十分なスワップ空間がありません。

`3 CL_EINVAL`
　無効な引数

`6 CL_EACCESS`
　アクセス権がありません

`35 CL_EIO`
　I/O エラー

`36 CL_ENOENT`
　そのようなオブジェクトはありません。

`39 CL_EEXIST`
　オブジェクトは存在します。

例 43 デバイスグループの変更

次の例では、デバイスグループ devgrp1 の preference プロパティーを true に設定して、numsecondaries プロパティーを 2 に設定する方法を示します。このコマンドはまた、希望ノードリスト phys-schost-1,phys-schost-2,phys-schost-3 も指定しています。

```
# cldevicegroup set -p preferenced=true -p numsecondaries=2 \
-n phys-schost-1,phys-schost-2,phys-schost-3 devgrp1
```

例 44 raw ディスクデバイスグループの変更

次の例では、既存の raw ディスクデバイスグループ rawdevgrp1 を変更する方法を示します。このコマンドは、新しいメンバーデバイスリスト内のデバイス d3 および d4 を指定し、localonly 属性を true に設定します。ノード phys-schost-1 は、ローカル専用 raw ディスクデバイスグループで許可される唯一のプライマリノードです。

```
# cldevicegroup set -d d3,d4 \
-p localonly=true -n phys-schost-1 rawdevgrp1
```

例 45 デバイスグループの numsecondaries 属性のリセット

次の例では、デバイスグループ devgrp1 の numsecondaries 属性に何も値を指定しないことによって、適切なシステムデフォルト値にリセットする方法を示します。

```
# cldevicegroup set -p numsecondaries= devgrp1
```

例 46 デバイスグループのスイッチオーバー

次の例では、デバイスグループ devgrp1 を新しいマスターノード phys-schost-2 に切り替える方法を示します。

```
# cldevicegroup switch -n phys-schost-2 devgrp1
```

例 47 デバイスグループの無効化

次の例では、デバイスグループ devgrp1 を無効にする方法を示します。

```
# cldevicegroup disable devgrp1
```

例 48 デバイスグループのオフライン

次の例では、デバイスグループ devgrp1 をオフラインにしてから無効にする方法を示します。

```
# cldevicegroup offline devgrp1
# cldevicegroup disable devgrp1
```

例 49 デバイスグループのそのプライマリノードでのオンライン化

次の例では、デバイスグループ devgrp1 をそのデフォルトのプライマリノードでオンラインにする方法を示します。このコマンドはまず、デバイスグループを有効にします。

```
# cldevicegroup online -e devgrp1
```

例 50 デバイスグループの指定されたノードでのオンライン化

次の例では、デバイスグループ devgrp1 を新しいプライマリノードとして指定された phys-schost-2 でオンラインにする方法を示します。

```
# cldevicegroup switch -n phys-schost-2 devgrp1
```

例 51 新しいノードのデバイスグループへの追加

次の例では、新しいノード phys-schost-3 をデバイスグループ devgrp1 に追加する方法を示します。このデバイスグループは、デバイスグループタイプ svm ではありません。

```
# cldevicegroup add-node -n phys-schost-3 devgrp1
```

例 52 デバイスグループの削除

次の例では、デバイスグループ devgrp1 を Oracle Solaris Cluster 構成から削除する方法を示します。このデバイスグループは、デバイスグループタイプ svm ではありません。

```
# cldevicegroup delete devgrp1
```

例 53 レプリケーション情報とデバイスグループ構成の同期化

次の例では、デバイスグループ devgrp1 のディスクが使用するレプリケーション構成を Oracle Solaris Cluster ソフトウェアに認識させる方法を示します。

```
# cldevicegroup sync devgrp1
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core

属性タイプ	属性値
インタフェースの安定性	発展中

19 ページのIntro(1CL), 61 ページのcldevice(1CL), 551 ページのcluster(1CL), Unresolved link to " metaset1M", 1447 ページのclconfiguration(5CL), Unresolved link to " rbac5", 1479 ページのdid(7)

Unresolved link to " Oracle Solaris Cluster システム管理 "

スーパーユーザーはこのコマンドのすべての形式を実行できます。

また、任意のユーザーは次のオプションを指定してこのコマンドを実行できます。

- -? (ヘルプ) オプション
- -V (バージョン) オプション

スーパーユーザー以外のユーザーがほかのサブコマンドを指定してこのコマンドを実行するには、RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
add-device	solaris.cluster.modify
add-node	solaris.cluster.modify
create	solaris.cluster.modify
delete	solaris.cluster.modify
disable	solaris.cluster.modify
enable	solaris.cluster.modify
export	solaris.cluster.read
list	solaris.cluster.read
offline	solaris.cluster.admin
online	solaris.cluster.admin
remove-device	solaris.cluster.modify
remove-node	solaris.cluster.modify
set	solaris.cluster.modify
show	solaris.cluster.read

サブコマンド	RBAC の承認
status	solaris.cluster.read
switch	solaris.cluster.modify
sync	solaris.cluster.admin

名前

cldevicegroup, cldg — Oracle Solaris Cluster デバイスグループの管理

```
/usr/cluster/bin/cldevicegroup -V

/usr/cluster/bin/cldevicegroup [subcommand] -?

/usr/cluster/bin/cldevicegroup subcommand [options] -v
     [devicegroup …]

/usr/cluster/bin/cldevicegroup add-device -d device
     [,…] devicegroup

/usr/cluster/bin/cldevicegroup add-node -n node[,…] [-t
     devicegroup-type[,…]] {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup create -n node[,…] -t
     devicegroup-type [-d device[,…]] [-p name=value]
     devicegroup ...

/usr/cluster/bin/cldevicegroup create -i {- | clconfigfile} [-d
     device[,…]] [-n node[,…]] [-p name=value] [-t
     devicegroup-type[,…]] {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup delete [-t devicegroup-type[,…]]
     {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup disable [-t devicegroup-type[,…]]
     {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup enable [-t devicegroup-type[,…]]
     {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup export [-n node[,…]] [-o
     {- | clconfigfile}] [-t devicegroup-type[,…]]
     {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup list [-n node[,…]] [-t
     devicegroup-type[,…]] [+ | devicegroup ...]

/usr/cluster/bin/cldevicegroup offline [-t devicegroup-type[,…]]
     {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup online [-e] [-n node] [-t
     devicegroup-type[,…]] {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup remove-device -d device
     [,…] devicegroup

/usr/cluster/bin/cldevicegroup remove-node -n node[,…]
     [-t devicegroup-type[,…]] {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup set -p name=value [-p name=value]…
     [-d device[,…]] [-n node[,…]] [-t devicegroup-type[,…]]
```

```
     {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup show [-n node[,…]] [-t
     devicegroup-type[,…]] [+ | devicegroup ...]

/usr/cluster/bin/cldevicegroup status [-n node[,…]] [-t
     devicegroup-type[,…]] [+ | devicegroup ...]

/usr/cluster/bin/cldevicegroup switch -n node [-t
     devicegroup-type[,…]] {+ | devicegroup ...}

/usr/cluster/bin/cldevicegroup sync [-t devicegroup-type[,…]]
     {+ | devicegroup ...}
```

`cldevicegroup` コマンドは、Oracle Solaris Cluster デバイスグループを管理します。`cldg` コマンドは、`cldevicegroup` コマンドの短縮形式です。次の 2 つのコマンドは同一です。どちらの形式のコマンドも使用できます。

このコマンドの一般的な形式は次のとおりです。

`cldevicegroup` [*subcommand*] [*options*] [*operands*]

subcommand を省略できるのは、*options* で `-?` オプションまたは `-V` オプションが指定されている場合のみです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式については、このマニュアルページの「オプション」セクションを参照してください。

`list`、`show`、および `status` を除いて、ほとんどのサブコマンドには少なくとも 1 つのオペランドが必要です。多くのサブコマンドは、適用できる**すべての**オブジェクトを示すオペランドとしてプラス記号 (+) を受け入れます。詳細は、このマニュアルページの SYNOPSIS およびほかのセクションを参照してください。

それぞれのサブコマンドはすべてのデバイスグループタイプに使用できますが、次のサブコマンドを除きます。

- `add-device` および `remove-device` サブコマンドは、`rawdisk` タイプでのみ有効です。
- `add-node`、`create`、`delete`、および `remove-node` サブコマンドは、`rawdisk` タイプでのみ有効です。

このコマンドは、大域ゾーンだけで使用できます。

サブコマンド

サポートされるサブコマンドには次のものがあります。

add-device

新しいメンバーディスクデバイスを既存の raw ディスクデバイスグループに追加します。

add-device サブコマンドは、rawdisk タイプの既存のデバイスグループだけで使用できます。デバイスグループタイプについての詳細は、-t オプションの説明を参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

ディスクデバイスを raw ディスクデバイスグループから削除する方法については、remove-device サブコマンドの説明を参照してください。

add-node

新しいノードを既存のデバイスグループに追加します。

このサブコマンドは、rawdisk デバイスグループタイプだけをサポートします。Oracle Solaris Cluster コマンドを使用して、svm または sds デバイスグループにノードを追加することはできません。代わりに、Solaris Volume Manager コマンドを使用して、Solaris Volume Manager ディスクセットにノードを追加してください。ディスクセットは、svm または sds デバイスグループとして、Oracle Solaris Cluster ソフトウェアに自動的に登録されます。デバイスグループタイプについての詳細は、-t オプションの説明を参照してください。

デバイスグループの preferenced プロパティーが true に設定されている場合、このサブコマンドはそのデバイスグループに使用できません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

ノードをデバイスグループから削除する方法については、remove-node サブコマンドの説明を参照してください。

create

新しいデバイスグループを作成します。

このサブコマンドは、rawdisk デバイスグループタイプだけをサポートします。Oracle Solaris Cluser コマンドでは、svm または sds デバイスグループは作成できません。その代わりに、Solaris Volume Manager コマンドを使用して、Solaris Volume Manager ディスクセットを作成します。ディスクセットは、svm または sds デバイスグループとして、Oracle Solaris Cluster ソフトウェアに自動的に登録されます。デバイスグループタイプについての詳細は、-t オプションの説明を参照してください。

-i オプションで構成ファイルを指定する場合、プラス記号 (+) をオペランドとして指定できます。このオペランドを使用するとき、このコマンドは、構成ファイルで指定された、まだ存在していないすべてのデバイスグループを作成します。

rawdisk デバイスグループタイプの場合、-d オプションを create サブコマンドと一緒に使用して、1 つまたは複数のデバイスをデバイスグループに指定します。デバイスを指定するとき、コマンドの呼び出しごとに 1 つの -d オプションを使用します。-i オプションを使用しないかぎり、1 つのコマンド呼び出しで複数の raw ディスクデバイスグループは作成できません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

デバイスグループを削除する方法については、delete サブコマンドの説明を参照してください。

delete

デバイスグループを削除します。

このサブコマンドは、rawdisk デバイスグループタイプだけをサポートします。

Oracle Solaris Cluster コマンドでは、svm または sds デバイスグループは削除できません。その代わりに、svm または sds デバイスグループを削除するには、Solaris Volume Manager コマンドを使用して、配下の Solaris Volume Manager ディスクセットを削除します。

削除する前に、デバイスグループはオフラインになっている必要があります。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

デバイスグループを作成する方法については、create サブコマンドの説明を参照してください。

disable

オフラインのデバイスグループを無効にします。

デバイスグループの無効な状態は、リブートしても変わりません。

オンラインのデバイスグループを無効にするには、まず offline サブコマンドを使用して、そのデバイスグループをオフラインにする必要があります。

デバイスグループがオンラインである場合、disable アクションは失敗して、指定されたデバイスグループを無効にできません。

switch サブコマンドまたは online サブコマンドを使用して、無効になっているデバイスグループをオンラインにすることはできません。まず、enable サブコマンドを使用して、デバイスグループの無効な状態をクリアしてください。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

デバイスグループを有効にする方法については、enable サブコマンドの説明を参照してください。

enable

デバイスグループを有効にします。

デバイスグループの無効な状態は、リブートしても変わりません。

無効なデバイスグループをオンラインにする前には、まず、enable サブコマンドを使用して、デバイスグループの無効な状態をクリアしてください。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

デバイスグループを無効にする方法については、disable サブコマンドの説明を参照してください。

export

デバイスグループ構成情報をエクスポートします。

-o オプションでファイル名を指定する場合、構成情報はその新しいファイルに書き込まれます。-o オプションを指定しない場合、出力は標準出力に書き込まれます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

list

デバイスグループのリストを表示します。

デフォルトでは、このサブコマンドは、autogen プロパティーが false に設定されているクラスタにあるすべてのデバイスグループのリストを表示します。クラスタ内のすべてのデバイスグループを表示するには、-v オプションも指定します。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

offline

デバイスグループをオフラインにします。

デバイスグループがオンラインである場合は、disable サブコマンドを実行する前に、offline サブコマンドを実行することによって、そのデバイスグループをオフラインにする必要があります。

オフラインのデバイスグループを起動するには、次のアクションのいずれかを実行します。

- 明示的な online サブコマンドまたは switch サブコマンドを発行します。

- そのデバイスグループ内のデバイスにアクセスします。
- そのデバイスグループに依存するファイルシステムをマウントします。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

デバイスグループをオンラインにする方法については、online サブコマンドの説明を参照してください。

online

あらかじめ指定したノードのデバイスグループをオンラインにします。

デバイスグループが無効な場合、そのデバイスグループをオンラインにする前に、次のいずれかの方法により、そのデバイスグループを有効にしてください。

- -e オプションを online サブコマンドと一緒に使用します。
- online サブコマンドを実行する前に、enable サブコマンドを実行します。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

デバイスグループをオフラインにする方法については、offline サブコマンドの説明を参照してください。

remove-device

メンバーディスクデバイスを raw ディスクデバイスグループから削除します。

remove-device サブコマンドは、rawdisk デバイスグループタイプだけに有効です。このサブコマンドは、svm または sds デバイスグループタイプに対しては有効ではありません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

ディスクデバイスを raw ディスクデバイスグループに追加する方法については、add-device サブコマンドの説明を参照してください。

remove-node

既存のデバイスグループからノードを削除します。

このサブコマンドは、rawdisk デバイスグループタイプだけをサポートします。Oracle Solaris Cluster コマンドを使用して、svm または sds デバイスグループからノードを削除す

ることはできません。代わりに、Solaris Volume Manager コマンドを使用して、Solaris Volume Manager ディスクセットからノードを削除してください。ディスクセットは、svm または sds デバイスグループとして、Oracle Solaris Cluster ソフトウェアに自動的に登録されます。デバイスグループタイプについての詳細は、-t オプションの説明を参照してください。

preferenced プロパティーが true に設定されている場合、remove-node サブコマンドはそのデバイスグループに使用できません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

ノードをデバイスグループに追加する方法については、add-node サブコマンドの説明を参照してください。

set

デバイスグループに関連付けられている属性を変更します。

rawdisk デバイスグループタイプの場合、-d オプションを set サブコマンドと一緒に使用して、指定したデバイスグループのメンバーディスクデバイスの新しいリストを指定します。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

show

デバイスグループの構成レポートを作成します。

デフォルトでは、このサブコマンドは、autogen プロパティーが false に設定されているクラスタにあるすべてのデバイスグループについて報告します。クラスタ内のすべてのデバイスグループを表示するには、-v オプションも指定します。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

status

デバイスグループのステータスレポートを作成します。

デフォルトでは、このサブコマンドは、autogen プロパティーが false に設定されているクラスタにあるすべてのデバイスグループについて報告します。クラスタ内のすべてのデバイスグループを表示するには、-v オプションも指定します。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

switch

Oracle Solaris Cluster 構成内のプライマリノードから別のノードにデバイスグループを転送します。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

sync

クラスタリングソフトウェアとデバイスグループ情報の同期を取ります。

このサブコマンドは、所有者、グループ、またはアクセス権などのボリューム属性を変更する場合に常に使用します。

また、sync サブコマンドを使用して、デバイスグループ構成をレプリケーション構成または非レプリケーション構成に変更します。

レプリケーション用に構成されたディスクを含む Solaris Volume Manager ディスクセットを作成したあとは、対応する svm または sds デバイスグループに sync サブコマンドを実行してください。Solaris Volume Manager ディスクセットは、svm または sds デバイスグループとして Oracle Solaris Cluster ソフトウェアに自動的に登録されますが、その時点でレプリケーション情報の同期はとられません。

新たに作成した rawdisk デバイスグループタイプの場合、そのディスクのレプリケーション情報を手動で同期する必要はありません。raw ディスクデバイスグループを Oracle Solaris Cluster ソフトウェアに登録すると、ディスク上のすべてのレプリケーション情報が自動的に検出されます。

+ オペランドを指定すると、autogen プロパティーが false に設定されているデバイスグループだけが影響を受けます。ブート時にシステムによって自動的に作成される、autogen プロパティーが true に設定されたデバイスグループにこのコマンドを適用するには、デバイスグループごとに明示的に指定してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

次のオプションがサポートされています。

-?
--help

ヘルプ情報を表示します。

このオプションは、単独でもサブコマンド付きでも使用できます。

- このオプションを単独で使用する場合、使用可能なサブコマンドのリストが出力されます。
- このオプションをサブコマンドを付けて使用する場合、そのサブコマンドの使用法オプションが出力されます。

このオプションを使用すると、ほかの処理は実行されません。

-d *device*[,…]
--device=*device*[,…]
--device *device*[,…]

指定されている raw ディスクデバイスグループのメンバーになるディスクデバイスのリストを指定します。

-d オプションは、タイプ rawdisk のデバイスグループに対する create および set サブコマンドでのみ有効です。常に、ノードリスト全体を指定してください。このオプションを使用して、メンバーディスクリストに個々のディスクを追加したり、メンバーディスクリストから個々のディスクを削除したりすることはできません。

ディスクは、DID グローバルデバイス名 (たとえば、d3) のみで指定します。詳細は、1479 ページのdid(7) のマニュアルページを参照してください。

-e
--enable

デバイスグループを有効にします。このオプションは、online サブコマンドと一緒に使用するときだけに有効です。

指定したデバイスグループがすでに有効である場合、-e オプションは無視され、このコマンドはデバイスグループをオンラインにする処理に進みます。

-i {- | *clconfigfile*}
--input={- | *clconfigfile*}
--input {- | *clconfigfile*}

デバイスグループの作成に使用する構成情報を指定します。この情報は、1447 ページのclconfiguration(5CL) のマニュアルページに定義されている形式に準拠している必要があります。この情報は、ファイルに含めることも、標準入力を介して指定することもできます。標準入力を指定するには、ファイル名の代わりにマイナス記号 (-) を指定します。

-i オプションは、完全修飾したデバイスグループリストに含まれるデバイスグループだけに影響します。

コマンドで指定するオプションは、構成ファイルで設定されている任意のオプションより優先されます。構成パラメータがクラスタ構成ファイルに存在しない場合、これらのパラメータをコマンド行で指定してください。

`-n` *node*[,…]
`--node=`*node*[,…]
`--node` *node*[,…]

ノードまたはノードリストを指定します。

デフォルトでは、ノードリストの順番は、デバイスグループのプライマリノードとして引き継がれるべき優先順位を示します。例外は、Oracle Solaris Cluster の制御範囲の外にあるローカル専用ディスクグループの場合で、プライマリノードとセカンダリノードの概念は当てはまりません。

デバイスグループの `preferenced` プロパティーが `false` に設定されている場合、ノードリストの順番は無視されます。その代わりに、グループ内で最初にデバイスにアクセスしたノードが自動的に、そのグループのプライマリノードになります。デバイスグループノードリストの `preferenced` プロパティーの設定については、`-p` オプションを参照してください。

`-n` オプションは、`svm` または `sds` デバイスグループのノードリストを指定するのには使用できません。その代わりに、Solaris ボリュームマネージャーのコマンドまたはユーティリティーを使用して、配下のディスクセットのノードリストを指定してください。

`create` および `set` サブコマンドは `-n` オプションを使用して、デバイスグループタイプ `rawdisk` 専用の潜在的なプライマリノードのリストを指定します。デバイスグループの完全なノードリストを指定してください。`-n` オプションを使用して、ノードリストに個別ノードを追加したり、ノードリストから個別ノードを削除したりすることはできません。

`switch` サブコマンドは `-n` オプションを使用して、新しいデバイスグループのプライマリノードとして単一のノードを指定します。

`export`、`list`、`show`、および `status` サブコマンドは `-n` オプションを使用して、指定されたノード上でオンラインでないデバイスグループを出力から除外します。

プライマリノードとセカンダリノードの概念は、Oracle Solaris Cluster の制御範囲の外にある `localonly` ディスクグループには適用されません。

`-o` {- | *clconfigfile*}
`--output=`{- | *clconfigfile*}
`--output` {- | *clconfigfile*}

1447 ページの`clconfiguration`(5CL) のマニュアルページで規定されている形式で、デバイスグループ構成を表示します。この情報は、ファイルまたは標準出力のどちらにでも書き込むことができます。

このオプションの引数としてファイル名を指定する場合、このコマンドは新しいファイルを作成して、そのファイルに構成情報を出力します。同じ名前のファイルがすでにある場合、このコマンドはエラーで終了します。既存のファイルに変更は行われません。

このオプションの引数としてマイナス記号 (`-`) を指定すると、このコマンドは標準出力に構成情報を表示します。このコマンドのほかの標準出力はすべて抑制されます。

`-o` オプションは、`export` サブコマンドだけで有効です。

-p *name= value*
--property=*name =value*
--property *name=value*

デバイスグループプロパティーに値を設定します。

-p オプションは、create および set サブコマンドでのみ有効です。-p*name-=value* の複数のインスタンスが許可されます。

次のプロパティーがサポートされています。

autogen

autogen プロパティーは true または false の値を持つことができます。手動で作成したデバイスグループの場合、デフォルトは false です。システム作成デバイスグループの場合、デフォルトは true です。

autogen プロパティーは、list、show、および status サブコマンドのインジケータです。これらのサブコマンドは、-v オプションを使用しないかぎり、autogen プロパティーが true に設定されているデバイスをリストに載せません。

このプロパティーは、デバイスグループタイプ rawdisk だけに有効です。デバイスグループタイプについての詳細は、-t オプションを参照してください。

failback

failback プロパティーは true または false の値を持つことができます。デフォルトは false です。

failback プロパティーは、デバイスグループのプライマリノードがクラスタメンバーシップから切り離され、あとで再結合された場合のシステムの動作を指定します。

デバイスグループのプライマリノードがクラスタメンバーシップから切り離された段階で、デバイスグループはセカンダリノードに処理を継続します。そして障害の発生したノードがクラスタメンバーシップに再結合すると、デバイスグループは、そのままセカンダリノードにマスターされ続けるか、あるいは、オリジナルのプライマリノードにフェイルバックするか、のいずれかの動作を取ります。

- failback プロパティーが true に設定されている場合、デバイスグループは、オリジナルのプライマリノードにマスターされるようになります。
- failback プロパティーが false に設定されている場合、デバイスグループは、セカンダリノードにマスターされ続けます。

デバイスグループ作成中、failback プロパティーはデフォルトでは無効にされます。set 処理中、failback プロパティーは変更されません。

localonly

localonly プロパティーは true または false の値を持つことができます。デフォルトは false です。

localonly プロパティーは、rawdisk タイプのディスクグループに対してのみ有効です。

ディスクグループが特定のノードだけでマスターされるようにする場合、localonly=true というプロパティーの設定で、ディスクグループを構成します。ローカル専用ディスクグループは、Oracle Solaris Cluster ソフトウェアの制御範囲の外にあります。ローカル専用ディスクグループのノードリストには、1 つのノードだけ指定できます。ディスクグループの localonly プロパティーを true に設定するとき、そのディスクグループのノードリストに指定できるのは 1 つのノードだけです。

numsecondaries

numsecondaries プロパティーには、0 より大きく、ノードリスト内のノードの合計数より小さい整数値を指定してください。デフォルトは 1 です。

このプロパティーの設定は、デバイスグループのセカンダリノードの希望数を動的に変更する場合に使用できます。デバイスグループのセカンダリノードは、現在のプライマリノードに問題が生じた場合に、新たなプライマリノードとしての機能を引き継ぐことができます。

numsecondaries プロパティーを使用することで、可用性のレベルを維持しながら、デバイスグループのセカンダリノードの指定数を変更できます。デバイスグループのセカンダリノードリストからノードを削除すると、そのノードはプライマリノードとして機能を引き継ぐことはできなくなります。

numsecondaries プロパティーは、現在クラスタモードであるデバイスグループのノードだけに適用されます。このノードは、デバイスグループの preferenced プロパティーと一緒に使用できる必要があります。デバイスグループの preferenced プロパティーが true に設定されている場合、まず、優先順位のもっとも低いノードがセカンダリノードリストから削除されます。優先フラグがついているノードがデバイスグループの中に 1 つもないと、クラスタはノードをランダムに選択し、削除します。

デバイスグループにおけるセカンダリノードの実際の数が目標のレベル未満になると、セカンダリノードリストから削除された資格のある各ノードがリストに戻されます。セカンダリノードリストに戻される資格のある各ノードは、次の条件のすべてに適合する必要があります。

- そのノードは現在クラスタにあります。
- そのノードはそのデバイスグループに属しています。
- そのノードは現在、プライマリノードまたはセカンダリノードではありません。

二次ノードリストに戻されるのは、デバイスグループの優先順位がもっとも高いノードからです。セカンダリノードの目標数に到達するまで、優先順位が高いノードから順にセカンダリノードリストに戻されます。

あるノードがクラスタに参加したとき、そのノードの優先順位がそのデバイスグループの既存のセカンダリノードよりも高い場合、優先順位が低いノードがセカンダリノードリストから削除されます。削除されるノードは新たに追加されたノードにより置換されます。この置換は、クラスタ内に存在するセカンダリノードの実際の数が目標数を超えたときだけ発生します。

デバイスグループのノードリストの preferenced プロパティーを設定する方法についての詳細は、preferenced プロパティーを参照してください。

preferenced

preferenced プロパティーは true または false の値を持つことができます。デフォルトは true です。

デバイスグループの作成中、preferenced プロパティーが true に設定されている場合、ノードリストはノードの優先順位も示します。ノードの優先順位は、各ノードが、デバイスグループのプライマリノードとしてテイクオーバーしようとする順序を決定します。

デバイスグループの作成中、このプロパティーが false に設定されている場合、デバイスグループ内のデバイスに最初にアクセスしたノードが自動的にプライマリノードになります。指定されたノードリスト内のノードの順番は意味を持ちません。ノードリストを再指定せずにこのプロパティーに true を再設定しても、ノードの順番は再度有効にはなりません。

preferenced=true プロパティーを指定し、かつ -n オプションを使用してデバイスグループのノードリスト全体を優先順位どおりに指定しないかぎり、set 操作中にノードの優先順位は変更されません。

-t *devicegroup-type*[,…]
--type=*devicegroup-type*[,…]
--type *devicegroup-type*[,…]

デバイスグループタイプまたデバイスグループタイプのリストを指定します。

create サブコマンドの場合、1 つのデバイスグループタイプだけを指定できます。これにより、このオプションで指定したタイプのデバイスグループが作成されます。

これ以外で -t オプションを受け付けることができるサブコマンドの場合、このオプションにより、コマンドに指定したデバイスグループのリストが指定したデバイスグループタイプだけに限定されます。

必ずしもすべてのサブコマンドとオプションがすべてのデバイスグループタイプに対して有効であるわけではありません。たとえば、create サブコマンドは rawdisk デバイスグループタイプに対してのみ有効であり、svm または sds デバイスグループタイプには有効ではありません。

-t オプションでサポートされるデバイスグループタイプは次のとおりです。

rawdisk

raw ディスクデバイスグループを指定します。

raw ディスクとは、ボリュームマネージャーのボリュームやメタデバイスの一部ではないディスクのことです。raw ディスクデバイスグループを使用すると、デバイスグループ内にディスクセットを定義できます。デフォルトでは、システムブート時に、raw ディスクデバイスグループが、構成内のデバイス ID (DID) 擬似ドライバデバイスごとに作成されます。慣例により、初期化時には、raw ディスクデバイスグループの名前が割り当てられます。これらの名前は、DID デバイス名から派生されます。raw ディスクデバイスグループに追加されるノードごとに、cldevicegroup コマンドは、グループ内のすべてのデバイスがそのノードに物理的に接続されていることを確認します。

create サブコマンドは、raw ディスクデバイスグループを作成して、複数のディスクデバイスをデバイスグループに追加します。新しい raw ディスクデバイスグループを作成

する前に、新しいデバイスグループに追加する各デバイスを、ブート時にそのデバイスが作成されたデバイスグループから削除してください。これにより、これらのデバイスが含まれる新しい raw ディスクデバイスグループを作成できます。-n オプションで潜在的なプライマリノード優先順位リストを指定することに加えて、これらのデバイスのリストを -d オプションで指定します。

指定した単一のノード上でデバイスグループをマスターするには、-p オプションを使用して、localonly=true というプロパティーの設定で、デバイスグループを構成します。ローカル専用デバイスグループを作成するとき、ノードリストには 1 つのノードだけを指定できます。

delete サブコマンドは、デバイスグループ名をクラスタデバイスグループ構成から削除します。

set サブコマンドは、次の変更を raw ディスクデバイスグループに行います。

- 潜在的なプライマリノードの優先順位を変更する
- 新しいノードリストを指定する
- フェイルバックを有効または無効にする
- 二次ノードの目標数を設定する
- 複数のグローバルデバイスをデバイスグループに追加する

raw ディスクデバイス名が raw ディスクデバイスグループに登録されている場合、その raw ディスクデバイス名は Solaris Volume Manager デバイスグループには登録できません。

sds

もともと Solstice DiskSuite™ ソフトウェアで作成されていたデバイスグループを指定します。複数所有者ディスクセットを除いて、このデバイスグループタイプは Solaris Volume Manager デバイスグループタイプ svm と同等です。詳細は、svm デバイスグループタイプの説明を参照してください。

svm

Solaris ボリュームマネージャーデバイスグループを指定します。

Solaris ボリュームマネージャーデバイスグループは、次のコンポーネントで定義されます。

- 名前
- グループにアクセスできるノード
- ディスクセット内のデバイスのグローバルなリスト
- 潜在的なプライマリノードの優先順位やフェイルバックの動作などのアクションを制御するプロパティーの集合

Solaris Volume Manager には、多重ホストまたは共有ディスクセットの概念があります。共有ディスクセットとは、2 つ以上のホストとディスクドライブからなるグループのことです。このディスクドライブはすべてのホストからアクセス可能であり、すべての

ホスト上でデバイス名が同じです。こうしたデバイス名の統一は、raw ディスクデバイスを用いてディスクセットを構築することにより行います。デバイス ID 疑似ドライバ (DID) を使用することで、多重ホストに使用するディスクの名前はクラスタ内で整合性が保たれます。Solaris Volume Manager デバイスグループのノードリストに構成できるのは、すでにディスクセットの一部として構成されているホストだけです。共有ディスクセットにドライブを追加する場合、そのドライブが、ほかの何らかの共有ディスクセットに属していてはいけません。

Solaris Volume Manager の `metaset` コマンドはディスクセットを作成して、そのディスクセットを Solaris Volume Manager デバイスグループとして、Oracle Solaris Cluster ソフトウェアに自動的に登録します。デバイスグループを作成したあと、`cldevicegroup` コマンドの `set` サブコマンドを使用して、ノード優先順位リストと `preferenced`、`failback`、および `numsecondaries` プロパティーを設定する必要があります。

1 つのデバイスグループには、1 つの Solaris Volume Manager ディスクセットだけを割り当てることができます。デバイスグループ名は常に、ディスクセットの名前と一致する必要があります。

Solaris Volume Manager デバイスグループにノードを追加または削除するのに、`add-node` または `remove-node` サブコマンドは使用できません。その代わりに、Solaris Volume Manager の `metaset` コマンドを使用して、配下の Solaris Volume Manager ディスクセットで、ノードを追加または削除します。

Solaris Volume Manager のデバイスグループをクラスタ構成から削除するのに、`delete` サブコマンドは使用できません。その代わりに、Solaris Volume Manager の `metaset` コマンドを使用して、配下の Solaris Volume Manager ディスクセットを削除します。

`export`、`list`、`show`、`status`、および `sync` サブコマンドだけが、Solaris Volume Manager 複数所有者ディスクセットで機能します。Solaris Volume Manager デバイスグループの配下のディスクセットを追加または削除するには、Solaris Volume Manager のコマンドまたはユーティリティーを使用してください。

`-V`
`--version`

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンドやオペランドなどのオプションは無視されます。`-V` オプションは、コマンドのバージョンを表示するだけです。その他の処理は行いません。

`-v`
`--verbose`

詳細なメッセージを標準出力に表示します。

このオプションは、このコマンドの任意の形式に使用できます。

次のオペランドを指定できます。

devicegroup　　デバイスグループを指定します。

`cldevicegroup` コマンドは、Oracle Solaris Cluster デバイスグループ名だけをオペランドとして受け付けます。複数のデバイスグループ名を受け付けるほとんどの形式のコマンドの場合、プラス記号 (+) を使用するとすべての可能なデバイスグループを指定できます。

注記 - + オペランドは、手動で作成されたデバイスグループだけを含み、`autogen` プロパティーに `true` が設定されている自動的に作成されたデバイスグループをすべて無視します。Oracle Solaris Cluster ソフトウェアは、システムがブートされるたびに、このようなデバイスグループを自動的に作成します。これらの「非表示の」デバイスグループにコマンドを適用するには、各デバイスグループを明示的に指定する必要があります。

このコマンドセットにあるすべてのコマンドの終了ステータスコードの完全なリストについては、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (`CL_NOERR`) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

`0 CL_NOERR`
　エラーなし

`1 CL_ENOMEM`
　十分なスワップ空間がありません。

`3 CL_EINVAL`
　無効な引数

`6 CL_EACCESS`
　アクセス権がありません

`35 CL_EIO`
　I/O エラー

`36 CL_ENOENT`
　そのようなオブジェクトはありません。

`39 CL_EEXIST`
　オブジェクトは存在します。

例 54 デバイスグループの変更

次の例では、デバイスグループ devgrp1 の preference プロパティーを true に設定して、numsecondaries プロパティーを 2 に設定する方法を示します。このコマンドはまた、希望ノードリスト phys-schost-1,phys-schost-2,phys-schost-3 も指定しています。

```
# cldevicegroup set -p preferenced=true -p numsecondaries=2 \
-n phys-schost-1,phys-schost-2,phys-schost-3 devgrp1
```

例 55 raw ディスクデバイスグループの変更

次の例では、既存の raw ディスクデバイスグループ rawdevgrp1 を変更する方法を示します。このコマンドは、新しいメンバーデバイスリスト内のデバイス d3 および d4 を指定し、localonly 属性を true に設定します。ノード phys-schost-1 は、ローカル専用 raw ディスクデバイスグループで許可される唯一のプライマリノードです。

```
# cldevicegroup set -d d3,d4 \
-p localonly=true -n phys-schost-1 rawdevgrp1
```

例 56 デバイスグループの numsecondaries 属性のリセット

次の例では、デバイスグループ devgrp1 の numsecondaries 属性に何も値を指定しないことによって、適切なシステムデフォルト値にリセットする方法を示します。

```
# cldevicegroup set -p numsecondaries= devgrp1
```

例 57 デバイスグループのスイッチオーバー

次の例では、デバイスグループ devgrp1 を新しいマスターノード phys-schost-2 に切り替える方法を示します。

```
# cldevicegroup switch -n phys-schost-2 devgrp1
```

例 58 デバイスグループの無効化

次の例では、デバイスグループ devgrp1 を無効にする方法を示します。

```
# cldevicegroup disable devgrp1
```

例 59 デバイスグループのオフライン

次の例では、デバイスグループ devgrp1 をオフラインにしてから無効にする方法を示します。

```
# cldevicegroup offline devgrp1
# cldevicegroup disable devgrp1
```

例 60 デバイスグループのそのプライマリノードでのオンライン化

次の例では、デバイスグループ devgrp1 をそのデフォルトのプライマリノードでオンラインにする方法を示します。このコマンドはまず、デバイスグループを有効にします。

```
# cldevicegroup online -e devgrp1
```

例 61 デバイスグループの指定されたノードでのオンライン化

次の例では、デバイスグループ devgrp1 を新しいプライマリノードとして指定された phys-schost-2 でオンラインにする方法を示します。

```
# cldevicegroup switch -n phys-schost-2 devgrp1
```

例 62 新しいノードのデバイスグループへの追加

次の例では、新しいノード phys-schost-3 をデバイスグループ devgrp1 に追加する方法を示します。このデバイスグループは、デバイスグループタイプ svm ではありません。

```
# cldevicegroup add-node -n phys-schost-3 devgrp1
```

例 63 デバイスグループの削除

次の例では、デバイスグループ devgrp1 を Oracle Solaris Cluster 構成から削除する方法を示します。このデバイスグループは、デバイスグループタイプ svm ではありません。

```
# cldevicegroup delete devgrp1
```

例 64 レプリケーション情報とデバイスグループ構成の同期化

次の例では、デバイスグループ devgrp1 のディスクが使用するレプリケーション構成を Oracle Solaris Cluster ソフトウェアに認識させる方法を示します。

```
# cldevicegroup sync devgrp1
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core

属性タイプ	属性値
インタフェースの安定性	発展中

19 ページのIntro(1CL), 61 ページのcldevice(1CL), 551 ページのcluster(1CL), Unresolved link to " metaset1M", 1447 ページのclconfiguration(5CL), Unresolved link to " rbac5", 1479 ページのdid(7)

Unresolved link to " Oracle Solaris Cluster システム管理 "

スーパーユーザーはこのコマンドのすべての形式を実行できます。

また、任意のユーザーは次のオプションを指定してこのコマンドを実行できます。

- -? (ヘルプ) オプション
- -V (バージョン) オプション

スーパーユーザー以外のユーザーがほかのサブコマンドを指定してこのコマンドを実行するには、RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
add-device	solaris.cluster.modify
add-node	solaris.cluster.modify
create	solaris.cluster.modify
delete	solaris.cluster.modify
disable	solaris.cluster.modify
enable	solaris.cluster.modify
export	solaris.cluster.read
list	solaris.cluster.read
offline	solaris.cluster.admin
online	solaris.cluster.admin
remove-device	solaris.cluster.modify
remove-node	solaris.cluster.modify
set	solaris.cluster.modify
show	solaris.cluster.read

サブコマンド	RBAC の承認
status	solaris.cluster.read
switch	solaris.cluster.modify
sync	solaris.cluster.admin

名前

clinterconnect, clintr — Oracle Solaris Cluster インターコネクトの管理

```
/usr/cluster/bin/clinterconnect -V

/usr/cluster/bin/clinterconnect [subcommand] -?

/usr/cluster/bin/clinterconnect subcommand [options] -v
     [endpoint[,endpoint] …]

/usr/cluster/bin/clinterconnect add [-d] endpoint[,endpoint] …

/usr/cluster/bin/clinterconnect add -i {- | clconfigfile} [-d]
     [-n node[,…]] {+ | endpoint[,endpoint] …}

/usr/cluster/bin/clinterconnect disable [-n node[,…]]
     {+ | endpoint[,endpoint] …}

/usr/cluster/bin/clinterconnect enable [-n node[,…]]
     {+ | endpoint[,endpoint] …}

/usr/cluster/bin/clinterconnect export [-o {- | configfile}] [-n
     node[,…]] [+ | endpoint[,endpoint] …]

/usr/cluster/bin/clinterconnect remove [-l] endpoint[,endpoint] ...

/usr/cluster/bin/clinterconnect show [-n node[,…]]
     [+ | endpoint[,endpoint] ...]

/usr/cluster/bin/clinterconnect status [-n node[,…]]
     [+ | endpoint[,endpoint] ...]
```

`clinterconnect` コマンドは、クラスタインターコネクトの構成を管理し、構成とステータス情報を表示します。`clintr` コマンドは、`clinterconnect` コマンドの短い形式です。`clinterconnect` コマンドと `clintr` コマンドは同じものです。どちらの形式のコマンドも使用できます。

クラスタインターコネクトには 2 つの終端があり、それらはケーブルで接続されます。終端は、ノード上のアダプタであることも、スイッチ (接続点とも呼ぶ) であることもあります。ケーブルでは、アダプタとスイッチが接続されることも、特定のトポロジにある 2 つのアダプタが接続されることもあります。クラスタトポロジマネージャーは、使用可能なケーブルを使用し、ノード間にエンドツーエンドのインターコネクトパスを構築します。このコマンドに指定するクラスタインターコネクトコンポーネントの名前は、実際の物理的な構成を正確に反映する必要があります。正確に反映しないと、システムは終端間のクラスタインターコネクトパスを構築できません。クラスタインターコネクトが正常に機能しない場合、クラスタノードはお互いに通信できず、ノードにパニックが発生するなどの状態になる可能性があります。

clinterconnect コマンドは、オンラインであり、クラスタモードであるクラスタノードから実行してください。

このコマンドの一般的な形式は次のとおりです。

clinterconnect [*subcommand*] [*options*] [*operands*]

subcommand を省略できるのは、*options* で -? オプションまたは -V オプションが指定されている場合のみです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式については、このマニュアルページの「オプション」セクションを参照してください。

このコマンドの一部の形式を非大域ゾーンで使用できます。このコマンドの有効な使用方法の詳細については、個々のサブコマンドの説明を参照してください。管理を容易にするため、このコマンドは大域ゾーンで使用します。

サブコマンド

サポートされるサブコマンドには次のものがあります。

add

コマンドへのオペランドとして指定された新しいクラスタインターコネクトコンポーネントを追加します。

このサブコマンドは、大域ゾーンでのみ使用します。

ケーブル、スイッチ、またはアダプタのどれを追加するのかは、オペランドの構文により決定されます。詳細は、このマニュアルページの「オペランド」セクションを参照してください。

add サブコマンドは、アダプタと別のノード上のアダプタ間、あるいは、アダプタとインターコネクトスイッチ間のインターコネクトケーブルを構成するのに使用します。ケーブルを構成するアダプタ終端またはスイッチ終端は、すでに存在している必要はありません。このサブコマンドは、アダプタまたはスイッチを構成に追加するのにも使用できます。

アダプタまたはスイッチを構成に追加するとき、このコマンドはまた、アダプタまたはスイッチを有効にします。ケーブルを追加するとき、このコマンドはまた、ケーブルの各終端がまだ有効になっていない場合、ケーブルの各終端を有効にします。

2 ノードクラスタでは、各終端にあるアダプタにケーブルを追加する場合、仮想スイッチも作成します。

-d オプションは、無効な状態にある終端を追加するのに使用します。

-i オプションで構成ファイルを指定する場合、プラス記号 (+) をオペランドとして指定できます。このオペランドを使用するとき、このコマンドは、構成ファイルで指定されており、まだクラスタに存在していないすべてのインターコネクトコンポーネントを作成します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

インターコネクトコンポーネントの削除については、remove コマンドを参照してください。

disable

コマンドへのオペランドとして指定されたインターコネクトコンポーネントを無効にします。

このサブコマンドは、大域ゾーンでのみ使用します。

ケーブル、スイッチ、またはアダプタのどれを無効にするのかは、オペランドの構文により決定されます。詳細は、このマニュアルページの「オペランド」セクションを参照してください。

有効なケーブルに接続されているアダプタまたはスイッチを無効にしようとすると、その操作はエラーになります。接続されているアダプタまたはスイッチを無効にする前に、まず、ケーブルを無効にしてください。

ケーブルを無効にするとき、このコマンドはまた、ケーブルに関連する各終端 (アダプタまたはスイッチポートの場合あり) も無効にします。また、すべてのスイッチポートが無効な状態の場合、このコマンドはスイッチを無効にします。

アクティブなクラスタノードの最後のクラスタインターコネクトパスのケーブルまたは終端を無効にしようとすると、その操作はエラーになります。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

インターコネクトコンポーネントの有効化については、enable サブコマンドを参照してください。

enable

コマンドへのオペランドとして指定されたインターコネクトコンポーネントを有効にします。

このサブコマンドは、大域ゾーンでのみ使用します。

ケーブル、スイッチ、またはアダプタのどれを有効にするのかは、オペランドの構文により決定されます。詳細は、このマニュアルページの「オペランド」セクションを参照してください。

ケーブルを有効にするとき、このコマンドはまた、ケーブルに関連する各終端 (アダプタまたはスイッチポートの可能性あり) も有効にします。

インターコネクトコンポーネントの無効化については、disable サブコマンドを参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

export

クラスタインターコネクトの構成情報をエクスポートします。

このサブコマンドは、大域ゾーンでのみ使用します。

-o オプションでファイル名を指定すると、構成情報はその新しいファイルに書き込まれます。-o オプションを指定しない場合、出力は標準出力に書き込まれます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

remove

コマンドへのオペランドとして指定されたクラスタインターコネクトコンポーネントを削除します。

このサブコマンドは、大域ゾーンでのみ使用します。

ケーブル、スイッチ、またはアダプタのどれを削除するのかは、オペランドの構文により決定されます。詳細は、このマニュアルページの「オペランド」セクションを参照してください。

次の動作は、ケーブルを削除するときに適用されます。

- ケーブルを削除する前に、まず、ケーブルを無効にしてください。
- 有効なケーブルを削除しようとすると、その操作はエラーになります。
- 無効なケーブルを削除する場合、次の状況を除いて、ケーブルの終端も削除されます。
  - スイッチが別のケーブルによって使用されている。
  - -l オプションも指定している。

次の動作は、アダプタ終端またはスイッチ終端を削除するときに適用されます。

- ケーブルに関連付けられていない終端を削除すると、指定された終端が削除されます。
- ケーブルに関連付けられている終端を削除しようとすると、削除操作はエラーになります。これは、ケーブルが有効または無効であるかに関わらず発生します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

インターコネクトコンポーネントの追加については、add サブコマンドを参照してください。

show

コマンドへのオペランドとして指定されたインターコネクトコンポーネントの構成を表示します。

このサブコマンドは、大域ゾーンまたはゾーンクラスタで使用できます。

構成情報には、そのコンポーネントが有効または無効であるかも含まれます。デフォルトでは、すべてのインターコネクトコンポーネントの構成が出力されます。

show サブコマンドは、すべてのコンポーネントを指定するプラス記号 (+) をオペランドとして受け入れます。-Z オプションを使用すると、指定した排他的 IP ゾーンクラスタのプライベートネットワーク構成情報を表示できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

status

インターコネクトパスのステータスを表示します。デフォルトでは、システムのすべてのインターコネクトパスのステータスが表示されます。-Z オプションを使用すると、指定した排他的 IP ゾーンクラスタのプライベートネットワーク構成情報のステータスを表示できます。

このサブコマンドは、大域ゾーンまたは排他的 IP ゾーンクラスタで使用できます。

次に、インターコネクトパスの可能な状態を示します。

`faulted` インターコネクトパスの機能を妨げるエラーが検出されました。

`Path online` インターコネクトパスはオンラインで、サービスを提供しています。

`waiting` インターコネクトパスは `Path online` 状態に移行中です。

インターコネクトコンポーネントが有効または無効であるかを判定するには、`show` サブコマンドを使用します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

次のオプションがサポートされています。

`-?`
`--help`

ヘルプ情報を表示します。このオプションを使用する場合、ほかの処理は実行されません。

このオプションは、単独でもサブコマンド付きでも使用できます。

- このオプションを単独で使用する場合、使用可能なサブコマンドのリストが出力されます。
- このオプションをサブコマンドを付けて使用する場合、そのサブコマンドの使用法オプションが出力されます。

`-d`

無効な状態にある終端を追加することを指定します。

`-i` {`-` | *clconfigfile*}
`--input=`{`-` | *clconfigfile*}
`--input` {`-` | *clconfigfile*}

ケーブルを追加または変更するのに使用される構成情報を指定します。この情報は、1447 ページの`clconfiguration`(5CL) のマニュアルページに定義されている形式に準拠している必要があります。この情報は、ファイルに含めることも、標準入力を介して指定することもできます。標準入力を指定するには、ファイル名の代わりにマイナス記号 (`-`) を指定します。

コマンドで指定するオプションは、クラスタ構成ファイルで設定されている任意のオプションより優先されます。必要な要素がクラスタ構成ファイルに存在しない場合、これらの要素をコマンド行で指定してください。

マイナス記号 (`-`) 引数をこのオプションと一緒に使用すると、構成が標準入力として提供されることを指定できます。

`-l`
`--limited`

ケーブル削除処理がケーブルだけを削除し、その終端を削除しないことを指定します。

`-l` オプションは、`remove` サブコマンドでのみ有効です。このオプションを `remove` サブコマンドとともに指定しない場合、このコマンドは、指定されたケーブルだけでなく、関連付けられたアダプタもすべて削除します。さらに、ケーブル削除処理がスイッチへの最後の接続を削除する場合、このコマンドはまた、そのスイッチも構成から削除します。

`-n` *node*[,…]
`--node=`*node*[,…]
`--node` *node*[,…]

ノードまたはノードリストを指定します。このオプションは、指定したノードだけに接続されているアダプタとケーブルに処理を制限します。

ノードは、ノード名またはノード ID のどちらででも指定できます。

`-o` {- | *clconfigfile*}
`--output=`{- | *clconfigfile*}
`--output` {- | *clconfigfile*}

1447 ページの`clconfiguration`(5CL) のマニュアルページで規定されている形式で、インターコネクト構成を表示します。

`export` サブコマンドだけが `-o` オプションを受け付けます。

このオプションの引数としてファイル名を指定する場合、このコマンドは新しいファイルを作成して、そのファイルに構成情報を出力します。同じ名前のファイルがすでにある場合、このコマンドはエラーで終了します。既存のファイルに変更は行われません。

このオプションの引数としてマイナス記号 (-) を指定すると、このコマンドは標準出力に構成情報を表示します。このコマンドのほかの標準出力はすべて抑制されます。

`-V`
`--version`

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンドやオペランドなどのオプションは無視されます。`-V` オプションは、コマンドのバージョンを表示するだけです。その他の処理は行いません。

`-v`
`--verbose`

詳細なメッセージを標準出力に表示します。デフォルトでは、`show` および `status` コマンドは詳細出力を表示します。

このオプションは、このコマンドの任意の形式に使用できます。

-Z {*zoneclustername*}
--zoneclustername={*zoneclustername* }
--zoneclustername { *zoneclustername*}

操作する 1 つまたは複数のクラスタを指定します。

このオプションは、show および status サブコマンドでサポートされています。

このオプションを指定する場合は、次のいずれかの引数も指定する必要があります。

zoneclustername このオプションと一緒に使用するコマンドが、*zoneclustername* という名前のゾーンクラスタ内でのみで動作するように指定します。

global このオプションを使用するコマンドが、グローバルクラスタでのみ機能するように指定します。

このコマンドは、インターコネクト終端またはコンマ区切りの終端のペアをオペランドとして受け付けます。終端は、アダプタであることも、スイッチであることもあります。コンマ区切りの終端のペアはケーブルを示します。

複数のインターコネクトコンポーネントを受け付ける形式のコマンドの場合、プラス記号 (+) 引数を使用すると、すべての可能なコンポーネントを指定できます。

次のオペランドがサポートされています。

node:*adapter*

アダプタ終端を指定します。

アダプタ終端は、ノード名とアダプタ名を持ちます。アダプタ名の構成は、インターコネクト名の直後に物理ユニット番号 (net0 など) が続いたものです。これらの処理を行うにあたって、アダプタをホストするノードは、クラスタ内でアクティブとなっている必要はありません。

次のタイプのアダプタは、クラスタトランスポートアダプタとして構成できます。

Ethernet Ethernet アダプタは別の Ethernet アダプタまたは Ethernet スイッチに接続できます。

InfiniBand InfiniBand アダプタは InfiniBand スイッチだけに接続できます。

デフォルトでは、アダプタは dlpi トランスポートタイプを使用して構成されます。

タグ付き VLAN アダプタを指定するには、物理デバイス名と VLAN インスタンス番号から派生されたタグ付き VLAN アダプタ名を使用します。VLAN インスタンス番号は、VLAN ID に 1000 を掛けて、元の物理ユニット番号を足したものです。たとえば、物理デバイス net2 上の VLAN ID 11 は、タグ付き VLAN アダプタ名 net11002 に変換されます。

switch[@*port*]

スイッチ終端を指定します。

個々のインターコネクトスイッチの名前は、クラスタの名前空間全体で一意でなければなりません。英字、数字、またはその両方の組み合わせを使用できます。スイッチ名の最初の文字は英字にする必要があります。

スイッチ終端の *port* コンポーネントを指定しない場合、このコマンドはデフォルトのポート名を想定します。デフォルトのポート名は、ケーブルのもう一方に接続されているノードのノード ID と同じです。

次のタイプのスイッチをクラスタトランスポートスイッチとして構成できます。

Ethernet　　Ethernet スイッチを Ethernet アダプタと一緒に使用します。

InfiniBand　　InfiniBand スイッチを InfiniBand アダプタと一緒に使用します。

デフォルトでは、スイッチは `switch` タイプを使用して構成されます。

node:*adapter*, *node*:*adapter*
node:*adapter*,*switch* [@*port*]

ケーブルを指定します。

ケーブルは、アダプタ終端またはスイッチ終端のコンマ区切りのペアです。終端の順番は重要でありません。ケーブルオペランドは、完全なクラスタインターコネクトを追加するのに使用します。ケーブルを追加するとき、`clinterconnect` コマンドは両方の終端を自動的に作成するため、アダプタ終端またはスイッチ終端を別に作成する必要はありません。

このコマンドセットにあるすべてのコマンドの終了ステータスコードの完全なリストについては、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (`CL_NOERR`) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

`0 CL_NOERR`

エラーなし

`1 CL_ENOMEM`

十分なスワップ空間がありません。

`3 CL_EINVAL`

無効な引数

`6 CL_EACCESS`

アクセス権がありません

35 CL_EIO

I/O エラー

36 CL_ENOENT

そのようなオブジェクトはありません。

37CL_EOP

操作が許可されていません

38 CL_EBUSY

オブジェクトはビジーです

39 CL_EEXIST

オブジェクトは存在します。

例 65 直接接続クラスタインターコネクトケーブルの作成

次の例では、ノード phys-schost-1 上のアダプタ net0 とノード phys-schost-2 上のアダプタ net0 の間のポートを接続するケーブルを追加する方法を示します。

```
# clinterconnect add phys-schost-1:net0,phys-schost-2:net0
```

例 66 スイッチとアダプタ間でのケーブルの作成

次の例では、ノード phys-schost-1 上のアダプタ net0 とスイッチ ether_switch の間にケーブルを追加する方法を示します。

```
# clinterconnect add phys-schost-1:net0,ether_switch
```

例 67 ケーブルの無効化

次の例では、ノード phys-schost-1 上のアダプタ net0 とスイッチ ether_switch の間に接続されているケーブルを無効にする方法を示します。

```
# clinterconnect disable phys-schost-1:net0,ether_switch
```

例 68 クラスタインターコネクトケーブルの削除

次の例では、ノード phys-schost-1 上のアダプタ net0 とスイッチ ether_switch の間に接続されているケーブルを取り外す方法を示します。

```
# clinterconnect remove phys-schost-1:net0,ether_switch
```

例 69 タグ付き VLAN アダプタとスイッチ間でのケーブルの作成

次の例では、ノード `phys-schost-1` 上のタグ付き VLAN アダプタ `net73002` と VLAN 対応スイッチ `switch1` の間にケーブルを追加する方法を示します。このアダプタの物理名は `net2` で、VLAN ID は `73` です。

```
# clinterconnect add phys-schost-1:net73002,switch1
```

例 70 スイッチの有効化

次の例では、スイッチ終端 `switch1` を有効にする方法を示します。

```
# clinterconnect enable switch1
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページの`Intro`(1CL), 551 ページの`cluster`(1CL),
1447 ページの`clconfiguration`(5CL), Unresolved link to " rbac5"

Unresolved link to " Oracle Solaris Cluster 4.2 Hardware Administration Manual "

Unresolved link to " Oracle Solaris Cluster ソフトウェアのインストール "

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のオプションを指定してこのコマンドを実行できます。

- `-?` (ヘルプ) オプション
- `-V` (バージョン) オプション

スーパーユーザー以外のユーザーがほかのサブコマンドを指定してこのコマンドを実行するには、RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
add	solaris.cluster.modify
disable	solaris.cluster.modify
enable	solaris.cluster.modify
export	solaris.cluster.read
remove	solaris.cluster.modify
show	solaris.cluster.read
status	solaris.cluster.read

名前

clinterconnect, clintr — Oracle Solaris Cluster インターコネクトの管理

```
/usr/cluster/bin/clinterconnect -V

/usr/cluster/bin/clinterconnect [subcommand] -?

/usr/cluster/bin/clinterconnect subcommand [options] -v
     [endpoint[,endpoint] …]

/usr/cluster/bin/clinterconnect add [-d] endpoint[,endpoint] …

/usr/cluster/bin/clinterconnect add -i {- | clconfigfile} [-d]
     [-n node[,…]] {+ | endpoint[,endpoint] …}

/usr/cluster/bin/clinterconnect disable [-n node[,…]]
     {+ | endpoint[,endpoint] …}

/usr/cluster/bin/clinterconnect enable [-n node[,…]]
     {+ | endpoint[,endpoint] …}

/usr/cluster/bin/clinterconnect export [-o {- | configfile}] [-n
     node[,…]] [+ | endpoint[,endpoint] …]

/usr/cluster/bin/clinterconnect remove [-l] endpoint[,endpoint] ...

/usr/cluster/bin/clinterconnect show [-n node[,…]]
     [+ | endpoint[,endpoint] ...]

/usr/cluster/bin/clinterconnect status [-n node[,…]]
     [+ | endpoint[,endpoint] ...]
```

`clinterconnect` コマンドは、クラスタインターコネクトの構成を管理し、構成とステータス情報を表示します。`clintr` コマンドは、`clinterconnect` コマンドの短い形式です。`clinterconnect` コマンドと `clintr` コマンドは同じものです。どちらの形式のコマンドも使用できます。

クラスタインターコネクトには 2 つの終端があり、それらはケーブルで接続されます。終端は、ノード上のアダプタであることも、スイッチ (接続点とも呼ぶ) であることもあります。ケーブルでは、アダプタとスイッチが接続されることも、特定のトポロジにある 2 つのアダプタが接続されることもあります。クラスタトポロジマネージャーは、使用可能なケーブルを使用し、ノード間にエンドツーエンドのインターコネクトパスを構築します。このコマンドに指定するクラスタインターコネクトコンポーネントの名前は、実際の物理的な構成を正確に反映する必要があります。正確に反映しないと、システムは終端間のクラスタインターコネクトパスを構築できません。クラスタインターコネクトが正常に機能しない場合、クラスタノードはお互いに通信できず、ノードにパニックが発生するなどの状態になる可能性があります。

clinterconnect コマンドは、オンラインであり、クラスタモードであるクラスタノードから実行してください。

このコマンドの一般的な形式は次のとおりです。

clinterconnect [*subcommand*] [*options*] [*operands*]

subcommand を省略できるのは、*options* で -? オプションまたは -V オプションが指定されている場合のみです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式については、このマニュアルページの「オプション」セクションを参照してください。

このコマンドの一部の形式を非大域ゾーンで使用できます。このコマンドの有効な使用方法の詳細については、個々のサブコマンドの説明を参照してください。管理を容易にするため、このコマンドは大域ゾーンで使用します。

サブコマンド

サポートされるサブコマンドには次のものがあります。

add

コマンドへのオペランドとして指定された新しいクラスタインターコネクトコンポーネントを追加します。

このサブコマンドは、大域ゾーンでのみ使用します。

ケーブル、スイッチ、またはアダプタのどれを追加するのかは、オペランドの構文により決定されます。詳細は、このマニュアルページの「オペランド」セクションを参照してください。

add サブコマンドは、アダプタと別のノード上のアダプタ間、あるいは、アダプタとインターコネクトスイッチ間のインターコネクトケーブルを構成するのに使用します。ケーブルを構成するアダプタ終端またはスイッチ終端は、すでに存在している必要はありません。このサブコマンドは、アダプタまたはスイッチを構成に追加するのにも使用できます。

アダプタまたはスイッチを構成に追加するとき、このコマンドはまた、アダプタまたはスイッチを有効にします。ケーブルを追加するとき、このコマンドはまた、ケーブルの各終端がまだ有効になっていない場合、ケーブルの各終端を有効にします。

2 ノードクラスタでは、各終端にあるアダプタにケーブルを追加する場合、仮想スイッチも作成します。

-d オプションは、無効な状態にある終端を追加するのに使用します。

-i オプションで構成ファイルを指定する場合、プラス記号 (+) をオペランドとして指定できます。このオペランドを使用するとき、このコマンドは、構成ファイルで指定されており、まだクラスタに存在していないすべてのインターコネクトコンポーネントを作成します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

インターコネクトコンポーネントの削除については、remove コマンドを参照してください。

disable

コマンドへのオペランドとして指定されたインターコネクトコンポーネントを無効にします。

このサブコマンドは、大域ゾーンでのみ使用します。

ケーブル、スイッチ、またはアダプタのどれを無効にするのかは、オペランドの構文により決定されます。詳細は、このマニュアルページの「オペランド」セクションを参照してください。

有効なケーブルに接続されているアダプタまたはスイッチを無効にしようとすると、その操作はエラーになります。接続されているアダプタまたはスイッチを無効にする前に、まず、ケーブルを無効にしてください。

ケーブルを無効にするとき、このコマンドはまた、ケーブルに関連する各終端 (アダプタまたはスイッチポートの場合あり) も無効にします。また、すべてのスイッチポートが無効な状態の場合、このコマンドはスイッチを無効にします。

アクティブなクラスタノードの最後のクラスタインターコネクトパスのケーブルまたは終端を無効にしようとすると、その操作はエラーになります。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

インターコネクトコンポーネントの有効化については、enable サブコマンドを参照してください。

enable

コマンドへのオペランドとして指定されたインターコネクトコンポーネントを有効にします。

このサブコマンドは、大域ゾーンでのみ使用します。

ケーブル、スイッチ、またはアダプタのどれを有効にするのかは、オペランドの構文により決定されます。詳細は、このマニュアルページの「オペランド」セクションを参照してください。

ケーブルを有効にするとき、このコマンドはまた、ケーブルに関連する各終端 (アダプタまたはスイッチポートの可能性あり) も有効にします。

インターコネクトコンポーネントの無効化については、disable サブコマンドを参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

export

クラスタインターコネクトの構成情報をエクスポートします。

このサブコマンドは、大域ゾーンでのみ使用します。

-o オプションでファイル名を指定すると、構成情報はその新しいファイルに書き込まれます。-o オプションを指定しない場合、出力は標準出力に書き込まれます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

remove

コマンドへのオペランドとして指定されたクラスタインターコネクトコンポーネントを削除します。

このサブコマンドは、大域ゾーンでのみ使用します。

ケーブル、スイッチ、またはアダプタのどれを削除するのかは、オペランドの構文により決定されます。詳細は、このマニュアルページの「オペランド」セクションを参照してください。

次の動作は、ケーブルを削除するときに適用されます。

- ケーブルを削除する前に、まず、ケーブルを無効にしてください。
- 有効なケーブルを削除しようとすると、その操作はエラーになります。
- 無効なケーブルを削除する場合、次の状況を除いて、ケーブルの終端も削除されます。
  - スイッチが別のケーブルによって使用されている。
  - -l オプションも指定している。

次の動作は、アダプタ終端またはスイッチ終端を削除するときに適用されます。

- ケーブルに関連付けられていない終端を削除すると、指定された終端が削除されます。
- ケーブルに関連付けられている終端を削除しようとすると、削除操作はエラーになります。これは、ケーブルが有効または無効であるかに関わらず発生します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

インターコネクトコンポーネントの追加については、add サブコマンドを参照してください。

show

コマンドへのオペランドとして指定されたインターコネクトコンポーネントの構成を表示します。

このサブコマンドは、大域ゾーンまたはゾーンクラスタで使用できます。

構成情報には、そのコンポーネントが有効または無効であるかも含まれます。デフォルトでは、すべてのインターコネクトコンポーネントの構成が出力されます。

show サブコマンドは、すべてのコンポーネントを指定するプラス記号 (+) をオペランドとして受け入れます。-Z オプションを使用すると、指定した排他的 IP ゾーンクラスタのプライベートネットワーク構成情報を表示できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

status

インターコネクトパスのステータスを表示します。デフォルトでは、システムのすべてのインターコネクトパスのステータスが表示されます。-Z オプションを使用すると、指定した排他的 IP ゾーンクラスタのプライベートネットワーク構成情報のステータスを表示できます。

このサブコマンドは、大域ゾーンまたは排他的 IP ゾーンクラスタで使用できます。

次に、インターコネクトパスの可能な状態を示します。

`faulted` インターコネクトパスの機能を妨げるエラーが検出されました。

`Path online` インターコネクトパスはオンラインで、サービスを提供しています。

`waiting` インターコネクトパスは `Path online` 状態に移行中です。

インターコネクトコンポーネントが有効または無効であるかを判定するには、`show` サブコマンドを使用します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

次のオプションがサポートされています。

`-?`
`--help`

ヘルプ情報を表示します。このオプションを使用する場合、ほかの処理は実行されません。

このオプションは、単独でもサブコマンド付きでも使用できます。

- このオプションを単独で使用する場合、使用可能なサブコマンドのリストが出力されます。
- このオプションをサブコマンドを付けて使用する場合、そのサブコマンドの使用法オプションが出力されます。

`-d`

無効な状態にある終端を追加することを指定します。

`-i` {`-` | *clconfigfile*}
`--input=`{`-` | *clconfigfile*}
`--input` {`-` | *clconfigfile*}

ケーブルを追加または変更するのに使用される構成情報を指定します。この情報は、1447 ページの`clconfiguration`(5CL) のマニュアルページに定義されている形式に準拠している必要があります。この情報は、ファイルに含めることも、標準入力を介して指定することもできます。標準入力を指定するには、ファイル名の代わりにマイナス記号 (`-`) を指定します。

コマンドで指定するオプションは、クラスタ構成ファイルで設定されている任意のオプションより優先されます。必要な要素がクラスタ構成ファイルに存在しない場合、これらの要素をコマンド行で指定してください。

マイナス記号 (`-`) 引数をこのオプションと一緒に使用すると、構成が標準入力として提供されることを指定できます。

`-l`
`--limited`

ケーブル削除処理がケーブルだけを削除し、その終端を削除しないことを指定します。

`-l` オプションは、`remove` サブコマンドでのみ有効です。このオプションを `remove` サブコマンドとともに指定しない場合、このコマンドは、指定されたケーブルだけでなく、関連付けられたアダプタもすべて削除します。さらに、ケーブル削除処理がスイッチへの最後の接続を削除する場合、このコマンドはまた、そのスイッチも構成から削除します。

`-n` *node*[,…]
`--node=`*node*[,…]
`--node` *node*[,…]

ノードまたはノードリストを指定します。このオプションは、指定したノードだけに接続されているアダプタとケーブルに処理を制限します。

ノードは、ノード名またはノード ID のどちらででも指定できます。

`-o` {- | *clconfigfile*}
`--output=`{- | *clconfigfile*}
`--output` {- | *clconfigfile*}

1447 ページの`clconfiguration`(5CL) のマニュアルページで規定されている形式で、インターコネクト構成を表示します。

`export` サブコマンドだけが `-o` オプションを受け付けます。

このオプションの引数としてファイル名を指定する場合、このコマンドは新しいファイルを作成して、そのファイルに構成情報を出力します。同じ名前のファイルがすでにある場合、このコマンドはエラーで終了します。既存のファイルに変更は行われません。

このオプションの引数としてマイナス記号 (-) を指定すると、このコマンドは標準出力に構成情報を表示します。このコマンドのほかの標準出力はすべて抑制されます。

`-V`
`--version`

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンドやオペランドなどのオプションは無視されます。`-V` オプションは、コマンドのバージョンを表示するだけです。その他の処理は行いません。

`-v`
`--verbose`

詳細なメッセージを標準出力に表示します。デフォルトでは、`show` および `status` コマンドは詳細出力を表示します。

このオプションは、このコマンドの任意の形式に使用できます。

-Z {*zoneclustername*}
--zoneclustername={*zoneclustername* }
--zoneclustername { *zoneclustername*}

操作する 1 つまたは複数のクラスタを指定します。

このオプションは、show および status サブコマンドでサポートされています。

このオプションを指定する場合は、次のいずれかの引数も指定する必要があります。

zoneclustername このオプションと一緒に使用するコマンドが、*zoneclustername* という名前のゾーンクラスタ内でのみで動作するように指定します。

global このオプションを使用するコマンドが、グローバルクラスタでのみ機能するように指定します。

このコマンドは、インターコネクト終端またはコンマ区切りの終端のペアをオペランドとして受け付けます。終端は、アダプタであることも、スイッチであることもあります。コンマ区切りの終端のペアはケーブルを示します。

複数のインターコネクトコンポーネントを受け付ける形式のコマンドの場合、プラス記号 (+) 引数を使用すると、すべての可能なコンポーネントを指定できます。

次のオペランドがサポートされています。

node:*adapter*

アダプタ終端を指定します。

アダプタ終端は、ノード名とアダプタ名を持ちます。アダプタ名の構成は、インターコネクト名の直後に物理ユニット番号 (net0 など) が続いたものです。これらの処理を行うにあたって、アダプタをホストするノードは、クラスタ内でアクティブとなっている必要はありません。

次のタイプのアダプタは、クラスタトランスポートアダプタとして構成できます。

Ethernet Ethernet アダプタは別の Ethernet アダプタまたは Ethernet スイッチに接続できます。

InfiniBand InfiniBand アダプタは InfiniBand スイッチだけに接続できます。

デフォルトでは、アダプタは dlpi トランスポートタイプを使用して構成されます。

タグ付き VLAN アダプタを指定するには、物理デバイス名と VLAN インスタンス番号から派生されたタグ付き VLAN アダプタ名を使用します。VLAN インスタンス番号は、VLAN ID に 1000 を掛けて、元の物理ユニット番号を足したものです。たとえば、物理デバイス net2 上の VLAN ID 11 は、タグ付き VLAN アダプタ名 net11002 に変換されます。

switch[@*port*]

スイッチ終端を指定します。

個々のインターコネクトスイッチの名前は、クラスタの名前空間全体で一意でなければなりません。英字、数字、またはその両方の組み合わせを使用できます。スイッチ名の最初の文字は英字にする必要があります。

スイッチ終端の *port* コンポーネントを指定しない場合、このコマンドはデフォルトのポート名を想定します。デフォルトのポート名は、ケーブルのもう一方に接続されているノードのノード ID と同じです。

次のタイプのスイッチをクラスタトランスポートスイッチとして構成できます。

Ethernet　　Ethernet スイッチを Ethernet アダプタと一緒に使用します。

InfiniBand　　InfiniBand スイッチを InfiniBand アダプタと一緒に使用します。

デフォルトでは、スイッチは `switch` タイプを使用して構成されます。

node:*adapter*, *node*:*adapter*
node:*adapter*,*switch* [@*port*]

ケーブルを指定します。

ケーブルは、アダプタ終端またはスイッチ終端のコンマ区切りのペアです。終端の順番は重要でありません。ケーブルオペランドは、完全なクラスタインターコネクトを追加するのに使用します。ケーブルを追加するとき、`clinterconnect` コマンドは両方の終端を自動的に作成するため、アダプタ終端またはスイッチ終端を別に作成する必要はありません。

このコマンドセットにあるすべてのコマンドの終了ステータスコードの完全なリストについては、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (`CL_NOERR`) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

`0 CL_NOERR`

エラーなし

`1 CL_ENOMEM`

十分なスワップ空間がありません。

`3 CL_EINVAL`

無効な引数

`6 CL_EACCESS`

アクセス権がありません

35 CL_EIO

I/O エラー

36 CL_ENOENT

そのようなオブジェクトはありません。

37CL_EOP

操作が許可されていません

38 CL_EBUSY

オブジェクトはビジーです

39 CL_EEXIST

オブジェクトは存在します。

例 71 直接接続クラスタインターコネクトケーブルの作成

次の例では、ノード phys-schost-1 上のアダプタ net0 とノード phys-schost-2 上のアダプタ net0 の間のポートを接続するケーブルを追加する方法を示します。

```
# clinterconnect add phys-schost-1:net0,phys-schost-2:net0
```

例 72 スイッチとアダプタ間でのケーブルの作成

次の例では、ノード phys-schost-1 上のアダプタ net0 とスイッチ ether_switch の間にケーブルを追加する方法を示します。

```
# clinterconnect add phys-schost-1:net0,ether_switch
```

例 73 ケーブルの無効化

次の例では、ノード phys-schost-1 上のアダプタ net0 とスイッチ ether_switch の間に接続されているケーブルを無効にする方法を示します。

```
# clinterconnect disable phys-schost-1:net0,ether_switch
```

例 74 クラスタインターコネクトケーブルの削除

次の例では、ノード phys-schost-1 上のアダプタ net0 とスイッチ ether_switch の間に接続されているケーブルを取り外す方法を示します。

```
# clinterconnect remove phys-schost-1:net0,ether_switch
```

例 75 タグ付き VLAN アダプタとスイッチ間でのケーブルの作成

次の例では、ノード `phys-schost-1` 上のタグ付き VLAN アダプタ `net73002` と VLAN 対応スイッチ `switch1` の間にケーブルを追加する方法を示します。このアダプタの物理名は `net2` で、VLAN ID は `73` です。

```
# clinterconnect add phys-schost-1:net73002,switch1
```

例 76 スイッチの有効化

次の例では、スイッチ終端 `switch1` を有効にする方法を示します。

```
# clinterconnect enable switch1
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページの`Intro`(1CL), 551 ページの`cluster`(1CL),
1447 ページの`clconfiguration`(5CL), Unresolved link to " rbac5"

Unresolved link to " Oracle Solaris Cluster 4.2 Hardware Administration Manual "

Unresolved link to " Oracle Solaris Cluster ソフトウェアのインストール "

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のオプションを指定してこのコマンドを実行できます。

- `-?` (ヘルプ) オプション
- `-V` (バージョン) オプション

スーパーユーザー以外のユーザーがほかのサブコマンドを指定してこのコマンドを実行するには、RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
add	solaris.cluster.modify
disable	solaris.cluster.modify
enable	solaris.cluster.modify
export	solaris.cluster.read
remove	solaris.cluster.modify
show	solaris.cluster.read
status	solaris.cluster.read

名前

clsnmpmib, clmib — Oracle Solaris Cluster SNMP MIB の管理

```
/usr/cluster/bin/clsnmpmib -V

/usr/cluster/bin/clsnmpmibsubcommand [] -?

/usr/cluster/bin/clsnmpmib [subcommand] [options] -v [mib]

/usr/cluster/bin/clsnmpmib disable [-n node[,…]] {+ | mib …}

/usr/cluster/bin/clsnmpmib enable [-n node[,…]] {+ | mib …}

/usr/cluster/bin/clsnmpmib export [-n node[,…]]
     [-o {- | clconfigfile}] [+ | mib …]

/usr/cluster/bin/clsnmpmib list [-n node[,…]] [+ | mib …]

/usr/cluster/bin/clsnmpmib set [-p name=value] […]
     [-n node[,…] {+ | mib …}

/usr/cluster/bin/clsnmpmib show [-n node[,…]] [+ | mib …]
```

`clsnmpmib` コマンドは、現在のノード上にある既存の Oracle Solaris Cluster SNMP (Simple Network Management Protocol) の管理情報ベース (MIB) を管理します。MIB を管理できる SNMP ホストを作成するには、491 ページの`clsnmphost`(1CL) のマニュアルページを参照してください。SNMPv3 プロトコルを使用しているときに MIB にアクセスできる SNMP バージョン 3 (SNMPv3) ユーザーを定義するには、509 ページの`clsnmpuser`(1CL) のマニュアルページを参照してください。

このコマンドの一般的な形式は次のとおりです。

`clsnmpmib` [*subcommand*] [*options*] [*operands*]

subcommand を省略できるのは、*options* で `-?` または `-V` オプションが指定されている場合のみです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式は、OPTIONS セクションにオプションの説明とともに記載されています。

詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

Oracle Solaris Cluster の MIB

Oracle Solaris Cluster は現在、1 つの MIB (イベント MIB) をサポートしています。Oracle Solaris Cluster SNMP の イベント MIB は、SNMP マネージャーにクラスタイベントをリアルタイムで通知します。有効になっている場合、Oracle Solaris Cluster のイベント MIB は、`clsnmphost` コマンドで定義されているすべてのホストにトラップ通知を自動的に送信します。Oracle Solaris Cluster イベント MIB は、ポート 11162 にトラップ通知を送信します。SNMP ツリーはポート 11161 に表示されます。

`clsnmpmib set` サブコマンドを使用して、`min_severity` または `log_number` の値を指定できます。クラスタは多数のイベント通知を生成するので、重要度が `min_severity` 以上のイベントのみがトラップ通知として送信されます。デフォルトでは、`min_severity` 値は `NOTICE` に設定されます。`log_number` の値には、古いエントリを破棄するまでに MIB テーブルに記録するイベントの数を指定します。MIB は、トラップが送信された最新のイベントの読み取り専用テーブルを維持しています。イベントの数は、`log_number` 値によって制限されます。この情報はリブートされると消滅します。

このコマンドは、大域ゾーンだけで使用できます。

サポートされるサブコマンドには次のものがあります。

`disable`

指定されたノード上の 1 つ以上のクラスタの MIB を無効にします。

このサブコマンドは、大域ゾーンだけで使用できます。

`-n` オプションを指定しないと、現在のノード上の MIB だけが無効にされます。MIB が無効になると、MIB のテーブルにアクセスできず、MIB はトラップ通知を一切送信しません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`enable`

指定されたノード上にある 1 つ以上のクラスタの MIB を有効にします。

このサブコマンドは、大域ゾーンだけで使用できます。

`-n` オプションを指定しないと、現在のノード上の MIB だけが有効にされます。有効にされる MIB を制限するには、*mib* オペランドを使用します。

MIB を有効にすると、その機能すべてが有効になります。ただし、すべての MIB 機能が完全に機能するには、さらにいくつかの構成が必要な場合があります。たとえば、ホストが構成されていないと、MIB はトラップ通知を送信できません。SNMP ホストの構成については、491 ページの`clsnmphost`(1CL) のマニュアルページを参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`export`

クラスタの MIB の構成情報をエクスポートします。

このサブコマンドは、大域ゾーンだけで使用できます。

MIB 構成情報をエクスポートする 1 つ以上のノードを指定するには、`-n` オプションを使用します。`export` を `-n` オプションなしで使用した場合、このサブコマンドは、現在のノードの MIB 構成情報のみをエクスポートします。デフォルトでは、このサブコマンドは現在のノードのすべての MIB から構成情報をエクスポートします。出力をさらに詳細化するには、構成情報が必要な 1 つ以上の MIB の名前を指定します。

`export` サブコマンドからの出力形式の詳細は、1447 ページの`clconfiguration`(5CL) のマニュアルページを参照してください。デフォルトでは、すべての出力が標準出力に送信されます。出力先をファイルに変更するには、`-o` オプションを使用して、そのあとにファイル名を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`list`

指定されたノード上のクラスタの MIB のリストを表示します。

このサブコマンドは、大域ゾーンだけで使用できます。

一覧表示するクラスタの MIB のノードを指定するには、`-n` オプションを使用します。`list` サブコマンドを `-n` オプションなしで使用すると、このサブコマンドは現在のノード上の MIB だけを一覧表示します。一覧表示する MIB を制限するには、一覧表示する 1 つ以上の MIB の名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`set`

指定されたノード上の 1 つ以上の MIB で使用される SNMP プロトコル、`min_severity`、または `log_number` 設定を変更します。

このサブコマンドは、大域ゾーンだけで使用できます。

デフォルトでは、このサブコマンドはノード上のすべての MIB を変更します。ノードを指定しないと、現在のノード上の MIB の SNMP プロパティーのみが変更されます。SNMP プロパティーは、`-p` オプションを使って指定する必要があります。すべての MIB では、デフォルトのプロパティー値 `protocol:SNMPv2`、`min_severity:NOTICE`、`log_number:100` を使用します。`set` サブコマンドは、*mib* オペランドを使用して MIB 名を指定しないかぎり、すべての MIB のプロトコル、`min_severity`、または `log_number` 設定を変更します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

show

指定されたノード上の MIB の情報を表示します。

このサブコマンドは、大域ゾーンだけで使用できます。

show サブコマンドは、MIB の名前、その SNMP プロトコルバージョン、min_severity 値、または log_number 値を表示します。デフォルトでは、このサブコマンドはノード上のすべての MIB の情報を表示します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

次のオプションがサポートされています。

-?
--help

ヘルプ情報を表示します。

このオプションはサブコマンド付きでもサブコマンドなしでも指定できます。

- このオプションをサブコマンドなしで使用すると、使用可能なサブコマンドのリストが表示されます。
- このオプションをサブコマンドとともに使用すると、サブコマンドの使用オプションが表示されます。

このオプションを使用すると、ほかの処理は実行されません。

-n *node*[,…]
--node[s] *node*[,…]

ノードまたはノードリストを指定します。各ノードはノード名またはノード ID で指定できます。clsnmpmib コマンドのすべての形式で、このオプションが受け付けられます。-n オプションを使用すると、処理が実行されるノードを指定できます。- n オプションがない場合、このコマンドは現在のノードを想定します。

-o {- | *clconfigfile*}
--output {- | *clconfigfile*}

クラスタの MIB 構成についての情報が書き込まれる場所を指定します。この書き込み先は、ファイルでも、標準出力でもかまいません。標準出力を指定するには、ファイル名の代わりにマイナス記号 (-) を指定します。標準出力を指定すると、コマンドに対するほかのすべての標準出力が抑制されます。-o オプションを指定しない場合、出力は標準出力に送信されます。このオプションは、export サブコマンドの場合にだけ指定できます。

構成情報は、1447 ページの`clconfiguration`(5CL) のマニュアルページで定義されている形式で書き込まれます。

`-p` *name*= *value*
`--property=`*name* =*value*
`--property` *name*=*value*

`version` MIB で使用する SNMP プロトコルのバージョンを指定します。Oracle Solaris Cluster は、SNMPv2 および SNMPv3 プロトコルバージョンをサポートしています。

`min_severity` 重要度の最小値を指定します。`min_severity` 値に等しいか、それを上回る値を持つイベントのみが MIB テーブルに記録され、それらのイベントについて、トラップが構成済みのホストに送信されます。

`log_number` 古いエントリを破棄するまでに MIB テーブルに記録するイベントの数を指定します。

`-p` *name*=*value* は複数回指定できます。

このオプションとともに次のプロパティーを設定できます。

`version`

MIB で使用する SNMP プロトコルのバージョンを指定します。*value* は次のように指定します。

- `version=SNMPv2`
- `version=snmpv2`
- `version=2`
- `version=SNMPv3`
- `version=snmpv3`
- `version=3`

`min_severity`

MIB で使用する重要度の最小値を指定します。この値は次のように指定します。

- `min_severity=NOTICE`
- `min_severity=WARNING`
- `min_severity=ERROR`
- `min_severity=CRITICAL`
- `min_severity=FATAL`

大文字と小文字のどちらの値も指定できます。

log_number

古いエントリを破棄するまでに MIB テーブルに記録するイベントの数を指定します。デフォルト値は 100 です。値は 100-500 の範囲である必要があります。この値は次のように指定します。

- log_number=*number*

-V
--version

コマンドのバージョンを表示します。

このオプションは、サブコマンド、オペランド、またはその他のオプションと一緒に指定しないでください。指定すると、一緒に指定されたサブコマンド、オペランド、またはその他のオプションは無視されます。-V オプションは、コマンドのバージョンだけを表示します。その他の処理は行いません。

-v
--verbose

詳細情報を標準出力に出力します。

このオプションはどの形式のコマンドととも指定できますが、一部のサブコマンドは拡張出力を発生させないことがあります。たとえば、export サブコマンドでは、冗長オプションを指定すると拡張出力は生成されません。

次のオペランドがサポートされています。

mib サブコマンドが適用される単数または複数の MIB の名前を指定します。このオペランドを指定しない場合は、デフォルトのプラス記号 (+) がサブコマンドで使用されます (すべての MIB を意味します)。*mib* オペランドを使用する場合は、その他のすべてのコマンド行オプションのあとにあるスペースで区切られたリスト内で MIB を指定してください。

\+ クラスタのすべての MIB。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

18 CL_EINTERNAL

内部エラーが発生しました

内部エラーは、ソフトウェアの欠陥またはその他の欠陥を示しています。

35 CL_EIO

I/O エラー

物理的な入出力エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

例 77 MIB の一覧表示

次のコマンドは、クラスタノード上にあるすべての MIB を一覧表示します。

```
# clsnmpmib list
Event
```

例 78 MIB の有効化

次のコマンドは、現在のノード上にある Event MIB を有効にします。

```
# clsnmpmib enable event
```

クラスタの MIB の名前では大文字と小文字は区別されません。

例 79 プロトコルの変更

次のコマンドは、クラスタノード phys-cluster-2 の Event MIB のプロトコルを SNMPv3 に変更します。

```
# clsnmpmib set -n phys-cluster-2 -p version=SNMPv3 Event
```

-n オプションを使用すると、ノード名の代わりにノード ID を使用できます。

例 80 構成の表示

次のコマンドは、クラスタノード phys-cluster-1 および phys-cluster-2 上の構成情報を表示します。

```
# clsnmpmib show -n phys-cluster-1,phys-cluster-2
--- SNMP MIB Configuration on myhost ---

SNMP MIB Name:                 phys-cluster-1
  State:                         Event
  Enabled:                       yes
  Protocol:                      SNMPv3
  min_severity:                  1
  log_number:                    100
SNMP MIB Name:                 phys-cluster-2
  State:                         Event
  Enabled:                       yes
  Protocol:                      SNMPv3
  min_severity:                  3
  log_number:                    250
```

例 81 Min Severity 値の変更

次のコマンドは、クラスタノード phys-cluster-2 上のイベント MIB の min_severity を WARNING に変更します。

```
# clsnmpmib set -n phys-cluster-2 -p min_severity=WARNING Event
```

-n オプションを使用すると、ノード名の代わりにノード ID を使用できます。

例 82 Log_Number 値の変更

次のコマンドは、クラスタノード phys-cluster-2 上のイベント MIB の log_number を 250 に変更します。

```
# clsnmpmib set -n phys-cluster-2 -p log_number=250 Event
```

-n オプションを使用すると、ノード名の代わりにノード ID を使用できます。

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

/usr/cluster/lib/mib/sun-cluster-event-mib.mib

Oracle Solaris Cluster SNMP イベント MIB 定義ファイル

491 ページのclsnmphost(1CL), 509 ページのclsnmpuser(1CL), 19 ページのIntro(1CL), 551 ページのcluster(1CL), 809 ページのsceventmib(1M), 1047 ページのscha_calls(3HA), Unresolved link to " attributes5", Unresolved link to " rbac5", 1447 ページのclconfiguration(5CL)

スーパーユーザーはこのコマンドのすべての形式を実行できます。

すべてのユーザーがこのコマンドに -? (ヘルプ) オプションまたは -V (バージョン) オプションを指定して実行できます。

clsnmpmib コマンドをほかのサブコマンドとともに実行する場合、スーパーユーザー以外のユーザーは RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
disable	solaris.cluster.modify
enable	solaris.cluster.modify
export	solaris.cluster.read

サブコマンド	RBAC の承認
list	solaris.cluster.read
set	solaris.cluster.modify
show	solaris.cluster.read

名前

clnasdevice, clnas — Oracle Solaris Cluster 向け NAS デバイスへのアクセスの管理

```
/usr/cluster/bin/clnasdevice -V

/usr/cluster/bin/clnasdevice [subcommand] -?

/usr/cluster/bin/clnasdevice subcommand [options] -v [nasdevice[…]]

/usr/cluster/bin/clnasdevice add -t type {-p name=value
     [,…] | -u userid} [-f passwdfile] [-Z {zoneclustername | global}]
     nasdevice

/usr/cluster/bin/clnasdevice add -i {- | clconfigfile}[-t type]
     [-p name=value | -u userid] {-f passwdfile} [-Z {zoneclustername | global}]
     {nasdevice

/usr/cluster/bin/clnasdevice add-dir -d directory[,…] [-Z
     {zoneclustername | global}] nasdevice

/usr/cluster/bin/clnasdevice add-dir -i {- | clconfigfile} [-d all |
     directory [,…]] [-f passwordfile] [-Z {zoneclustername | global}]
     {nasdevice

/usr/cluster/bin/clnasdevice export [-o {- | clconfigfile}] [-t
     type[,…]] [-d all | directory[,…]] [+ | nasdevice[…]]

/usr/cluster/bin/clnasdevice find-dir [-d {all |directory[,...]
     [-t type[,…]] [-Z {zoneclustername[,…] | global | all}]
     [+ | nasdevice[…]]

/usr/cluster/bin/clnasdevice list [-t type[,…]] [-Z
     {zoneclustername[,...] | global | all}] [+ | nasdevice[…]]

/usr/cluster/bin/clnasdevice remove [-t type[,…]] [-Z
     {zoneclustername | global}] [-F ] {+ | nasdevice[…]}

/usr/cluster/bin/clnasdevice remove-dir -d all | directory[,…]
     [-Z {zoneclustername | global}] nasdevice

/usr/cluster/bin/clnasdevice set {-p name=value[,…] | -u
     userid} [-f passwdfile] [-Z {zoneclustername | global}]
     nasdevice

/usr/cluster/bin/clnasdevice show [-d {all | directory[,...] [-t
     type[,…]] [-Z {zoneclustername[,…] | global | all}]
     [+ | nasdevice[…]]
```

`clnasdevice` コマンドは、NAS デバイスとそのディレクトリまたはプロジェクトに関する Oracle Solaris Cluster 構成情報を管理します。

clnas コマンドは、clnasdevice コマンドの短い形式です。clnas コマンドと clnasdevice コマンドは同じものです。どちらの形式のコマンドも使用できます。

このコマンドの一般的な形式は次のとおりです。

clnasdevice [*subcommand*] [*options*] [*operands*]

subcommand を省略できるのは、*options* で -? オプションまたは -V オプションが指定されている場合のみです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式は、このマニュアルページの「オプション」セクションのオプションの説明で紹介されています。

clnasdevice コマンドを使用して NAS デバイスをクラスタに構成する前に、NAS デバイスが次の状態であることを確認してください。

- NAS デバイスが設定されており、機能している。
- NAS デバイスがブートされ、動作している。
- NAS デバイスのディレクトリが作成され、クラスタノードに使用可能である。
- NAS デバイスが定足数デバイスになる予定である場合、その定足数デバイスの LUN が作成されている必要があります。NAS 定足数デバイスの構成については、clquorum のマニュアルページを参照してください。

NAS デバイスのベンダーによっては、デバイスをクラスタに構成する前に、追加のタスクを行う必要がある場合もあります。これらのタスクについての詳細は、「オプション」セクションの -t オプションを参照してください。NAS デバイスの設定手順やそのディレクトリのエクスポート手順については、NAS デバイスのドキュメントを参照してください。

NAS デバイスが完全に機能し、クラスタにストレージを提供する準備ができたあとは、clnasdevice コマンドを使用して、クラスタ内の NAS デバイス構成情報を管理します。これを行わないと、クラスタは NAS デバイスとそのエクスポートされるディレクトリを検出できません。結果として、クラスタはこれらのディレクトリにある情報の整合性を保護できません。

clnasdevice コマンドを使用して、次のような管理タスクを行います。

- NAS デバイス構成の作成
- NAS タイプ固有プロパティーの更新
- NAS デバイスディレクトリのクラスタ構成からの削除

■ NAS デバイスのクラスタ構成からの削除

`clnasdevice` コマンドは、アクティブなクラスタノードだけで実行できます。コマンドを実行して得られる結果は、実行するノードに関係なく、常に同じです。

`clnasdevice` コマンドは、ゾーンクラスタで、`export` を除くすべてのサブコマンドを指定して使用することができます。すべてのサブコマンド (`export` を除く) で `-Z` オプションを使用して、操作を制限する特定のゾーンクラスタの名前を指定することもできます。

グローバルクラスタノードからすべてのゾーンクラスタ情報にアクセスできますが、特定のゾーンクラスタは他のゾーンクラスタを認識しません。特定のゾーンクラスタに操作を制限できない場合、使用するサブコマンドは現在のクラスタでのみ機能します。

サポートされるサブコマンドには次のものがあります。

`add`

Oracle Solaris Cluster 構成に NAS デバイスを追加します。

`-t` オプションは、NAS デバイスのベンダーを指定するのに使用します。詳細は、「オプション」セクションの `-t` オプションの説明を参照してください。

NAS デバイスのタイプによって、追加のプロパティーを設定する必要がある場合があります。このような追加のプロパティーについても、「オプション」セクションの `-t` オプションの説明を参照してください。

スーパーユーザー以外のユーザーがこのコマンドを使用するには、`solaris.cluster.modify` 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" を参照してください。

`remove` サブコマンドの説明も参照してください。

`add-dir`

すでに構成されている NAS デバイスの指定したディレクトリまたはプロジェクトをクラスタ構成に追加します。このサブコマンドを使用する前に、デバイス上でこれらのディレクトリまたはプロジェクトを作成し、クラスタで使用できるようにしなければなりません。ディレクトリまたはプロジェクトの作成については、NAS デバイスのドキュメントを参照してください。

NAS デバイスのディレクトリまたはプロジェクトの追加は、次のいずれかの方法で行なうことができます。

- `clnasdevice add` コマンドを使用して、NAS デバイスをクラスタに構成します。その後、`clnasdevice add-dir` コマンドを使用してデバイスのディレクトリまたはプロジェクトをクラスタ内に構成します。
- `clnasdevice add-dir -i` *configurationfile* 形式のコマンドを使用して、デバイスの追加とそのディレクトリまたはプロジェクトの構成を 1 ステップで行ないます。この方法で

ディレクトリまたはプロジェクトを追加するには、-f オプションを使用してパスワードファイルを指定します。このオプションの詳細は、「オプション」セクションを参照してください。詳細は、1447 ページのclconfiguration(5CL) のマニュアルページを参照してください。

NAS デバイス上に新しいディレクトリまたはプロジェクトを作成し、それをクラスタノードから使用できるようにする場合は常に、この add-dir サブコマンドを使用してそれらのディレクトリまたはプロジェクトをクラスタ構成に追加する必要があります。add-dir サブコマンドを使用してクラスタに追加できる使用可能なディレクトリやプロジェクトのリストを表示するには、find-dir サブコマンドを使用します。

スーパーユーザー以外のユーザーがこのコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

remove-dir サブコマンドの説明も参照してください。

export

クラスタ NAS デバイス構成情報をエクスポートします。-o オプションでファイルを指定すると、そのファイルに構成情報が書き込まれます。-o オプションを使用しない場合、出力は標準出力 (stdout) に書き込まれます。

export サブコマンドはクラスタ構成情報を変更しません。

スーパーユーザー以外のユーザーがこのコマンドを使用するには、solaris.cluster.read RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

find-dir

NAS デバイスに設定されており、クラスタによって使用される可能性のある sun_uss プロジェクトを表示します。これらのプロジェクトは、add-dir サブコマンドでクラスタ構成にまだ追加されていません。出力にリストされているプロジェクトは、add-dir サブコマンドを使用するときの -d オプションの候補になることができます。

特定のタイプの NAS デバイスを表示するには、-t オプションを指定します。

sun_uss プロジェクトとそのプロジェクト内のファイルシステムを表示するには、-v オプションを指定します。

特定の sun_uss NAS デバイスプロジェクトを表示するには、-d オプションを指定します。

特定の sun_uss NAS デバイスプロジェクトとそのプロジェクト内のファイルシステムを表示するには、-v オプションと -d オプションを指定します。

find-dir サブコマンドはクラスタ構成情報を変更しません。

スーパーユーザー以外のユーザーがこのコマンドを使用するには、solaris.cluster.read RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

list

クラスタに構成されている NAS デバイス構成を表示します。

クラスタに構成されているデバイスのディレクトリとデバイスタイプを表示するには、詳細オプション -v を使用します。

特定のタイプの NAS デバイスを表示するには、-t オプションを使用します。

スーパーユーザー以外のユーザーがこのコマンドを使用するには、solaris.cluster.read RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

remove

指定されている単数または複数の NAS デバイスを、Oracle Solaris Cluster 構成から削除します。

強制オプション -F を指定しない場合、remove-dir サブコマンドを使用して、NAS デバイスディレクトリを構成から削除しておいてください。

強制オプション -F を指定する場合、このコマンドは、NAS デバイスとそのディレクトリをクラスタ構成から削除します。「-**オプション**」セクションの *F* の説明を参照してください。

スーパーユーザー以外のユーザーがこのコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

add サブコマンドの説明も参照してください。

remove-dir

指定した NAS ディレクトリまたはプロジェクトを Oracle Solaris Cluster 構成から削除します。

remove-dir サブコマンドは、-d オプションで指定したエクスポート済みディレクトリまたはプロジェクトを削除します。-d all を使用すると、このサブコマンドによって、指定した NAS デバイスのすべてのディレクトリまたはプロジェクトが削除されます。

NAS デバイスからディレクトリまたはプロジェクトを削除した場合、この remove-dir サブコマンドを使用して、クラスタ構成からそのディレクトリまたはプロジェクトを削除する必要があります。クラスタ構成内の NAS ディレクトリまたはプロジェクトは、NAS デバイスからエクスポートした既存のディレクトリまたはプロジェクトと一致していなければなりません。

スーパーユーザー以外のユーザーがこのコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

add-dir サブコマンドの説明も参照してください。

set

特定の NAS デバイスの指定されたプロパティーを設定します。

スーパーユーザー以外のユーザーがこのコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

show

オプションが何も指定されていない場合は、次の情報を表示します。

- Oracle Solaris Cluster に構成されている現在のすべての NAS デバイスのリスト
- 各 NAS デバイスの使用可能なディレクトリ

- 各 NAS デバイスに関連付けられたすべてのプロパティー

特定のタイプの NAS デバイスを表示するには、-t オプションを指定します。特定のデバイスに関する情報を表示するには、NAS デバイスのホスト名をオペランドとしてコマンドに渡します。

指定されたプロジェクトに含まれているファイルシステムを表示するには、show サブコマンドで -d および -v オプションを使用します。all キーワードを使用して、NAS デバイスのすべてのプロジェクトまたは個々のプロジェクトを表示できます。

スーパーユーザー以外のユーザーがこのコマンドを使用するには、solaris.cluster.read RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

次のオプションがサポートされています。

-?
--help

ヘルプ情報を表示します。このオプションを使用するとき、ほかのオプションの処理は実行されません。

このオプションはサブコマンド付きでもサブコマンドなしでも指定できます。

サブコマンドなしでこのオプションを指定すると、このコマンドのサブコマンドのリストが表示されます。

サブコマンド付きでこのオプションを指定すると、サブコマンドの使用方法が表示されます。

-d *directory*[,…]
--directory=*directory*-[,…]
--directory *directory*-[,…]
-d *project*[,…]
--directory=*project*-[,…]
--directory *project*-[,…]

sun_uss NAS デバイスのプロジェクトを指定します。sun_uss NAS デバイスでは、ファイルシステムを作成する前に NAS デバイスでプロジェクトを作成する必要があります。プロジェクト名を / で始めることはできません。ファイルシステムはプロジェクト内で作成する必要があります。*プロジェクト*は sun_uss NAS デバイスの用語です。プロジェクトでは任意の数のファイルシステムを作成できます。このオプションは、add-dir、remove-dir、export、および show サブコマンドでのみ使用します。

このオプションは、特別なキーワード all を受け入れます。-d all オプションを使用すると、指定した NAS デバイス上のすべてのディレクトリが指定されます。

- remove-dir サブコマンドと一緒に使用する場合、指定したデバイス上のすべてのディレクトリが削除されます。
- export サブコマンドと一緒に使用する場合、指定したデバイス上のすべてのディレクトリの構成情報が、指定した出力に表示されます。

- `add-dir` サブコマンドと `-i` *configfile* オプションを使用すると、構成ファイルに一覧表示されている指定された NAS デバイス上のすべてのディレクトリが追加されます。
- `show` サブコマンドおよび `find-dir` サブコマンドを使用する際に、sun_uss NAS デバイスの `-v` オプションを指定すると、`-d` オプションで指定したプロジェクトに含まれるファイルシステムが表示されます。`all` キーワードを使用して、NAS デバイスのすべてのプロジェクトまたは個々のプロジェクトを表示できます。

`-F`
`--force`

指定した NAS デバイスを強制的に削除します。

強制オプションを一緒に指定できるのは、`remove` サブコマンドだけです。この強制オプションを `remove` サブコマンドと一緒に使用すると、NAS デバイスとその構成されているディレクトリが Oracle Solaris Cluster 構成から削除されます。

`-f` *passwd-file*
`--passwdfile=`*passwd-file*
`--passwdfile` *passwd-file*

NAS デバイスにログインするときに使用するパスワードが格納されているパスワードファイルを指定します。

セキュリティー上の理由により、パスワードはコマンド行オプションには指定できません。パスワードのセキュリティーを強化するには、パスワードをテキストファイルに格納し、`-f` オプションでこのファイルを指定してください。パスワード用の入力ファイルを指定しないと、パスワードの入力が求められます。

入力ファイルの読み取り可能アクセス権を root だけに与え、このファイルにグループからも一般ユーザーからもアクセスできないようにします。

`clnasdevice add` を `-i` オプションと一緒に使用するとき、*clconfigfile* にパスワードが格納されていない場合、`-f` *passwdfile* オプションを必ず指定してください。

入力ファイルでは、次の制限を守ってください。

- パスワードは単一行に指定します。パスワードを複数行にまたがって入力してはいけません。
- 行頭にあるブランクやタブは無視されます。
- コメントは引用符で囲まれていない # 記号で始まります。コメントは次の新しい行に続きます。

  すべてのコメントはパーサーによって無視されます。
- デバイスユーザーのパスワードに入力ファイルを使用する場合は、# 記号をパスワードの一部に使用できません。

`-i` *clconfigfile*
`--input={-` | *clconfigfile*`}`
`--input {-` | *clconfigfile*`}`

NAS デバイスを作成または変更するのに使用される構成情報を指定します。この情報は、1447 ページの`clconfiguration`(5CL) のマニュアルページに定義されている形式に準拠している必要があります。この情報はファイルに含まれている場合と、標準入力 `stdin` を通して指定される場合があります。標準入力を指定するには、ファイル名の代わりに `-` を指定します。

コマンド行と *clconfigfile* ファイルで同じプロパティーを指定する場合、コマンド行で設定したプロパティーが優先されます。

`clnasdevice add` を `-i` オプションと一緒に使用するとき、`-f` *passwdfile* オプションを必ず指定してください。

`-o` {`-` | *clconfigfile*}
`--output={-` | *clconfigfile*`}`
`--output {-` | *clconfigfile*`}`

1447 ページの`clconfiguration`(5CL) のマニュアルページで定義されている形式で、NAS デバイス構成情報を書き込みます。この情報は、ファイルまたは標準出力 (`stdout`) のどちらにでも書き込むことができます。標準出力を指定するには、ファイル名の代わりに `-` を指定します。

`-p` *name*`=`*value*`[,…]`
`--property=`*name*`=`*value*`[,…]`
`--property` *name value*`[,…]`

ある NAS デバイスタイプに固有のプロパティーを指定します。

`add` サブコマンドを使用して新しい NAS デバイスをクラスタ構成に追加する場合は、このオプションを指定する必要があります。`set` サブコマンドで NAS デバイスのプロパティーを変更する場合も、このオプションを指定する必要があります。詳細は、`-t` オプションの説明を参照してください。

`-t` *nas-device-type*
`--type=`*nas-device-type*
`--type` *nas-device-type*

NAS デバイスタイプを指定します。このオプションは、NAS デバイスを Oracle Solaris Cluster 構成に追加するときに指定してください。NAS デバイスタイプは、ベンダー名で指定します。たとえば、OracleZFS Storage Appliance NAS デバイスタイプは `sun_uss` です。

異なるタイプの NAS デバイスには異なるプロパティーがあるか、またはプロパティーがまったくない場合もあります。

`sun_uss`	OracleZFS Storage Appliance NAS デバイスを指定します。 `-p userid=osc_agent [-f passwd-file]` または `-u userid [-f passwdfile]`

`userid` は `osc_agent` である必要があります。sun_uss を使用する前に、クライアントコードをダウンロードし、すべてのクライアントノードにインストールする必要があります。この `osc_agent userid` は、デバイスでいずれかのワークフローを実行すると作成されます。この `userid` は、`userid` を入力として受け取る `clnasdevice` サブコマンドを使用する前に、デバイスで作成されている必要があります。

`userid` とパスワードプロパティーは必須です。

`-p "nodeIPs{`*node*`}"=[` *IP*`]`

このプロパティーは、各ノードの IP を指定します。クラスタノード名の IP とは異なる IP を使用して NAS デバイスにアクセスする場合、`nodeIPs`*node* プロパティーを使用してこの IP を指定できます。このプロパティーは省略可能です。IP を指定しない場合は、クラスタノード名の IP が使用されます。これらの IP は、NAS デバイス上のプロジェクトの `NFS Access Mode` で指定された IP と一致する必要があります。

プロパティーの値を指定しない場合 (例: `-p "nodeIPs{`*node*`}"=`)、指定したノードの IP がクラスタ構成から削除され、クラスタノード名の IP が使用されます。

sun_uss NAS デバイスとそのプロジェクトを追加する前に、必要な設定を行う必要があります。この設定タスクには、クライアントコードのダウンロードとクラスタノードへのインストールが含まれます。デバイスで `Configure for Oracle Solaris Cluster NFS` ワークフローを実行して、`userid osc_agent` とそのパスワードを作成します。プロジェクトを作成し、`Share Mode` を none または read-only に設定します (read-write モードもサポートされていますがお勧めしません)。`NFS Access Mode` では、ネットワーク概念を使用し、クラスタノードの IP に対する read-write アクセスを付与する必要があります。

NAS デバイスとそのエクスポートされたディレクトリをクラスタ構成に追加する前に、次に示す手順を実行しておく必要があります。

- NAS デバイスを設定します。
- ディレクトリを設定し、クラスタノードに使用可能にします。
- NAS デバイスで管理タスクを実行するときに使用するユーザー ID とパスワードを決定します。

また、NAS デバイスは起動され、動作している必要があります。詳細は、Unresolved link to " Oracle Solaris Cluster With Network-Attached Storage Device Manual "を参照してください。

`-u` *userid*
`--userid=`*userid*
`--userid` *userid*

NAS デバイスにログインするのに使用するユーザー ID を指定します。

クラスタは、ログインし、デバイス上で管理処理を実行するために、ユーザー ID を把握する必要があります。

あるいは、-p オプションでもユーザー ID を指定できます。詳細は、-p オプションを参照してください。

このオプションを一緒に指定できるのは、add および set サブコマンドだけです。

-V
--version

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンドやオペランドなどのオプションは無視されます。コマンドのバージョンが表示されます。ほかの処理は実行されません。

-v
--verbose

詳細な情報を標準出力 stdout に表示します。

-Z {*zoneclustername* | global | all}
--zoneclustername={*zoneclustername* | global | all}

nas-device-type が登録されている、処理するクラスタを指定します。

このオプションは、export コマンド以外のすべてのサブコマンドでサポートされています。

このオプションを指定する場合は、次のいずれかの引数も指定する必要があります。

zoneclustername
: このオプションと一緒に使用するコマンドが、*zoneclustername* という名前のゾーンクラスタ内でのみ、指定された各 *nas-device-type* に対して動作することを指定します。

global
: このオプションと一緒に使用するコマンドが、グローバルクラスタ内でのみ、指定された各 *nas-device-type* に対して動作することを指定します。

all
: グローバルクラスタ内でこの引数を使用した場合は、この引数と一緒に使用するコマンドが、すべてのクラスタ (グローバルクラスタおよびすべてのゾーンクラスタを含む) 内で、指定された各 *nas-device-type* に対して動作することを指定します。

  ゾーンクラスタ内でこの引数を使用した場合は、この引数と一緒に使用するコマンドが、そのゾーンクラスタ内でのみ、指定された各 *nas-device-type* に対して動作することを指定します。

次のオペランドがサポートされています。

nasdevice

NAS デバイスの名前。NAS デバイス名は、ネットワークで通信を行うために NAS デバイスが使用するホスト名です。NAS デバイスと通信するために、クラスタは NAS デバイス

の NAS ホスト名を知っておく必要があります。サブコマンドが複数の NAS デバイスを受け入れる場合は、プラス記号 + を使用して、すべての NAS デバイスを指定できます。add および add-dir サブコマンドの場合、プラス記号オペランドは、指定された構成ファイルにあるすべての NAS デバイスを指定します。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

次の終了値が返される可能性があります。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

15 CL_EPROP

無効なプロパティーです

-p、-y、または -x オプションで指定したプロパティーまたは値が存在しないか、許可されていません。

18 CL_EINTERNAL

内部エラーが発生しました

内部エラーは、ソフトウェアの欠陥またはその他の欠陥を示しています。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

41 CL_ETYPE

無効なタイプです

-t または -p オプションで指定したタイプは存在しません。

例 83 OracleZFS Storage Appliance からクラスタまたはゾーンクラスタへの NAS デバイスの追加

次の clnasdevice コマンドは、タイプが sun_uss の OracleZFS Storage Appliance uss7110-01 を構成に追加します。

```
# clnasdevice add -t sun_uss -p userid=osc_agent -f passwd-file uss7110-01
```

例 84 クラスタへの NAS プロジェクトの追加

次の clnasdevice コマンドは、すでに構成されている NAS デバイスである uss7110-01 に 2 つのプロジェクトを追加します。

```
# clnasdevice add-dir -d pool-0/local/nassa-pl,pool-0/local/nassa-p2 uss7110-01
```

例 85 クラスタまたはゾーンクラスタから NAS デバイスを削除する

次の clnasdevice コマンドは、NAS デバイス uss7110-01 と、残りのプロジェクトがある場合はそのプロジェクトをすべて、sun というクラスタ構成から削除します。

```
# clnasdevice remove -F uss7110-01
```

次の clnasdevice コマンドは、NAS デバイス uss7110-01 とその残りのすべてのプロジェクト (存在する場合) をゾーンクラスタの zc 構成から削除します。

```
# clnasdevice remove -Z ZC -F uss7110-01
```

次の例は、nodeIPs プロパティーを更新する方法を示しています。

```
# clnasdevice set -p "nodeIPs{cluster-1}"=10.155.55.145 \
-p "nodeIPs{cluster-2}"=10.155.55.146 uss7110-01
```

次の例では、クラスタ構成から現在の IP の設定を削除して、クラスタノード名の IP が使用されるようにします。

```
# clnasdevice set -p "nodeIPs{cluster-1}"= -p "nodeIPs{cluster-2}"= uss7110-01
```

例 86 クラスタに追加されていない NAS デバイスプロジェクトを表示する

次の clnasdevice コマンドは、クラスタにまだ追加されていない NAS デバイスプロジェクト名を表示します。

```
# clnasdevice find-dir uss7110-01
Nas Device:           uss7110-01
Type:                 sun_uss
Unconfigured Project: pool-0/local/nassa-p2
Unconfigured Project: pool-0/local/nassa-p1
```

例 87 クラスタまたはゾーンクラスタに構成されている NAS デバイスを表示する

次の clnasdevice コマンドは、クラスタに構成されているすべての NAS デバイスの名前を表示します。デバイスとそのディレクトリのリストを表示するには、詳細オプションか show サブコマンドを使用します。

```
# clnasdevice list
uss7110-01
```

次の clnasdevice コマンドは、ゾーンクラスタ ZC に構成されているすべての NAS デバイスの名前を表示します。デバイスとそのディレクトリのリストを表示するには、詳細オプションか show サブコマンドを使用します。

```
# clnasdevice list -Z ZC
ZC:uss7110-01
```

次の clnasdevice コマンドは、ゾーンクラスタ ZC に構成されているすべての NAS デバイスの名前を表示します。デバイスとそのディレクトリのリストを表示するには、詳細オプションか show サブコマンドを使用します。

```
# clnasdevice list -Z all
global:uss7110-01
ZC:uss7110-01
```

例 88 NAS デバイスとそのプロジェクトを表示する

次の `clnasdevice` コマンドは、クラスタに構成されているすべての NAS デバイスの名前とプロジェクトファイルシステムを表示します。

```
# clnasdevice show -v -d all uss7110-01
Nas Device:      uss7110-01
Type:            sun_uss
Project:         pool-0/local/nassa-p1
  File System:   /export/nassa-p1/nassa-p1-fs1
  File System:   /export/nassa-p1/nassa-p1-fs2
  File System:   /export/nassa-p1/nassa-p1-fs3
Project:         pool-0/local/nassa-p2
  File System:   /export/nassa-p2/nassa-p2-fs1
  File System:   /export/nassa-p2/nassa-p2-fs2
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページの`Intro`(1CL), 551 ページの`cluster`(1CL)

Unresolved link to " Oracle Solaris Cluster With Network-Attached Storage Device Manual "

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のサブコマンドとオプションを指定してこのコマンドを実行できます。

- `-?` オプション
- `-V` オプション

スーパーユーザー以外のユーザーがほかのサブコマンドを指定してこのコマンドを実行するには、RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
`add`	`solaris.cluster.modify`

サブコマンド	RBAC の承認
add-dir	solaris.cluster.modify
export	solaris.cluster.read
find-dir	solaris.cluster.read
list	solaris.cluster.read
set	solaris.cluster.modify
remove	solaris.cluster.modify
remove-dir	solaris.cluster.modify
show	solaris.cluster.read

名前

clnasdevice, clnas — Oracle Solaris Cluster 向け NAS デバイスへのアクセスの管理

```
/usr/cluster/bin/clnasdevice -V

/usr/cluster/bin/clnasdevice [subcommand] -?

/usr/cluster/bin/clnasdevice subcommand [options] -v [nasdevice[…]]

/usr/cluster/bin/clnasdevice add -t type {-p name=value
     [,…] | -u userid} [-f passwdfile] [-Z {zoneclustername | global}]
     nasdevice

/usr/cluster/bin/clnasdevice add -i {- | clconfigfile}[-t type]
     [-p name=value | -u userid] {-f passwdfile} [-Z {zoneclustername | global}]
     {nasdevice

/usr/cluster/bin/clnasdevice add-dir -d directory[,…] [-Z
     {zoneclustername | global}] nasdevice

/usr/cluster/bin/clnasdevice add-dir -i {- | clconfigfile} [-d all |
     directory [,…]] [-f passwordfile] [-Z {zoneclustername | global}]
     {nasdevice

/usr/cluster/bin/clnasdevice export [-o {- | clconfigfile}] [-t
     type[,…]] [-d all | directory[,…]] [+ | nasdevice[…]]

/usr/cluster/bin/clnasdevice find-dir [-d {all |directory[,...]
     [-t type[,…]] [-Z {zoneclustername[,…] | global | all}]
     [+ | nasdevice[…]]

/usr/cluster/bin/clnasdevice list [-t type[,…]] [-Z
     {zoneclustername[,...] | global | all}] [+ | nasdevice[…]]

/usr/cluster/bin/clnasdevice remove [-t type[,…]] [-Z
     {zoneclustername | global}] [-F ] {+ | nasdevice[…]}

/usr/cluster/bin/clnasdevice remove-dir -d all | directory[,…]
     [-Z {zoneclustername | global}] nasdevice

/usr/cluster/bin/clnasdevice set {-p name=value[,…] | -u
     userid} [-f passwdfile] [-Z {zoneclustername | global}]
     nasdevice

/usr/cluster/bin/clnasdevice show [-d {all | directory[,...] [-t
     type[,…]] [-Z {zoneclustername[,…] | global | all}]
     [+ | nasdevice[…]]
```

`clnasdevice` コマンドは、NAS デバイスとそのディレクトリまたはプロジェクトに関する Oracle Solaris Cluster 構成情報を管理します。

clnas コマンドは、clnasdevice コマンドの短い形式です。clnas コマンドと clnasdevice コマンドは同じものです。どちらの形式のコマンドも使用できます。

このコマンドの一般的な形式は次のとおりです。

clnasdevice [*subcommand*] [*options*] [*operands*]

subcommand を省略できるのは、*options* で -? オプションまたは -V オプションが指定されている場合のみです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式は、このマニュアルページの「オプション」セクションのオプションの説明で紹介されています。

clnasdevice コマンドを使用して NAS デバイスをクラスタに構成する前に、NAS デバイスが次の状態であることを確認してください。

- NAS デバイスが設定されており、機能している。
- NAS デバイスがブートされ、動作している。
- NAS デバイスのディレクトリが作成され、クラスタノードに使用可能である。
- NAS デバイスが定足数デバイスになる予定である場合、その定足数デバイスの LUN が作成されている必要があります。NAS 定足数デバイスの構成については、clquorum のマニュアルページを参照してください。

NAS デバイスのベンダーによっては、デバイスをクラスタに構成する前に、追加のタスクを行う必要がある場合もあります。これらのタスクについての詳細は、「オプション」セクションの -t オプションを参照してください。NAS デバイスの設定手順やそのディレクトリのエクスポート手順については、NAS デバイスのドキュメントを参照してください。

NAS デバイスが完全に機能し、クラスタにストレージを提供する準備ができたあとは、clnasdevice コマンドを使用して、クラスタ内の NAS デバイス構成情報を管理します。これを行わないと、クラスタは NAS デバイスとそのエクスポートされるディレクトリを検出できません。結果として、クラスタはこれらのディレクトリにある情報の整合性を保護できません。

clnasdevice コマンドを使用して、次のような管理タスクを行います。

- NAS デバイス構成の作成
- NAS タイプ固有プロパティーの更新
- NAS デバイスディレクトリのクラスタ構成からの削除

■ NAS デバイスのクラスタ構成からの削除

`clnasdevice` コマンドは、アクティブなクラスタノードだけで実行できます。コマンドを実行して得られる結果は、実行するノードに関係なく、常に同じです。

`clnasdevice` コマンドは、ゾーンクラスタで、`export` を除くすべてのサブコマンドを指定して使用することができます。すべてのサブコマンド (`export` を除く) で `-Z` オプションを使用して、操作を制限する特定のゾーンクラスタの名前を指定することもできます。

グローバルクラスタノードからすべてのゾーンクラスタ情報にアクセスできますが、特定のゾーンクラスタは他のゾーンクラスタを認識しません。特定のゾーンクラスタに操作を制限できない場合、使用するサブコマンドは現在のクラスタでのみ機能します。

サポートされるサブコマンドには次のものがあります。

`add`

Oracle Solaris Cluster 構成に NAS デバイスを追加します。

`-t` オプションは、NAS デバイスのベンダーを指定するのに使用します。詳細は、「オプション」セクションの `-t` オプションの説明を参照してください。

NAS デバイスのタイプによって、追加のプロパティーを設定する必要がある場合があります。このような追加のプロパティーについても、「オプション」セクションの `-t` オプションの説明を参照してください。

スーパーユーザー以外のユーザーがこのコマンドを使用するには、`solaris.cluster.modify` 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" を参照してください。

`remove` サブコマンドの説明も参照してください。

`add-dir`

すでに構成されている NAS デバイスの指定したディレクトリまたはプロジェクトをクラスタ構成に追加します。このサブコマンドを使用する前に、デバイス上でこれらのディレクトリまたはプロジェクトを作成し、クラスタで使用できるようにしなければなりません。ディレクトリまたはプロジェクトの作成については、NAS デバイスのドキュメントを参照してください。

NAS デバイスのディレクトリまたはプロジェクトの追加は、次のいずれかの方法で行なうことができます。

- `clnasdevice add` コマンドを使用して、NAS デバイスをクラスタに構成します。その後、`clnasdevice add-dir` コマンドを使用してデバイスのディレクトリまたはプロジェクトをクラスタ内に構成します。
- `clnasdevice add-dir -i` *configurationfile* 形式のコマンドを使用して、デバイスの追加とそのディレクトリまたはプロジェクトの構成を 1 ステップで行ないます。この方法で

ディレクトリまたはプロジェクトを追加するには、-f オプションを使用してパスワードファイルを指定します。このオプションの詳細は、「オプション」セクションを参照してください。詳細は、1447 ページのclconfiguration(5CL) のマニュアルページを参照してください。

NAS デバイス上に新しいディレクトリまたはプロジェクトを作成し、それをクラスタノードから使用できるようにする場合は常に、この add-dir サブコマンドを使用してそれらのディレクトリまたはプロジェクトをクラスタ構成に追加する必要があります。add-dir サブコマンドを使用してクラスタに追加できる使用可能なディレクトリやプロジェクトのリストを表示するには、find-dir サブコマンドを使用します。

スーパーユーザー以外のユーザーがこのコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

remove-dir サブコマンドの説明も参照してください。

export

クラスタ NAS デバイス構成情報をエクスポートします。-o オプションでファイルを指定すると、そのファイルに構成情報が書き込まれます。-o オプションを使用しない場合、出力は標準出力 (stdout) に書き込まれます。

export サブコマンドはクラスタ構成情報を変更しません。

スーパーユーザー以外のユーザーがこのコマンドを使用するには、solaris.cluster.read RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

find-dir

NAS デバイスに設定されており、クラスタによって使用される可能性のある sun_uss プロジェクトを表示します。これらのプロジェクトは、add-dir サブコマンドでクラスタ構成にまだ追加されていません。出力にリストされているプロジェクトは、add-dir サブコマンドを使用するときの -d オプションの候補になることができます。

特定のタイプの NAS デバイスを表示するには、-t オプションを指定します。

sun_uss プロジェクトとそのプロジェクト内のファイルシステムを表示するには、-v オプションを指定します。

特定の sun_uss NAS デバイスプロジェクトを表示するには、-d オプションを指定します。

特定の sun_uss NAS デバイスプロジェクトとそのプロジェクト内のファイルシステムを表示するには、-v オプションと -d オプションを指定します。

find-dir サブコマンドはクラスタ構成情報を変更しません。

スーパーユーザー以外のユーザーがこのコマンドを使用するには、solaris.cluster.read RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

list

クラスタに構成されている NAS デバイス構成を表示します。

クラスタに構成されているデバイスのディレクトリとデバイスタイプを表示するには、詳細オプション -v を使用します。

特定のタイプの NAS デバイスを表示するには、-t オプションを使用します。

スーパーユーザー以外のユーザーがこのコマンドを使用するには、solaris.cluster.read RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

remove

指定されている単数または複数の NAS デバイスを、Oracle Solaris Cluster 構成から削除します。

強制オプション -F を指定しない場合、remove-dir サブコマンドを使用して、NAS デバイスディレクトリを構成から削除しておいてください。

強制オプション -F を指定する場合、このコマンドは、NAS デバイスとそのディレクトリをクラスタ構成から削除します。「-**オプション**」セクションの *F* の説明を参照してください。

スーパーユーザー以外のユーザーがこのコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

add サブコマンドの説明も参照してください。

remove-dir

指定した NAS ディレクトリまたはプロジェクトを Oracle Solaris Cluster 構成から削除します。

remove-dir サブコマンドは、-d オプションで指定したエクスポート済みディレクトリまたはプロジェクトを削除します。-d all を使用すると、このサブコマンドによって、指定した NAS デバイスのすべてのディレクトリまたはプロジェクトが削除されます。

NAS デバイスからディレクトリまたはプロジェクトを削除した場合、この remove-dir サブコマンドを使用して、クラスタ構成からそのディレクトリまたはプロジェクトを削除する必要があります。クラスタ構成内の NAS ディレクトリまたはプロジェクトは、NAS デバイスからエクスポートした既存のディレクトリまたはプロジェクトと一致していなければなりません。

スーパーユーザー以外のユーザーがこのコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

add-dir サブコマンドの説明も参照してください。

set

特定の NAS デバイスの指定されたプロパティーを設定します。

スーパーユーザー以外のユーザーがこのコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

show

オプションが何も指定されていない場合は、次の情報を表示します。

- Oracle Solaris Cluster に構成されている現在のすべての NAS デバイスのリスト
- 各 NAS デバイスの使用可能なディレクトリ

- 各 NAS デバイスに関連付けられたすべてのプロパティー

特定のタイプの NAS デバイスを表示するには、-t オプションを指定します。特定のデバイスに関する情報を表示するには、NAS デバイスのホスト名をオペランドとしてコマンドに渡します。

指定されたプロジェクトに含まれているファイルシステムを表示するには、show サブコマンドで -d および -v オプションを使用します。all キーワードを使用して、NAS デバイスのすべてのプロジェクトまたは個々のプロジェクトを表示できます。

スーパーユーザー以外のユーザーがこのコマンドを使用するには、solaris.cluster.read RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

次のオプションがサポートされています。

-?
--help

ヘルプ情報を表示します。このオプションを使用するとき、ほかのオプションの処理は実行されません。

このオプションはサブコマンド付きでもサブコマンドなしでも指定できます。

サブコマンドなしでこのオプションを指定すると、このコマンドのサブコマンドのリストが表示されます。

サブコマンド付きでこのオプションを指定すると、サブコマンドの使用方法が表示されます。

-d *directory*[,…]
--directory=*directory*-[,…]
--directory *directory*-[,…]
-d *project*[,…]
--directory=*project*-[,…]
--directory *project*-[,…]

sun_uss NAS デバイスのプロジェクトを指定します。sun_uss NAS デバイスでは、ファイルシステムを作成する前に NAS デバイスでプロジェクトを作成する必要があります。プロジェクト名を / で始めることはできません。ファイルシステムはプロジェクト内で作成する必要があります。*プロジェクト*は sun_uss NAS デバイスの用語です。プロジェクトでは任意の数のファイルシステムを作成できます。このオプションは、add-dir、remove-dir、export、および show サブコマンドでのみ使用します。

このオプションは、特別なキーワード all を受け入れます。-d all オプションを使用すると、指定した NAS デバイス上のすべてのディレクトリが指定されます。

- remove-dir サブコマンドと一緒に使用する場合、指定したデバイス上のすべてのディレクトリが削除されます。
- export サブコマンドと一緒に使用する場合、指定したデバイス上のすべてのディレクトリの構成情報が、指定した出力に表示されます。

- `add-dir` サブコマンドと `-i` *configfile* オプションを使用すると、構成ファイルに一覧表示されている指定された NAS デバイス上のすべてのディレクトリが追加されます。
- `show` サブコマンドおよび `find-dir` サブコマンドを使用する際に、sun_uss NAS デバイスの `-v` オプションを指定すると、`-d` オプションで指定したプロジェクトに含まれるファイルシステムが表示されます。`all` キーワードを使用して、NAS デバイスのすべてのプロジェクトまたは個々のプロジェクトを表示できます。

`-F`
`--force`

指定した NAS デバイスを強制的に削除します。

強制オプションを一緒に指定できるのは、`remove` サブコマンドだけです。この強制オプションを `remove` サブコマンドと一緒に使用すると、NAS デバイスとその構成されているディレクトリが Oracle Solaris Cluster 構成から削除されます。

`-f` *passwd-file*
`--passwdfile=`*passwd-file*
`--passwdfile` *passwd-file*

NAS デバイスにログインするときに使用するパスワードが格納されているパスワードファイルを指定します。

セキュリティー上の理由により、パスワードはコマンド行オプションには指定できません。パスワードのセキュリティーを強化するには、パスワードをテキストファイルに格納し、`-f` オプションでこのファイルを指定してください。パスワード用の入力ファイルを指定しないと、パスワードの入力が求められます。

入力ファイルの読み取り可能アクセス権を root だけに与え、このファイルにグループからも一般ユーザーからもアクセスできないようにします。

`clnasdevice add` を `-i` オプションと一緒に使用するとき、*clconfigfile* にパスワードが格納されていない場合、`-f` *passwdfile* オプションを必ず指定してください。

入力ファイルでは、次の制限を守ってください。

- パスワードは単一行に指定します。パスワードを複数行にまたがって入力してはいけません。
- 行頭にあるブランクやタブは無視されます。
- コメントは引用符で囲まれていない # 記号で始まります。コメントは次の新しい行に続きます。

  すべてのコメントはパーサーによって無視されます。
- デバイスユーザーのパスワードに入力ファイルを使用する場合は、# 記号をパスワードの一部に使用できません。

`-i` *clconfigfile*
`--input={-` | *clconfigfile*`}`
`--input {-` | *clconfigfile*`}`

NAS デバイスを作成または変更するのに使用される構成情報を指定します。この情報は、1447 ページの`clconfiguration`(5CL) のマニュアルページに定義されている形式に準拠している必要があります。この情報はファイルに含まれている場合と、標準入力 `stdin` を通して指定される場合があります。標準入力を指定するには、ファイル名の代わりに `-` を指定します。

コマンド行と *clconfigfile* ファイルで同じプロパティーを指定する場合、コマンド行で設定したプロパティーが優先されます。

`clnasdevice add` を `-i` オプションと一緒に使用するとき、`-f` *passwdfile* オプションを必ず指定してください。

`-o` {`-` | *clconfigfile*}
`--output={-` | *clconfigfile*`}`
`--output {-` | *clconfigfile*`}`

1447 ページの`clconfiguration`(5CL) のマニュアルページで定義されている形式で、NAS デバイス構成情報を書き込みます。この情報は、ファイルまたは標準出力 (`stdout`) のどちらにでも書き込むことができます。標準出力を指定するには、ファイル名の代わりに `-` を指定します。

`-p` *name*`=`*value*`[,…]`
`--property=`*name*`=`*value*`[,…]`
`--property` *name value*`[,…]`

ある NAS デバイスタイプに固有のプロパティーを指定します。

`add` サブコマンドを使用して新しい NAS デバイスをクラスタ構成に追加する場合は、このオプションを指定する必要があります。`set` サブコマンドで NAS デバイスのプロパティーを変更する場合も、このオプションを指定する必要があります。詳細は、`-t` オプションの説明を参照してください。

`-t` *nas-device-type*
`--type=`*nas-device-type*
`--type` *nas-device-type*

NAS デバイスタイプを指定します。このオプションは、NAS デバイスを Oracle Solaris Cluster 構成に追加するときに指定してください。NAS デバイスタイプは、ベンダー名で指定します。たとえば、OracleZFS Storage Appliance NAS デバイスタイプは `sun_uss` です。

異なるタイプの NAS デバイスには異なるプロパティーがあるか、またはプロパティーがまったくない場合もあります。

`sun_uss`	OracleZFS Storage Appliance NAS デバイスを指定します。 `-p userid=osc_agent [-f passwd-file]` または `-u userid [-f passwdfile]`

`userid` は `osc_agent` である必要があります。sun_uss を使用する前に、クライアントコードをダウンロードし、すべてのクライアントノードにインストールする必要があります。この `osc_agent userid` は、デバイスでいずれかのワークフローを実行すると作成されます。この `userid` は、`userid` を入力として受け取る `clnasdevice` サブコマンドを使用する前に、デバイスで作成されている必要があります。

`userid` とパスワードプロパティーは必須です。

`-p "nodeIPs{`*node*`}"=[` *IP*`]`

このプロパティーは、各ノードの IP を指定します。クラスタノード名の IP とは異なる IP を使用して NAS デバイスにアクセスする場合、`nodeIPs`*node* プロパティーを使用してこの IP を指定できます。このプロパティーは省略可能です。IP を指定しない場合は、クラスタノード名の IP が使用されます。これらの IP は、NAS デバイス上のプロジェクトの `NFS Access Mode` で指定された IP と一致する必要があります。

プロパティーの値を指定しない場合 (例: `-p "nodeIPs{`*node*`}"=`)、指定したノードの IP がクラスタ構成から削除され、クラスタノード名の IP が使用されます。

sun_uss NAS デバイスとそのプロジェクトを追加する前に、必要な設定を行う必要があります。この設定タスクには、クライアントコードのダウンロードとクラスタノードへのインストールが含まれます。デバイスで `Configure for Oracle Solaris Cluster NFS` ワークフローを実行して、`userid osc_agent` とそのパスワードを作成します。プロジェクトを作成し、`Share Mode` を none または read-only に設定します (read-write モードもサポートされていますがお勧めしません)。`NFS Access Mode` では、ネットワーク概念を使用し、クラスタノードの IP に対する read-write アクセスを付与する必要があります。

NAS デバイスとそのエクスポートされたディレクトリをクラスタ構成に追加する前に、次に示す手順を実行しておく必要があります。

- NAS デバイスを設定します。
- ディレクトリを設定し、クラスタノードに使用可能にします。
- NAS デバイスで管理タスクを実行するときに使用するユーザー ID とパスワードを決定します。

また、NAS デバイスは起動され、動作している必要があります。詳細は、Unresolved link to " Oracle Solaris Cluster With Network-Attached Storage Device Manual "を参照してください。

`-u` *userid*
`--userid=`*userid*
`--userid` *userid*

NAS デバイスにログインするのに使用するユーザー ID を指定します。

クラスタは、ログインし、デバイス上で管理処理を実行するために、ユーザー ID を把握する必要があります。

あるいは、-p オプションでもユーザー ID を指定できます。詳細は、-p オプションを参照してください。

このオプションを一緒に指定できるのは、add および set サブコマンドだけです。

-V
--version

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンドやオペランドなどのオプションは無視されます。コマンドのバージョンが表示されます。ほかの処理は実行されません。

-v
--verbose

詳細な情報を標準出力 stdout に表示します。

-Z {*zoneclustername* | global | all}
--zoneclustername={*zoneclustername* | global | all}

nas-device-type が登録されている、処理するクラスタを指定します。

このオプションは、export コマンド以外のすべてのサブコマンドでサポートされています。

このオプションを指定する場合は、次のいずれかの引数も指定する必要があります。

zoneclustername
: このオプションと一緒に使用するコマンドが、*zoneclustername* という名前のゾーンクラスタ内でのみ、指定された各 *nas-device-type* に対して動作することを指定します。

global
: このオプションと一緒に使用するコマンドが、グローバルクラスタ内でのみ、指定された各 *nas-device-type* に対して動作することを指定します。

all
: グローバルクラスタ内でこの引数を使用した場合は、この引数と一緒に使用するコマンドが、すべてのクラスタ (グローバルクラスタおよびすべてのゾーンクラスタを含む) 内で、指定された各 *nas-device-type* に対して動作することを指定します。

  ゾーンクラスタ内でこの引数を使用した場合は、この引数と一緒に使用するコマンドが、そのゾーンクラスタ内でのみ、指定された各 *nas-device-type* に対して動作することを指定します。

次のオペランドがサポートされています。

nasdevice

NAS デバイスの名前。NAS デバイス名は、ネットワークで通信を行うために NAS デバイスが使用するホスト名です。NAS デバイスと通信するために、クラスタは NAS デバイス

の NAS ホスト名を知っておく必要があります。サブコマンドが複数の NAS デバイスを受け入れる場合は、プラス記号 + を使用して、すべての NAS デバイスを指定できます。add および add-dir サブコマンドの場合、プラス記号オペランドは、指定された構成ファイルにあるすべての NAS デバイスを指定します。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

次の終了値が返される可能性があります。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

15 CL_EPROP

無効なプロパティーです

-p、-y、または -x オプションで指定したプロパティーまたは値が存在しないか、許可されていません。

18 CL_EINTERNAL

内部エラーが発生しました

内部エラーは、ソフトウェアの欠陥またはその他の欠陥を示しています。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

41 CL_ETYPE

無効なタイプです

-t または -p オプションで指定したタイプは存在しません。

例 89 OracleZFS Storage Appliance からクラスタまたはゾーンクラスタへの NAS デバイスの追加

次の clnasdevice コマンドは、タイプが sun_uss の OracleZFS Storage Appliance uss7110-01 を構成に追加します。

```
# clnasdevice add -t sun_uss -p userid=osc_agent -f passwd-file uss7110-01
```

例 90 クラスタへの NAS プロジェクトの追加

次の clnasdevice コマンドは、すでに構成されている NAS デバイスである uss7110-01 に 2 つのプロジェクトを追加します。

```
# clnasdevice add-dir -d pool-0/local/nassa-pl,pool-0/local/nassa-p2 uss7110-01
```

例 91 クラスタまたはゾーンクラスタから NAS デバイスを削除する

次の clnasdevice コマンドは、NAS デバイス uss7110-01 と、残りのプロジェクトがある場合はそのプロジェクトをすべて、sun というクラスタ構成から削除します。

```
# clnasdevice remove -F uss7110-01
```

次の clnasdevice コマンドは、NAS デバイス uss7110-01 とその残りのすべてのプロジェクト (存在する場合) をゾーンクラスタの zc 構成から削除します。

```
# clnasdevice remove -Z ZC -F uss7110-01
```

次の例は、nodeIPs プロパティーを更新する方法を示しています。

```
# clnasdevice set -p "nodeIPs{cluster-1}"=10.155.55.145 \
-p "nodeIPs{cluster-2}"=10.155.55.146 uss7110-01
```

次の例では、クラスタ構成から現在の IP の設定を削除して、クラスタノード名の IP が使用されるようにします。

```
# clnasdevice set -p "nodeIPs{cluster-1}"= -p "nodeIPs{cluster-2}"= uss7110-01
```

例 92 クラスタに追加されていない NAS デバイスプロジェクトを表示する

次の clnasdevice コマンドは、クラスタにまだ追加されていない NAS デバイスプロジェクト名を表示します。

```
# clnasdevice find-dir uss7110-01
Nas Device:           uss7110-01
Type:                 sun_uss
Unconfigured Project: pool-0/local/nassa-p2
Unconfigured Project: pool-0/local/nassa-p1
```

例 93 クラスタまたはゾーンクラスタに構成されている NAS デバイスを表示する

次の clnasdevice コマンドは、クラスタに構成されているすべての NAS デバイスの名前を表示します。デバイスとそのディレクトリのリストを表示するには、詳細オプションか show サブコマンドを使用します。

```
# clnasdevice list
uss7110-01
```

次の clnasdevice コマンドは、ゾーンクラスタ ZC に構成されているすべての NAS デバイスの名前を表示します。デバイスとそのディレクトリのリストを表示するには、詳細オプションか show サブコマンドを使用します。

```
# clnasdevice list -Z ZC
ZC:uss7110-01
```

次の clnasdevice コマンドは、ゾーンクラスタ ZC に構成されているすべての NAS デバイスの名前を表示します。デバイスとそのディレクトリのリストを表示するには、詳細オプションか show サブコマンドを使用します。

```
# clnasdevice list -Z all
global:uss7110-01
ZC:uss7110-01
```

例 94 NAS デバイスとそのプロジェクトを表示する

次の `clnasdevice` コマンドは、クラスタに構成されているすべての NAS デバイスの名前とプロジェクトファイルシステムを表示します。

```
# clnasdevice show -v -d all uss7110-01
Nas Device:      uss7110-01
Type:            sun_uss
Project:         pool-0/local/nassa-p1
  File System:   /export/nassa-p1/nassa-p1-fs1
  File System:   /export/nassa-p1/nassa-p1-fs2
  File System:   /export/nassa-p1/nassa-p1-fs3
Project:         pool-0/local/nassa-p2
  File System:   /export/nassa-p2/nassa-p2-fs1
  File System:   /export/nassa-p2/nassa-p2-fs2
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページの`Intro`(1CL), 551 ページの`cluster`(1CL)

Unresolved link to " Oracle Solaris Cluster With Network-Attached Storage Device Manual "

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のサブコマンドとオプションを指定してこのコマンドを実行できます。

- `-?` オプション
- `-V` オプション

スーパーユーザー以外のユーザーがほかのサブコマンドを指定してこのコマンドを実行するには、RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
`add`	`solaris.cluster.modify`

サブコマンド	RBAC の承認
add-dir	solaris.cluster.modify
export	solaris.cluster.read
find-dir	solaris.cluster.read
list	solaris.cluster.read
set	solaris.cluster.modify
remove	solaris.cluster.modify
remove-dir	solaris.cluster.modify
show	solaris.cluster.read

名前

clnode — Oracle Solaris Cluster ノードの管理

```
/usr/cluster/bin/clnode -V

/usr/cluster/bin/clnode [subcommand] -?

/usr/cluster/bin/clnode subcommand [options] -v [node …]

/usr/cluster/bin/clnode add -n sponsornode[-i {- | clconfigfile}] -c clustername] [-G globaldevfs]
[-e endpoint,endpoint] node

/usr/cluster/bin/clnode create-loadlimit -p limitname=value[-p
     softlimit=value] [-p hardlimit=value] {+ | node[:zone] …}

/usr/cluster/bin/clnode clear [-F] node...

/usr/cluster/bin/clnode delete-loadlimit -p limitname=value
     {+ | node[:zone] …}

/usr/cluster/bin/clnode evacuate [-T seconds] {+ | node …}

/usr/cluster/bin/clnode export [-o {- | clconfigfile}][+ | node …]

/usr/cluster/bin/clnode list [-Z {zoneclustername | global | all}]
     [+ | node …]

/usr/cluster/bin/clnode rename -n newnodename[node]

/usr/cluster/bin/clnode remove [-n sponsornode][-G globaldevfs]
     [-F] [node]

/usr/cluster/bin/clnode set [-p name=value] […] {+ | node …}

/usr/cluster/bin/clnode set-loadlimit -p limitname=value[-p
     softlimit=value] [-p hardlimit=value] {+ | node[:zone] …}

/usr/cluster/bin/clnode show [-p name[,…]][-Z {zoneclustername |
     global | all}][+ | node …]

/usr/cluster/bin/clnode show-rev [node]

/usr/cluster/bin/clnode status [-m][-Z {zoneclustername| global |
     all}][+ | node …]
```

このコマンドは次のことを行います。

- ノードをクラスタに追加します
- ノードをクラスタから削除します

- すべてのリソースグループとデバイスグループのスイッチオーバーを試みます
- ノードのプロパティーの変更
- ノードの負荷制限を管理します
- 1 つまたは複数のノードのステータスと構成の報告またはエクスポート

clnode コマンドのほとんどのサブコマンドはクラスタモードで機能します。これらのサブコマンドのほとんどは、クラスタの任意のノードから実行できます。ただし、add および remove サブコマンドは例外です。これらのコマンドは非クラスタモードで実行してください。

add および remove サブコマンドを実行するときには、追加または削除しようとしているノードから実行してください。clnode add コマンドはまた、ノード自身もクラスタに参加できるように初期化します。clnode remove コマンドは、削除されるノードでクリーンアップ処理も実行します。

subcommand を省略できるのは、*options* が -? オプションまたは -V オプションの場合のみです。

このコマンドの各オプションには長形式と短形式があります。各オプションの両方の形式は、オプション に説明とともに記載されています。

clnode コマンドに短形式はありません。

このコマンドは、ゾーンクラスタでいくつかの形式で使用できます。クラスタでのこのコマンドの有効な使用方法の詳細については、個々のサブコマンドの説明を参照してください。

サブコマンド

サポートされるサブコマンドには次のものがあります。

add

ノードをクラスタに構成および追加します。

このサブコマンドは、大域ゾーンだけで使用できます。 このサブコマンドは、大域ゾーンだけで使用できます。

このサブコマンドは非クラスタモードで実行してください。

ノードを構成および追加するには、-n *sponsornode* オプションを使用してください。このオプションは、既存のアクティブなノードをスポンサノードとして指定します。ノードをクラスタに構成するときには常に、スポンサノードが必要です。

-c *clustername* を指定しない場合、このサブコマンドは、最初に追加するノードの名前を新しいクラスタ名として使用します。

オペランド *node* はオプションです。しかし、オペランドを指定する場合、そのオペランドはサブコマンドを実行するノードのホスト名でなければなりません。

注記 - Oracle Solaris Cluster ソフトウェアをインストールするには、`pkg install` コマンドを実行します。次に、`scinstall` ユーティリティーを実行して新しいクラスタを作成するか、または既存のクラスタにノードを追加します。手順については、Unresolved link to " Oracle Solaris Cluster ソフトウェアのインストール "を参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`clear`

`remove` サブコマンドを実行したあと、クラスタノードについての情報が残っていれば、クリーンアップまたはクリアします。

このサブコマンドは、大域ゾーンだけで使用できます。このサブコマンドは、大域ゾーンだけで使用できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`create-loadlimit`

ノードに負荷制限を追加します。

このサブコマンドは、大域ゾーンまたはゾーンクラスタで使用できます。

オプションの `-p` オプションを参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`delete-loadlimit`

ノードの既存の負荷制限を削除します。

このサブコマンドは、大域ゾーンまたはゾーンクラスタで使用できます。

オプションの `-p` オプションを参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`evacuate`

すべてのリソースグループおよびデバイスグループを、指定ノードから新しいプライマリノードのセットに切り替えることを試みます。

このサブコマンドは、大域ゾーンまたはゾーンクラスタノードで使用できます。

システムは、各グループの構成済みの設定に従って、新しいプライマリノードの選択を試みます。退避されたすべてのリソースグループが同じプライマリノードによって再マスターされるとは限りません。指定されたノードから 1 つまたは複数のリソースグループまたはデバイス

グループを退避できない場合、このサブコマンドは失敗します。失敗した場合、このサブコマンドはエラーメッセージを発行して、ゼロ以外の終了コードで終了します。このサブコマンドがデバイスグループのプライマリ所有権をほかのノードに変更できない場合、元のノードがそのデバイスグループのプライマリ所有権を保持します。RGM が退避したリソースグループを新しいプライマリ所有権で起動できない場合、その退避したリソースグループがオフラインになる可能性があります。

このサブコマンドの -T オプションを使用して、リソースグループが切り替わらないようにする秒数を指定できます。値を指定しない場合、デフォルトでは、60 秒が使用されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`export`

ノード構成情報をファイルまたは標準出力 (stdout) にエクスポートします。

このサブコマンドは、大域ゾーンだけで使用できます。このサブコマンドは、大域ゾーンだけで使用できます。

-o オプションとファイルの名前を指定すると、構成情報はそのファイルに書き込まれます。

-o オプションとファイルの名前を指定しない場合、構成情報は標準出力に書き込まれます。

このサブコマンドはクラスタ構成データを変更しません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`list`

クラスタに構成されているノードの名前を表示します。

このサブコマンドと -Z オプションを指定すると、指定したノード (特に 1 つ以上のクラスタ) の名前が次のように列挙されます。

- すべての広域クラスタノードとゾーンクラスタノード
- すべての広域クラスタノードのみ
- 名前を指定したゾーンクラスタノードのみ

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

node オペランドを指定しない場合、または、プラス記号オペランド (+) を指定する場合、このサブコマンドはすべてのノードメンバーを表示します。

このサブコマンドはクラスタモードで実行してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.read 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`remove`

ノードをクラスタから削除します。

このサブコマンドは、大域ゾーンだけで使用できます。このサブコマンドは、大域ゾーンだけで使用できます。

このサブコマンドは非クラスタモードで実行してください。

クラスタからノードを削除するには、次のガイドラインに従います。これらのガイドラインに従わない場合、ノードを削除すると、クラスタの定足数が漏洩する場合があります。

- `-F` オプションも指定する場合を除き、いずれかの定足数デバイスから削除するノードは構成解除してください。
- 削除されるノードがアクティブなクラスタメンバーでないことを確認します。
- 1 つ以上の共有定足数デバイスが構成されている場合を除き、3 ノードクラスタからノードを削除しないでください。

このサブコマンドは、クラスタ構成デバイスから、ノード参照のサブセットを削除しようとします。`-F` オプションを指定すると、このサブコマンドは、クラスタ構成データベースから、すべてのノード参照を削除しようとします。

注記 - ノードからクラスタソフトウェアを削除するには、`scinstall -r` コマンドを実行する必要があります。詳細は、Unresolved link to " Oracle Solaris Cluster ソフトウェアのインストール "を参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`rename`

ノードの名前を新しいノード名に変更します。

このサブコマンドは、大域ゾーンだけで使用できます。このサブコマンドは非クラスタモードで実行してください。

注記 - このコマンドは、Oracle Solaris ホスト名を変更したノードと同じノードで実行する必要があります。

ノードの名前を *newnodename* に変更するには、`-n` *newnodename* オプションを使用します。現在のアクティブな Oracle Solaris ノードの名前を、*oldnodename* から変更する必要があります。このコマンドを正常に実行するには、クラスタ内のすべてのノードが非クラスタノードでなければなりません。

オペランドはオプションであり、サブコマンドを実行するノードのホスト名でなければなりません。

注記 - ノードの名前を変更する前に、まず Oracle Solaris のホスト名変更手順を実行し、クラスタ内のクラスタノードの名前を変更する必要があります。手順については、Unresolved link to " Managing System Information, Processes, and Performance in Oracle Solaris 11.2 のHow to Change a System's Identity"を参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`set`

指定されたノードに関連するプロパティーを変更します。

このサブコマンドは、大域ゾーンだけで使用できます。このサブコマンドは、大域ゾーンだけで使用できます。

オプションの `-p` オプションを参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`set-loadlimit`

ノードの既存の負荷制限を変更します。

このサブコマンドは、大域ゾーンまたはゾーンクラスタで使用できます。

オプションの `-p` オプションを参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`show`

指定されたノード (1 つまたは複数) の構成、または、それらノードのプロパティーについての情報を表示します。

このサブコマンドと `-Z` オプションを指定すると、指定した 1 つ以上のノード (特に 1 つ以上のクラスタ) の構成情報またはプロパティー情報が次のように表示されます。

- すべての広域クラスタノードとゾーンクラスタノード
- すべての広域クラスタノードのみ
- 名前を指定したゾーンクラスタノードのみ

このサブコマンドは、大域ゾーンだけで使用できます。このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

オペランドを指定しない場合やプラス記号 (+) を指定した場合、このサブコマンドは、すべてのクラスタノードの情報を表示します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`show-rev`

ノードにインストールされている Solaris Cluster パッケージの名前とリリース情報を表示します。

このサブコマンドは、大域ゾーンだけで使用できます。

このサブコマンドは非クラスタモードおよびクラスタモードで実行できます。非クラスタモードで実行する場合、このサブコマンドを実行したノードの名前だけを指定でき、そのノードについての情報を取得できます。クラスタモードで実行する場合、クラスタ内の任意のノードの名前を指定でき、そのノードについての情報を取得できます。

このサブコマンドを `-v` オプションと一緒に使用すると、このサブコマンドは、パッケージの名前、バージョン、およびそれらのパッケージに適用されているパッチを表示します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`status`

指定したノードまたは Internet Protocol (IP) ネットワークマルチパス (IPMP) グループのステータスを表示します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

オペランドを指定しない場合やプラス記号 (`+`) を指定した場合、このサブコマンドは、すべてのクラスタノードのステータスを表示します。ノードのステータスは `Online` または `Offline` のどちらかです。

このサブコマンドで `-m` オプションを指定すると、Oracle Solaris IPMP グループのみが表示されます。

このサブコマンドで詳細オプション `-v` を指定すると、クラスタノードと Oracle Solaris IPMP グループの両方のステータスが表示されます。

このサブコマンドと `-Z` オプションを指定すると、指定した 1 つ以上のノード (特に 1 つ以上のクラスタ) のステータス情報が次のように表示されます。

- すべての広域クラスタノードとゾーンクラスタノード
- すべての広域クラスタノードのみ
- 名前を指定したゾーンクラスタノードのみ

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

注記 - このセクションでは、各オプションの短い形式と長い形式の両方が示されています。

次のオプションがサポートされています。

-?
--help

ヘルプ情報を表示します。

このオプションは*サブコマンド*付きでもサブコマンドなしでも指定できます。

subcommand を指定しない場合、使用可能なすべてのサブコマンドのリストが表示されます。

subcommand を指定する場合、そのサブコマンドの使用法が表示されます。

このオプションとその他のオプションを指定すると、その他のオプションは無視されます。

-c *clustername*
--clustername=*clustername*
--clustername *clustername*

ノードを追加するクラスタの名前を指定します。

このオプションを一緒に指定できるのは、add サブコマンドだけです。

このオプションを指定する場合、指定する *clustername* は既存のクラスタの名前と一致する必要があります。そうでない場合、エラーが発生します。

-e *endpoint*,*endpoint*
--endpoint=*endpoint*,*endpoint*
--endpoint *endpoint*,*endpoint*

トランスポート接続を指定します。

このオプションを一緒に指定できるのは、add サブコマンドだけです。このオプションは、クラスタトランスポートトポロジを確立するのに指定します。トポロジを確立するには、アダプタとスイッチを接続するケーブルを構成します。終端としては、アダプタまたはスイッチを指定できます。ケーブルを示すには、コンマ区切りの終端のペアを指定します。ケーブルは、現在のノード上のクラスタトランスポートアダプタから、次のいずれかへの接続を確立します。

- クラスタトランスポートスイッチ上のポート。トランスポート接続点とも呼ぶ。
- すでにクラスタに含まれる別のノード上のアダプタ。

-e オプションを指定しない場合、add サブコマンドはデフォルトケーブルの構成を試みます。ただし、clnode コマンドの 1 つのインスタンス内で複数のトランスポートアダプタまたはスイッチを構成する場合、clnode はデフォルトを構築できません。デフォルトでは、シングル構成のトランスポートアダプタからシングル構成 (つまり、デフォルト) のトランスポートスイッチまでに、1 つのケーブルが構成されます。

-e オプションを指定するたびに、必ず、コンマ区切りの終端のペアを指定してください。終端のペアはそれぞれ、1 つのケーブルを定義します。個々の終端はそれぞれ、次のいずれかで指定されます。

- アダプタ終端:

  node:*adapter*

- スイッチ終端:

  switch[@*port*]

タグ付き VLAN アダプタを指定するには、物理デバイス名と VLAN インスタンス番号から派生されたタグ付き VLAN アダプタ名を使用します。VLAN インスタンス番号は、VLAN ID に 1000 を掛けて、元の物理ユニット番号を足したものです。たとえば、物理デバイス `net2` 上の VLAN ID 11 は、タグ付き VLAN アダプタ名 `net11002` に変換されます。

スイッチ終端のポートコンポーネントを指定しない場合、デフォルトのポートが割り当てられます。

`-F`
`--force`

指定されたノード上にグローバルマウントが残っているかどうかを確認せずに、そのノードを強制的に削除またはクリアします。

このオプションは、`clear` または `remove` サブコマンドとともにのみ使用します。

`-G {lofi | `*special* `| `*mount-point*`}`
`--globaldevfs={lofi | `*special* `| `*mount-point*`}`
`--globaldevfs {lofi | `*special* `| `*mount-point*`}`

グローバルデバイスのマウントポイントとして、`lofi` デバイス、raw *special* ディスクデバイス、または専用ファイルシステムを指定します。

このオプションを一緒に指定できるのは、`add` または `remove` サブコマンドだけです。

ノードをクラスタメンバーに追加するには、事前に各クラスタノードのローカルファイルシステムを `/global/.devices/node@`*nodeID* に対してグローバルにマウントしておく必要があります。しかし、`clnode` コマンドを実行するまで、ノード ID は未知です。デフォルトでは、`clnode add` コマンドは、`/globaldevices` にマウントされた、または `-G` オプションで指定したマウントポイントにマウントされた空のファイルシステムを検索します。このようなファイルシステムが指定されている場合、`clnode add` コマンドは `/etc/vfstab` ファイルに対して必要な変更を行います。指定したファイルシステムは、`/globaldevices` にマウントし直されます。ノード ID のマウントが見つからない場合、`clnode` コマンドは、`vfstab` ファイルにエントリを追加しようとします。Unresolved link to " vfstab4" のマニュアルページを参照してください。

`/global/.devices/node@`*nodeID* がマウントされておらず、空の `/globaldevices` ファイルシステムが提供されない場合は、コマンドが失敗します。

`-G lofi` が指定されている場合は、`/.globaldevices` ファイルが作成されます。`lofi` デバイスはそのファイルに関連付けられ、`lofi` デバイスにグローバルデバイスファイルシステムが作成されます。`/etc/vfstab` ファイルには `/global/.devices/node@`*nodeID* エントリは追加されません。`lofi` デバイスの詳細は、Unresolved link to " lofi7D" のマニュアルページを参照してください。

raw *special* ディスクデバイス名が指定され、`/global/.devices/node@`*nodeID* がマウントされていない場合、ファイルシステムは、`newfs` コマンドを使用してデバイス上に作成され

ます。デバイス名にすでにマウントされているファイルシステムを指定すると、エラーになります。

目安として、専用ファイルシステムのサイズは最低でも 512M バイトは必要です。このようなパーティションやファイルシステムが使用できない場合やサイズが不足している場合は、Oracle Solaris OS をインストールし直す必要が生じることもあります。

`lofi` デバイスに作成される名前空間向けに、ルートファイルシステムに 100 M バイトの空き領域が必要です。

このオプションは `remove` サブコマンドと一緒に使用して、以前の `/global/.devices` マウントポイントを復元するのに使用する新しいマウントポイント名を指定します。

広域デバイスの名前空間が専用パーティションにマウントされている場合に、このオプションを `remove` サブコマンドと一緒に使用することにより、以前の `/global/.devices` マウントポイントの復元に使用する新しいマウントポイント名を指定します。広域デバイスの名前空間が専用パーティションにマウントされている場合に `-G` オプションを指定しないと、デフォルトでそのマウントポイントの名前が `/globaldevices` に変更されます。

`-i` {`-` | *clconfigfile*}
`--input=`{`-` | *clconfigfile*}
`--input` {`-` | *clconfigfile*}

ノード構成情報をファイルまたは標準入力 (`stdin`) から読み取ります。この構成情報の形式は、1447 ページの`clconfiguration`(5CL) のマニュアルページで説明されています。

このオプションにファイル名を指定する場合、ノード構成情報はファイルから読み取られます。このオプションで `-` を指定した場合、構成情報は標準入力 (`stdin`) から読み取られます。

`-m`

IPMP グループを指定します。IPMP グループのステータスだけを表示する場合に、`status` サブコマンドと使用してください。

`-n` *newnodename*
`--newnodename=`*newnodename*
`--newnodename` *newnodename*

新しいノード名を指定します。

このオプションは、`rename` サブコマンドでのみ使用できます。

現在のノードの新しいノード名を指定できます。`rename` サブコマンドを使用してノード名を *newnodename* に変更する場合、現在のノードのホスト名がすでに *newnodename* に変更されている必要があります。

`-n` *sponsornode*
`--sponsornode=`*sponsornode*
`--sponsornode` *sponsornode*

スポンサノードの名前を指定します。

sponsornode には、名前またはノード識別子を指定できます。`add` サブコマンドを使用してノードをクラスタに追加するとき、クラスタに追加する最初のアクティブなノードがスポン

サノードになります。この時点から、そのノードがそのクラスタの *sponsornode* として残ります。`remove` サブコマンドを使用してノードを削除するとき、削除するノード以外の任意のアクティブなノードをスポンサノードとして指定できます。

デフォルトでは、*sponsornode* をサブコマンドに指定すると、*sponsornode* が属するクラスタがそのサブコマンドの影響を受けます。

`-o` {`-` | *clconfigfile*}
`--output=`{`-` | *clconfigfile*}
`--output` {`-` | *clconfigfile*}

ノード構成情報をファイルまたは標準出力 (`stdout`) に書き込みます。この構成情報の形式は、1447 ページの`clconfiguration`(5CL) のマニュアルページで説明されています。

このオプションにファイル名を指定する場合、このオプションは新しいファイルを作成します。次に、ノード構成情報はそのファイルに格納されます。`-` をこのオプションとともに指定すると、構成情報は標準出力 (`stdout`) に送信されます。このコマンドのほかの標準出力はすべて抑制されます。

このオプションを一緒に指定できるのは、`export` サブコマンドだけです。

`-p` *name*
`--property=`*name*
`--property` *name*

`show` サブコマンドで情報を表示するノードのプロパティーを指定します。

`set` サブコマンドで追加または変更できるプロパティーについては、`-p` *name*= *value* オプションの説明を参照してください。

このオプションに指定できるプロパティーは次のとおりです。

`privatehostname`

プライベートホスト名は、プライベートクラスタインターコネクト経由による指定されたノードの IP アクセスに使用されます。ノードをクラスタに追加する場合、このオプションはデフォルトでプライベートホスト名 `clusternode` *nodeid*`-priv` を使用します。

`reboot_on_path_failure`

このプロパティーに設定できる値は、`enabled` と `disabled` です。

`-p` *name*=*value*
`--property=`*name*=*value*
`--property` *name*=*value*

`set` サブコマンドで追加または変更するノードのプロパティーを指定します。

`-p` *name*= *value* は複数回指定できます。

`show` サブコマンドで情報を表示できるプロパティーについては、`-p` *name* オプションの説明を参照してください。

このオプションで変更できるプロパティーは次のとおりです。

`defaultpsetmin`

デフォルトプロセッサセットリソースで使用可能な CPU の最小数を設定します。

デフォルト値は 1、最小値も 1 です。最大値は、このプロパティーを設定しているマシンの CPU 数です。

`globalzoneshares`

大域ゾーンに割り当てられるシェアの数を設定します。

1 から 65535 まで (両端の値を含む) の値を指定できます。この上限を理解するには、Unresolved link to " prctl1" のマニュアルページの `zone.cpu-shares` 属性を参照してください。`globalzoneshares` のデフォルト値は 1 です。

`hardlimit`

ノードのリソースグループ負荷の必須上限を定義します。ノードの合計負荷は、強い制限値を超えることができません。

`hardlimit` プロパティーは、符号のない整数です。`softlimit` プロパティーは、符号のない整数です。`hardlimit` プロパティーのデフォルト値は、*null* です。null または空の値は、対応する `limitname` がそのノードで無制限であることを示します。空以外の値が指定されている場合、1,000 万を超えることはできません。

`limitname`

`limitname` プロパティーは文字列です。名前には、強い負荷制限値と弱い負荷制限値という 2 つの値が関連付けられており、これらはそれぞれ `hardlimit` プロパティーと `softlimit` プロパティーで指定されています。

各 `limitname` プロパティーに負荷係数を割り当てる方法については、305 ページの`clresourcegroup`(1CL) のマニュアルページを参照してください。また、`clresourcegroup` コマンドを使用して、優先順位とプリエンプションモードを決定することもできます。リソースグループ負荷をすべてのノードにわたって分散する方法については、551 ページの`cluster`(1CL) のマニュアルページを参照してください。

`privatehostname`

プライベートクラスタトランスポート経由での特定のノードへの IP アクセスに使用されます。ノードをクラスタに追加する場合、このオプションはデフォルトでプライベートホスト名 `clusternode`*nodeid* `-priv` を使用します。

プライベートホスト名を変更する前に、すべてのノードで、このプライベートホスト名を使用するすべてのリソースまたはアプリケーションを無効にしてください。Unresolved link to " Oracle Solaris Cluster システム管理 のノードのプライベートホスト名を変更する"にある「プライベートホスト名を変更する」というタイトルの例を参照してください。

`hosts` データベースやネーミングサービスのデータサービスには、プライベートホスト名を格納*しないでください*。Unresolved link to " hosts4" のマニュアルページを参照してください。特殊な `nsswitch` コマンドは、プライベートホスト名の全ホスト名検索を実行します。Unresolved link to " nsswitch.conf4" のマニュアルページを参照してください。

value を指定しない場合、このオプションは、デフォルトのプライベートホスト名 `clusternode`*nodeid*`-priv` を使用します。

`reboot_on_path_failure`

次の条件に適合する場合、モニターしているすべてのディスクパスが失敗したとき、ノードの自動リブートを有効にします。

- ノード上ですべてのモニター対象の共有ディスクパスが失敗した。
- 少なくとも 1 つのディスクがクラスタ内の異なるノードからアクセス可能である。`scdpm` デーモンは、プライベートインターコネクトを使用して、ディスクがクラスタ内の別のノードからアクセス可能かどうかをチェックします。プライベートインターコネクトが無効な場合、`scdpm` デーモンは別のノードからディスクのステータスを取得できません。

このプロパティーを変更するには、`set` サブコマンドだけを使用できます。このプロパティーは、`enabled` または `disabled` に設定できます。

ノードがリブートすると、そのノード上でマスターされているすべてのリソースグループとデバイスグループが別のノード上で再起動します。

ノードが自動リブートしたあと、ノード上のすべてのモニター対象共有ディスクパスがアクセス不能のままである場合、そのノードは再び自動リブートしません。ただし、ノードがリブートしたあとにモニター対象ディスクパスのどれかが使用可能になり、その後、すべてのモニター対象共有ディスクパスでふたたび問題が発見されると、ノードはふたたび自動的にリブートします。

`reboot_on_path_failure` プロパティーを有効にすると、ローカルディスクパスの状態は、ノードのリブートが必要かどうか決定するときには考慮されません。モニターされた共有ディスクのみが影響を受けます。

このプロパティーを `disabled` に設定し、ノード上のすべてのモニター対象共有ディスクパスが失敗した場合、そのノードはリブート*しなくなります*。

`softlimit`

ノードのリソースグループ負荷の推奨される上限を定義します。たとえば、負荷を分散するためのクラスタ容量が不足している場合など、ノードの合計負荷は、弱い制限値を超えることができます。弱い負荷制限値を超えると、クラスタのステータスを表示するコマンドやツールで、条件にフラグが付きます。

`softlimit` プロパティーは、符号のない整数です。`softlimit` プロパティーのデフォルト値は、*0* です。弱い制限値の値が *0* であるということは、弱い制限値が設定されていないことを意味します。ステータスコマンドからは、*弱い制限値の超過*に関する警告は表示されません。`softlimit` プロパティーの最大値は 1,000 万です。特定の負荷制限の `softlimit` プロパティーは、`hardlimit` の値以下でなければなりません。

-T *seconds*
--time=*seconds*
--time *seconds*

あるノードからリソースグループを退避したあと、そのノードにリソースグループがスイッチバックしないようにする時間を秒数で指定します。

このオプションを一緒に指定できるのは、evacuate サブコマンドだけです。*seconds* には、0 から 65535 までの整数値を指定する必要があります。値を指定しない場合、デフォルトでは、60 秒が使用されます。

退避が完了したあと 60 秒間または指定した秒数退避ノードになっていると、、リソースグループはフェイルオーバーできなかったり、自動的にオンラインになったりします。

ただし、switch または online サブコマンドを使用してリソースグループをオンライン、または退避されたノードリブートに切り替えると、退避タイマーはただちに期限切れになり、自動フェイルオーバーが再度許可されます。

-v
--verbose

詳細情報を標準出力 (stdout) で表示します。

-V
--version

コマンドのバージョンを表示します。

このオプションをほかのオプション、サブコマンド、またはオペランドと一緒に指定する場合、これらはすべて無視されます。コマンドのバージョンだけが表示されます。ほかの処理は行われません。

-Z {*zoneclustername* | global | all}
--zonecluster={*zoneclustername* | global | all}
--zonecluster {*zoneclustername* | global | all}

情報を表示する 1 つまたは複数のノードが存在する 1 つまたは複数のクラスタを指定します。

このオプションを指定する場合は、次のいずれかの引数も指定する必要があります。

zoneclustername　　*zoneclustername* という名前のゾーンクラスタノードだけに関する情報を表示するように指定します。

global　　広域クラスタノードだけに関する情報を表示するように指定します。

all　　*すべての*広域クラスタノードとゾーンクラスタノードに関する情報を表示するように指定します。

次のオペランドがサポートされています。

node　　管理対象のノードの名前です。

add サブコマンドを使用するときは、*node* のホスト名を指定します。ほかのサブコマンドを使用するときは、*node* のノード名またはノード識別子を指定します。

+ クラスタ内のすべてのノードです。

このコマンドセットにあるすべてのコマンドの終了ステータスコードの完全なリストについては、19 ページのIntro(1CL) のマニュアルページを参照してください。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

15 CL_EPROP

無効なプロパティーです

-p、-y、または -x オプションで指定したプロパティーまたは値が存在しないか、許可されていません。

35 CL_EIO

I/O エラー

物理的な入出力エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

37 CL_EOP

操作が許可されていません

サポートされていない構成に対する操作を実行しようとしたか、サポートされていない操作を実行しました。

例 95 ノードのクラスタへの追加

次のコマンドは、このコマンドを実行したノードを構成し、既存のクラスタに追加します。この例では、デフォルトで、グローバルデバイスのマウントポイントとして /globaldevices を使用します。デフォルトでは、この例はまた、プライベートホスト名として clusternode1-priv を使用します。

このコマンドは、クラスタ cluster-1 を指定し、スポンサノードが phys-schost-1 であることを指定しています。このコマンドはまた、アダプタ net1 がトランスポートスイッチ switch1 に接続されていることも指定しています。最後に、このコマンドは、アダプタ net2 がトランスポートスイッチ switch2 に接続されていることを指定しています。

```
# clnode add -c cluster-1 -n phys-schost-1 \
-e phys-schost-2:net1,switch1 -e phys-schost-2:net2,switch2
```

例 96 クラスタからのノードの削除

次のコマンドは、ノードをクラスタから削除します。このコマンドは、このコマンドを実行したノードを削除します。このノードは、非クラスタモードです。

```
# clnode remove
```

例 97 ノードに関連するプライベートホスト名の変更

次のコマンドは、ノード phys-schost-1 のプライベートホスト名をデフォルト設定に変更します。

```
# clnode set -p privatehost=phys-schost-1
```

例 98 すべてのノードに対するプライベートホスト名設定の変更

次のコマンドは、すべてのノードのプライベースホスト名設定をデフォルト値に変更します。この場合は、+ がプラス記号オペランドであることを示すために、等号 (=) とプラス記号 (+) の間に空白を挿入*する必要があります*。

```
# clnode set -p privatehost= +
```

例 99 グローバルクラスタノードまたはゾーンクラスタノードでの負荷制限の設定

次のコマンドは、グローバルクラスタのすべてのノードの既存の負荷制限を変更します。この例では、3 つの負荷制限 mem_load、disk_load、および cpu_load を定義し、それぞれに弱い制限値と強い制限値を設定しています。mem_load 負荷制限には弱い制限値 11 があり、disk_load には弱い制限値がなく、cpu_load には強い制限値がありません。この例の + オペランドは、すべてのノードの負荷制限を変更します。

```
# clnode set-loadlimit -p limitname=mem_load -p softlimit=11 -p hardlimit=20 +

# clnode set-loadlimit -p limitname=disk_load -p hardlimit=20 +

# clnode set-loadlimit -p limitname=cpu_load -p softlimit=8 node1:zone1 node2:zone2
```

次のコマンドは、大域ゾーンから、ゾーンクラスタノードの負荷制限を変更します。この例では、ゾーンクラスタノードで強い制限値を使用して負荷制限を定義しています。

```
# clnode set-loadlimit -
Z zoneclustername
 -p limitname=zc_disk_load -p
 hardlimit=15
zc-node1
```

例 100 クラスタ内のすべてのノードのステータスの表示

次のコマンドは、クラスタ内のすべてのノードのステータスを表示します。

```
# clnode status
=== Cluster Nodes ===

--- Node Status ---

Node Name                                       Status
---------                                       ------
phys-schost-1                                   Online
phys-schost-2                                   Online
```

例 101 クラスタ内のすべてのノードの冗長なステータスの表示

次のコマンドは、クラスタ内のすべてのノードの冗長なステータスを表示します。

```
# clnode status -v
=== Cluster Nodes ===

--- Node Status ---

Node Name                                                       Status
---------                                                       ------
phys-schost-1                                                   Online
phys-schost-2                                                   Online

--- Node IPMP Group Status ---

Node Name        Group Name        Status        Adapter        Status
---------        ----------        ------        -------        ------
phys-schost-1    sc_ipmp0          Online        net0           Online
phys-schost-2    sc_ipmp0          Online        net0           Online




--- Load Limit Status ---

Node Name      Load Limit Name   Soft Limit/Hard Limit   Load   Status

phys-schost-1  mem_load          30/50                   23     OK
               disk_load         10/15                   14     Softlimit Exceeded
               cpu_load          2/unlimited             1      OK
phys-schost-2  disk_load         90/97                   11     OK
               cpu_load          unlimited/unlimited     0      OK
```

例 102 すべてのノードの負荷制限ステータスを表示する

次のコマンドは、クラスタ内のすべてのノードの負荷制限ステータスを表示します。

```
# clnode status -l

--- Load Limit Status ---

Node Name      Load Limit Name   Soft Limit/Hard Limit   Load   Status

phys-schost-1  mem_load          30/50                   23     OK
               disk_load         10/15                   14     Softlimit Exceeded
               cpu_load          2/unlimited             1      OK
phys-schost-2  disk_load         90/97                   11     OK
               cpu_load          unlimited/unlimited     0      OK
```

例 103 クラスタ内のすべての広域クラスタノードとゾーンクラスタノードのステータスの表示

次のコマンドは、クラスタ内のすべての広域クラスタノードとゾーンクラスタノードのステータスを表示します。

```
# clnode status -Z all

=== Cluster Nodes ===

--- Node Status ---

Node Name                                       Status
---------                                       ------
global:phys-schost-1                            Online
global:phys-schost-2                            Online
global:phys-schost-4                            Online
global:phys-schost-3                            Online


=== Zone Cluster Nodes ===

--- Node Status ---

Node Name                                       Status
---------                                       ------
cz2:phys-schost-1                               Online
cz2:phys-schost-3                               Offline
```

例 104 クラスタ内のすべてのノードについての構成情報の表示

次のコマンドは、クラスタ内のすべてのノードについての構成情報を表示します。

```
# clnode show
=== Cluster Nodes ===

Node Name:                                      phys-schost-1
  Node ID:                                         1
  Enabled:                                         yes
  privatehostname:                                 clusternode1-priv
  reboot_on_path_failure:                          disabled
  globalzoneshares:                                1
  defaultpsetmin:                                  1
  quorum_vote:                                     1
  quorum_defaultvote:                              1
  quorum_resv_key:                                 0x4487349A00000001
  Transport Adapter List:                          net2, net3

Node Name:                                      phys-schost-2
  Node ID:                                         2
  Enabled:                                         yes
  privatehostname:                                 clusternode2-priv
  reboot_on_path_failure:                          disabled
```

```
  globalzoneshares:                               1
  defaultpsetmin:                                 1
  quorum_vote:                                    1
  quorum_defaultvote:                             1
  quorum_resv_key:                                0x4487349A00000002
  Transport Adapter List:                         net2, net3
```

例 105 クラスタ内の特定のノードについての構成情報の表示

次のコマンドは、クラスタ内の `phys-schost-1` に関する構成情報を表示します。

```
# clnode show phys-schost-1
=== Cluster Nodes ===

Node Name:                                      phys-schost-1
  Node ID:                                        1
  Enabled:                                        yes
  privatehostname:                                clusternode1-priv
  reboot_on_path_failure:                         disabled
  globalzoneshares:                               1
  defaultpsetmin:                                 1
  quorum_vote:                                    1
  quorum_defaultvote:                             1
  quorum_resv_key:                                0x4487349A00000001
  Transport Adapter List:                         net2, net3
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/system/core`
インタフェースの安定性	発展中

Unresolved link to " prctl1", 37 ページの`claccess`(1CL), 305 ページの`clresourcegroup`(1CL), 551 ページの`cluster`(1CL), 19 ページの`Intro`(1CL), Unresolved link to " newfs1M", Unresolved link to " su1M", Unresolved link to " hosts4", 819 ページの`scinstall`(1M), Unresolved link to " nsswitch.conf4", Unresolved link to " vfstab4", Unresolved link to " attributes5", Unresolved link to " rbac5", 1447 ページの`clconfiguration`(5CL), Unresolved link to " lofi7D"

Unresolved link to " Oracle Solaris Cluster システム管理 のクラスタの管理の概要"で、プライベートホスト名の変更方法を示す例を参照してください。

スーパーユーザーはこのコマンドのすべての形式を実行できます。

すべてのユーザーがこのコマンドに -? (ヘルプ) オプションまたは -V (バージョン) オプションを指定して実行できます。

スーパーユーザー以外のユーザーが clnode コマンドにサブコマンドを付けて実行するには、RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
add	solaris.cluster.modify
clear	solaris.cluster.modify
create-loadlimit	solaris.cluster.modify
delete-loadlimit	solaris.cluster.modify
evacuate	solaris.cluster.admin
export	solaris.cluster.read
list	solaris.cluster.read
remove	solaris.cluster.modify
rename	solaris.cluster.modify
set	solaris.cluster.modify
set-loadlimit	solaris.cluster.modify
show	solaris.cluster.read
show-rev	solaris.cluster.read
status	solaris.cluster.read

名前

clpstring, clps — Oracle Solaris Cluster プライベート文字列の管理

```
/usr/cluster/bin/clpstring -V

/usr/cluster/bin/clpstring subcommand -?

/usr/cluster/bin/clpstring subcommand [options] -v [pstring-name[...]

/usr/cluster/bin/clpstring create -b object-instance [-f
     stringvalue-file] [-t object-type] [-Z {zoneclustername | global}]
     pstring-name

/usr/cluster/bin/clpstring delete [-F] [-Z {zoneclustername[,…] |
     global | all}] + | pstring-name …

/usr/cluster/bin/clpstring list [-b object-instance[,…]] [-t
     type[,…]] [-Z {zoneclustername[,…] | global | all}] [+ |
     pstring-name[…]]

/usr/cluster/bin/clpstring set [-f stringvalue-file] [-Z
     {zoneclustername | global}] pstring-name

/usr/cluster/bin/clpstring show [-b object-instance[,…]] [-t
     type[,…]] [-Z {zoneclustername[,…] | global | all}] [+ |
     pstring-name[…]]
```

`clpstring` コマンドは、Oracle Solaris Cluster プライベート文字列を管理します。プライベート文字列は一意の名前で識別され、669 ページの`scha_cluster_get`(1HA) コマンドを使用してのみ取得可能なエンコードされた値が含まれます。

プライベート文字列は、リソースなどのクラスタオブジェクトで使用され、非公開の値をセキュアに保管および取得します。一般的な用途には、エージェントで使用される内部パスワードがあります。

`clps` コマンドは `clpstring` コマンドの短い形式です。`clpstring` コマンドと `clps` コマンドは同じものです。どちらの形式のコマンドも使用できます。

このコマンドの一般的な形式は次のとおりです。

```
clpstring [subcommand] [options] [operands]
```

options に `-?` または `-V` オプションを指定する場合だけは、*subcommand* を省略できます。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式は、このマニュアルページの「オプション」セクションのオプションの説明で紹介されています。

`clpstring` コマンドは、次の管理タスクに使用します:

- 存在またはまだ存在しない可能性のあるクラスタオブジェクトインスタンスで使用するためのプライベート文字列を作成する
- プライベート文字列の値を更新する
- プライベート文字列をクラスタ構成から削除する
- プライベート文字列の仕様を表示する

`clpstring` コマンドは、アクティブなクラスタノードのみで実行できます。コマンドの実行結果は、実行するノードに関係なく常に同じです。

`clpstring` コマンドのすべてのサブコマンドは、大域ゾーンとゾーンクラスタの両方で実行できます。大域ゾーンで実行する場合、`-Z` オプションを使用すると、操作を制限する特定のゾーンクラスタの名前を指定できます。

グローバルクラスタノードからすべてのゾーンクラスタ情報にアクセスできますが、個別のゾーンクラスタはほかのゾーンクラスタを認識しません。特定のゾーンクラスタに操作を制限できない場合、使用するサブコマンドは現在のクラスタでのみ機能します。

サブコマンド

サポートされるサブコマンドには次のものがあります。

`create`

Oracle Solaris Cluster オブジェクトインスタンスで使用するためのプライベート文字列を作成します。

このプライベート文字列を使用するクラスタオブジェクトインスタンスを指定するには、`-b` オプションを使用します。オブジェクトインスタンスがクラスタ構成に存在しなくても、そのインスタンスのプライベート文字列を作成できます。クラスタオブジェクトインスタンスのタイプを指定するには、`-t` オプションを使用します。デフォルトのオブジェクト型は `resource` です。

プライベート文字列の値を含むファイルを指定するには、`-f` オプションを使用します。`-f` が指定されていない場合は、コマンドによって、プライベート文字列の値を指定するように求められます。詳細は オプション セクションで確認できます。

スーパーユーザー以外のユーザーが `create` サブコマンドを使用するには、`solaris.cluster.modify` 役割に基づくアクセス制御 (RBAC) の承認が必要です。詳細は、Unresolved link to " rbac5" のマニュアルページを参照してください。

`delete` サブコマンドの説明も参照してください。

`delete`

Oracle Solaris Cluster 構成の指定したプライベート文字列を削除します。

force オプションの `-F` を指定しない場合は、プライベート文字列が作成されたクラスタオブジェクトインスタンスをすでに削除している必要があります。`-F` オプションを指定すると、関連付けられたオブジェクトインスタンスがクラスタ構成にまだ存在し、プライベート文字列を使用している場合でも、コマンドによってプライベート文字列が削除されます。詳細は、オプション の `-F` を参照してください。

スーパーユーザー以外のユーザーが `delete` サブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。

`create` サブコマンドの説明も参照してください。

`list`

クラスタに作成されたすべてのプライベート文字列を表示しますが、その値は表示しません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。

`set`

指定したプライベート文字列の値を設定します。`-f` オプションを使用すると、プライベート文字列の値のソースを指定できます。`-f` が指定されていない場合は、コマンドによって値が求められます。プライベート文字列の値については、オプション セクションの `-f` の説明を参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。

`show`

プライベート文字列の仕様を表示しますが、その値は表示しません。仕様には、プライベート文字列の名前、それに関連するオブジェクトインスタンス、およびインスタンスのオブジェクト型が含まれます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。

次のオプションがサポートされています。

`-?`
`--help`

ヘルプ情報を表示します。このオプションを使用する場合、ほかの処理は実行されません。

このオプションを指定するとき、サブコマンドは指定してもしなくてもかまいません。

サブコマンドなしでこのオプションを指定すると、このコマンドのサブコマンドのリストが表示されます。

サブコマンド付きでこのオプションを指定すると、サブコマンドの使用方法が表示されます。

-b *object-instance*
--object-instance=*object-instance*
--object-instance *object-instance*

プライベート文字列を使用または使用を目的とするオブジェクトインスタンスの名前を指定します。オブジェクト型が resource のオブジェクトインスタンスのみが現在サポートされています。

-F
--force

指定したプライベート文字列を強制的に削除します。このオプションは、delete サブコマンドの場合にだけ指定できます。

このオプションを delete サブコマンドとともに使用する場合、指定したプライベート文字列を使用するオブジェクトインスタンスがクラスタにまだ存在する場合でも、このプライベート文字列を削除します。プライベート文字列を削除する前に、クラスタからオブジェクトインスタンスを通常どおりに削除します。

-f *stringvalue-file*
--stringvalue-file=*stringvalue-file*
--stringvalue-file *stringvalue-file*

プライベート文字列の値を含むファイルを指定します。ファイル名は、コマンドを実行するノードからアクセスできるフルパスである必要があります。

セキュリティー上の理由により、プライベート文字列の値はコマンド行オプションには指定できません。値のセキュリティーを維持するには、値をテキストファイルに格納し、-f オプションを使用して、このファイルのフルパスを指定します。文字列値のファイルの所有者を root にし、ファイルの読み取り可能アクセス権を root のみに設定して、グループからも一般ユーザーからもアクセスできないようにします。セキュリティーをさらに強化するために、コマンドを実行して値をプライベート文字列に設定したら、ファイルを削除できます。

-f オプションを指定しない場合は、同じ値が入力されたことを確認するために、コマンドによって、プライベート文字列の値を 2 回入力するように求められます。エコーを無効にした制御端末から値を読み取ります。

-f - (-f の後に空白とダッシュ) を指定すると、プライベート文字列の値を標準入力から 1 回のみ直接読み取ることができます。プライベート文字列の値は、入力時に画面でエコーされるか、コマンドがスクリプト化されている場合はスクリプトに表示されるため、プライベート文字列の値をこのように設定する場合は注意してください。

プライベート文字列の値の入力には次の要件があります:

- 文字列の長さは 257 文字以下にする必要があります。
- この文字列に NULL 文字を含めることはできません。

-t *object-type*
--object-type=*object-type*
--object-type *object-type*

オブジェクトインスタンスのタイプを指定します。デフォルトタイプは resource で、現在はプライベート文字列を使用できる唯一のオブジェクト型であるため、-t オプションは必要ありません。

-V
--version

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンドやオペランドなどのオプションは無視されます。-V オプションは、コマンドのバージョンだけを表示します。その他の処理は行いません。

-v
--verbose

詳細メッセージを標準出力に表示します。

このオプションは、このコマンドの任意の形式とともに指定できます。

-Z {*zoneclustername* | global | all}
--zoneclustername={*zoneclustername* | global | all}

プライベート文字列を作成するクラスタまたはプライベート文字列が存在するクラスタを指定します。

このオプションは、すべてのサブコマンドでサポートされています。

このオプションを指定する場合は、次のいずれかの引数も指定する必要があります。

zoneclustername — このオプションを使用するコマンドが、*zoneclustername* という名前のゾーンクラスタ内でのみ、指定されたすべてのプライベート文字列で機能するように指定します。

global — このオプションを使用するコマンドが、グローバルクラスタでのみ、指定されたすべてのプライベート文字列で機能するように指定します。

all — 広域クラスタでこの引数を使用する場合、それを使用するコマンドが広域クラスタとすべてのゾーンクラスタを含め、すべてのクラスタで指定されたすべてのリソースで機能するように指定します。

次のオペランドだけがサポートされています。

pstring-name — プライベート文字列の名前を指定します。プライベート文字列を作成する場合は、指定する名前がクラスタ全体で一意である必要があります。サブコマンドで複数のプライベート文字列を受け入れる場合、すべてのプライベート文字列を指定するには、*pstring-name* の代わりに、プラス記号 (+) を使用できます。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

次の終了コードが返されます。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

18 CL_EINTERNAL

内部エラーが発生しました

内部エラーは、ソフトウェアの欠陥またはその他の欠陥を示しています。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

37 CL_EOP

操作が許可されていません

サポートされていない構成に対する操作を実行しようとしたか、サポートされていない操作を実行しました。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

41 CL_ETYPE

無効なタイプです

-t または -p オプションで指定したタイプは存在しません。

これらの終了値は、1047 ページのscha_calls(3HA) のマニュアルページで説明されているリターンコードと互換性があります。

例 106 グローバルクラスタまたはゾーンクラスタのリソースにプライベート文字列を作成する

次のコマンドは、リソースインスタンスのプライベート文字列をグローバルクラスタに作成します。

```
# clpstring create -b resource1 -t resource -v pstring1
Enter string value:
Enter string value again:
Private string "pstring1" is created for the global cluster.
```

次のコマンドは大域ゾーンで実行され、zc1 という名前のゾーンクラスタにプライベート文字列を作成します。プライベート文字列の値は、/pvalue.file ファイルで指定されます。

```
# clpstring create -Z zc1 -b resource2 -f /pvalue.file pstring2
```

例 107 グローバルクラスタまたはゾーンクラスタの構成からのプライベート文字列の削除

次のコマンドは、オブジェクトインスタンスがクラスタにまだ存在するかどうかに関係なく、クラスタ構成からすべてのプライベート文字列を削除します。

```
# clpstring delete -F +
```

次のコマンドは、zc1 という名前のゾーンクラスタから、指定したプライベート文字列を削除します。

```
# clpstring delete -Z zc1 pstring1
```

例 108 クラスタに作成されたプライベート文字列の仕様の表示

次のコマンドは、クラスタ内のプライベート文字列を表示します。

```
# clpstring show
=== Private Strings ===

Pstring Name:                                   pstring1
  Object Instance:                                 resource1
  Object Type:                                     resource

Pstring Name:                                   pstring2
  Object Instance:                                 object2
  Object Type:                                     resource
```

例 109 グローバルクラスタとゾーンクラスタのプライベート文字列の一覧表示

次のコマンドは、グローバルクラスタおよびすべてのゾーンクラスタ内のプライベート文字列の名前を表示します。

```
# clpstring list -Z all
global:pstring1
global:pstring2
zc1:pstring1
zc1:pstring2
zc2:pstring
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページの`Intro`(1CL), 551 ページの`cluster`(1CL), 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5", Unresolved link to " rbac5"

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のオプションを指定してこのコマンドを実行できます。

■ -? オプション

■ -V オプション

サブコマンドを指定してこのコマンドを実行する場合、スーパーユーザー以外のユーザーは RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
create	solaris.cluster.modify
delete	solaris.cluster.modify
list	solaris.cluster.read
set	solaris.cluster.modify
show	solaris.cluster.read

名前

clpstring, clps — Oracle Solaris Cluster プライベート文字列の管理

```
/usr/cluster/bin/clpstring -V

/usr/cluster/bin/clpstring subcommand -?

/usr/cluster/bin/clpstring subcommand [options] -v [pstring-name[...]

/usr/cluster/bin/clpstring create -b object-instance [-f
     stringvalue-file] [-t object-type] [-Z {zoneclustername | global}]
     pstring-name

/usr/cluster/bin/clpstring delete [-F] [-Z {zoneclustername[,…] |
     global | all}] + | pstring-name …

/usr/cluster/bin/clpstring list [-b object-instance[,…]] [-t
     type[,…]] [-Z {zoneclustername[,…] | global | all}] [+ |
     pstring-name[…]]

/usr/cluster/bin/clpstring set [-f stringvalue-file] [-Z
     {zoneclustername | global}] pstring-name

/usr/cluster/bin/clpstring show [-b object-instance[,…]] [-t
     type[,…]] [-Z {zoneclustername[,…] | global | all}] [+ |
     pstring-name[…]]
```

`clpstring` コマンドは、Oracle Solaris Cluster プライベート文字列を管理します。プライベート文字列は一意の名前で識別され、669 ページの`scha_cluster_get`(1HA) コマンドを使用してのみ取得可能なエンコードされた値が含まれます。

プライベート文字列は、リソースなどのクラスタオブジェクトで使用され、非公開の値をセキュアに保管および取得します。一般的な用途には、エージェントで使用される内部パスワードがあります。

`clps` コマンドは `clpstring` コマンドの短い形式です。`clpstring` コマンドと `clps` コマンドは同じものです。どちらの形式のコマンドも使用できます。

このコマンドの一般的な形式は次のとおりです。

```
clpstring [subcommand] [options] [operands]
```

options に `-?` または `-V` オプションを指定する場合だけは、*subcommand* を省略できます。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式は、このマニュアルページの「オプション」セクションのオプションの説明で紹介されています。

`clpstring` コマンドは、次の管理タスクに使用します:

- 存在またはまだ存在しない可能性のあるクラスタオブジェクトインスタンスで使用するためのプライベート文字列を作成する
- プライベート文字列の値を更新する
- プライベート文字列をクラスタ構成から削除する
- プライベート文字列の仕様を表示する

`clpstring` コマンドは、アクティブなクラスタノードのみで実行できます。コマンドの実行結果は、実行するノードに関係なく常に同じです。

`clpstring` コマンドのすべてのサブコマンドは、大域ゾーンとゾーンクラスタの両方で実行できます。大域ゾーンで実行する場合、`-Z` オプションを使用すると、操作を制限する特定のゾーンクラスタの名前を指定できます。

グローバルクラスタノードからすべてのゾーンクラスタ情報にアクセスできますが、個別のゾーンクラスタはほかのゾーンクラスタを認識しません。特定のゾーンクラスタに操作を制限できない場合、使用するサブコマンドは現在のクラスタでのみ機能します。

サブコマンド

サポートされるサブコマンドには次のものがあります。

`create`

Oracle Solaris Cluster オブジェクトインスタンスで使用するためのプライベート文字列を作成します。

このプライベート文字列を使用するクラスタオブジェクトインスタンスを指定するには、`-b` オプションを使用します。オブジェクトインスタンスがクラスタ構成に存在しなくても、そのインスタンスのプライベート文字列を作成できます。クラスタオブジェクトインスタンスのタイプを指定するには、`-t` オプションを使用します。デフォルトのオブジェクト型は `resource` です。

プライベート文字列の値を含むファイルを指定するには、`-f` オプションを使用します。`-f` が指定されていない場合は、コマンドによって、プライベート文字列の値を指定するように求められます。詳細は オプション セクションで確認できます。

スーパーユーザー以外のユーザーが `create` サブコマンドを使用するには、`solaris.cluster.modify` 役割に基づくアクセス制御 (RBAC) の承認が必要です。詳細は、Unresolved link to " rbac5" のマニュアルページを参照してください。

`delete` サブコマンドの説明も参照してください。

`delete`

Oracle Solaris Cluster 構成の指定したプライベート文字列を削除します。

force オプションの `-F` を指定しない場合は、プライベート文字列が作成されたクラスタオブジェクトインスタンスをすでに削除している必要があります。`-F` オプションを指定すると、関連付けられたオブジェクトインスタンスがクラスタ構成にまだ存在し、プライベート文字列を使用している場合でも、コマンドによってプライベート文字列が削除されます。詳細は、オプション の `-F` を参照してください。

スーパーユーザー以外のユーザーが `delete` サブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。

`create` サブコマンドの説明も参照してください。

`list`

クラスタに作成されたすべてのプライベート文字列を表示しますが、その値は表示しません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。

`set`

指定したプライベート文字列の値を設定します。`-f` オプションを使用すると、プライベート文字列の値のソースを指定できます。`-f` が指定されていない場合は、コマンドによって値が求められます。プライベート文字列の値については、オプション セクションの `-f` の説明を参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。

`show`

プライベート文字列の仕様を表示しますが、その値は表示しません。仕様には、プライベート文字列の名前、それに関連するオブジェクトインスタンス、およびインスタンスのオブジェクト型が含まれます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。

次のオプションがサポートされています。

`-?`
`--help`

ヘルプ情報を表示します。このオプションを使用する場合、ほかの処理は実行されません。

このオプションを指定するとき、サブコマンドは指定してもしなくてもかまいません。

サブコマンドなしでこのオプションを指定すると、このコマンドのサブコマンドのリストが表示されます。

サブコマンド付きでこのオプションを指定すると、サブコマンドの使用方法が表示されます。

-b *object-instance*
--object-instance=*object-instance*
--object-instance *object-instance*

プライベート文字列を使用または使用を目的とするオブジェクトインスタンスの名前を指定します。オブジェクト型が resource のオブジェクトインスタンスのみが現在サポートされています。

-F
--force

指定したプライベート文字列を強制的に削除します。このオプションは、delete サブコマンドの場合にだけ指定できます。

このオプションを delete サブコマンドとともに使用する場合、指定したプライベート文字列を使用するオブジェクトインスタンスがクラスタにまだ存在する場合でも、このプライベート文字列を削除します。プライベート文字列を削除する前に、クラスタからオブジェクトインスタンスを通常どおりに削除します。

-f *stringvalue-file*
--stringvalue-file=*stringvalue-file*
--stringvalue-file *stringvalue-file*

プライベート文字列の値を含むファイルを指定します。ファイル名は、コマンドを実行するノードからアクセスできるフルパスである必要があります。

セキュリティー上の理由により、プライベート文字列の値はコマンド行オプションには指定できません。値のセキュリティーを維持するには、値をテキストファイルに格納し、-f オプションを使用して、このファイルのフルパスを指定します。文字列値のファイルの所有者を root にし、ファイルの読み取り可能アクセス権を root のみに設定して、グループからも一般ユーザーからもアクセスできないようにします。セキュリティーをさらに強化するために、コマンドを実行して値をプライベート文字列に設定したら、ファイルを削除できます。

-f オプションを指定しない場合は、同じ値が入力されたことを確認するために、コマンドによって、プライベート文字列の値を 2 回入力するように求められます。エコーを無効にした制御端末から値を読み取ります。

-f - (-f の後に空白とダッシュ) を指定すると、プライベート文字列の値を標準入力から 1 回のみ直接読み取ることができます。プライベート文字列の値は、入力時に画面でエコーされるか、コマンドがスクリプト化されている場合はスクリプトに表示されるため、プライベート文字列の値をこのように設定する場合は注意してください。

プライベート文字列の値の入力には次の要件があります:

- 文字列の長さは 257 文字以下にする必要があります。
- この文字列に NULL 文字を含めることはできません。

`-t` *object-type*
`--object-type=`*object-type*
`--object-type` *object-type*

オブジェクトインスタンスのタイプを指定します。デフォルトタイプは `resource` で、現在はプライベート文字列を使用できる唯一のオブジェクト型であるため、`-t` オプションは必要ありません。

`-V`
`--version`

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンドやオペランドなどのオプションは無視されます。`-V` オプションは、コマンドのバージョンだけを表示します。その他の処理は行いません。

`-v`
`--verbose`

詳細メッセージを標準出力に表示します。

このオプションは、このコマンドの任意の形式とともに指定できます。

`-Z` {*zoneclustername* | `global` | `all`}
`--zoneclustername=`{*zoneclustername* | `global` | `all`}

プライベート文字列を作成するクラスタまたはプライベート文字列が存在するクラスタを指定します。

このオプションは、すべてのサブコマンドでサポートされています。

このオプションを指定する場合は、次のいずれかの引数も指定する必要があります。

zoneclustername	このオプションを使用するコマンドが、*zoneclustername* という名前のゾーンクラスタ内でのみ、指定されたすべてのプライベート文字列で機能するように指定します。
`global`	このオプションを使用するコマンドが、グローバルクラスタでのみ、指定されたすべてのプライベート文字列で機能するように指定します。
`all`	広域クラスタでこの引数を使用する場合、それを使用するコマンドが広域クラスタとすべてのゾーンクラスタを含め、すべてのクラスタで指定されたすべてのリソースで機能するように指定します。

次のオペランドだけがサポートされています。

pstring-name	プライベート文字列の名前を指定します。プライベート文字列を作成する場合は、指定する名前がクラスタ全体で一意である必要があります。サブコマンドで複数のプライベート文字列を受け入れる場合、すべてのプライベート文字列を指定するには、*pstring-name* の代わりに、プラス記号 (+) を使用できます。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

次の終了コードが返されます。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

18 CL_EINTERNAL

内部エラーが発生しました

内部エラーは、ソフトウェアの欠陥またはその他の欠陥を示しています。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

37 CL_EOP

操作が許可されていません

サポートされていない構成に対する操作を実行しようとしたか、サポートされていない操作を実行しました。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

41 CL_ETYPE

無効なタイプです

-t または -p オプションで指定したタイプは存在しません。

これらの終了値は、1047 ページのscha_calls(3HA) のマニュアルページで説明されているリターンコードと互換性があります。

例 110 グローバルクラスタまたはゾーンクラスタのリソースにプライベート文字列を作成する

次のコマンドは、リソースインスタンスのプライベート文字列をグローバルクラスタに作成します。

```
# clpstring create -b resource1 -t resource -v pstring1
Enter string value:
Enter string value again:
Private string "pstring1" is created for the global cluster.
```

次のコマンドは大域ゾーンで実行され、zc1 という名前のゾーンクラスタにプライベート文字列を作成します。プライベート文字列の値は、/pvalue.file ファイルで指定されます。

```
# clpstring create -Z zc1 -b resource2 -f /pvalue.file pstring2
```

例 111 グローバルクラスタまたはゾーンクラスタの構成からのプライベート文字列の削除

次のコマンドは、オブジェクトインスタンスがクラスタにまだ存在するかどうかに関係なく、クラスタ構成からすべてのプライベート文字列を削除します。

```
# clpstring delete -F +
```

次のコマンドは、zc1 という名前のゾーンクラスタから、指定したプライベート文字列を削除します。

```
# clpstring delete -Z zc1 pstring1
```

例 112 クラスタに作成されたプライベート文字列の仕様の表示

次のコマンドは、クラスタ内のプライベート文字列を表示します。

```
# clpstring show
=== Private Strings ===

Pstring Name:                                   pstring1
  Object Instance:                                 resource1
  Object Type:                                     resource

Pstring Name:                                   pstring2
  Object Instance:                                 object2
  Object Type:                                     resource
```

例 113 グローバルクラスタとゾーンクラスタのプライベート文字列の一覧表示

次のコマンドは、グローバルクラスタおよびすべてのゾーンクラスタ内のプライベート文字列の名前を表示します。

```
# clpstring list -Z all
global:pstring1
global:pstring2
zc1:pstring1
zc1:pstring2
zc2:pstring
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページの`Intro`(1CL), 551 ページの`cluster`(1CL), 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5", Unresolved link to " rbac5"

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のオプションを指定してこのコマンドを実行できます。

■ -? オプション

■ -V オプション

サブコマンドを指定してこのコマンドを実行する場合、スーパーユーザー以外のユーザーは RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
create	solaris.cluster.modify
delete	solaris.cluster.modify
list	solaris.cluster.read
set	solaris.cluster.modify
show	solaris.cluster.read

名前

clquorum, clq — Oracle Solaris Cluster 定足数デバイスおよびプロパティーの管理

```
/usr/cluster/bin/clquorum -V

/usr/cluster/bin/clquorum subcommand -?

/usr/cluster/bin/clquorum subcommand [options] -v devicename[…]

/usr/cluster/bin/clquorum add [-a] [-t type] [-p name=value [,…]] devicename[…]

/usr/cluster/bin/clquorum add -i {- | clconfigfile} [-t type] [-p name=value[,…]] {+
 | devicename[…]}

/usr/cluster/bin/clquorum disable [-t type[,…]] {+ | devicename...}

/usr/cluster/bin/clquorum enable [-t type[,…]] {+ | devicename[…]}

/usr/cluster/bin/clquorum export [-o {- | clconfigfile}] [-t type[,…]] {+ | devicename[…]}

/usr/cluster/bin/clquorum list [-t type[,…]] [-n node[,…]] [+ | devicename[…]]

/usr/cluster/bin/clquorum remove -F [-t type[,…]]
     {+ | devicename[…]}

/usr/cluster/bin/clquorum reset

/usr/cluster/bin/clquorum show [-t type[,…]] [-n node[,…]]
     [+ | devicename[…]]

/usr/cluster/bin/clquorum status [-t type[,…]] [-n node[,…]]
     [+ | devicename[…]]
```

`clquorum` コマンドは、クラスタ定足数デバイスとクラスタ定足数プロパティーを管理します。`clq` コマンドは、`clquorum` コマンドの短縮形式です。`clquorum` コマンドと `clq` コマンドは同じものです。どちらの形式のコマンドも使用できます。

このコマンドの一般的な形式は次のとおりです。

`clquorum` [*subcommand*] [*options*] [*operands*]

subcommand を省略できるのは、*options* で `-?` オプションまたは `-V` オプションが指定されている場合のみです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式は、このマニュアルページの「オプション」セクションのオプションの説明で紹介されています。

定足数デバイスは、スプリットブレーンおよび記憶喪失状態からクラスタを保護するために必要です。(スプリットブレインおよび記憶喪失状態については、『*Oracle Solaris Cluster の概念*』の定足数および定足数デバイスに関するセクションを参照してください)。各定足数デバイスは、SCSI ケーブルまたは IP ネットワークによって、2 つ以上のノードに接続する必要があります。

定足数デバイスは、共有 SCSI ストレージデバイス、共有 NAS ストレージデバイス (Oracle ZFS Storage Appliance)、定足数サーバーのいずれかです。定足数デバイスがユーザーデータを格納する場合に、そのデバイスを定足数デバイスとして追加または削除しても、格納されているデータは影響を受けません。ただし、レプリケーションストレージデバイスを使用する場合、定足数デバイスはレプリケートされないボリューム上に置きます。

ノードと定足数デバイスが保守状態にある場合を除き、ノードと定足数デバイスは両方、クラスタ定足数構成に参加します。ノードまたは定足数デバイスが保守状態の場合、その投票数は常に 0 で、このノードまたは定足数デバイスは、定足数構成に参加しません。

`clquorum` コマンドを使用すると、次に示すタスクを実行できます。

- 定足数デバイスを Oracle Solaris Cluster 構成に追加する
- 定足数デバイスを Oracle Solaris Cluster 構成から削除する
- 定足数プロパティーを管理する

サブコマンド

サポートされるサブコマンドには次のものがあります。

`add`

指定した共有デバイスを定足数デバイスとして追加します。

このサブコマンドは、大域ゾーンだけで使用できます。

個々の定足数デバイスは、クラスタ内の最低 2 つのノードに接続します。定足数デバイスは、クラスタ構成内の接続パスを使用して、デバイスが接続されているすべてのノードに追加されます。その後、定足数デバイス-クラスタノード間の接続が変更された場合は、ユーザーがパスを更新します。パスは、定足数デバイスを削除し、構成に追加し直すことによって更新できます。この状態は、定足数デバイスに接続されているノードをさらに追加する場合や、1 つ以上のノードから定足数デバイスを切断する場合に発生することがあります。定足数の管理の詳細は、Unresolved link to " Oracle Solaris Cluster システム管理 の第 6 章定足数の管理"を参照してください。

定足数デバイスには、タイプがいくつかあります。詳細は、オプション セクションの `-t` オプションを参照してください。デフォルトのタイプは `shared_disk` です。

スーパーユーザー以外のユーザーは、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" を参照してください。

remove サブコマンドの説明も参照してください。

disable

定足数デバイスまたはノードを定足数保守状態に置きます。

このサブコマンドは、大域ゾーンだけで使用できます。

保守状態では、共有デバイスまたはノードの投票数は 0 です。そのような共有デバイスまたはノードは、定足数構成にもはや参加しません。さらに、保守状態のノードの場合、そのノードに接続されているすべての定足数デバイスの投票数は、1 減分されます。

この機能は、保守のために長期間ノードまたはデバイスを停止する必要がある場合に便利です。installmode が設定されていない場合、ノードをクラスタにブートし直すと、ノードは、保守モードを解除します。

ノードを保守状態にする前に、ノードを停止する必要があります。

スーパーユーザー以外のユーザーは、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

enable サブコマンドの説明も参照してください。

enable

定足数デバイスまたはノードを定足数保守状態から解除します。

このサブコマンドは、大域ゾーンだけで使用できます。

enable サブコマンドは、定足数デバイスまたはノードの保守モードを解除します。このサブコマンドは、定足数デバイスまたはノードの構成済み定足数投票数をデフォルトにリセットします。これにより、共有デバイスまたはノードは、定足数構成に参加できるようになります。

定足数デバイスをリセットすると、定足数デバイスの投票数は *N*-1 に変更されます。この計算では、*N* は、デバイスに接続されているゼロ以外の投票数を持つノードの数です。ノードのリセット後、投票数はそのデフォルト値にリセットされます。次に、ノードに接続されている定足数デバイスの投票数が、1 増分されます。

インストールモード設定 installmode が有効でない場合は、各ノードの定足数構成が、ブート時に自動的に有効になります。

スーパーユーザー以外のユーザーは、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

disable サブコマンドの説明も参照してください。

export

クラスタ定足数の構成情報をエクスポートします。

このサブコマンドは、大域ゾーンだけで使用できます。

-o オプションを使用してファイルを指定する場合、構成情報は、そのファイルに書き込まれます。ファイルを指定しない場合、この情報は標準出力 (stdout) に書き込まれます。

export サブコマンドは、クラスタ構成データをまったく変更しません。

スーパーユーザー以外のユーザーは、solaris.cluster.read RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

list

クラスタ内で構成されている定足数デバイスの名前を表示します。

このサブコマンドのすべての形式を大域ゾーンで使用できます。ゾーンクラスタでは、このサブコマンドを -? または -V オプションと一緒でのみ、またはオプションなしで使用できます。

オプションを指定しない場合、list サブコマンドは、クラスタ内で構成されているすべての定足数デバイスを表示します。-t オプションを指定すると、このサブコマンドは、指定されているタイプの定足数デバイスだけを表示します。-n オプションを指定すると、このサブコマンドは、指定されているいずれかのノードに接続されているすべての定足数デバイスの名前を表示します。

スーパーユーザー以外のユーザーは、solaris.cluster.read RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

remove

指定された定足数デバイス (1 つまたは複数) を、Oracle Solaris Cluster 定足数構成から削除します。

このサブコマンドは、大域ゾーンだけで使用できます。

このサブコマンドで強制オプション -F を使用すると、2 ノードクラスタの最後の定足数デバイスを削除できます。-F オプションを指定しないと、remove サブコマンドは、2 ノードクラスタの最後の定足数デバイスを削除しません。

remove サブコマンドは、物理デバイスを切断したり、削除したりするわけではありません。また、何らかのデータが存在する場合も、デバイス上のユーザーデータに影響を与えることはありません。installmode が有効でないかぎり、2 ノードクラスタの最後の定足数デバイスは削除できません。

削除できるのは、定足数デバイスだけです。このサブコマンドを使用して、クラスタノードを削除できません。

スーパーユーザー以外のユーザーは、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

add サブコマンドの説明も参照してください。

reset

定足数構成全体をリセットし、デフォルトの投票数にします。

このサブコマンドは、大域ゾーンだけで使用できます。

installmode が有効な場合、このノードはリセットによってクリアーされます。1 つ以上の定足数デバイスが正常に構成されている場合を除き、2 ノードクラスタでは installmode をリセットできません。

スーパーユーザー以外のユーザーは、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

installmode プロパティーの説明については、551 ページのcluster(1CL) の -p オプションも参照してください。

show

定足数デバイスのプロパティーを表示します。

このサブコマンドは、大域ゾーンだけで使用できます。

オプションを指定しない場合、show サブコマンドは、クラスタ内のすべての定足数デバイスのプロパティーを表示します。

-t オプションを使用してタイプを指定すると、このサブコマンドは、そのタイプのデバイスのプロパティーだけを表示します。オプション の -t を参照してください。

-n オプションを使用してノードを指定すると、このサブコマンドは、指定されるずれかのノードに接続されている定足数デバイスのプロパティーを表示します。

スーパーユーザー以外のユーザーは、solaris.cluster.read RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

status

定足数デバイスの現在のステータスおよび投票数を確認および表示します。

このサブコマンドのすべての形式を大域ゾーンで使用できます。ゾーンクラスタでは、このサブコマンドを -? または -V オプションと一緒でのみ、またはオプションなしで使用できます。

このサブコマンドを大域ゾーンで使用すると、指定したノードに接続されている定足数デバイスのステータスをすぐに確認できます。ノードに接続されていない定足数デバイスでは、このサブコマンドは前回のクラスタ再構成中に True であったステータスを表示します。

オプションを指定しない場合、status サブコマンドは、クラスタ内のすべての定足数デバイスに関する情報を表示します。

-t オプションを使用してタイプを指定すると、このサブコマンドは、そのタイプのデバイスの情報だけを表示します。オプション セクションの -t を参照してください。

-n オプションを使用してノードを指定すると、このサブコマンドは、指定されるずれかのノードに接続されている定足数デバイスのプロパティーを表示します。

スーパーユーザー以外のユーザーは、solaris.cluster.read RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

次のオプションがサポートされています。

-?
--help

ヘルプ情報を表示します。このオプションを使用する場合、ほかの処理は実行されません。

このオプションを指定するとき、サブコマンドは指定してもしなくてもかまいません。

このオプションをサブコマンドなしで指定すると、このコマンドのサブコマンドのリストが表示されます。

サブコマンド付きでこのオプションを指定すると、サブコマンドの使用方法が表示されます。

`-a`
`--autoconfig`

共有ディスクを使用する 2 ノードクラスタの場合、定足数デバイスが構成されていないときは、1 つの定足数デバイスを自動的に選択して構成します。

クラスタにあるすべての共有ディスクは、定足数デバイスとしての資格を備えていなければなりません。`autoconfig` サブコマンドは、使用できるデバイスが定足数デバイスになる資格があるかどうかをチェックしません。`autoconfig` サブコマンドは、共有ディスクだけをチェックします。

スーパーユーザー以外のユーザーは、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`-F`

指定した定足数デバイスを強制的に削除します。

強制オプションを一緒に指定できるのは、`remove` サブコマンドだけです。強制オプションにより、2 ノードクラスタの最後の定足数デバイスを削除したり、失敗した定足数デバイスを削除したりすることができます。`remove` サブコマンドでこのオプションを使用すると、定足数サブシステムは削除中に定足数デバイスにタッチしません。

`-i` *clconfigfile*
`--input=`*clconfigfile*
`--input` *clconfigfile*

定足数デバイスの管理に使用する構成情報を指定します。この情報は、1447 ページの`clconfiguration`(5CL) のマニュアルページに定義されている形式に準拠している必要があります。

ほかのコマンド行オプションでサブコマンドとともに `-i` を使用すると、コマンド行オプションの引数が、構成ファイル内の設定を上書きします。

`-n` *node-name*
`--node=`*node-name*
`--node` *node-name*

定足数デバイスの接続先ノード名を指定します。このオプションは、指定されているノードに接続されている定足数デバイスに対して表示される情報を制限するために、`list`、`status`、および `show` サブコマンドで使用されます。

ノード名または *node-name* に対応するノード ID を指定できます。

`-o` {- | *clconfigfile*}
`--output=`{- | *clconfigfile*}
`--output` {- | *clconfigfile*}

定足数デバイスの構成情報をファイルまたは標準出力 (`stdout`) に書き込みます。この構成情報の形式は、1447 ページの`clconfiguration`(5CL) のマニュアルページで説明さ

れている形式に準拠します。標準出力を指定するには、ファイル名の代わりに - を指定します。

-p *name*=*value*[,…]
--property=*name*=*value*[,…]
--property *name*=*value*[,…]

デバイスタイプ固有の定足数デバイスのプロパティーを指定します。このオプションは、add サブコマンドとともに使用します。これらプロパティーのリストと説明については、-t オプションの説明を参照してください。

-t *device-type*
--type=*device-type*
--type *device-type*

定足数デバイスタイプを指定します。このオプションが指定されている場合、オペランドは、指定されているタイプでなければなりません。

add、export、および remove サブコマンドの現在サポートされている定足数デバイスタイプは、次のとおりです。

- shared_disk によって指定されている共有ローカルディスク (SCSI-2、SCSI-3、ソフトウェア定足数 (フェンシングが無効化されている SCSI ディスク) のいずれか)
- quorum_server によって指定されている、Oracle Solaris Cluster Quorum Server マシン上で実行されている定足数サーバープロセス

デフォルトのタイプは shared_disk です。

add サブコマンドは、-t*node* を定足数タイプとして受け付けません。

enable、disable、list、show、および status サブコマンドの場合、タイプ node、shared_disk、または quorum_server を指定できます。これらの異なるタイプの定足数デバイスは、次に示すプロパティーを持っています。

node

定足数構成に参加するノードに対して、固有のプロパティーは設定されていません。

このタイプは、enable、disable、list、status、および show サブコマンドでのみ使用されます。タイプ node の定足数デバイスの追加には使用できません。

quorum_server

quorum_server タイプの定足数デバイスは、次に示すプロパティーを持っています。

qshost=*quorum-server-host*: 定足数サーバーが動作するマシンの名前を指定します。このホストは、ネットワーク上のマシンまたはホスト名の IP アドレスとすることができます。ホスト名を指定した場合は、そのマシンの IP アドレスが、/etc/hosts ファイルまたは /etc/inet/ipnodes ファイル、あるいはその両方に指定されている必要があります。

port=*port*: 定足数サーバーがクラスタノードと通信するために使用するポート番号を指定します。

定足数サーバーを追加する前に、定足数サーバーソフトウェアをホストマシンにインストールし、定足数サーバーを起動して実行しておく必要があります。詳細は、Unresolved link to " Oracle Solaris Cluster ソフトウェアのインストール "を参照してください。

`shared_disk`

SCSI-2、SCSI-3、またはソフトウェア定足数デバイスを構成するには、このタイプを使用します。`shared_disk` 定足数デバイスに対して、固有のプロパティーは設定されていません。`autoconfig` サブコマンドは、このタイプの定足数デバイスだけを受け付けます。

`-V`
`--version`

コマンドのバージョンを表示します。

このオプションは、ほかのサブコマンド、オプション、またはオペランドと一緒に使用しないでください。サブコマンド、オプション、またはオペランドは、無視されます。`-V` オプションは、コマンドのバージョンだけを表示します。その他の処理は行いません。

`-v`
`--verbose`

詳細な情報を標準出力 (`stdout`) に表示します。

次のオペランドがサポートされています。

devicename

`add`、`export`、および `remove` サブコマンドの場合のみ、オペランドは共有ディスク (SCSI、定足数サーバー、または NAS 定足数デバイス) の名前です。`add` サブコマンドの場合、`-i` を使用して `clconfigurationfile` を指定しないときは、少なくとも 1 つの定足数デバイスをオペランドとして指定する必要があります。

`disable`、`enable`、`list`、`status`、および `show` サブコマンドの場合のみ、オペランドをノードの名前または共有ディスク (SCSI、定足数サーバー、または NAS 定足数デバイス) の名前とすることができます。

どの場合でも、オペランドのタイプは `-t` オプションの値と一致している必要があります (そのオプションを指定する場合)。

devicename オペランドとして、次の値を使用します。

- ノードの場合、オペランドはノード名またはノード ID です。
- SCSI 定足数デバイスの場合、オペランドはデバイス識別子または DID フルパス名である必要があります (たとえば、`d1` や `/dev/did/rdsk/d1`)。
- 定足数サーバーの定足数デバイスの場合、オペランドは、単数または複数の定足数サーバーの識別子を指定する必要があります。これは定足数サーバーのインスタンス名でもよく、すべての定足数デバイスで一意の必要があります。

+

disable、enable、list、status、および show サブコマンドの場合のみ、クラスタに対して構成されているすべての定足数デバイスを指定します。-t オプションを使用する場合、プラス記号 (+) オペランドでは該当するタイプを持つすべてのデバイスを指定します。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

次の終了値が返される可能性があります。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

18 CL_EINTERNAL

内部エラーが発生しました

内部エラーは、ソフトウェアの欠陥またはその他の欠陥を示しています。

35 CL_EIO

I/O エラー

物理的な入出力エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

41 CL_ETYPE

無効なタイプです

-t または -p オプションで指定したタイプは存在しません。

例 114 SCSI 定足数デバイスの追加

clquorum コマンドは、すべてのクラスタノードに接続されている SCSI 定足数デバイスを構成します。

```
# clquorum add /dev/did/rdsk/d4s2
```

add サブコマンドの使用時は、shared_disk タイプがデフォルトです。shared_disk 定足数デバイスを追加するには、-t shared_disk を指定する必要はありません。

例 115 定足数サーバーの追加

次の clquorum コマンドは、定足数サーバー qs1 を構成します。

```
# clquorum add -t quorum_server -p qshost=10.11.114.81 -p port=9000 qs1
```

例 116 定足数デバイスの削除

次の clquorum コマンドは、定足数デバイス d4 を削除します。

```
# clquorum remove d4
```

定足数デバイスを削除するために使用するコマンドは、デバイスタイプが shared_disk と quorum_server のいずれでも同じです。

例 117 定足数デバイスの保守状態への移行

次の clquorum コマンドは、定足数デバイス qs1 を保守状態に移行し、デバイスが保守状態であることを確認します。

```
# clquorum disable qs1
# clquorum status qs1

=== Cluster Quorum ===

--- Quorum Votes by Device ---

Device Name       Present      Possible      Status
-----------       -------      --------      ------
qs1                1            1             Offline
```

例 118 定足数デバイスの定足数投票数のリセット

次の clquorum コマンドは、定足数デバイス d4 の構成済みの定足数投票数をデフォルトにリセットします。

```
# clquorum enable d4
```

例 119 クラスタ内で構成されている定足数デバイスの表示

次の clquorum コマンドは、簡潔な形式と詳細形式で、定足数デバイスを表示します。

```
# clquorum list
d4
pcow1
pcow2

# clquorum list -v
Quorums               Type
-------               ----
d4                    shared_disk
pcow1                 node
pcow2                 node
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページのIntro(1CL), 551 ページのcluster(1CL), 1447 ページのclconfiguration(5CL)

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のオプションを指定してこのコマンドを実行できます。

- -? オプション
- -V オプション

スーパーユーザー以外のユーザーがほかのサブコマンドを指定してこのコマンドを実行するには、RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
add	solaris.cluster.modify
disable	solaris.cluster.modify
enable	solaris.cluster.modify
export	solaris.cluster.read
list	solaris.cluster.read
remove	solaris.cluster.modify
reset	solaris.cluster.modify
show	solaris.cluster.read
status	solaris.cluster.read

名前

clquorum, clq — Oracle Solaris Cluster 定足数デバイスおよびプロパティーの管理

```
/usr/cluster/bin/clquorum -V

/usr/cluster/bin/clquorum subcommand -?

/usr/cluster/bin/clquorum subcommand [options] -v devicename[…]

/usr/cluster/bin/clquorum add [-a] [-t type] [-p name=value [,…]] devicename[…]

/usr/cluster/bin/clquorum add -i {- | clconfigfile} [-t type] [-p name=value[,…]] {+
 | devicename[…]}

/usr/cluster/bin/clquorum disable [-t type[,…]] {+ | devicename...}

/usr/cluster/bin/clquorum enable [-t type[,…]] {+ | devicename[…]}

/usr/cluster/bin/clquorum export [-o {- | clconfigfile}] [-t type[,…]] {+ | devicename[…]}

/usr/cluster/bin/clquorum list [-t type[,…]] [-n node[,…]] [+ | devicename[…]]

/usr/cluster/bin/clquorum remove -F [-t type[,…]]
     {+ | devicename[…]}

/usr/cluster/bin/clquorum reset

/usr/cluster/bin/clquorum show [-t type[,…]] [-n node[,…]]
     [+ | devicename[…]]

/usr/cluster/bin/clquorum status [-t type[,…]] [-n node[,…]]
     [+ | devicename[…]]
```

`clquorum` コマンドは、クラスタ定足数デバイスとクラスタ定足数プロパティーを管理します。`clq` コマンドは、`clquorum` コマンドの短縮形式です。`clquorum` コマンドと `clq` コマンドは同じものです。どちらの形式のコマンドも使用できます。

このコマンドの一般的な形式は次のとおりです。

`clquorum` [*subcommand*] [*options*] [*operands*]

subcommand を省略できるのは、*options* で `-?` オプションまたは `-V` オプションが指定されている場合のみです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式は、このマニュアルページの「オプション」セクションのオプションの説明で紹介されています。

定足数デバイスは、スプリットブレーンおよび記憶喪失状態からクラスタを保護するために必要です。(スプリットブレインおよび記憶喪失状態については、『*Oracle Solaris Cluster の概念*』の定足数および定足数デバイスに関するセクションを参照してください)。各定足数デバイスは、SCSI ケーブルまたは IP ネットワークによって、2 つ以上のノードに接続する必要があります。

定足数デバイスは、共有 SCSI ストレージデバイス、共有 NAS ストレージデバイス (Oracle ZFS Storage Appliance)、定足数サーバーのいずれかです。定足数デバイスがユーザーデータを格納する場合に、そのデバイスを定足数デバイスとして追加または削除しても、格納されているデータは影響を受けません。ただし、レプリケーションストレージデバイスを使用する場合、定足数デバイスはレプリケートされないボリューム上に置きます。

ノードと定足数デバイスが保守状態にある場合を除き、ノードと定足数デバイスは両方、クラスタ定足数構成に参加します。ノードまたは定足数デバイスが保守状態の場合、その投票数は常に 0 で、このノードまたは定足数デバイスは、定足数構成に参加しません。

`clquorum` コマンドを使用すると、次に示すタスクを実行できます。

- 定足数デバイスを Oracle Solaris Cluster 構成に追加する
- 定足数デバイスを Oracle Solaris Cluster 構成から削除する
- 定足数プロパティーを管理する

サブコマンド

サポートされるサブコマンドには次のものがあります。

`add`

指定した共有デバイスを定足数デバイスとして追加します。

このサブコマンドは、大域ゾーンだけで使用できます。

個々の定足数デバイスは、クラスタ内の最低 2 つのノードに接続します。定足数デバイスは、クラスタ構成内の接続パスを使用して、デバイスが接続されているすべてのノードに追加されます。その後、定足数デバイス-クラスタノード間の接続が変更された場合は、ユーザーがパスを更新します。パスは、定足数デバイスを削除し、構成に追加し直すことによって更新できます。この状態は、定足数デバイスに接続されているノードをさらに追加する場合や、1 つ以上のノードから定足数デバイスを切断する場合に発生することがあります。定足数の管理の詳細は、Unresolved link to " Oracle Solaris Cluster システム管理 の第 6 章定足数の管理"を参照してください。

定足数デバイスには、タイプがいくつかあります。詳細は、オプション セクションの `-t` オプションを参照してください。デフォルトのタイプは `shared_disk` です。

スーパーユーザー以外のユーザーは、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" を参照してください。

remove サブコマンドの説明も参照してください。

disable

定足数デバイスまたはノードを定足数保守状態に置きます。

このサブコマンドは、大域ゾーンだけで使用できます。

保守状態では、共有デバイスまたはノードの投票数は 0 です。そのような共有デバイスまたはノードは、定足数構成にもはや参加しません。さらに、保守状態のノードの場合、そのノードに接続されているすべての定足数デバイスの投票数は、1 減分されます。

この機能は、保守のために長期間ノードまたはデバイスを停止する必要がある場合に便利です。installmode が設定されていない場合、ノードをクラスタにブートし直すと、ノードは、保守モードを解除します。

ノードを保守状態にする前に、ノードを停止する必要があります。

スーパーユーザー以外のユーザーは、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

enable サブコマンドの説明も参照してください。

enable

定足数デバイスまたはノードを定足数保守状態から解除します。

このサブコマンドは、大域ゾーンだけで使用できます。

enable サブコマンドは、定足数デバイスまたはノードの保守モードを解除します。このサブコマンドは、定足数デバイスまたはノードの構成済み定足数投票数をデフォルトにリセットします。これにより、共有デバイスまたはノードは、定足数構成に参加できるようになります。

定足数デバイスをリセットすると、定足数デバイスの投票数は *N*-1 に変更されます。この計算では、*N* は、デバイスに接続されているゼロ以外の投票数を持つノードの数です。ノードのリセット後、投票数はそのデフォルト値にリセットされます。次に、ノードに接続されている定足数デバイスの投票数が、1 増分されます。

インストールモード設定 installmode が有効でない場合は、各ノードの定足数構成が、ブート時に自動的に有効になります。

スーパーユーザー以外のユーザーは、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

disable サブコマンドの説明も参照してください。

export

クラスタ定足数の構成情報をエクスポートします。

このサブコマンドは、大域ゾーンだけで使用できます。

-o オプションを使用してファイルを指定する場合、構成情報は、そのファイルに書き込まれます。ファイルを指定しない場合、この情報は標準出力 (stdout) に書き込まれます。

export サブコマンドは、クラスタ構成データをまったく変更しません。

スーパーユーザー以外のユーザーは、solaris.cluster.read RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

list

クラスタ内で構成されている定足数デバイスの名前を表示します。

このサブコマンドのすべての形式を大域ゾーンで使用できます。ゾーンクラスタでは、このサブコマンドを -? または -V オプションと一緒でのみ、またはオプションなしで使用できます。

オプションを指定しない場合、list サブコマンドは、クラスタ内で構成されているすべての定足数デバイスを表示します。-t オプションを指定すると、このサブコマンドは、指定されているタイプの定足数デバイスだけを表示します。-n オプションを指定すると、このサブコマンドは、指定されているいずれかのノードに接続されているすべての定足数デバイスの名前を表示します。

スーパーユーザー以外のユーザーは、solaris.cluster.read RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

remove

指定された定足数デバイス (1 つまたは複数) を、Oracle Solaris Cluster 定足数構成から削除します。

このサブコマンドは、大域ゾーンだけで使用できます。

このサブコマンドで強制オプション -F を使用すると、2 ノードクラスタの最後の定足数デバイスを削除できます。-F オプションを指定しないと、remove サブコマンドは、2 ノードクラスタの最後の定足数デバイスを削除しません。

remove サブコマンドは、物理デバイスを切断したり、削除したりするわけではありません。また、何らかのデータが存在する場合も、デバイス上のユーザーデータに影響を与えることはありません。installmode が有効でないかぎり、2 ノードクラスタの最後の定足数デバイスは削除できません。

削除できるのは、定足数デバイスだけです。このサブコマンドを使用して、クラスタノードを削除できません。

スーパーユーザー以外のユーザーは、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

add サブコマンドの説明も参照してください。

reset

定足数構成全体をリセットし、デフォルトの投票数にします。

このサブコマンドは、大域ゾーンだけで使用できます。

installmode が有効な場合、このノードはリセットによってクリアーされます。1 つ以上の定足数デバイスが正常に構成されている場合を除き、2 ノードクラスタでは installmode をリセットできません。

スーパーユーザー以外のユーザーは、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

installmode プロパティーの説明については、551 ページのcluster(1CL) の -p オプションも参照してください。

show

定足数デバイスのプロパティーを表示します。

このサブコマンドは、大域ゾーンだけで使用できます。

オプションを指定しない場合、show サブコマンドは、クラスタ内のすべての定足数デバイスのプロパティーを表示します。

-t オプションを使用してタイプを指定すると、このサブコマンドは、そのタイプのデバイスのプロパティーだけを表示します。オプション の -t を参照してください。

-n オプションを使用してノードを指定すると、このサブコマンドは、指定されるずれかのノードに接続されている定足数デバイスのプロパティーを表示します。

スーパーユーザー以外のユーザーは、solaris.cluster.read RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

status

定足数デバイスの現在のステータスおよび投票数を確認および表示します。

このサブコマンドのすべての形式を大域ゾーンで使用できます。ゾーンクラスタでは、このサブコマンドを -? または -V オプションと一緒でのみ、またはオプションなしで使用できます。

このサブコマンドを大域ゾーンで使用すると、指定したノードに接続されている定足数デバイスのステータスをすぐに確認できます。ノードに接続されていない定足数デバイスでは、このサブコマンドは前回のクラスタ再構成中に True であったステータスを表示します。

オプションを指定しない場合、status サブコマンドは、クラスタ内のすべての定足数デバイスに関する情報を表示します。

-t オプションを使用してタイプを指定すると、このサブコマンドは、そのタイプのデバイスの情報だけを表示します。オプション セクションの -t を参照してください。

-n オプションを使用してノードを指定すると、このサブコマンドは、指定されるずれかのノードに接続されている定足数デバイスのプロパティーを表示します。

スーパーユーザー以外のユーザーは、solaris.cluster.read RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

次のオプションがサポートされています。

-?
--help

ヘルプ情報を表示します。このオプションを使用する場合、ほかの処理は実行されません。

このオプションを指定するとき、サブコマンドは指定してもしなくてもかまいません。

このオプションをサブコマンドなしで指定すると、このコマンドのサブコマンドのリストが表示されます。

サブコマンド付きでこのオプションを指定すると、サブコマンドの使用方法が表示されます。

`-a`
`--autoconfig`

共有ディスクを使用する 2 ノードクラスタの場合、定足数デバイスが構成されていないときは、1 つの定足数デバイスを自動的に選択して構成します。

クラスタにあるすべての共有ディスクは、定足数デバイスとしての資格を備えていなければなりません。`autoconfig` サブコマンドは、使用できるデバイスが定足数デバイスになる資格があるかどうかをチェックしません。`autoconfig` サブコマンドは、共有ディスクだけをチェックします。

スーパーユーザー以外のユーザーは、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`-F`

指定した定足数デバイスを強制的に削除します。

強制オプションを一緒に指定できるのは、`remove` サブコマンドだけです。強制オプションにより、2 ノードクラスタの最後の定足数デバイスを削除したり、失敗した定足数デバイスを削除したりすることができます。`remove` サブコマンドでこのオプションを使用すると、定足数サブシステムは削除中に定足数デバイスにタッチしません。

`-i` *clconfigfile*
`--input=`*clconfigfile*
`--input` *clconfigfile*

定足数デバイスの管理に使用する構成情報を指定します。この情報は、1447 ページの`clconfiguration`(5CL) のマニュアルページに定義されている形式に準拠している必要があります。

ほかのコマンド行オプションでサブコマンドとともに `-i` を使用すると、コマンド行オプションの引数が、構成ファイル内の設定を上書きします。

`-n` *node-name*
`--node=`*node-name*
`--node` *node-name*

定足数デバイスの接続先ノード名を指定します。このオプションは、指定されているノードに接続されている定足数デバイスに対して表示される情報を制限するために、`list`、`status`、および `show` サブコマンドで使用されます。

ノード名または *node-name* に対応するノード ID を指定できます。

`-o` {- | *clconfigfile*}
`--output=`{- | *clconfigfile*}
`--output` {- | *clconfigfile*}

定足数デバイスの構成情報をファイルまたは標準出力 (`stdout`) に書き込みます。この構成情報の形式は、1447 ページの`clconfiguration`(5CL) のマニュアルページで説明さ

れている形式に準拠します。標準出力を指定するには、ファイル名の代わりに - を指定します。

-p *name=value*[,…]
--property=*name=value*[,…]
--property *name=value*[,…]

デバイスタイプ固有の定足数デバイスのプロパティーを指定します。このオプションは、add サブコマンドとともに使用します。これらプロパティーのリストと説明については、-t オプションの説明を参照してください。

-t *device-type*
--type=*device-type*
--type *device-type*

定足数デバイスタイプを指定します。このオプションが指定されている場合、オペランドは、指定されているタイプでなければなりません。

add、export、および remove サブコマンドの現在サポートされている定足数デバイスタイプは、次のとおりです。

- shared_disk によって指定されている共有ローカルディスク (SCSI-2、SCSI-3、ソフトウェア定足数 (フェンシングが無効化されている SCSI ディスク) のいずれか)
- quorum_server によって指定されている、Oracle Solaris Cluster Quorum Server マシン上で実行されている定足数サーバープロセス

デフォルトのタイプは shared_disk です。

add サブコマンドは、-t*node* を定足数タイプとして受け付けません。

enable、disable、list、show、および status サブコマンドの場合、タイプ node、shared_disk、または quorum_server を指定できます。これらの異なるタイプの定足数デバイスは、次に示すプロパティーを持っています。

node

定足数構成に参加するノードに対して、固有のプロパティーは設定されていません。

このタイプは、enable、disable、list、status、および show サブコマンドでのみ使用されます。タイプ node の定足数デバイスの追加には使用できません。

quorum_server

quorum_server タイプの定足数デバイスは、次に示すプロパティーを持っています。

qshost=*quorum-server-host*: 定足数サーバーが動作するマシンの名前を指定します。このホストは、ネットワーク上のマシンまたはホスト名の IP アドレスとすることができます。ホスト名を指定した場合は、そのマシンの IP アドレスが、/etc/hosts ファイルまたは /etc/inet/ipnodes ファイル、あるいはその両方に指定されている必要があります。

port=*port*: 定足数サーバーがクラスタノードと通信するために使用するポート番号を指定します。

定足数サーバーを追加する前に、定足数サーバーソフトウェアをホストマシンにインストールし、定足数サーバーを起動して実行しておく必要があります。詳細は、Unresolved link to " Oracle Solaris Cluster ソフトウェアのインストール "を参照してください。

`shared_disk`

SCSI-2、SCSI-3、またはソフトウェア定足数デバイスを構成するには、このタイプを使用します。`shared_disk` 定足数デバイスに対して、固有のプロパティーは設定されていません。`autoconfig` サブコマンドは、このタイプの定足数デバイスだけを受け付けます。

`-V`
`--version`

コマンドのバージョンを表示します。

このオプションは、ほかのサブコマンド、オプション、またはオペランドと一緒に使用しないでください。サブコマンド、オプション、またはオペランドは、無視されます。`-V` オプションは、コマンドのバージョンだけを表示します。その他の処理は行いません。

`-v`
`--verbose`

詳細な情報を標準出力 (`stdout`) に表示します。

次のオペランドがサポートされています。

devicename

`add`、`export`、および `remove` サブコマンドの場合のみ、オペランドは共有ディスク (SCSI、定足数サーバー、または NAS 定足数デバイス) の名前です。`add` サブコマンドの場合、`-i` を使用して `clconfigurationfile` を指定しないときは、少なくとも 1 つの定足数デバイスをオペランドとして指定する必要があります。

`disable`、`enable`、`list`、`status`、および `show` サブコマンドの場合のみ、オペランドをノードの名前または共有ディスク (SCSI、定足数サーバー、または NAS 定足数デバイス) の名前とすることができます。

どの場合でも、オペランドのタイプは `-t` オプションの値と一致している必要があります (そのオプションを指定する場合)。

devicename オペランドとして、次の値を使用します。

- ノードの場合、オペランドはノード名またはノード ID です。
- SCSI 定足数デバイスの場合、オペランドはデバイス識別子または DID フルパス名である必要があります (たとえば、`d1` や `/dev/did/rdsk/d1`)。
- 定足数サーバーの定足数デバイスの場合、オペランドは、単数または複数の定足数サーバーの識別子を指定する必要があります。これは定足数サーバーのインスタンス名でもよく、すべての定足数デバイスで一意の必要があります。

+

disable、enable、list、status、および show サブコマンドの場合のみ、クラスタに対して構成されているすべての定足数デバイスを指定します。-t オプションを使用する場合、プラス記号 (+) オペランドでは該当するタイプを持つすべてのデバイスを指定します。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

次の終了値が返される可能性があります。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

18 CL_EINTERNAL

内部エラーが発生しました

内部エラーは、ソフトウェアの欠陥またはその他の欠陥を示しています。

35 CL_EIO

I/O エラー

物理的な入出力エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

41 CL_ETYPE

無効なタイプです

-t または -p オプションで指定したタイプは存在しません。

例 120 SCSI 定足数デバイスの追加

clquorum コマンドは、すべてのクラスタノードに接続されている SCSI 定足数デバイスを構成します。

```
# clquorum add /dev/did/rdsk/d4s2
```

add サブコマンドの使用時は、shared_disk タイプがデフォルトです。shared_disk 定足数デバイスを追加するには、-t shared_disk を指定する必要はありません。

例 121 定足数サーバーの追加

次の clquorum コマンドは、定足数サーバー qs1 を構成します。

```
# clquorum add -t quorum_server -p qshost=10.11.114.81 -p port=9000 qs1
```

例 122 定足数デバイスの削除

次の clquorum コマンドは、定足数デバイス d4 を削除します。

```
# clquorum remove d4
```

定足数デバイスを削除するために使用するコマンドは、デバイスタイプが shared_disk と quorum_server のいずれでも同じです。

例 123 定足数デバイスの保守状態への移行

次の clquorum コマンドは、定足数デバイス qs1 を保守状態に移行し、デバイスが保守状態であることを確認します。

```
# clquorum disable qs1
# clquorum status qs1

=== Cluster Quorum ===

--- Quorum Votes by Device ---

Device Name       Present      Possible      Status
-----------       -------      --------      ------
qs1                1            1             Offline
```

例 124 定足数デバイスの定足数投票数のリセット

次の clquorum コマンドは、定足数デバイス d4 の構成済みの定足数投票数をデフォルトにリセットします。

```
# clquorum enable d4
```

例 125 クラスタ内で構成されている定足数デバイスの表示

次の clquorum コマンドは、簡潔な形式と詳細形式で、定足数デバイスを表示します。

```
# clquorum list
d4
pcow1
pcow2

# clquorum list -v
Quorums               Type
-------               ----
d4                    shared_disk
pcow1                 node
pcow2                 node
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページのIntro(1CL), 551 ページのcluster(1CL), 1447 ページのclconfiguration(5CL)

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のオプションを指定してこのコマンドを実行できます。

- -? オプション
- -V オプション

スーパーユーザー以外のユーザーがほかのサブコマンドを指定してこのコマンドを実行するには、RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
add	solaris.cluster.modify
disable	solaris.cluster.modify
enable	solaris.cluster.modify
export	solaris.cluster.read
list	solaris.cluster.read
remove	solaris.cluster.modify
reset	solaris.cluster.modify
show	solaris.cluster.read
status	solaris.cluster.read

名前

clreslogicalhostname, clrslh — Oracle Solaris Cluster 論理ホスト名用リソースの管理

```
/usr/cluster/bin/clreslogicalhostname [subcommand] -?

/usr/cluster/bin/clreslogicalhostname -V

/usr/cluster/bin/clreslogicalhostname [subcommand [options]] -v
     [lhresource]...

/usr/cluster/bin/clreslogicalhostname create -g resourcegroup
     [-h lhost[,…]] [-N netif@node[,…]] [-p name=value]
     [-Z {zoneclustername | global}] [-d] lhresource

/usr/cluster/bin/clreslogicalhostname create -i
     {- | clconfiguration} [-a] [-g resourcegroup[,…]] [-p
     name=value] [-d] {+ | lhresource...}

/usr/cluster/bin/clreslogicalhostname delete [-g resourcegroup[,…]]
     [-Z {zoneclustername | global}] [-F] {+ | lhresource...}

/usr/cluster/bin/clreslogicalhostname disable [-g resourcegroup[,…]]
     [-R] [-n node[,…]] [-Z {zoneclustername | global}]
     {+ | lhresource...}

/usr/cluster/bin/clreslogicalhostname enable [-g resourcegroup[,…]]
     [-R] [-n node[,…]] [-Z {zoneclustername | global}]
     {+ | lhresource...}

/usr/cluster/bin/clreslogicalhostname export [-o {- | configfile}]
     [+ | lhresource...]

/usr/cluster/bin/clreslogicalhostname list [-s state[,…]]
     [-g resourcegroup[,…]] [-Z {zoneclustername
     [,...] | global | all}] [+ | lhresource...]

/usr/cluster/bin/clreslogicalhostname list-props [-l listtype]
     [-p name[,…]] [-Z {zoneclustername [,...] | global | all}]
     [+ | lhresource...]

/usr/cluster/bin/clreslogicalhostname monitor [-g resourcegroup[,…]]
     [-Z zoneclustername | all | global] {+ | lhresource...}

/usr/cluster/bin/clreslogicalhostname reset [-f errorflag] [-g
     resourcegroup[,…]] [-Z {zoneclustername | global}]
     {+ | lhresource...}

/usr/cluster/bin/clreslogicalhostname set [-i
     {- | clconfiguration}] [-g resourcegroup[,…]] [-p name
     [+|-]=value] [-Z {zoneclustername}] {+ | lhresource...}

/usr/cluster/bin/clreslogicalhostname show [-g resourcegroup[,…]]
     [-p name[,…]] [-Z {zoneclustername [,...] | global | all}]
```

```
     [+ | lhresource...]

/usr/cluster/bin/clreslogicalhostname status [-s state[,…]]
     [-n node[,…]] [-g resourcegroup[,…]] [-Z
     {zoneclustername [,...] | global | all}] [+ | lhresource...]

/usr/cluster/bin/clreslogicalhostname unmonitor [-g
     resourcegroup[,…]] [-Z {zoneclustername | global}]
     {+ | lhresource...}
```

clreslogicalhostname コマンドは、Oracle Solaris Cluster 論理ホスト名用のリソースを管理します。clrslh コマンドは、clreslogicalhostname コマンドの短い形式です。clreslogicalhostname コマンドと clrslh コマンドは同じものです。どちらの形式のコマンドも使用できます。

clreslogicalhostname コマンドには、論理ホスト名リソースを作成するための便利なオプションが組み込まれています。clreslogicalhostname コマンドでは、Solaris IP マルチパス (IPMP) グループの自動作成もサポートされています。

clreslogicalhostname コマンドの一部のサブコマンドは、リソース構成を変更します:

- disable
- enable
- monitor
- reset
- set
- unmonitor

clreslogicalhostname コマンドの一部のサブコマンドは、リソースに関する情報だけを取得します。これらのサブコマンドは、グローバルクラスタまたはゾーンクラスタから使用できます: 次のコマンドは、リソースに関する情報のみを取得します:

- export
- list
- list-props
- show
- status

このコマンドからの予想不能な結果を避けるには、コマンドのすべての書式をグローバルクラスタノードから実行してください。

このコマンドの一般的な形式は次のとおりです。

`clreslogicalhostname` [*subcommand*] [*options*] [*operands*]

subcommand を省略できるのは、*options* で `-?`、`-o`、`-V`、または `-v` オプションが指定されている場合のみです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式については、このマニュアルページの「オプション」セクションを参照してください。

ゾーンクラスタでの操作

`clreslogicalhostname` コマンドは、ゾーンクラスタで、`export` を除くすべてのサブコマンドを指定して使用することができます。

`export` 以外のすべてのサブコマンドで `-Z` オプションを使用して、操作を制限する特定のゾーンクラスタの名前を指定することもできます。また、論理ホスト名リソース (*zoneclustername* : *lhresource*) にゾーンクラスタ名を付けて、操作を特定のゾーンクラスタに制限することもできます。

グローバルクラスタノードからすべてのゾーンクラスタ情報にアクセスできますが、特定のゾーンクラスタは他のゾーンクラスタを認識しません。特定のゾーンクラスタに操作を制限できない場合、使用するサブコマンドは現在のクラスタでのみ機能します。

サポートされるサブコマンドには次のものがあります。

`create`

コマンドに対するオペランドとして指定された論理ホスト名リソースを作成します。

`create` を `-i` オプションと使用して構成ファイルを指定した場合、サブコマンドはプラス記号 (+) をオペランドとして受け付けます。`+` オペランドを使用すると、構成ファイル内に存在しないすべてのリソースが作成されます。

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

デフォルトでは、リソースはモニタリング対象となり、有効な状態で作成されます。ただし、リソースがオンライン状態になり、モニターされるのは、リソースのリソースグループがオンラインになったあとだけです。無効な状態でリソースを作成するには、`-d` オプションを指定します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから、ゾーンクラスタ内の論理ホスト名リソースを作成するには、`-Z` オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。

delete サブコマンドの説明も参照してください。

delete

コマンドに対するオペランドとして指定された論理ホスト名リソースを削除します。このサブコマンドに対しオペランドとしてプラス記号 (+) を指定すると、すべてのリソースが削除されます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、削除するリソースを限定することができます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを削除します。

- デフォルトでは、リソースは次の条件が満たされる場合に *のみ* 削除されます。
- リソースが無効なとき
- リソースに対するすべての依存性が削除されているとき
- 指定したすべてのリソースを確実に削除するには、-F オプションを指定します。-F オプションの効果は、次のとおりです。
- 指定したすべてのリソース (無効になっていないリソースも含む) が削除されます。
- 指定したすべてのリソースが、他のリソースのリソース依存性設定から削除されます。

リソースは、コマンド行でリソースを指定した順序とは無関係に、リソース間の依存性を満たすのに必要な順序に従って削除されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから、ゾーンクラスタ内の論理ホスト名リソースを削除するには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

create サブコマンドの説明も参照してください。

disable

コマンドに対するオペランドとして指定された論理ホスト名リソースを無効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースが無効になります。

-g オプションを指定すると、オペランドのリストをフィルタリングし、無効にするリソースを限定することができます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを無効にします。

必要なリソース依存性をすべて確実に満たすには、-R オプションを指定します。-R オプションは、コマンドに対するオペランドとして指定したリソースに依存しているリソース (コマンドに対するオペランドとして指定しなかったリソースも含まれる) をすべて無効にします。-g オプションと -t オプションは、リソースの依存関係を満たすためだけに無効化されるリソースには適用されません。

リソースは、コマンド行でリソースを指定した順序とは無関係に、リソース間の依存性を満たすのに必要な順序に従って無効化されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから、ゾーンクラスタ内で登録された論理ホスト名リソースを無効にするには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

enable サブコマンドの説明も参照してください。

enable

コマンドに対するオペランドとして指定された論理ホスト名リソースを有効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースが有効になります。

-g オプションを指定すると、オペランドのリストをフィルタリングし、有効にするリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを有効にします。

必要なリソース依存性をすべて確実に満たすには、-R オプションを指定します。-R オプションは、コマンドに対するオペランドとして指定したリソースに依存しているリソース (コマンドに対するオペランドとして指定しなかったリソースも含まれる) をすべて有効にします。-g オプションは、リソースの依存関係を満たすためだけに有効化されるリソースには適用されません。

リソースは、コマンド行でリソースを指定した順序とは無関係に、リソース間の依存性を満たすのに必要な順序に従って有効化されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから、ゾーンクラスタ内で登録された論理ホスト名リソースを有効にするには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

disable サブコマンドの説明も参照してください。

export

1447 ページのclconfiguration(5CL) のマニュアルページで規定されている形式で、論理ホスト名リソース構成をエクスポートします。

このサブコマンドは、大域ゾーンだけで使用できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

list

コマンドに対するオペランドとして指定された論理ホスト名リソースのリストを表示します。デフォルトでは、すべてのリソースが表示されます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、表示するリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを表示します。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、指定したリソースグループ内のすべてのリソースまたは指定したリソースタイプのインスタンスであるすべてのリソースを指定できます。オペランドが指定されていない場合、指定されているリソースグループ内のすべてのリソースまたは指定されているリソースタイプのインスタンスであるすべてのリソースが表示されます。

-v オプションを指定すると、該当するリソースグループおよびリスト内の各リソースのリソースタイプも表示されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから、ゾーンクラスタ内で登録された論理ホスト名リソースを表示するには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

`list-props`

コマンドに対するオペランドとして指定された論理ホスト名リソースのプロパティーのリストを表示します。デフォルトでは、すべてのリソースの拡張プロパティーが表示されます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、プロパティーが表示されるリソースを限定できます。

-g *resourcegrouplist*
: *resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内の論理ホスト名リソースのみのプロパティーを表示します。

-l オプションでは、表示するリソースプロパティーのタイプを指定します。

`-l all`
: 標準プロパティーと拡張プロパティーを表示するように指定します。

`-l extension`
: 拡張プロパティーだけを表示するように指定します。デフォルトでは、拡張プロパティーだけが表示されます。

`-l standard`
: 標準プロパティーだけを表示するように指定します。

-l オプションを指定しない場合、-p オプションまたは -y オプションを使用して標準プロパティーを明示的に指定しないかぎり、拡張プロパティーだけが表示されます。

-p オプションは、表示するリソースプロパティーのセットを制限します。-p オプションを指定すると、*namelist* で指定したプロパティーだけが表示されます。*namelist* では、標準プロパティーと拡張プロパティーを指定できます。

-v オプションを指定すると、各プロパティーの説明も表示されます。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、指定したリソースグループ内のすべてのリソースまたは指定したリソースタイプのインスタンスであるすべてのリソー

スを指定できます。オペランドを指定しないと、指定したリソースグループ内のすべてのリソースのプロパティーまたは指定したリソースタイプのインスタンスであるすべてのリソースのプロパティーが表示されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

ゾーンクラスタの論理ホスト名リソースのプロパティーのリストを表示するには、-z オプションを使用してゾーンクラスタ名を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

`monitor`

コマンドに対するオペランドとして指定された論理ホスト名リソースのモニタリングを有効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースに対してモニタリングが有効になります。

-g オプションを指定すると、オペランドのリストをフィルタリングし、モニター対象のリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけをモニターします。

- リソースは、モニタリングが有効になっている場合、次の条件が満たされているときだけモニタリングされます。
- リソースが有効な状態にある。
- 該当リソースが含まれるリソースグループが、1 つ以上のクラスタノード上でオンライン状態にある。

注記 - リソースに対するモニタリングを有効にしても、そのリソースは有効*になりません*。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタのリソースをモニターするには、-z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.admin` RBAC の承認が必要です。

`unmonitor` サブコマンドの説明も参照してください。

`reset`

コマンドに対するオペランドとして指定された論理ホスト名リソースに関連付けられているエラーフラグをクリアします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースのエラーフラグがクリアされます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、リセットするリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけをリセットします。

デフォルトでは、reset サブコマンドによって STOP_FAILED エラーフラグがクリアーされます。クリアするエラーフラグを明示的に指定するには、-f オプションを使用します。-f オプションが受け付けるエラーフラグは、STOP_FAILED だけです。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから、ゾーンクラスタ内の論理ホスト名リソースをリセットするには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

set

コマンドに対するオペランドとして指定された論理ホスト名リソースの指定プロパティーを変更します。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースの指定したプロパティーが変更されます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、変更するリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを変更します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから、ゾーンクラスタ内の論理ホスト名リソースのプロパティーを設定するには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

show

コマンドに対するオペランドとして指定された論理ホスト名リソースの構成を表示します。デフォルトでは、すべてのリソースの構成が表示されます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、構成が表示されるリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースのみの構成を表示します。

-p オプションは、表示するリソースプロパティーのセットを制限します。-p オプションを指定すると、*namelist* で指定したプロパティーだけが表示されます。*namelist* では、標準プロパティーと拡張プロパティーを指定できます。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、指定したリソースグループ内のすべてのリソースまたは指定したリソースタイプのインスタンスであるすべてのリソースを指定できます。オペランドを指定しないと、指定したリソースグループ内のすべてのリソースの構成または指定したリソースタイプのインスタンスであるすべてのリソースの構成が表示されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから、ゾーンクラスタ内の論理ホスト名リソースの構成を表示するには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

`status`

コマンドに対するオペランドとして指定された論理ホスト名リソースのステータスを表示します。デフォルトでは、すべてのリソースのステータスが表示されます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、ステータスが表示されるリソースを限定できます。

`-g` *resourcegrouplist* — *resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースのみのステータスを表示します。

`-n` *nodelist* — *nodelist* 内のノード上でホストされている、オペランドのリスト内のリソースについてのみステータスを表示します。

`-s` *statelist* — *statelist* 内の状態にある、オペランドのリスト内のリソースだけのステータスを表示します。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、指定したリソースグループ内のすべてのリソースまたは指定したリソースタイプのインスタンスであるすべてのリソースを指定できます。オペランドを指定しないと、指定したリソースグループ内のすべてのリソースのステータスまたは指定したリソースタイプのインスタンスであるすべてのリソースのステータスが表示されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから、ゾーンクラスタ内の論理ホスト名リソースのステータスを表示するには、`-Z` オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

`unmonitor`

コマンドに対するオペランドとして指定された論理ホスト名リソースのモニタリングを無効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースに対するモニタリングが無効になります。

無効になっているリソースのモニタリングを無効にしても、リソースは影響を受けません。リソースとそのモニターは、すでにオフライン状態です。

注記 - リソースに対するモニタリングを無効にしても、そのリソースは無効に*なりません*。ただし、リソースを無効にする場合、モニタリングを無効にする必要はありません。無効なリソースとそのモニターは、オフライン状態が維持されます。

`-g` オプションを指定すると、オペランドのリストをフィルタリングし、モニタリングを無効にするリソースを限定できます。`-g` オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースのモニタリングを無効にします。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタへの論理ホスト名リソースのモニタリングをオフにするには、-z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

disable サブコマンドおよび monitor サブコマンドの説明も参照してください。

次のオプションがサポートされています。

-?
--help

ヘルプ情報を表示します。このオプションを使用する場合、ほかの処理は実行されません。

このオプションを指定するとき、サブコマンドは指定してもしなくてもかまいません。

サブコマンドなしでこのオプションを指定すると、このコマンドのサブコマンドのリストが表示されます。

サブコマンド付きでこのオプションを指定すると、サブコマンドの使用方法が表示されます。

特定のサブコマンドを指定した場合のこのオプションの効果は、次のようになります。

create

-g オプションとともに指定された場合、指定されたリソースグループのすべてのリソースプロパティーのヘルプ情報を表示します。

set

コマンドに対するオペランドとして指定したリソースのプロパティーに関する情報を表示します。

-a
--automatic

クラスタ構成情報からリソースが作成される場合、次の処理も自動的に実行します。

- リソースタイプの登録
- リソースグループの作成
- オペランドのリスト内で指定されているリソースの依存先リソースの作成
- クラスタ構成情報には、次の処理をすべて実行するのに必要な十分な情報が含まれている必要があります。
- リソースタイプの登録を有効にする
- リソースグループの作成を有効にする
- リソースの作成を有効にする

このオプションを指定できるのは、create サブコマンドの場合だけです。このオプションを指定する場合は、-i オプションも指定し、構成ファイルを指定します。

-d
--disable

リソースの作成時にリソースを無効にします。このオプションを指定できるのは、create サブコマンドの場合だけです。デフォルトでは、リソースは作成されたあと、有効な状態になります。

リソースは、有効化しても、オンライン状態になるとは限りません。リソースは、リソースのリソースグループが 1 つ以上のノードでオンライン状態になったあとでのみオンライン状態になります。

-f *errorflag*
--flag *errorflag*

reset サブコマンドによってクリアするエラーフラグを明示的に指定します。このオプションを指定できるのは、reset サブコマンドの場合だけです。デフォルトでは、reset サブコマンドはエラーフラグ STOP_FAILED をクリアします。

-f オプションが受け付けるエラーフラグは、STOP_FAILED だけです。

-F
--force

無効状態でないリソースの削除が、強制的に実行されます。このオプションは、delete サブコマンドの場合にだけ指定できます。

-g *resourcegroup*[,…]
--resourcegroup *resourcegroup*[,…]

1 つのリソースグループまたはリソースグループのリストを指定します。

create 以外のサブコマンドの場合、コマンドは -g オプションで指定したリソースグループのメンバーである、オペランドのリスト内のリソースにだけ作用します。

特定のサブコマンドを指定した場合のこのオプションの効果は、次のようになります。

create	指定したリソースグループ内でリソースを作成するように指定します。-create サブコマンドを指定して g を使用する場合、リソースグループは 1 つしか指定することができません。

-h *lhost*[,…]
--logicalhost *lhost*[,…]

このリソースが表す論理ホスト名のリストを指定します。複数の論理ホスト名を新しい論理ホスト名リソースと関連付ける場合または論理ホスト名がリソース自体と同じ名前でない場合は、-h オプションを使用します。リスト内のすべての論理ホスト名は、同じサブネット上に置きます。-h オプションを指定しない場合、リソースはリソース名と同じ名前の 1 台の論理ホスト名を表します。

-h は、-p とともに HostnameList プロパティーを設定する代わりに使用できます。ただし、-h を使用し、同じコマンド内で HostnameList を明示的に設定することはできません。

-h は、create サブコマンドの場合にだけ使用できます。

注記 - ゾーンクラスタの場合、すべての論理ホスト名または対応する IP アドレスを、ゾーンクラスタ構成内のグローバルスコープのネットプロパティーで指定する必要があります。指定しない場合、リソースグループの作成に失敗します。

グローバルスコープのネットプロパティーについての詳細は、613 ページの`clzonecluster`(1CL) のマニュアルページを参照してください。

`-i` {- | *clconfiguration*}
`--input` {- | *clconfiguration*}

論理ホスト名リソースの作成または変更に使用する構成情報を指定します。この情報は、1447 ページの`clconfiguration`(5CL) のマニュアルページに定義されている形式に準拠している必要があります。この情報は、ファイルに含めることも、標準入力を使って指定することもできます。標準入力を指定するには、ファイル名の代わりに - を指定します。

コマンドに対するオペランドとして指定したリソースだけが、作成または変更されます。コマンドで指定したオプションは、構成情報で設定されているオプションより優先されます。構成パラメータは、構成情報内で設定されていない場合、コマンド行で指定します。

特定のサブコマンドを指定した場合のこのオプションの効果は、次のようになります。

`create`	`-a` オプションと指定された場合は、必要なすべてのリソースタイプを登録し、必要なすべてのリソースグループを作成します。登録および構成に必要なすべての情報を指定します。その他の構成データはすべて無視します。

`-l` *listtype*
`--listtype` *listtype*

`list-props` サブコマンドによって表示するリソースプロパティーのタイプを指定します。このオプションは、`list-props` サブコマンドの場合にだけ指定できます。

listtype に対して、次のリストから値を 1 つ指定する必要があります。

`all`	標準プロパティーと拡張プロパティーを表示するように指定します。
`extension`	拡張プロパティーだけを表示するように指定します。デフォルトでは、拡張プロパティーだけが表示されます。
`standard`	標準プロパティーだけを表示するように指定します。

`-l` オプションを指定しないと、`-p` オプションを使って標準プロパティーを明示的に指定しないかぎり、拡張プロパティーしか表示されません。

`-n` *node*[,…]
`--node` *node*[,…]

ターゲットのグローバルクラスタまたはゾーンクラスタに、ノードまたはノードのリストを指定します。各ノードは、ノード名またはノード ID として指定できます。`-z` オプションが指定されて

いる場合は、-n オプションで、グローバルクラスタホスト名ではなくゾーンクラスタホスト名のみを指定できます。-Z オプションが指定されていない場合は、-n オプションでグローバルクラスタホスト名のみを指定できます。

このオプションとともに指定できるサブコマンドは、次のとおりです。

`disable`	指定したノード上でホストされている、オペランドのリスト内のリソースのみを無効にします。
`enable`	指定したノード上でホストされている、オペランドのリスト内のリソースのみを有効にします。
`status`	指定したノード上でホストされている、オペランドのリスト内のリソースについてのみステータスを報告します。

`-N` *netif@node*[,…]
`--netiflist` *netif@node*[,…]

リソースプロパティーを指定します。-N オプションを使用すると、プロパティーの p オプションを使用せずに -NetIfList プロパティーを設定できます。-N を指定しない場合、clreslogicalhostname コマンドは、利用できる IPMP グループまたはパブリックアダプタと、HostnameList プロパティーと関連付けられているサブネットに基づいて、NetIfList プロパティーを設定しようと試みます。

NetIfList プロパティーは、*ipmpgroup@node*[,…]、または *publicNIC@node*[,…] の形式で指定できます。-N を使用しない場合、またはそれを *publicNIC@node* とともに使用する場合、clreslogicalhostname コマンドは必要な IPMP グループを作成しようと試みます。システムは基本デフォルトで単一アダプタの IPMP グループを作成します。このグループは、あとで標準 Solaris IPMP インタフェースを使用して変更できます。IPMP グループは、グローバルクラスタノードの場合のみ、自動的に作成されます。

-N は、NetIfList プロパティーを -p とともに直接設定する代わりに使用できます。ただし、同じコマンド内で -N を使用して、NetIfList を明示的に設定できません。

-N は、create サブコマンドの場合にだけ使用できます。

`-o` {- | *clconfiguration*}
`--output` {- | *clconfiguration*}

リソース構成情報の書き込み先を指定します。この書き込み先は、ファイルでも、標準出力でもかまいません。標準出力を指定するには、ファイル名の代わりに - を指定します。標準出力を指定すると、コマンドに対するほかのすべての標準出力が抑制されます。このオプションは、export サブコマンドの場合にだけ指定できます。

構成情報は、コマンドに対するオペランドとして指定したリソースの場合だけ書き込まれます。この情報は、1447 ページのclconfiguration(5CL) のマニュアルページに定義されている形式で書き込まれます。

-p *name=value*
-p *name+=array-values*
-p *name-=array-values*
--property *name=value*
--property *name+=value-values*
--property *name-=value-values*

リソースの標準プロパティーと拡張プロパティーを設定します。このオプションは、create サブコマンドおよび set サブコマンドの場合にだけ指定できます。

標準プロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

リソースタイプの拡張プロパティーについては、リソースタイプに関するドキュメントを参照してください。

このオプションとともに使用する演算子は、次のとおりです。

=	プロパティーに、指定した値を設定します。create サブコマンドおよび set サブコマンドがこの演算子を受け付けます。
+=	1 つまたは複数の値を文字列配列値に追加します。この演算子は、set サブコマンドでのみ使用できます。この演算子は、文字列配列値に対してだけ指定できます。
-=	1 つまたは複数の値が、文字列配列値から削除されます。この演算子は、set サブコマンドでのみ使用できます。この演算子は、文字列配列値に対してだけ指定できます。

ノード単位のプロパティーをクラスタノードのサブセット上でのみ設定する場合は、中括弧で囲まれたノードのリストを次のようにプロパティー名の後に付加し、プロパティーが設定の対象となるノードを指定します。

name{*nodelist*}

nodelist は、ノード名またはノード ID をコンマで区切ったリストです。ノード単位のプロパティーの詳細は、1335 ページのrt_properties(5) のマニュアルページを参照してください。

-p *name*[,…]
--property *name*[,…]

list-props サブコマンドおよび show サブコマンドのプロパティーのリストを指定します。

このオプションは、リソースの標準プロパティーおよび拡張プロパティーに対して使用できます。

標準プロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

リソースタイプの拡張プロパティーについては、リソースタイプに関するドキュメントを参照してください。

このオプションを指定しない場合、list-props サブコマンドおよび show サブコマンドは、-v オプションも指定されているかどうかに応じて、すべてまたはほとんどのリソースプロパティーを一覧表示します。

-R
--recursive

必要な依存性がすべて満たされるように、リソースの有効化または無効化を再帰的に実行します。このオプションは、disable サブコマンドおよび enable サブコマンドでのみ指定できます。

このオプションをこれらのサブコマンドとともに指定した場合の効果は、次のとおりです。

disable	コマンドに対するオペランドとして指定したリソースに依存しているリソース (コマンドに対するオペランドとして指定しなかったリソースも含まれる) をすべて無効にします。
enable	コマンドに対するオペランドとして指定したリソースの依存先リソース (コマンドに対するオペランドとして指定しなかったリソースも含まれる) がすべて有効になります。

-s *state*[,…]
--state *state*[,…]

list サブコマンドおよび status サブコマンドの状態のリストを指定します。

このオプションは出力を制限し、ノードリスト内の 1 つまたは複数のノード上で指定されている状態の 1 つにあるリソースだけが含まれるようにします。

可能な状態は、次のとおりです。

- degraded
- detached
- faulted
- monitor_failed
- not_online - online または online_not_monitored 以外のすべてのステータスを指定します
- offline
- online
- online_not_monitored
- start_failed
- stop_failed
- unknown
- unmonitored

- `wait`

`-V`
`--version`

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンドやオペランドなどのオプションは無視されます。`-V` オプションは、コマンドのバージョンを表示するだけです。その他の処理は行いません。

`-v`
`--verbose`

詳細なメッセージを標準出力に表示します。

このオプションは、このコマンドの任意の形式とともに指定できます。

`-v` オプションと `-o -` オプションを同時に指定してはいけません。`-v` オプ ションは無視されます。`-o-` オプションは、その他のすべての標準出力を抑制します。

`-Z` {*zoneclustername* | global | all}
`--zoneclustername=`{*zoneclustername* | global | all}
`--zoneclustername` {*zoneclustername* | global | all}

クラスタ、またはリソースが存在するクラスタや処理するクラスタを指定します。

このオプションは、`export` サブコマンドを除くすべてのサブコマンドでサポートされています。

このオプションを指定する場合は、次のいずれかの引数も指定する必要があります。

zoneclustername	このオプションを使用しているコマンドが、*zoneclustername* という名前のゾーンクラスタ内のみの指定されたすべてのリソースに対して動作することを指定します。
`global`	このオプションを使用するコマンドが、広域クラスタでのみ指定されたすべてのリソースで機能するように指定します。
`all`	広域クラスタでこの引数を使用する場合、それを使用するコマンドが広域クラスタとすべてのゾーンクラスタを含め、すべてのクラスタで指定されたすべてのリソースで機能するように指定します。 この引数をゾーンクラスタで使用すると、このオプションを使用するコマンドが、ゾーンクラスタ内だけで指定されたすべてのリソースに対して機能するように指定されます。

次のオペランドを指定できます。

resource	Oracle Solaris Cluster のリソース名をオペランドとして受け付けるように指定します。サブコマンドで複数のリソースを指定できる場合は、プラス記号 (+) を使用すると、すべての論理ホスト名リソースを指定できます。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

9 CL_ESTATE

オブジェクトの状態が不正です

その時点で変更できない、または常時変更できないプロパティー、リソースグループ、またはその他のオブジェクトを変更しようとしました。

10 CL_EMETHOD

リソースのメソッドが失敗しました

リソースのメソッドが失敗しました。次のいずれかの理由のために、メソッドは失敗しました。

- validate メソッドは、リソースを作成しようとしたときに、あるいは、リソースのプロパティーを変更しようとしたときに失敗しました。
- validate 以外のメソッドは、リソースを有効、無効、または削除しようとしたときに失敗しました。

15 CL_EPROP

無効なプロパティーです

-p、-y、または -x オプションで指定したプロパティーまたは値が存在しないか、許可されていません。

35 CL_EIO

I/O エラー

物理的な入出力エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

これらの終了値は、1047 ページのscha_calls(3HA) のマニュアルページで説明されているリターンコードと互換性があります。

例 126 論理ホスト名リソースの作成

このコマンドは、rg-failover という名前のリソースグループ内に logicalhost1 という名前のリソースを作成します。リソースは有効な状態で作成され、モニタリングも有効になっています。

```
# clreslogicalhostname create -g rg-failover logicalhost1
```

次の 2 つのコマンドはどちらも、ゾーンクラスタ ZC 内に logicalhost1 という名前のリソースを作成します。これらのコマンドは、グローバルクラスタノードまたはゾーンクラスタ ZC から実行できます。コマンドをゾーンクラスタから実行する場合、任意でゾーンクラスタ名を持つリソースのスコープを明示的に定義できます。

```
# clreslogicalhostname create -g rg-failover -Z ZC logicalhost1

# clreslogicalhostname create -g rg-failover ZC:logicalhost1
```

例 127 異なる論理ホスト名を持つ論理ホスト名リソースの作成

このコマンドは、rg-failover という名前のリソースグループ内に、rs-logicalhost1 という名前のリソースを作成します。

論理ホスト名はリソース名と同じではありませんが、論理ホストの名前と IP アドレスは同じままです。

```
# clreslogicalhostname create -g rg-failover \
-h logicalhost1 rs-logicalhost1
```

例 128 論理ホスト名リソースの IPMP グループの指定

このコマンドは、logicalhost1 リソースの IPMP グループを設定します。

```
# clreslogicalhostname create -g rg-failover \
-N ipmp0@black,ipmp0@white logicalhost1
```

例 129 論理ホスト名リソースの削除

このコマンドは、logicalhost1 という名前のリソースを削除します。

```
# clreslogicalhostname delete logicalhost1
```

例 130 論理ホスト名リソースの一覧表示

このコマンドは、すべての論理ホスト名リソースを一覧表示します。

```
# clreslogicalhostname list
logicalhost1
logicalhost2
```

例 131 論理ホスト名リソース、リソースグループ、およびリソースタイプの一覧表示

このコマンドは、すべての論理ホスト名リソースをそれらのリソースグループおよびリソースタイプと合わせて一覧表示します。

```
# clreslogicalhostname list -v
Resources    Resource Groups Resource Types
---------    --------------- --------------
logicalhost1 rg-failover-1   SUNW.LogicalHostname
logicalhost2 rg-failover-2   SUNW.LogicalHostname
```

例 132 論理ホスト名リソースの拡張プロパティーの一覧表示

このコマンドは、すべての論理ホスト名リソースの拡張プロパティーを一覧表示します。

```
# clreslogicalhostname list-props -v
Properties       Descriptions
----------       ------------
NetIfList        List of IPMP groups on each node
HostnameList     List of hostnames this resource manages
CheckNameService Name service check flag
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページの`Intro`(1CL), 551 ページの`cluster`(1CL), 273 ページの`clresource`(1CL), 347 ページの`clressharedaddress`(1CL), 305 ページの`clresourcegroup`(1CL), 333 ページの`clresourcetype`(1CL), 1047 ページの`scha_calls`(3HA), 1447 ページの`clconfiguration`(5CL), Unresolved link to " rbac5", 1287 ページの`r_properties`(5)

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のオプションを指定してこのコマンドを実行できます。

- `-?` オプション
- `-V` オプション

サブコマンドを指定してこのコマンドを実行する場合、スーパーユーザー以外のユーザーは RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
`create`	`solaris.cluster.modify`
`delete`	`solaris.cluster.modify`
`disable`	`solaris.cluster.admin`
`enable`	`solaris.cluster.admin`
`export`	`solaris.cluster.read`
`list`	`solaris.cluster.read`

サブコマンド	RBAC の承認
list-props	solaris.cluster.read
monitor	solaris.cluster.admin
reset	solaris.cluster.admin
set	solaris.cluster.modify
show	solaris.cluster.read
status	solaris.cluster.read
unmonitor	solaris.cluster.admin

名前

clresource, clrs — Oracle Solaris Cluster データサービスのリソースの管理

```
/usr/cluster/bin/clresource subcommand [-?]

/usr/cluster/bin/clresource -V

/usr/cluster/bin/clresource subcommand [options] -v [resource]...

/usr/cluster/bin/clresource clear [-f errorflag] [-g
     [resourcegroup,]...] [-t [resourcetype,]...] -n node
     [,…]] [-Z {zoneclustername | global}]
     {+ | resource...}

/usr/cluster/bin/clresource create -g resourcegroup -t
     resourcetype [-d] [-p "property-name[{node-specifier,...}]"=
     value] [-x "extension-property[{node-specifier,...}]"=value] [-y
     standard-property=value] [-Z {zoneclustername | global}]
     resource

/usr/cluster/bin/clresource create -i {- | clconfiguration} -t
     resourcetype [-a] [-d] [-g [resourcegroup,]...] [-p "
     property-name[{node-specifier,...}]"=value] [-x "
     extension-property[{node-specifier,...}]"=value] [-y
     standard-property=value] {+ | resource...}

/usr/cluster/bin/clresource delete [-F] [-g [resourcegroup,]...]
     [-t [resourcetype,...]] [-Z {zoneclustername | global}]
     {+ | resource...}

/usr/cluster/bin/clresource disable [-r] [-g [resourcegroup,]...]
     [-t [resourcetype,]...] [-n node[,…]]
     [-Z {zoneclustername | global}] {+ | resource...}

/usr/cluster/bin/clresource enable [-r] [-g [resourcegroup,]...]
     [-t [resourcetype,]...] [ -n node[,…]]
     [-Z {zoneclustername | global}] {+ | resource...}

/usr/cluster/bin/clresource export [-o {- | configfile}]
     [+ | resource...]

/usr/cluster/bin/clresource list [-g [resourcegroup,]...] [-t
     [resourcetype,]...] [ -n node[,…]] [-Z
     {zoneclustername [,...] | global | all}] [+ | resource...]

/usr/cluster/bin/clresource list-props [-l listtype] [-g
     [resourcegroup,]...] [-p "property-name[{node-specifier,…}]",...]
     [-t [resourcetype,]...] [-x "extension-property[{node-specifier,…}]"...]
     [-y "standard-property[{node-specifier,…}]",...] [-Z
```

```
     {zoneclustername [,...] | global | all}] [+ | resource...]

/usr/cluster/bin/clresource monitor [-g [resourcegroup,]...] [-t
     [resourcetype,]...] [ -n node[,…]] [-Z
     {zoneclustername | global}] {+ | resource...}

/usr/cluster/bin/clresource set [-g [resourcegroup,]...] [-p "
     property-name[{node-specifier,…}]"=value] [-t
     [resourcetype,]...] [-x "extension-property[{node-specifier,…}]"=
     value] [-y standard-property [+= | -=]value] [-Z
     {zoneclustername | global}] {+ | resource...}

/usr/cluster/bin/clresource show [-g [resourcegroup,]...] [-p
     property-name[{node-specifier,…}]",...] [-t [resourcetype,]...]
     [-x "extension-property[{node-specifier,…}]",...] [-y "
     standard-property[{node-specifier,…}]",...] [-Z
     {zoneclustername [,...] | global | all}] [+ | resource...]

/usr/cluster/bin/clresource status [-g [resourcegroup,]...] [-s
     [state,]...] [-t [resourcetype,]...] [ -n node[,…]]
     [-Z {zoneclustername [,...] | global | all}] [+ | resource...]

/usr/cluster/bin/clresource unmonitor [-g [resourcegroup,]...]
     [-t [resourcetype,]...] [ -n node[,…]]
     [-Z {zoneclustername | global}] {+ | resource...}
```

clresource コマンドは、Oracle Solaris Cluster データサービスのリソースを管理します。clrs コマンドは、clresource コマンドの短縮形式です。clresource コマンドと clrs コマンドは同じものです。どちらの形式のコマンドも使用できます。

このコマンドの一般的な形式は次のとおりです。

```
clresource [subcommand] [options] [operands]
```

options に -? または -V オプションを指定する場合だけは、*subcommand* を省略できます。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式は、このマニュアルページの「オプション」セクションのオプションの説明で紹介されています。

ゾーンクラスタでこのコマンドを使用する

clresource コマンドは、ゾーンクラスタで、export を除くすべてのサブコマンドを指定して使用することができます。

export 以外のすべてのサブコマンドで -Z オプションを使用して、操作を制限する特定のゾーンクラスタの名前を指定することもできます。また、リソース名 (*zoneclustername* : *resource*) にゾーンクラスタ名を付けて、操作を特定のゾーンクラスタに制限することもできます。

グローバルクラスタノードからすべてのゾーンクラスタ情報にアクセスできますが、特定のゾーンクラスタは他のゾーンクラスタを認識しません。特定のゾーンクラスタに操作を制限できない場合、使用するサブコマンドは現在のクラスタでのみ機能します。

ゾーンクラスタのリソースは、別のゾーンクラスタのリソース、またはグローバルクラスタのリソースに依存性を持ちます。また、グローバルクラスタからのリソースは、そのグローバルクラスタのゾーンクラスタのリソースに依存性を持ちます。このインタークラスタ依存性は、グローバルクラスタより設定できます。

インタークラスタ依存性は、次のコマンドで指定できます。

```
# clresource set -p resource_dependencies=target-
zc
:target-rs  source-zc:
source-rs
```

たとえば、ゾーンクラスタ ZC1 内のリソース R1 からゾーンクラスタ ZC2 内のリソース R2 への依存性を指定する必要がある場合は、次のコマンドを使用します。

```
# clresource set -p resource_dependencies=ZC2:R2 ZC1:R1
```

グローバルクラスタリソース R2 にゾーンクラスタ ZC1 リソース R1 の依存性を指定する必要がある場合、次のコマンドを使用してください。

```
# clresource set -p resource_dependencies=global:R2 ZC1:R1
```

既存のリソース依存性（強、弱、リスタート、オフラインリスタート）がサポートされています。

リソースのステータスとステータス

リソースのステータスとリソースのステータスは、ノード単位で維持されます。リソースは、各クラスタノード上で固有のステータス、固有のステータスを持ちます。

Resource Group Manager(RGM) は、リソースに対して起動されたメソッドに基づき、各ノード上でリソースの状態を設定します。たとえば、指定のノード上でリソースに対する STOP メソッドを正しく実行した場合、そのリソースのノード上での状態は OFFLINE になります。STOP メソッドが 0 以外またはタイムアウトで終了した場合、そのリソースの状態は Stop_failed になります。

リソースの可能な状態は、次のとおりです。

- Online
- Offline
- Start_failed

- Stop_failed
- Monitor_failed
- Online_not_monitored
- Starting
- Stopping
- Not_online

注記 - Offline や Start_failed などの状態の名前は、大文字と小文字が区別*されません*。状態の名前を指定する際には、大文字と小文字を任意に組み合わせることができます。

RGM は、リソースのステータスだけでなく、リソース自体が API を使って設定するリソースのステータスも維持します。Status Message のフィールドは、実際には、ステータスキーワードとステータスメッセージからなります。ステータスメッセージは、ステータスキーワードのあとに出力される任意のテキスト文字列で、リソースによって任意に設定されます。

リソースステータスの値には、次のものがあります。

DEGRADED	リソースはオンラインですが、何らかの理由でパフォーマンスまたは可用性が低下しています。
FAULTED	リソースの機能を妨げるエラーが検出されました。
OFFLINE	リソースはオフラインです。
ONLINE	リソースはオンラインでサービスを提供します。
UNKNOWN	現在のステータスは不明または遷移中です。

サポートされるサブコマンドには次のものがあります。

clear

コマンドに対するオペランドとして指定したリソースに関連付けられているエラーフラグをクリアします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースのエラーフラグがクリアされます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、エラーフラグがクリアされるリソースを限定できます。

-g *resourcegroup*	*resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけをクリアします。

-n *node*
: 指定した 1 つまたは複数のノード上のリソースをクリアします。-n オプションを指定しないと、すべてのノード上のリソースがクリアされます。

-t *resourcetype*
: *resourcetype* 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースだけをクリアします。

-Z {*zoneclustername* | `global`}
: 特定のクラスタまたは指定したクラスタ内のリソースだけをクリアします。グローバルクラスタからゾーンクラスタ内のリソースをクリアするには、-Z オプションを使用してゾーンクラスタを指定します。

デフォルトでは、`clear` サブコマンドは `STOP_FAILED` エラーフラグをクリアします。クリアするエラーフラグを明示的に指定するには、-f オプションを使用します。-f オプションが受け付けるエラーフラグは、`STOP_FAILED` だけです。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.admin` RBAC の承認が必要です。

`create`

コマンドに対するオペランドとして指定されたリソースを作成します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから特定のゾーンクラスタ内のリソースを作成するには、-Z オプションを使用してゾーンクラスタの名前を指定します。

`create` を -i オプションとともに使用して構成ファイルを指定した場合、このサブコマンドはプラス記号 (+) をオペランドとして受け付けます。+ オペランドを使用すると、存在しないすべてのリソースが構成ファイル内に作成されます。

デフォルトでは、リソースはモニタリング対象となり、有効な状態で作成されます。ただし、リソースは、そのリソースのリソースグループがオンライン状態になったあとでのみ、オンライン状態になり、モニター対象となります。無効な状態でリソースを作成するには、-d オプションを指定します。

リソースの作成時にプロパティー値を設定するには、次のオプションを使用します。

-p *property-name*= *value*
: 標準プロパティーまたは拡張プロパティーを、それらの名前が一意であるかぎり、設定します。

-x *extension-property*= *value*
: 拡張プロパティーを設定します。

-y *standard-property*= *value*
: 標準プロパティーを設定します。

node-specifier は、-p および -x オプションに対する任意の修飾子です。リソースが作成されると、指定した 1 つまたは複数のノード上にあるプロパティー*のみ*が設定されることを示します。指定したプロパティーは、クラスタ内のほかのノード上では設定されません。*node-specifier* を含めない場合、指定したプロパティーは、クラスタ内のすべてのノード上で設定されます。*node-specifier* の構文例を次に示します。

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

中括弧 ({}) は、指定したプロパティーをノード phys-schost-1 でのみ設定することを示します。ほとんどのシェルの場合、中括弧は引用符で囲みます。

次の構文を使用すると、2 つのノード上でプロパティーを設定できます。

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

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。

delete サブコマンドの説明も参照してください。

delete

コマンドに対するオペランドとして指定されたリソースを削除します。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースが削除されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドは、リソース間の依存性を満たすのに必要な順序で複数のリソースを削除します。このサブコマンドは、コマンド行でリソースを指定する順序を無視します。

同時に複数のリソースを削除する場合、このコマンドは、いくつかのステップに分けて実行されます。たとえばノードで問題が発見された場合などのように、コマンドが割り込まれたときには、いくつかのリソースが無効な構成のまま残される場合があります。問題を修正して、リソースの削除を終了するには、同じコマンドを正常なノードで再実行してください。

次のオプションを指定すると、オペランドのリストをフィルタリングし、削除するリソースを限定できます。

-g *resourcegroup*	*resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを削除します。
-t *resourcetype*	*resourcetype* 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースだけを削除します。
-Z {*zoneclustername* | global}	特定のクラスタまたは指定したクラスタ内のリソースだけを削除します。グローバルクラスタからゾーンクラスタ内のリソースを削除するには、-Z オプションを使用してゾーンクラスタを指定します。

デフォルトでは、リソースは次の条件が満たされる場合に *のみ* 削除されます。

- リソースが無効な状態である。
- リソースに対するすべての依存性が削除されている。

指定したリソースの削除を強制的に実行するには、-F オプションを指定します。このオプションは、次のような影響があるので、注意して使用してください。

- 指定したすべてのリソース (無効になっていないリソースも含む) が削除されます。
- 指定したすべてのリソースが、他のリソースのリソース依存性設定から削除されます。

このような影響により、クラスタ内でサービスが消失する場合があります。削除されていない依存リソースも、無効な状態またはエラー状態で残される場合があります。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。

`create` サブコマンドの説明も参照してください。

`disable`

コマンドに対するオペランドとして指定されたリソースを無効にします。このサブコマンドでプラス記号 (`+`) をオペランドとして指定すると、すべてのリソースが無効になります。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、無効にするリソースを限定できます。

`-g` *resourcegroup*
: *resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを無効にします。

`-n` *node*
: `-n` *node* を使用すると、1 つ以上のノードでリソースを無効化できます。

`-t` *resourcetype*
: *resourcetype* 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースだけを無効にします。

`-Z` {*zoneclustername* | `global`}
: 特定のクラスタまたは指定したクラスタ内のリソースだけを無効にします。グローバルクラスタからゾーンクラスタ内のリソースを削除するには、`-Z` オプションを使用してゾーンクラスタを指定します。

`-r` オプションは、コマンドに対するオペランドとして指定したリソースに依存しているすべてのリソースを無効にします。これらのリソースは、コマンドに対するオペランドとして指定しなかった場合でも無効になります。`-g` オプションと `-t` オプションは、リソース依存性を満たすためだけに無効化されるリソースには適用されません。

このサブコマンドは、リソースのモニタリングステータスに影響を与えません。リソースは、有効な状態のときにモニターされていると、無効化されたあともモニターされます。リソースは、あとで再度有効な状態にしても、モニターされます。

このサブコマンドは、リソース間の依存性を満たすのに必要な順序でリソースを無効にします。このサブコマンドでは、コマンド行でリソースが指定された順序は無視されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.admin` RBAC の承認が必要です。

`enable` サブコマンドの説明も参照してください。

`enable`

コマンドに対するオペランドとして指定されたリソースを有効にします。このサブコマンドでプラス記号 (`+`) をオペランドとして指定すると、すべてのリソースが有効になります。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、有効にするリソースを限定できます。

`-g` *resourcegroup*	*resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを有効にします。
`-n` *node*	`-n` *node* を使用すると、1 つ以上のノードでリソースを有効化できます。
`-t` *resourcetype*	*resourcetype* 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースだけを有効にします。
`-Z` {*zoneclustername* | `global`}	特定のクラスタまたは指定したクラスタ内のリソースだけを有効にします。グローバルクラスタからゾーンクラスタ内のリソースを有効にするには、`-Z` オプションを使用してゾーンクラスタを指定します。

必要なリソース依存性をすべて確実に満たすには、`-r` オプションを指定します。`-r` オプションは、コマンドに対するオペランドとして指定したリソースが依存しているすべてのリソースを有効にします。これらのリソースは、コマンドに対するオペランドとして指定しなかった場合でも有効になります。`-g` オプションおよび `-t` オプションは、リソース依存性を満たすためだけに有効化されるリソースには適用されません。

リソースは、リソース間の依存性を満たすのに必要な順序で有効化されます。このサブコマンドでは、コマンド行でリソースが指定された順序は無視されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.admin` RBAC の承認が必要です。

`disable` サブコマンドの説明も参照してください。

`export`

1447 ページの`clconfiguration`(5CL) のマニュアルページで規定されている形式で、クラスタリソース構成をエクスポートします。

このサブコマンドは、大域ゾーンだけで使用できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。

`list`

コマンドに対するオペランドとして指定したリソースのリストを表示します。デフォルトでは、すべてのリソースが表示されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、表示するリソースを限定できます。

`-g` *resourcegroup*	*resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを表示します。

-n *node*
: -n *node* を使用すると、1 つ以上のノードでオンライン状態のリソースのみを一覧表示できます。

-t *resourcetype*
: *resourcetype* 内のリソースタイプのインスタンスであるリソースだけを表示します。

-Z {*zoneclustername* | `global` | `all`}
: 特定のクラスタまたは指定したクラスタ内のリソースだけを表示します。グローバルクラスタからゾーンクラスタ内のリソースを表示するには、-Z オプションを使用してゾーンクラスタを指定します。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソース構成が表示されます。-g オプションまたは -t オプションを指定すると、表示する情報を特定のリソースグループまたはリソースタイプに制限することができます。オペランドが指定されていない場合、指定されているリソースグループ内のすべてのリソースまたは指定されているリソースタイプのインスタンスであるすべてのリソースが表示されます。

-v オプションを指定すると、リスト内の各リソースのリソースグループおよびリソースタイプも表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。

`list-props`

コマンドに対するオペランドとして指定したリソースのプロパティーのリストを表示します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、プロパティーが表示されるリソースを限定できます。

-g *resourcegroup*
: *resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースのプロパティーだけを表示します。

-t *resourcetype*
: *resourcetype* 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースのプロパティーだけを表示します。

-l オプションでは、表示するリソースプロパティーのタイプを指定します。

`-l all`
: 標準プロパティーと拡張プロパティーを表示するように指定します。

`-l extension`
: 拡張プロパティーだけを表示するように指定します。デフォルトでは、拡張プロパティーだけが表示されます。

`-l standard`
: 標準プロパティーだけを表示するように指定します。

-l オプションを指定しないと、拡張プロパティーだけが表示されます。標準プロパティーを表示するには、-p オプションまたは -y オプションを使用して、プロパティーを明示的に指定します。

次のオプションを指定すると、表示するリソースプロパティーのセットを限定できます。

-p *property-name*
: *property-name* で指定したプロパティーだけ表示します。*property-name* では、標準プロパティーと拡張プロパティーを指定できます。

-x *extension-property*
: *extension-property* で指定した 1 つまたは複数のノード上の拡張プロパティーだけを表示します。

-y *standard-property*
: *standard-property* で指定した標準プロパティーだけを表示します。

node-specifier は、-p、-x、および -y オプションに対する任意の修飾子です。指定した 1 つまたは複数のノード上にあるプロパティー*のみ*が表示されることを示します。指定したプロパティーは、クラスタ内のほかのノード上では表示されません。*node-specifier* を含めない場合、指定したプロパティーは、クラスタ内のすべてのノード上で表示されます。*node-specifier* の構文例を次に示します。

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

中括弧 ({}) は、指定したプロパティーをノード `phys-schost-1` でのみ表示するよう指定します。ほとんどのシェルの場合、中括弧は引用符で囲みます。

次の構文を使用すると、2 つのノード上でプロパティーを表示できます。

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

-v オプションを指定すると、各プロパティーの説明も表示されます。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースプロパティーが表示されます。オペランドを指定しないと、指定したリソースグループ内のすべてのリソースのプロパティーまたは指定したリソースタイプのインスタンスであるすべてのリソースのプロパティーが表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。

-Z {*zoneclustername* | `global` | `all`}
: 特定のクラスタまたは指定したクラスタ内のリソースのプロパティーを一覧表示します。グローバルクラスタからゾーンクラスタ内のリソースを一覧表示するには、-Z オプションを使用してゾーンクラスタを指定します。

`monitor`
: コマンドに対するオペランドとして指定したリソースのモニタリングを有効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースに対してモニタリングが有効になります。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、モニタリングを有効にするリソースを限定できます。

-g *resourcegroup*
: *resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースのモニタリングだけを有効にします。

-n *node*
: 1 つ以上のノードでオンライン状態のリソースのモニタリングのみを有効にします。

-t *resourcetype*
: *resourcetype* 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースのモニタリングだけを有効にします。

-Z {*zoneclustername* | `global`}
: 特定のクラスタまたは指定したクラスタ内のリソースのモニタリングだけを有効にします。グローバルクラスタからゾーンクラスタ内のリソースの監視を有効にするには、`-Z` オプションを使用してゾーンクラスタを指定します。

リソースは、モニタリングが有効になっている場合、次の条件が満たされているときだけモニタリングされます。

- リソースが有効な状態にある。
- 該当リソースが含まれるリソースグループが、1 つ以上のクラスタノード上でオンライン状態にある。

注記 - リソースに対するモニタリングを有効にする場合、該当リソースを有効にする必要は*ありません*。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.admin` RBAC の承認が必要です。

`unmonitor` サブコマンドの説明も参照してください。

`set`

コマンドに対するオペランドとして指定したリソースの、指定したプロパティーを設定します。このサブコマンドでプラス記号 (`+`) をオペランドとして指定すると、すべてのリソースの指定したプロパティーが変更されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、プロパティーが変更の対象となるリソースを限定できます。

-g *resourcegroup*
: *resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースのプロパティーだけを変更します。

-t *resourcetype*
: *resourcetype* 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースのプロパティーだけを変更します。

プロパティー値は、次のオプションを使用して設定します。

-p *property-name*= *value*
: 標準プロパティーまたは拡張プロパティーを、それらの名前が一意であるかぎり、設定します。

-x *extension-property*= *value*
: 拡張プロパティーを設定します。

-y *standard-property*= *value*
: 標準プロパティーを設定します。

node-specifier は、ノード単位の拡張プロパティーを更新するための -p および -x オプションに対する*オプションの*修飾子です。指定した 1 つまたは複数のノード上で*のみ*プロパティーが設定されることを示します。指定したプロパティーは、クラスタ内のほかのノード上では設定されません。*node-specifier* を含めない場合、指定したプロパティーは、クラスタ内のすべてのノード上で設定されます。*node-specifier* の構文例を次に示します:

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

中括弧 ({}) は、指定したプロパティーをノード `phys-schost-1` でのみ設定することを示します。ほとんどのシェルの場合、中括弧は引用符で囲みます。

次の構文を使用すると、2 つのノード上でプロパティーを設定できます。

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

-Z {*zoneclustername* | `global`}
: 特定のクラスタまたは指定したクラスタ内のリソースのプロパティーだけを設定します。グローバルクラスタからゾーンクラスタ内のリソースのプロパティーを設定するには、-Z オプションを使用してゾーンクラスタを指定します。

  スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。

`show`
: コマンドに対するオペランドとして指定したリソースの構成を表示します。デフォルトでは、すべてのリソースの構成が表示されます。

  このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

  次のオプションを指定すると、オペランドのリストをフィルタリングし、構成が表示されるリソースを限定できます。

  -g *resourcegroup*
  : *resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースの構成だけを表示します。

  -n *node*
  : -n *node* を使用すると、1 つ以上のノードでオンライン状態のリソースについてのみ構成を表示できます。

  -t *resourcetype*
  : *resourcetype* 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースの構成だけを表示します。

-Z {*zoneclustername* | `global` | `all`}

特定のクラスタまたは指定したクラスタ内のリソースだけを表示します。グローバルクラスタからゾーンクラスタ内のリソースを表示するには、-Z オプションを使用してゾーンクラスタを指定します。

次のオプションを指定すると、表示するリソースプロパティーのセットを限定できます。

-p *property-name*
: *property-name* で指定したプロパティーだけ表示します。*property-name* では、標準プロパティーと拡張プロパティーを指定できます。

-x *extension-property*
: *extension-property* で指定した 1 つまたは複数のノード上の拡張プロパティーだけを表示します。

-y *standard-property*
: *standard-property* で指定した標準プロパティーだけを表示します。

node-specifier は、-p、-x、および -y オプションに対する任意の修飾子です。指定した 1 つまたは複数のノード上にあるプロパティー*のみ*が表示されることを示します。指定したプロパティーは、クラスタ内のほかのノード上では表示されません。*node-specifier* を含めない場合、指定したプロパティーは、クラスタ内のすべてのノード上で表示されます。*node-specifier* の構文例を次に示します。

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

中括弧 ({}) は、指定したプロパティーをノード `phys-schost-1` でのみ表示するよう指定します。ほとんどのシェルの場合、中括弧は引用符で囲みます。

次の構文を使用すると、2 つのノード上でプロパティーを表示できます。

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

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソース構成が表示されます。-g オプションまたは -t オプションを指定すると、表示する情報を特定のリソースグループまたはリソースタイプに制限することができます。オペランドを指定しないと、指定したすべてのリソースの構成が表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。

`status`

コマンドに対するオペランドとして指定したリソースのステータスを表示します。デフォルトでは、すべてのリソースのステータスが表示されます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、ステータスが表示されるリソースを限定できます。

-g *resourcegroup*
: *resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースのステータスだけを表示します。

-n *node*

-n *node* を使用すると、1 つ以上のノードでオンライン状態のリソースについてのみステータスを表示できます。-n オプションおよび -s オプションは、同時に指定できません。

-s *state*

state のステータス内にある、オペランドのリスト内のリソースのステータスだけを表示します。-n オプションおよび -s オプションは、同時に指定できません。

-t *resourcetype*

resourcetype 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースのステータスだけを表示します。

-Z {*zoneclustername* | `global` | `all`}

特定のクラスタまたは指定したクラスタ内のリソースのステータスを表示します。グローバルクラスタからゾーンクラスタ内のリソースのステータスを表示するには、-Z オプションを使用してゾーンクラスタを指定します。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースのステータスが表示されます。-g オプションまたは -t オプションを指定すると、表示する情報を特定のリソースグループまたはリソースタイプに制限することができます。オペランドを指定しないと、指定したすべてのリソースのステータスが表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。

`unmonitor`

コマンドに対するオペランドとして指定したリソースのモニタリングを無効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースに対するモニタリングが無効になります。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

無効になっているリソースのモニタリングを無効にしても、リソースは影響を受けません。リソースとそのモニターは、すでにオフライン状態です。

注記 - リソースのモニタリングを無効にしても、リソースは無効に*なりません*。ただし、リソースを無効にする場合、モニタリングを無効にする必要はありません。無効なリソースとそのモニターは、オフライン状態が維持されます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、モニタリングを無効にするリソースを限定できます。

-g *resourcegroup*

resourcegroup 内のリソースグループのメンバーである、オペランドのリスト内のリソースのモニタリングだけを無効にします。

-n *node*

1 つ以上のノードでオンライン状態のリソースのモニタリングのみを無効にします。

-t *resourcetype*
: *resourcetype* 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースのモニタリングだけを無効にします。

-Z {*zoneclustername* | `global`}
: 特定のクラスタまたは指定したクラスタ内のリソースのモニタリングだけを無効にします。グローバルクラスタからゾーンクラスタ内のリソースのモニタリングを無効にするには、`-Z` オプションを使用してゾーンクラスタを指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.admin` RBAC の承認が必要です。

`disable` サブコマンドおよび `monitor` サブコマンドの説明も参照してください。

次のオプションがサポートされています。

`-?`
`--help`
: ヘルプ情報を表示します。このオプションを使用する場合、ほかの処理は実行されません。

このオプションを指定するとき、サブコマンドは指定してもしなくてもかまいません。

サブコマンドなしでこのオプションを指定すると、このコマンドのサブコマンドのリストが表示されます。

サブコマンド付きでこのオプションを指定すると、サブコマンドの使用方法が表示されます。

`-a`
`--automatic`
: リソースがクラスタ構成ファイル (1447 ページの`clconfiguration`(5CL)) から作成される場合は、次の追加の操作を自動的に実行します。

- リソースタイプの登録
- リソースグループの作成
- オペランドのリスト内で指定されているリソースの依存先リソースの作成

クラスタ構成情報には、次の処理をすべて実行するのに必要な十分な情報が含まれている必要があります。

- リソースタイプの登録を有効にする
- リソースグループの作成を有効にする
- リソースの作成を有効にする

このオプションを指定できるのは、`create` サブコマンドの場合だけです。このオプションを指定する場合は、`-i` オプションも指定し、構成ファイルを指定します。

-d
--disable

リソースの作成時にリソースを無効にします。このオプションを指定できるのは、create サブコマンドの場合だけです。デフォルトでは、リソースは作成されたあと、有効な状態になります。

リソースは、有効化しても、オンライン状態になるとは限りません。リソースは、リソースのリソースグループが 1 つ以上のノードでオンラインになってはじめてオンラインになります。

-f *errorflag*
--flag=*errorflag*
--flag *errorflag*

clear サブコマンドによってクリアするエラーフラグを明示的に指定します。このオプションは、clear サブコマンドの場合にだけ指定できます。デフォルトでは、clear サブコマンドは STOP_FAILED エラーフラグをクリアします。

-f オプションが受け付けるエラーフラグは、STOP_FAILED だけです。

-F
--force

無効状態でないリソースの削除が、強制的に実行されます。このオプションは、delete サブコマンドの場合にだけ指定できます。

このオプションは、次のような影響があるので、注意して使用してください。

- 指定したすべてのリソース (無効になっていないリソースも含む) が削除されます。
- 指定したすべてのリソースが、他のリソースのリソース依存性設定から削除されます。

このような影響により、クラスタ内でサービスが消失する場合があります。削除されていない依存リソースも、無効な状態またはエラー状態で残される場合があります。

-g *resourcegroup*[,…]
--resourcegroup=*resourcegroup*[,…]
--resourcegroup *resourcegroup*[,…]

1 つのリソースグループまたはリソースグループのリストを指定します。

create 以外のサブコマンドの場合、このコマンドは指定されたリソースグループのメンバーである、オペランドのリスト内のリソースに対してのみ作用します。-g オプションを使用して、リソースグループを指定します。

-g オプションと create サブコマンドを同時に指定すると、clresource は指定されたリソースグループ内に該当するリソースを作成します。このオプションを使用する場合は、リソースグループを 1 つしか指定することができません。

-i {- | *clconfiguration*}
--input={- | *clconfiguration*}
--input {- | *clconfiguration*}

リソースの作成または変更に使用する構成情報を指定します。この情報は、1447 ページのclconfiguration(5CL) のマニュアルページに定義されている形式に

準拠している必要があります。この情報は、ファイルに含めることも、標準入力を使って指定することもできます。標準入力を指定するには、ファイル名の代わりに - を指定します。

コマンドに対するオペランドとして指定したリソースだけが、作成または変更されます。コマンドで指定したオプションは、構成情報で設定されているオプションより優先されます。構成パラメータは、構成情報内で設定されていない場合、コマンド行で指定します。

-i オプションと create サブコマンドを同時に指定すると、clresource は必要なすべてのリソースタイプを登録し、必要なリソースグループをすべて作成します。登録および構成に必要なすべての情報を指定します。その他の構成データはすべて無視します。

-l *listtype*
--listtype=*listtype*
--listtype *listtype*

list-props サブコマンドによって表示するリソースプロパティーのタイプを指定します。このオプションは、list-props サブコマンドの場合にだけ指定できます。

listtype に対して、次のリストから値を 1 つ指定します。

all	標準プロパティーと拡張プロパティーを表示するように指定します。
extension	拡張プロパティーだけを表示するよう指定します。デフォルトでは、拡張プロパティーだけが表示されます。
standard	標準プロパティーだけを表示するように指定します。

-l オプションを指定しないと、拡張プロパティーだけが表示されます。標準プロパティーを表示するには、-p オプションまたは -y オプションを使用して、プロパティーを明示的に指定します。

-n *node*[,…]
--node=*node*[,…]
--node *node*[,…]

ターゲットのグローバルクラスタまたはゾーンクラスタに、ノードまたはノードのリストを指定します。各ノードはノード名またはノード ID で指定できます。

-Z オプションが指定されている場合は、-n オプションで、グローバルクラスタホスト名ではなくゾーンクラスタホスト名のみを指定できます。-Z オプションが指定されていない場合は、-n オプションでグローバルクラスタホスト名のみを指定できます。

このオプションとともに指定できるサブコマンドは、次のとおりです。

disable	指定したノード上でホストされている、オペランドのリスト内のリソースのみを無効にします。
enable	指定したノード上でホストされている、オペランドのリスト内のリソースのみを有効にします。
list	指定したノード上でホストされている、オペランドのリスト内のリソースのみのリストを表示します。

`monitor`
: 指定したノード上でホストされている、オペランドのリスト内のリソースのみをモニターします。

`show`
: 指定したノード上でホストされている、オペランドのリスト内のリソースについてのみ構成情報を表示します。

`status`
: 指定したノード上でホストされている、オペランドのリスト内のリソースについてのみステータスを報告します。

`unmonitor`
: 指定したノード上でホストされている、オペランドのリスト内のリソースについてのみモニターを解除します。

`-o` {- | *clconfiguration*}
`--output=`{- | *clconfiguration*}
`--output` {- | *clconfiguration*}

リソース構成情報の書き込み先を指定します。この書き込み先は、ファイルでも、標準出力でもかまいません。標準出力を指定するには、ファイル名の代わりに、ダッシュ (-) を指定します。標準出力を指定すると、コマンドに対するほかのすべての標準出力が抑制されます。このオプションは、`export` サブコマンドの場合にだけ指定できます。

構成情報は、コマンドに対するオペランドとして指定したリソースの場合だけ書き込まれます。この情報は、1447 ページの`clconfiguration`(5CL) のマニュアルページに定義されている形式で書き込まれます。

`-p` *property-name=value*
`-p` *property-name+=array-values*
`-p` *property-name-=array-values*
`--property=`*property-name=value*
`--property=`*property-name+=array-values*
`--property=`*property-name-=array-values*
`--property` *property-name=value*
`--property` *property-name+=array-values*
`--property` *property-name-=array-values*

コマンドに対するオペランドとして指定したリソースに対し、プロパティーの値を設定します。このオプションの代入形式を指定できるのは、`create` サブコマンドおよび `set` サブコマンドだけです。

`-p` オプションを使用すると、任意の標準プロパティーまたは拡張プロパティーを指定できます。あるリソースタイプの拡張プロパティーが同じリソースタイプの標準プロパティーと同じ名前を持っている場合、`-p` オプションを使用するとエラーが返されます。そのような場合、拡張プロパティーを指定するには `-x` オプション、標準プロパティーを指定するには `-y` オプションを使用します。

標準プロパティーについては、1287 ページの`r_properties`(5) のマニュアルページを参照してください。

リソースタイプの拡張プロパティーについては、リソースタイプに関するドキュメントを参照してください。

このオプションとともに使用する演算子は、次のとおりです。

=　　プロパティーに、指定した値を設定します。この演算子は、create サブコマンドおよび set サブコマンドで使用できます。

+=　　1 つまたは複数の値を文字列配列値に追加します。この演算子は、set サブコマンドでのみ使用できます。この演算子は、たとえば、Resource_dependencies のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。

-=　　1 つまたは複数の値を文字列配列値から削除します。この演算子は、set サブコマンドでのみ使用できます。この演算子は、たとえば、Resource_dependencies のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。

クラスタノードのサブセットに対してノード単位の拡張プロパティーを設定するには、プロパティーの設定時にノードを指定します。次のように、中括弧で囲まれたノードのリストをプロパティー名のあとに付加します。

name{*node*}

node は、ノード名またはノード ID をコンマで区切ったリストです。ノード単位の拡張プロパティーの詳細は、1335 ページのrt_properties(5) のマニュアルページを参照してください。

ノード単位のリソース依存関係をクラスタノードのサブセットに設定するには、各ノード単位の依存関係を次の形式で指定します:

```
myres1@node1,myres2@node2,myres3@node3
```

gds-rs リソースの場合は、次のコマンドで、ノード ptrancos1 上のリソース trancos-3-rs およびノード ptrancos2 上のリソース trancos-4-rs に対する依存関係を設定します。

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

phys-schost-1# clresource show -p resource_dependencies gds-rs
=== Resources ===
Resource: gds-rs
Resource_dependencies: trancos-3-rs@ptrancos1 trancos-4-rs@ptrancos2
```

ローカルノードスコープでリソースの依存関係を設定するには、LOCAL_NODE　修飾子を次の形式で指定します:

```
myres1{LOCAL_NODE},myres2{LOCAL_NODE}
```

gds-rs リソースの場合は、次のコマンドで、リソース trancos-3-rs に対するローカルノードの依存関係を設定します。

```
# clresource set -p resource_dependencies=trancos-3-rs{LOCAL_NODE} gds-rs

phys-schost-1# clresource show -p resource_dependencies gds-rs
=== Resources ===
```

Resource: gds-rs
Resource_dependencies: trancos-3-rs{LOCAL_NODE}

ノード単位のリソース依存関係および依存関係のスコープ修飾子の詳細は、1287 ページの`r_properties`(5) のマニュアルページを参照してください。

`-p` *property-name*[,…]
`--property=`*property-name*[,…]
`--property` *property-name*[,…]

`list-props` サブコマンドおよび `show` サブコマンドのプロパティーのリストを指定します。

`-p` オプションを使用すると、任意の標準プロパティーまたは拡張プロパティーを指定できます。あるリソースタイプの拡張プロパティーが同じリソースタイプの標準プロパティーと同じ名前を持っている場合、`-p` オプションを使用するとエラーが返されます。そのような場合、拡張プロパティーを指定するには `-x` オプション、標準プロパティーを指定するには `-y` オプションを使用します。

標準プロパティーについては、1287 ページの`r_properties`(5) のマニュアルページを参照してください。

リソースタイプの拡張プロパティーについては、リソースタイプに関するドキュメントを参照してください。

このオプションを指定しなかった場合、`list-props` サブコマンドおよび `show` サブコマンドは、`-v` オプションも指定されているかどうかに基づいて、すべてまたはほとんどのリソースプロパティーを一覧表示します。

`-r`
`--recursive`

必要な依存性がすべて満たされるように、リソースの有効化または無効化を再帰的に実行します。このオプションは、`disable` サブコマンドおよび `enable` サブコマンドの場合にだけ指定できます。

このオプションをこれらのサブコマンドとともに指定した場合の効果は、次のとおりです。

`disable`	コマンドに対するオペランドとして指定したリソースに依存しているリソースを、すべて無効にします。これらのリソースは、コマンドに対するオペランドとして指定しなかった場合でも無効になります。
`enable`	コマンドに対するオペランドとして指定したリソースが依存しているリソースを、すべて有効にします。これらのリソースは、コマンドに対するオペランドとして指定しなかった場合でも有効になります。

`-s` *state*[,…]
`--state=`*state*[,…]
`--state` *state*[,…]

`list` サブコマンドおよび `status` サブコマンドの状態のリストを指定します。

このオプションは出力を制限し、ノードリスト内の 1 つまたは複数のノード上で指定されている状態の 1 つにあるリソースだけが含まれるようにします。

可能な状態は、次のとおりです。

- `Online`
- `Offline`
- `Start_failed`
- `Stop_failed`
- `Monitor_failed`
- `Online_not_monitored`
- `Starting`
- `Stopping`
- `Not_online`

注記 - `Offline` や `Start_failed` などの状態の名前は、大文字と小文字が区別*されません*。状態の名前を指定する際には、大文字と小文字を任意に組み合わせることができます。

`-t` *resourcetype*[,…]
`--type=`*resourcetype*[,…]
`--type` *resourcetype*[,…]

1 つのリソースタイプまたはリソースタイプのリストを指定します。

`create` を除くこのオプションを使用できるすべてのサブコマンドの場合、コマンドは、次の両方の条件を満たすリソースに対してだけ作用します。

- リソースが、オペランドのリスト内にある。
- リソースが、`-t` オプションで指定するリソースタイプのインスタンスである。

`-t` オプションと `clresource create` を同時に指定すると、指定したタイプのリソースが作成されます。指定できるリソースタイプは、1 つだけです。

リソースタイプ名の形式については、Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド の有効な RGM 名"を参照してください。

`-u`

`+` オペランドを使用する場合、このオプションは、リソースグループが中断されたリソース上でコマンドが機能するように指定します。

`+` オペランドを指定した状態で `-u` オプションを指定しないと、コマンドはリソースグループが中断されたすべてのリソースを無視します。`-u` オプションは、`+` オペランドが `clear`、`disable`、`enable`、`monitor`、`set`、または `unmonitor` コマンドとともに指定されている場合に有効となります。

`+` オペランドが `clear`、`disable`、`enable`、`monitor`、`set`、または`unmonitor` サブコマンドとともに使用されている場合、`-u` オプションも指定しないかぎり、コマンドはリソースグループが中断されたすべてのリソースを無視します。

-V
--version

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンドやオペランドなどのオプションは無視されます。-V オプションは、コマンドのバージョンだけを表示します。その他の処理は行いません。

-v
--verbose

詳細メッセージを標準出力に表示します。

このオプションは、このコマンドの任意の形式とともに指定できます。

-v オプションと -o - オプションを同時に指定してはいけません。-v オプ ションは無視されます。-o - オプションは、ほかのすべての標準出力を抑制します。

-x *extension-property=value*
-x *extension-property+=array-value*
-x *extension-property-=array-value*
--extension-property=*extension-property=value*
--extension-property=*extension-property+=array-value*
--extension-property=*extension-property-=array-value*
--extension-property *extension-property=value*
--extension-property *extension-property+=array-value*
--extension-property *extension-property-=array-value*

コマンドに対するオペランドとして指定したリソースの拡張プロパティーの値を、設定または変更します。

一般に、-p オプションを使用すると、任意の標準プロパティーまたは拡張プロパティーを指定できます。あるリソースタイプの拡張プロパティーが同じリソースタイプの標準プロパティーと同じ名前を持っている場合、-p オプションを使用するとエラーが返されます。そのような場合、拡張プロパティーを指定するには -x オプション、標準プロパティーを指定するには -y オプションを使用します。

このオプションの代入形式を指定できるのは、create サブコマンドおよび set サブコマンドだけです。

リソースタイプの拡張プロパティーについては、リソースタイプに関するドキュメントを参照してください。

このオプションとともに使用する演算子は、次のとおりです。

=　　プロパティーに、指定した値を設定します。この演算子は、create サブコマンドおよび set サブコマンドで使用できます。

+=　　1 つまたは複数の値を文字列配列値に追加します。この演算子は、set サブコマンドでのみ使用できます。この演算子は、たとえば、Resource_dependencies のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。

-=

1 つまたは複数の値が、文字列配列値から削除されます。この演算子は、set サブコマンドでのみ使用できます。この演算子は、たとえば、Resource_dependencies のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。

クラスタノードのサブセットに対してノード単位の拡張プロパティーを設定するには、プロパティーの設定時にノードを指定します。次のように、中括弧で囲まれたノードのリストをプロパティー名のあとに付加します。

name{*node*}

node は、ノード名またはノード ID をコンマで区切ったリストです。ノード単位のプロパティーの詳細は、1335 ページのrt_properties(5) のマニュアルページを参照してください。

-x *extension-property*[,…]
--extension-property=*extension-property*[…]
--extension-property *extension-property*[,…]

list-props サブコマンドおよび show サブコマンドの拡張プロパティーのリストを指定します。

リソースタイプの拡張プロパティーについては、リソースタイプに関するドキュメントを参照してください。

-p オプションを使用すると、任意の標準プロパティーまたは拡張プロパティーを指定できます。あるリソースタイプの拡張プロパティーが同じリソースタイプの標準プロパティーと同じ名前を持っている場合、-p オプションを使用するとエラーが返されます。そのような場合、拡張プロパティーを指定するには -x オプション、標準プロパティーを指定するには -y オプションを使用します。

このオプションを指定しないと、list-props サブコマンドおよび show サブコマンドは、-v オプションも指定されているかどうかに基づいて、すべてまたはほとんどのリソースプロパティーを一覧表示します。

-y *standard-property=value*
-y *standard-property+=array-value*
-y *standard-property-=array-value*
--standard-property=*standard-property=value*
--standard-property=*standard-property+=array-value*
--standard-property=*standard-property-=array-value*
--standard-property *standard-property=value*
--standard-property *standard-property+=array-value*
--standard-property *standard-property-=array-value*

コマンドに対するオペランドとして指定したリソースの標準プロパティーの値を、設定または変更します。

-p オプションを使用すると、任意の標準プロパティーまたは拡張プロパティーを指定できます。あるリソースタイプの拡張プロパティーが同じリソースタイプの標準プロパティーと同じ名前を持っている場合、-p オプションを使用するとエラーが返されます。そのような場合、拡張プロパティーを指定するには -x オプション、標準プロパティーを指定するには -y オプションを使用します。

このオプションの代入形式を指定できるのは、create サブコマンドおよび set サブコマンドだけです。

標準プロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

このオプションとともに使用する演算子は、次のとおりです。

=
: プロパティーに、指定した値を設定します。この演算子は、create サブコマンドおよび set サブコマンドで使用できます。

+=
: 1 つまたは複数の値を文字列配列値に追加します。この演算子は、set サブコマンドでのみ使用できます。この演算子は、たとえば、Resource_dependencies のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。

-=
: 1 つまたは複数の値が、文字列配列値から削除されます。この演算子は、set サブコマンドでのみ使用できます。この演算子は、たとえば、Resource_dependencies のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。

-y *standard-property*[,…]
--standard-property=*standard-property*[,...]
--standard-property *standard-property*[,...]

list-props サブコマンドおよび show サブコマンドの標準プロパティーのリストを指定します。

標準プロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

-p オプションを使用すると、任意の標準プロパティーまたは拡張プロパティーを指定できます。あるリソースタイプの拡張プロパティーが同じリソースタイプの標準プロパティーと同じ名前を持っている場合、-p オプションを使用するとエラーが返されます。そのような場合、拡張プロパティーを指定するには -x オプション、標準プロパティーを指定するには -y オプションを使用します。

このオプションを指定しないと、list-props サブコマンドおよび show サブコマンドは、-v オプションも指定されているかどうかに基づいて、すべてまたはほとんどのリソースプロパティーを一覧表示します。

-Z {*zoneclustername* | global | all}
--zoneclustername={*zoneclustername* | global | all}
--zoneclustername {*zoneclustername* | global | all}

クラスタ、またはリソースが存在するクラスタや処理するクラスタを指定します。

このオプションは、export サブコマンド以外のすべてのサブコマンドでサポートされています。

このオプションを指定する場合は、次のいずれかの引数も指定する必要があります。

zoneclustername このオプションを使用するコマンドが、*zoneclustername* という名前のゾーンクラスタでのみ指定されたすべてのリソースで機能するように指定します。

`global` このオプションを使用するコマンドが、広域クラスタでのみ指定されたすべてのリソースで機能するように指定します。

`all` 広域クラスタでこの引数を使用する場合、それを使用するコマンドが広域クラスタとすべてのゾーンクラスタを含め、すべてのクラスタで指定されたすべてのリソースで機能するように指定します。

この引数をゾーンクラスタで使用すると、このオプションを使用するコマンドが、ゾーンクラスタ内だけで指定されたすべてのリソースに対して機能するように指定されます。

次のオペランドだけがサポートされています。

resource 管理対象のリソース (1 つまたは複数) を指定します。サブコマンドで複数のリソースを指定できる場合は、プラス記号 (+) を使用すると、すべてのリソースを指定できます。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (`CL_NOERR`) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

次の終了コードが返されます。

`0 CL_NOERR`

エラーなし

実行したコマンドは正常に終了しました。

`1 CL_ENOMEM`

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

`3 CL_EINVAL`

無効な引数

コマンドを間違って入力したか、`-i` オプションで指定したクラスタ構成情報の構文が間違っていました。

`6 CL_EACCESS`

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

9 CL_ESTATE

オブジェクトの状態が不正です

その時点で変更できない、または常時変更できないプロパティー、リソースグループ、またはその他のオブジェクトを変更しようとしました。

10 CL_EMETHOD

リソースのメソッドが失敗しました

リソースのメソッドが失敗しました。次のいずれかの理由のために、メソッドは失敗しました。

- `validate` メソッドは、リソースを作成しようとしたときに、あるいは、リソースのプロパティーを変更しようとしたときに失敗しました。
- `validate` 以外のメソッドは、リソースを有効、無効、または削除しようとしたときに失敗しました。

15 CL_EPROP

無効なプロパティーです

-p、-y、または -x オプションで指定したプロパティーまたは値が存在しないか、許可されていません。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

41 CL_ETYPE

無効なタイプです

-t または -p オプションで指定したタイプは存在しません。

これらの終了値は、1047 ページのscha_calls(3HA) のマニュアルページで説明されているリターンコードと互換性があります。

例 133 リソースの作成

この例では、rg-failover という名前のリソースグループ内で rs-nfs という名前のリソースを作成します。リソースは、リソースタイプ SUNW.nfs のインスタンスです。リソースは作成後に有効な状態になり、リソースモニタリングも有効になります。

```
# clresource create -g rg-failover -t SUNW.nfs rs-nfs
```

次の 2 つのコマンドのいずれかで、rg-failover という名前のリソースグループにあるゾーンクラスタ ZC 内に、rs-nfs という名前のリソースを作成します。これらのコマンドは、グローバルクラスタノードから、またはゾーンクラスタ ZC 内で実行できます。コマンドをゾーンクラスタから実行する場合、任意でゾーンクラスタ名を持つリソースのスコープを明示的に定義できます。

```
# clresource create -g rg-failover -t SUNW.nfs -Z ZC rs-nfs
```

```
# clresource create -g rg-failover -t SUNW.nfs ZC:rs-nfs
```

例 134 リソースモニタリングの有効化

この例では、rs-nfs という名前のリソースに対するモニタリングを有効にします。

```
# clresource monitor rs-nfs
```

リソースのモニタリングは、いったん有効になると、clresource unmonitor コマンドを使って明示的に無効にするまで、有効な状態のままとなります。リソースの無効化および有効化は、リソースがモニターされるかどうかに影響を与えません。

例 135 リソースの有効化

この例では、リソースグループ rg-failover および rg-failover2 の中のすべてのリソースを有効にします。

```
# clresource enable -g rg-failover,rg-failover2 +
```

このコマンドは、リソースがモニターされるかどうかに影響を与えません。

例 136 リソースプロパティーの設定

この例では、リソースタイプ SUNW.nfs の全インスタンスの r_description プロパティーを HA-NFS res に設定します。

```
# clresource set -t SUNW.nfs -p r_description="HA-NFS res" +
```

例 137 ノード単位リソースプロパティーの設定

この例では、リソース rs-oracle のノード単位プロパティー oracle_sid に、次のように別のノード上の別の値を設定します。

- ノード phys-schost-1 およびノード phys-schost-2 上で、このプロパティーには myora1 が設定されます。
- ノード phys-schost-3 上の場合、このプロパティーには myora2 が設定されます。

この例の場合、中括弧文字は、使用されるシェルに対して特別な意味があります。そのため、ノードリストが付加される各プロパティー名は、二重引用符で囲んであります。

```
# clresource set -p "oracle_sid{phys-schost-1,phys-schost-2}"=myora1 \
-p "oracle_sid{phys-schost-3}"=myora2 rs-oracle
```

例 138 ノード単位のリソース依存関係の設定

この例では、gds-rs のノード単位のリソース依存関係を設定して、それが 2 つの異なる論理ホストリソースに依存するようにします。

```
# clresource set -p resource_dependencies=node-3-rs@pnode1,node-4-rs@pnode2 gds-rs

# clresource show -p resource_dependencies gds-rs
Resource: gds-rs
Standard Properties:
Resource_dependencies: node-3-rs@pnode1,node-4-rs@pnode2
```

例 139 文字列配列プロパティーへの値の追加

この例では、値 rs-oracle を、リソース rs-myapp の文字列配列プロパティー resource_dependencies に追加します。この文字列配列プロパティー内の既存の値は、変更されません。

```
# clresource set -p resource_dependencies+=rs-oracle rs-myapp

# clresource show -p resource_dependencies rs-myapp
Resource: rs-myapp
Standard Properties:
Resource_dependencies: rs-nfs rs-oracle
```

例 140 リソースの削除

この例では、rs-nfs という名前のリソースを削除します。

```
# clresource delete rs-nfs
```

例 141 クラスタ構成全体の更新

この例では、次の順序で処理を実行することにより、クラスタ構成全体を更新します。

1. クラスタ内のリソースグループをすべてオフラインにしたあと、すべてのリソースを削除してから、すべてのリソースグループを削除します。
2. すべてのリソースタイプの登録解除
3. 構成ファイル /net/server/export/mycluster.xml の中で指定されているすべてのリソースの作成、そのリソースタイプの登録、および必要なすべてのリソースグループの作成

```
# clresourcegroup delete --force +
# clresourcetype unregister +
# clresource -i /net/server/export/mycluster.xml -a +
```

例 142 リソースの一覧表示

この例では、すべてのリソースを一覧表示します。

```
# clresource list
logicalhost1
rs-nfs-1
rs-nfs-2
logicalhost2
rs-apache-1
```

例 143 リソースおよびグループとタイプの一覧表示

この例では、すべてのリソースおよびそのリソースグループとリソースタイプを一覧表示します。

```
# clresource list -v

Resource Name      Resource Group    Resource Type
-------------      --------------    -------------
logicalhost1       rg-failover-1     SUNW.LogicalHostname
rs-nfs-1           rg-failover-1     SUNW.nfs
logicalhost2       rg-failover-2     SUNW.LogicalHostname
rs-nfs-2           rg-failover-2     SUNW.nfs
rs-apache-1        rg-failover-1     SUNW.apache
```

例 144 特定のタイプのリソースの一覧表示

この例では、リソースタイプ nfs のすべてのインスタンスを一覧表示します。

```
# clresource list -t nfs
rs-nfs-1
rs-nfs-2
```

例 145 リソースタイプの拡張プロパティーと説明の一覧表示

この例では、リソースタイプ nfs の拡張プロパティーと各拡張プロパティーの説明を一覧表示します。

```
# clresource list-props -t nfs -v
Properties              Descriptions
----------              ------------
Monitor_retry_count     Number of PMF restarts allowed for the fault monitor
Monitor_retry_interval  Time window (minutes) for fault monitor restarts
Rpcbind_nullrpc_timeout Timeout(seconds) to use when probing rpcbind
Nfsd_nullrpc_timeout    Timeout(seconds) to use when probing nfsd
Mountd_nullrpc_timeout  Timeout(seconds) to use when probing mountd
Statd_nullrpc_timeout   Timeout(seconds) to use when probing statd
Lockd_nullrpc_timeout   Timeout(seconds) to use when probing lockd
Rpcbind_nullrpc_reboot  Boolean to indicate if we should reboot system when
                          null rpc call on rpcbind fails
Nfsd_nullrpc_restart    Boolean to indicate if we should restart nfsd when
                          null rpc call fails
Mountd_nullrpc_restart  Boolean to indicate if we should restart mountd when
                          null rpc call fails
```

Line breaks in the Descriptions column are added to enhance the readability of this example. Actual output from the command does not contain these line breaks.

例 146 リソースの無効化および有効化によるリソース状態 Start_failed のクリア

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

Stop_failed リソース状態とは異なり、Start_failed リソース状態は、Oracle Solaris Cluster ソフトウェアがリソースグループ上で操作を実行することを*妨げません*。リソース状態

Start_failed は、*command* clear コマンドを発行しなくてもクリアすることができます。該当リソースを再起動するコマンドを実行するだけで済みます。

次のコマンドでは、リソース resource-1 上で発生したリソース状態 Start_failed を、リソースをいったん無効化したあと再度有効化することにより、クリアします。

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

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページのIntro(1CL), 251 ページのclreslogicalhostname(1CL), 305 ページのclresourcegroup(1CL), 333 ページのclresourcetype(1CL), 347 ページのclressharedaddress(1CL), 551 ページのcluster(1CL), 1047 ページのscha_calls(3HA), 1447 ページのclconfiguration(5CL), Unresolved link to " attributes5", 1287 ページのr_properties(5), Unresolved link to " rbac5"

Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド "

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のオプションを指定してこのコマンドを実行できます。

- -? オプション
- -V オプション

サブコマンドを指定してこのコマンドを実行する場合、スーパーユーザー以外のユーザーは RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
create	solaris.cluster.modify
delete	solaris.cluster.modify

サブコマンド	RBAC の承認
disable	solaris.cluster.admin
enable	solaris.cluster.admin
export	solaris.cluster.read
list	solaris.cluster.read
list-props	solaris.cluster.read
set	solaris.cluster.modify
monitor	solaris.cluster.admin
clear	solaris.cluster.admin
show	solaris.cluster.read
status	solaris.cluster.read
unmonitor	solaris.cluster.admin

名前

clresourcegroup, clrg — Oracle Solaris Cluster データサービスのリソースグループの管理

```
/usr/cluster/bin/clresourcegroup -V

/usr/cluster/bin/clresourcegroup [subcommand] -?

/usr/cluster/bin/clresourcegroup subcommand [options] -v [resourcegroup …]

/usr/cluster/bin/clresourcegroup add-node -n node[,...] [-S] [-Z {zoneclustername | global}]
 {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup create [-S] [-n node[,...]] [-p name=value] […] [-Z
 {zoneclustername | global}] {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup create -i {- | clconfigfile} [-S] [-n node [,...]] [-
p name=value] […] {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup delete [-F] [-Z {zoneclustername | global}] {+
 | resourcegroup...}

/usr/cluster/bin/clresourcegroup evacuate -n node[,...] [-T seconds] [-Z {zoneclustername
 | global}] {+}

/usr/cluster/bin/clresourcegroup export [-o {- | configfile}] [+ | resourcegroup...]

/usr/cluster/bin/clresourcegroup list [-n node[,...]] [-r resource[,...]] [-s state[,...]] [-
t resourcetype[,...]] [ [-Z {zoneclustername[,...] | global | all}] [+ | resourcegroup...]

/usr/cluster/bin/clresourcegroup manage [-Z {zoneclustername | global}] {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup offline [-n node [,...]] ] [-Z {zoneclustername | global}] {+
 | resourcegroup...}

/usr/cluster/bin/clresourcegroup online [-e] [-m] [-M] [-n node [,...]] [-Z {zoneclustername
 | global}] {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup quiesce [-k] [-Z {zoneclustername | global}] {+
 | resourcegroup...}

/usr/cluster/bin/clresourcegroup remaster [-Z {zoneclustername |
     global}] {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup remove-node -n node
     [,...]  [-Z {zoneclustername | global}]
     {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup restart [-n node[,...]]
     [-Z zoneclustername |global}] {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup resume [-Z {zoneclustername |
     global}] {+ | resourcegroup...}
```

```
/usr/cluster/bin/clresourcegroup set [-n node[,...]] -p name[+|-]=value […]
     [-Z {zoneclustername | global}] {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup show [-n node[,...]]
     [-p name[,...]] [-r resource[,...]] [-t resourcetype[,...]]
     [-Z {zoneclustername[,...] | global | all}]
     [+ | resourcegroup...]

/usr/cluster/bin/clresourcegroup status [-n node[,...]]
     [-r resource [,]...] [-s state [,]...] [-t resourcetype
     [,]...] [-Z {zoneclustername[,...] | global | all}]
     [+ | resourcegroup...]

/usr/cluster/bin/clresourcegroup suspend [-k] [-Z
     {zoneclustername | global}] {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup switch -n node[,...]
     [-e] [-m] [-M] [-Z {zoneclustername | global}]
     {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup unmanage [-Z {zoneclustername |
     global}] {+ | resourcegroup...}
```

このコマンドは、Oracle Solaris Cluster データサービスのリソースグループを管理します。

subcommand を省略できるのは、*options* が `-?` オプションまたは `-V` オプションの場合のみです。

このコマンドの各オプションには長形式と短形式があります。各オプションの両方の形式は、オプション に説明とともに記載されています。

`clrg` コマンドは、`clresourcegroup` コマンドの短い形式です。

`list`、`show`、および `status` 以外のサブコマンドでは、1 つ以上のオペランドが必要です。ただし、多くのサブコマンドはプラス記号のオペランド (+) を受け付けます。このオペランドは、サブコマンドを*すべての*適用できるオブジェクトに適用します。

このコマンドは、ゾーンクラスタでいくつかの形式で使用できます。このコマンドの有効な使用方法の詳細については、個々のサブコマンドの説明を参照してください。管理しやすいように、広域クラスタノードからこのコマンドを使用してください。

リソースとリソースグループ

リソースのステータス、リソースグループのステータス、リソースのステータスは、すべてノード単位で管理されます。たとえば、リソースは、各クラスタノード上で固有のステータス、固有のステータスを持ちます。

注記 - Offline や Start_failed などの状態の名前は、大文字と小文字が区別*されません*。状態の名前を指定する際には、大文字と小文字を任意に組み合わせることができます。

リソース状態は、そのリソース上でどのメソッドが呼び出されたかにのみ基づいて、各ノード上の Resource Group Manager (RGM) によって設定されます。たとえば、指定のノード上でリソースに対する STOP メソッドを正しく実行した場合、そのリソースのノード上での状態は Offline になります。STOP メソッドが 0 以外またはタイムアウトで終了した場合、そのリソースの状態は Stop_failed になります。

可能性のあるリソース状態には、Online、Offline、Start_failed、Stop_failed、Monitor_failed、Online_not_monitored、Starting、および Stopping があります。

可能性のあるリソースグループの状態には、Unmanaged、Online、Offline、Pending_online、Pending_offline、Error_stop_failed、Online_faulted、および Pending_online_blocked があります。

RGM は、リソースのステータスだけでなく、リソース自体が API を使って設定するリソースのステータスも維持します。Status Message のフィールドは、実際には、ステータスキーワードとステータスメッセージからなります。ステータスメッセージは、ステータスキーワードのあとに出力される任意のテキスト文字列で、リソースによって任意に設定されます。

リソースステータスの値には、次のものがあります。

Degraded	リソースはオンラインですが、何らかの理由でパフォーマンスまたは可用性が低下しています。
Faulted	リソースの機能を妨げるエラーが検出されました。
Offline	リソースはオフラインです。
Online	リソースはオンラインでサービスを提供します。
Unknown	現在のステータスは不明または遷移中です。

ゾーンクラスタでこのコマンドを使用する

clresourcegroup コマンドは、ゾーンクラスタで、export を除くすべてのサブコマンドを指定して使用することができます。

export 以外のすべてのサブコマンドで -Z オプションを使用して、操作を制限する特定のゾーンクラスタの名前を指定することもできます。また、リソースグループ (*zoneclustername*

:*resourcegroup*) にゾーンクラスタ名を付けて、操作を特定のゾーンクラスタに制限することもできます。

グローバルクラスタノードからすべてのゾーンクラスタ情報にアクセスできますが、特定のゾーンクラスタは他のゾーンクラスタを認識しません。特定のゾーンクラスタに操作を制限できない場合、使用するサブコマンドは現在のクラスタでのみ機能します。

ゾーンクラスタのリソースグループと別のゾーンクラスタまたはグローバルクラスタのリソースグループ間のアフィニティーを指定することができます。次のコマンドで別のゾーンクラスタのリソースグループ間のアフィニティーを指定することができます。

```
# clresourcegroup set -p RG_affinities={+|++|-|--}
target-zc:target-rg
source-zc:source-rg
```

アフィニティータイプは次のいずれかとなります。

- + (弱い肯定的)
- ++ (強い肯定的)
- +++ (フェイルオーバー委託付きの強い肯定的)
- - (弱い否定的)
- -- (強い否定的)

たとえば、ゾーンクラスタ ZC1 内のリソースグループ RG1 とゾーンクラスタ ZC2 内のリソースグループ RG2 の間に強い肯定的なアフィニティー (++) を指定する必要がある場合は、次のコマンドを使用します。

```
# clresourcegroup set -p RG_affinities=++ZC2:RG2 ZC1:RG1
```

ゾーンクラスタ ZC1 内のリソースグループ RG1 とゾーンクラスタ ZC2 内のリソースグループ RG2 の間にフェイルオーバー委託付きの強い肯定的なアフィニティー (+++) を指定するには、次のコマンドを使用します。

```
# clresourcegroup set -p RG_affinities=+++ZC2:RG2 ZC1:RG1
```

ゾーンクラスタ ZC1 内のリソースグループ RG1 とグローバルクラスタ内のリソースグループ RG2 の間に強い否定的なアフィニティー (--) を指定するには、次のコマンドを使用します。

```
# clresourcegroup set -p RG_affinities=--global:RG2 ZC1:RG1
```

リソースグループはクラスタノードまたはゾーンに自動的に分散されます。詳細は、1319 ページのrg_properties(5) のマニュアルページにある Load_factors、Priority、および Preemption_mode のエントリを参照してください。

サポートされるサブコマンドには次のものがあります。

add-node

ノードをリソースグループの Nodelist プロパティーの最後に追加します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

リスト内のノードとゾーンの順序は、それらのノードまたはゾーンでリソースグループがオンラインにされる優先順位を指定します。Nodelist プロパティー内の別の位置にノードを追加するには、set サブコマンドを使用します。

広域クラスタノードから特定のゾーンクラスタのリソースグループに対してノードを追加するために、-Z オプションを使用してゾーンクラスタの名前を指定できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

create

新しいリソースグループを作成します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

広域クラスタノードから特定のゾーンクラスタにリソースグループを作成するために、-Z オプションを使用してゾーンクラスタの名前を指定できます。

-i オプションで構成ファイルを指定する場合は、プラス記号のオペランド (+) を指定できます。このオペランドは、存在していないすべてのリソースをそのファイル内に作成することを指定します。

新しいリソースグループの Nodelist プロパティーを設定するには、次に示すオプションのいずれかを指定します。

- -n *node*
- -p Nodelist=-*node-*][,…]
- -i *clconfigfile*

リスト内のノードの順序は、それらのノードでリソースグループをオンラインにする優先順位を指定します。作成時にノードリストを指定しない場合、クラスタ内で構成されているすべてのノードが Nodelist プロパティーに設定されます。順序は任意です。

デフォルトでは、リソースグループは Failover が設定された RG_mode プロパティーで作成されます。ただし、-S オプションまたは -p RG_mode=Scalable オプションを使用するか、Maximum_primaries に 1 より大きい値を指定することで、スケーラブルリソースグループを作成できます。リソースグループの RG_mode プロパティーを設定できるのは、このグループを作成するときだけです。

リソースグループは、作成時には常に管理されない状態に置かれます。ただし、manage サブコマンドを実行したり、online または switch サブコマンドを -M> オプションとともに実行したりすると、RGM の状態は管理状態に変わります。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

delete

リソースグループを削除します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

広域クラスタノードから特定のゾーンクラスタのリソースグループを削除するために、-Z オプションを使用してゾーンクラスタの名前を指定できます。

このサブコマンドとともに正符号のオペランド (+) を指定すると、すべてのリソースグループを削除できます。

リソースグループがリソースを含む場合は、-F オプションを指定しないかぎり、そのリソースグループを削除できません。-F オプションを指定した場合は、各グループ内のすべてのリソースおよびそのグループが削除されます。すべての依存関係とアフィニティーも削除されます。

このサブコマンドは、リソースとリソースグループの依存関係を反映する順序で複数のリソースグループを削除します。コマンド行でリソースグループを指定する順序は関係ありません。

次の clresourcegroup delete コマンドはいくつかのステップで実行されます。

- 複数のリソースグループを同時に削除する場合
- リソースグループを -F オプションで削除する場合

たとえばノードで問題が発見された場合などのように、このいずれかのフォームのコマンドが割り込まれたときには、いくつかのリソースグループが無効な構成のまま残される場合があります。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

evacuate

-n オプションで指定したノード上のすべてのリソースグループをオフラインにします。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタノードから evacuate コマンドを実行すると、このサブコマンドはグローバルクラスタまたはゾーンクラスタのすべてのリソースグループを退避します。ゾーンクラスタで、このサブコマンドは指定したゾーンクラスタのリソースグループのみを退避します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを退避する場合は、-Z オプションを使用してゾーンクラスタの名前を指定できます。

リソースグループは、リソースとリソースグループの依存関係を反映する順序でオフラインにされます。

このサブコマンドの -T オプションを使用して、リソースグループが切り替わらないようにする秒数を指定できます。値を指定しない場合、デフォルトでは、60 秒が使用されます。

退避が完了したあと 60 秒間または指定した秒数の間、リソースグループは退避中ノードでフェイルオーバーできなかったり自動的にオンラインになれなくなったりします。

ただし、switch または online サブコマンドを使用して、リソースグループをオンラインまたは退避ノードリブートに切り替えると、退避タイマーはただちに期限切れになり、自動フェイルオーバーが再度許可されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`export`

リソースグループの構成情報をファイルまたは標準出力 (stdout) に書き込みます。

このサブコマンドは、大域ゾーンだけで使用できます。

この構成情報の形式は、1447 ページのclconfiguration(5CL) のマニュアルページで説明されています。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`list`

指定したリソースグループの修飾子オプションでフィルタリングされたリストを表示します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

-r *resource* を使用すると、リソースを含むリソースグループのみを含めることができます。-t *resourcetype* を使用すると、*resourcetype* 内のリソースタイプを含むリソースグループのみを含めることができます。-n *node* を使用すると、1 つ以上のノード内でオンラインであるリソースグループのみを含めることができます。

-s *state* を指定すると、指定した状態のグループのみが一覧表示されます。

オペランドを指定しない場合や正符号のオペランド (+) を指定した場合は、指定した任意の修飾子オプションでフィルタリングされたすべてのリソースグループが一覧表示されます。

詳細オプション -v を指定すると、リソースグループがオンラインかオフラインにかかわらず、そのステータスが表示されます。クラスタ内の 1 つのノードだけでオンラインの場合でも、リソースグループはオンラインとして一覧表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`manage`

指定したリソースグループを管理状態にします。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾーンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを管理するには、-Z オプションを使用してゾーンクラスタの名前を指定できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

offline

指定したリソースグループをオフライン状態にします。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾーンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを退避するには、-Z オプションを使用してゾーンクラスタの名前を指定できます。

-n オプションを指定すると、リソースグループは、指定したノードでのみオフラインになります。

-n オプションを指定しないと、リソースグループはすべてのノード上でオフラインになります。

リソースグループを offline サブコマンドでオフラインにした場合、リソースグループの Offline 状態はノードのリブートで解除されます。つまり、ノードを終了またはクラスタに接続すると、以前にオフラインにしたリソースグループが一部のノード上でオンラインになる可能性があります。すべてのリソースを無効にしていても、リソースグループはオンラインになります。

同様に、任意の RG_dependencies または強い RG_affinities を示すリソースグループは、別のリソースグループに切り替わると、自動的のオンラインになります。

リソースグループが自動的にオンラインになるのを防止するには、suspend サブコマンドを使用してリソースグループの自動復旧処理を中断します。自動復旧処理を再開するには、resume サブコマンドを使用します。

リソースグループは、リソースとリソースグループの依存関係を反映する順序でオフラインにされます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

online

指定したリソースグループをオンライン状態にします。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾー

ンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを退避するには、-Z オプションを使用してゾーンクラスタの名前を指定できます。

リソースグループをオンラインにするノードのリストを指定するには、-n オプションを使用します。-n オプションを指定しない場合、このサブコマンドは、リソースグループを現在のプライマリノードからオフラインにせずに、リソースグループを最優先ノードでオンラインにします。リソースグループごとのオンラインノードの総数は、Desired_primaries および Maximum_primaries プロパティーによって制限されます。ノードの優先順序は、Nodelist、RG_affinities、および Load_factors プロパティーによって決定されます。これらのプロパティーの詳細は、1319 ページのrg_properties(5) のマニュアルページを参照してください。

コマンド行で複数のリソースグループのオペランドが指定され、-n オプションが指定されていない場合、それらのリソースグループのオペランドには Priority プロパティーによって決定される順序でプライマリノードが割り当てられます。つまり、優先順位のもっとも高いリソースグループが最初にノードの割り当てを受けます。プライマリノードが割り当てられたあと、リソースの依存関係またはリソースグループの依存関係によって制約されている場合を除き、すべてのリソースグループオペランドが並列でオンラインになります。コマンド行でリソースグループを指定する順序は関係ありません。Priority プロパティーの詳細は、1319 ページのrg_properties(5) のマニュアルページを参照してください。

優先順位の低いリソースグループは、最優先ノードに割り当てられなかったり、負荷制限を超えている場合は優先順位の高いリソースグループによって強制的にオフラインにされたりすることがあります。詳細は、185 ページのclnode(1CL) のマニュアルページの loadlimit サブコマンドを参照してください。

switch サブコマンドと異なり、このサブコマンドは Nodelist プロパティーで一覧表示されているどのノードも Offline 状態にしようとしません。

このサブコマンドとともに -e オプションを指定すると、オンラインにされるリソースグループのセット内のすべてのリソースが有効になります。

-m オプションを指定すると、オンラインにされるリソースグループのセット内のすべてのリソースのモニタリングを有効にできます。ただし、最初に有効にされ、MONITOR_START メソッドと関連付けられている場合以外、リソースが実際にモニターされることはありません。

-M オプションを指定して、オンラインにされるすべてのリソースグループを管理状態に置くように指示することもできます。-M オプションが指定されていない場合、このサブコマンドは、管理されていないリソースグループには効果を持ちません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

quiesce

指定されたリソースグループを休止状態にします。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾーンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを退避するには、-Z オプションを使用してゾーンクラスタの名前を指定できます。

このコマンドは、START または STOP メソッドが失敗した場合に、あるノードから別のノードにリソースグループが連続して切り替わるのを阻止します。また、停止メソッドが失敗したとき、リソースの Failover_mode プロパティーが HARD に設定されている場合に通常実行されるノードのリブートも阻止します。その場合、リソースは代わりに STOP_FAILED 状態に移動します。

影響されるリソースグループ内のリソースの代わりに実行されているメソッドを終了するには、-k オプションを使用します。-k オプションを指定しないと、終了するか構成されているタイムアウトを超えるまで継続的にメソッドを実行できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

remaster

指定したリソースグループを、現在のプライマリノードから最優先ノードに切り替えます。各リソースグループのオンラインノードの合計数は、 Desired_primaries および Maximum_primaries プロパティーによって制限されます。ノードの優先順序は、Nodelist、RG_affinities、および Load_factors プロパティーによって決定されます。詳細は、185 ページのclnode(1CL) および 1319 ページのrg_properties(5) のマニュアルページを参照してください。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾーンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを退避するには、-Z オプションを使用してゾーンクラスタの名前を指定できます。

online サブコマンドと異なり、このサブコマンドでは、リソースグループを現在のマスターからオフラインに切り替えて、より優先されるマスター上でオンラインにすることができます。

コマンド行で複数のリソースグループオペランドを指定すると、Priority プロパティーで決定されている順序でリソースグループオペランドがプライマリノードに割り当てられます。優先順位がもっとも高いリソースグループに最初にノードが割り当てられます。コマンド行でリソースグループを指定する順序は関係ありません。詳細は、1319 ページのrg_properties(5) のマニュアルページを参照してください。

優先順位の低いリソースグループは、最優先ノードに割り当てられなかったり、負荷制限を超えている場合は優先順位の高いリソースグループによって強制的にオフラインにされたりすることがあります。詳細は、185 ページのclnode(1CL) のマニュアルページの loadlimit サブコマンドを参照してください。

このサブコマンドは、管理されていないリソースグループには影響を与えません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

remove-node

リソースグループの Nodelist プロパティーからノードを削除します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドは、グローバルクラスタノードまたはゾーンクラスタから使用できます。グローバルクラスタノードからゾーンクラスタ内のリソースグループのノードを削除するには、-z オプションを使用してゾーンクラスタの名前を指定できます。

ノードを削除すると、remove-node は Maximum_primaries または Desired_primaries プロパティーの値を Nodelist プロパティー内の新しいノード数にリセットする場合があります。remove-node が Maximum_primaries または Desired_primaries プロパティーの値をリセットするのは、いずれかの値が Nodelist プロパティー内の新しいノード数を超える場合のみです。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

restart

現在リソースグループをホストしているプライマリノードの同じセット上でリソースグループをオフラインにしてからオンラインに戻します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾーンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを退避するには、-z オプションを使用してゾーンクラスタの名前を指定できます。

-n オプションを指定すると、リソースグループは、指定するノードのリストにある現在のマスターでのみ再起動されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

resume

suspend サブコマンドによって以前中断されていた指定のリソースグループで自動復旧処理を再開します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾーンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定

のゾーンクラスタのリソースグループを退避するには、-Z オプションを使用してゾーンクラスタの名前を指定できます。

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

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.admin` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`set`

指定するリソースグループに関連付けられているプロパティーを変更します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾーンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを退避するには、-Z オプションを使用してゾーンクラスタの名前を指定できます。

`Nodelist` プロパティーは、`-p Nodelist=`*node* で、または便宜的に `-n` *node* で変更できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`show`

指定したリソースグループの修飾子オプションでフィルタリングされた構成レポートを生成します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

-r*resource* を使用すると、リソースを含むリソースグループのみを含めることができます。-t*resourcetype* を使用すると、*resourcetype* 内のリソースタイプを含むリソースグループのみを含めることができます。`-n` *node* を使用すると、1 つ以上のノード内でオンラインであるリソースグループのみを含めることができます。広域クラスタから -Z オプションを使用すると、指定されたゾーンクラスタでオンラインになるこれらのリソースグループのみを含めることができます。

-p オプションを使用すると、すべてのリソースグループプロパティーではなく、選択されたリソースグループプロパティーのセットを表示できます。

オペランドを指定しない場合や正符号のオペランド (+) を指定した場合は、指定した任意の修飾子オプションでフィルタリングされたすべてのリソースグループが一覧表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`status`

指定したリソースグループの修飾子オプションでフィルタリングされたステータスレポートを生成します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

ゾーンクラスタでこのコマンドを使用すると、このサブコマンドはゾーンクラスタのリソースグループにのみ適用されます。

`-r` *resource* を使用すると、リソースを含むリソースグループのみを含めることができます。`-t` *resourcetype* を使用すると、*resourcetype* 内のリソースタイプを含むリソースグループのみを含めることができます。`-n` *node* を使用すると、1 つ以上のノード内でオンラインであるリソースグループのみを含めることができます。`-Z` オプションを使用すると、広域クラスタノードからゾーンクラスタを指定して、指定されたゾーンクラスタでオンラインになるこれらのリソースグループのみを含めることができます。

`-s` *state* を指定すると、指定した状態のグループのみが一覧表示されます。

注記 - `status` サブコマンドでは、`-n` オプションまたは `-s` オプションのいずれかを指定できます。ただし、両方のオプションを同時に `status` サブコマンドとともに指定できません。

オペランドを指定しない場合や正符号のオペランド (+) を指定した場合は、指定した任意の修飾子オプションでフィルタリングされたすべてのリソースグループが一覧表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`suspend`

指定されたリソースグループ上の自動復旧アクションを中断し、そのリソースグループを休止します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾーンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを退避するには、`-Z` オプションを使用してゾーンクラスタの名前を指定できます。

自動復旧を再開するコマンドを明示的に実行するまで、*中断されたリソースグループが*自動的に再開またはフェイルオーバーされることはありません。オンラインかオフラインかにかかわらず、中断されたデータサービスは現在の状態のままです。リソースグループが一時停止されている間、`clresourcegroup`(1CL) または `clresource`(1CL) コマンドを `switch`、`online`、`offline`、`disable`、`enable` などのサブコマンドとともに使用すると、リ

ソースグループまたはそのリソースを特定のノード上の別の状態に手動で切り替えることができます。アプリケーションプロセスの強制終了やアプリケーション固有のコマンドの実行など、リソースに対して直接に操作するのではなく、clresourcegroup(1CL) または clresource(1CL) コマンドを使用してください。これにより、クラスタのフレームワークがリソースおよびリソースグループの現在のステータスを正確に把握することができ、resume サブコマンドが実行されたときに可用性が正しく復元されます。

リソースグループの自動回復を中断する必要があるのは、クラスタの問題を調査して修正する場合、あるいは、リソースグループサービスに保守を実行する場合などです。

また、-k オプションを指定すると、影響されるリソースグループ内のリソースの代わりに実行されているメソッドを即座に終了できます。-k オプションを使用することで、リソースグループの休止を早めることができます。-k オプションを指定しない場合、メソッドは終了するか、または構成されているタイムアウトを超えるまで実行を継続することを許可されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

switch

指定したリソースグループをマスターするノードまたはノードのセットを変更します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾーンクラスタのリソースグループでのみ正常に動作します。

リソースグループをオンラインにするノードのリストを指定するには、-n オプションを使用します。-Z オプションを使用すると、広域クラスタノードからゾーンクラスタを指定して、指定されたゾーンクラスタのリソースグループのリストのみを含めることができます。

リソースグループがまだオンラインでない場合は、-n オプションで指定されているノードのセット上でオンラインになります。ただし、これらのグループが新しいノード上でオンラインになる前に、-n オプションで指定されていないノード上でオンラインのグループがオフラインになります。

このサブコマンドとともに -e を指定すると、オンラインにされるリソースグループのセット内のすべてのリソースが有効になります。

-m を指定すると、オンラインにされるリソースグループのセット内のすべてのリソースのモニタリングを有効にできます。ただし、最初に有効にされ、MONITOR_START メソッドと関連付けられている場合以外、リソースが実際にモニターされることはありません。

-M オプションを指定して、オンラインにされるすべてのリソースグループを管理状態に置くように指示することもできます。-M オプションが指定されていない場合、このサブコマンドは、管理されていないリソースグループには効果を持ちません。

リソースグループは、リソースとリソースグループの依存関係を反映する順序でオンラインにされます。コマンド行でグループを指定する順序は関係ありません。

優先順位の低いリソースグループは、指定されたノードに切り替えられなかったり、負荷制限を超えている場合は優先順位の高いリソースグループによって強制的にオフラインに

されたりすることがあります。詳細は、185 ページのclnode(1CL) のマニュアルページの loadlimit サブコマンドを参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

unmanage

指定したリソースグループを管理されない状態にします。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、同じゾーンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを退避するには、-Z オプションを使用してゾーンクラスタの名前を指定できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

次のオプションがサポートされています。

注記 - このセクションでは、各オプションの短い形式と長い形式の両方が示されています。

-?
--help

ヘルプ情報を表示します。

このオプションは*サブコマンド*付きでもサブコマンドなしでも指定できます。

このオプションを *subcommand* なしで指定すると、使用可能なすべてのサブコマンドのリストが表示されます。

このオプションを *subcommand* とともに指定すると、*subcommand* の使用法が表示されます。

このオプションを create または set サブコマンドで指定した場合は、すべてのリソースグループプロパティーのヘルプ情報が表示されます。

このオプションをほかのオプション、サブコマンド、またはオペランドと一緒に指定する場合、これらはすべて無視されます。ほかの処理は行われません。

-e
--enable

グループがオンラインにされる時点でリソースグループ内のすべてのリソースを有効にします。

このオプションは switch および online サブコマンドとだけ使用できます。

`-F`
`--force`

リソースが有効またはオンラインであっても、リソースグループおよびそのすべてのリソースを強制的に削除します。また、このオプションは、ほかのリソースおよびほかのリソースグループ内のすべての依存関係プロパティー設定またはアフィニティープロパティー設定からもリソースとリソースグループの両方を削除します。

`-F` オプションを `delete` サブコマンドとともに使用する場合は、注意が必要です。依存関係やアフィニティーが設定されている場合などは、強制削除により、削除されたリソースグループを参照するその他のリソースグループが変更される場合があります。強制削除後、依存リソースが無効またはエラー状態のまま残されることもあります。このような現象が発生した場合は、影響を受ける依存リソースを再構成または再起動する必要がある場合があります。

`-i` {- | *clconfigfile*}
`--input=`{- | *clconfigfile*}
`--input` {- | *clconfigfile*}

clconfigfile ファイル内にある構成情報を使用することを指定します。1447 ページの`clconfiguration`(5CL) のマニュアルページを参照してください。

標準入力 stdin を通して構成情報を提供するには、このオプションとともにダッシュ `-` を指定します。

ほかのオプションを指定した場合は、*clconfigfile* 内のオプションや情報より指定したオプションが優先されます。

指定したリソースグループだけがこのオプションの影響を受けます。

`-k`
`--kill`

指定したリソースグループ内のリソースの代わりに実行されている RGM のリソースメソッドを終了します。

このオプションは、`quiesce` および `suspend` サブコマンドで使用できます。`-k` オプションを指定しないと、終了するか構成されているタイムアウトを超えるまで継続的にメソッドを実行できます。

`-m`
`--monitor`

リソースグループがオンラインにされる時点でリソースグループ内のすべてのリソースのモニタリングを有効にします。

ただし、最初に有効にされ、`MONITOR_START` メソッドと関連付けられている場合以外、リソースが実際にモニターされることはありません。

このオプションは `switch` および `online` サブコマンドとだけ使用できます。

-M
--manage

サブコマンドによってオンラインにされるすべてのリソースグループが管理状態となることを指定します。

-n *node*[,…]]
--node=*node*[,…]]
--node *node*[,…]]

ターゲットのグローバルクラスタまたはゾーンクラスタに、ノードまたはノードのリストを指定します。-Z オプションが指定されている場合は、-n オプションで、グローバルクラスタホスト名ではなくゾーンクラスタホスト名のみを指定できます。-Z オプションが指定されていない場合は、-n オプションでグローバルクラスタホスト名のみを指定できます。

node には、ノードの名前または識別子を指定できます。

list、show、および status サブコマンドとともに使用されると、このオプションは出力を限定します。ノードリスト内の 1 つ以上のノードで現在オンラインのリソースグループのみが対象となります。

このオプションを create、add-node、remove-node、および set サブコマンドで指定することは、Nodelist プロパティーを設定することと同じです。Nodelist プロパティー内のノードの順序は、それらのノードでグループがオンラインになる順序を指定します。create サブコマンドでノードリストを指定しない場合は、クラスタ内のすべてのノードが Nodelist プロパティーに設定されます。順序は任意です。

switch および online サブコマンドとともに使用するときは、このオプションによりリソースグループをオンラインにするノードを指定します。

evacuate および offline サブコマンドとともに使用するときは、このオプションによりリソースグループをオフラインにするノードを指定します。

restart サブコマンドとともに使用するときは、このオプションによりリソースグループを再起動するノードを指定します。リソースグループは、指定されたリストにある現在のマスターで再起動されます。

-o {- | *clconfigfile*}
--output={- | *clconfigfile*}
--output {- | *clconfigfile*}

リソースグループの構成情報をファイルまたは標準出力 stdout に書き込みます。この構成情報の形式は、1447 ページのclconfiguration(5CL) のマニュアルページで説明されています。

このオプションにファイル名を指定する場合、このオプションは新しいファイルを作成します。次に、ノード構成情報はそのファイルに格納されます。- をこのオプションとともに指定すると、構成情報は標準出力 (stdout) に送信されます。このコマンドのほかの標準出力はすべて抑制されます。

このオプションを一緒に指定できるのは、export サブコマンドだけです。

-p *name*
--property=*name*
--property *name*

リソースグループのプロパティーのリストを指定します。

このオプションは、show サブコマンドとともに使用します。

create または set サブコマンドで設定または変更できるプロパティーについては、-p*name=value* オプションの説明を参照してください。

このオプションを指定しないと、show サブコマンドはほとんどのリソースグループプロパティーを一覧表示します。このオプションを指定せず、-verbose オプションを show サブコマンドで指定した場合、このサブコマンドはすべてのリソースグループプロパティーを一覧表示します。

指定できるリソースグループプロパティーは、Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド のリソースグループプロパティー"に記載されています。

-p *name=value*
-p *name+=array-values*
-p *name=array-values*
--property=*name=value*
--property=*name+=array-values*
--property=*name-=array-values*
--property *name=value*
--property *name+=array-values*
--property *name-=array-values*

リソースグループプロパティーの値を設定または変更します。

このオプションは create および set サブコマンドとだけ使用できます。

show サブコマンドで情報を表示できるプロパティーについては、-p *name* オプションの説明を参照してください。

-p は複数のインスタンスが許可されます。

このオプションとともに使用する演算子は、次のとおりです。

=	プロパティーに、指定した値を設定します。create および set サブコマンドがこの演算子を受け付けます。
+=	1 つ以上の値をプロパティー値のリストに追加します。この演算子は、set サブコマンドでのみ使用できます。この演算子は、たとえば、Nodelist のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。
–=	1 つ以上の値をプロパティー値のリストから削除します。この演算子は、set サブコマンドでのみ使用できます。この演算子は、たとえば、Nodelist のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。

-r *resource*[,…]
--resource=*resource*[,…]
--resource *resource*[,…]

リソースまたはリソースのリストを指定します。

このオプションは、list、show、および status サブコマンドでのみ使用できます。このオプションは、これらのコマンドからの出力を制限します。リソースリスト内に 1 つ以上のリソースを含むリソースグループだけが出力されます。

-s *state*[,…]
--state=*state*[,…]
--state *state*[,…]

リソースグループの状態またはリソースグループの状態のリストを指定します。

このオプションを一緒に指定できるのは、evacuate サブコマンドだけです。このオプションは、指定した任意のノード上で指定した状態にあるリソースグループのみが表示されるように、出力を制限します。このオプションとともに次に示す引数を 1 つ以上指定できます。引数は、状態を表します。

Error_stop_failed

指定した任意のノード上で Error_stop_failed 状態のリソースグループがすべて表示されます。

Not_online

指定した任意のノード上で online 以外の状態の指定したリソースグループがすべて表示されます。

Offline

指定したすべてのノードで Offline 状態の場合にのみ、指定したリソースグループが表示されます。

Online

指定した任意のノード上で Online 状態の指定したリソースグループがすべて表示されます。

Online_faulted

指定した任意のノード上で Online_faulted 状態のリソースグループがすべて表示されます。

Pending_offline

指定した任意のノード上で Pending_offline 状態のリソースグループがすべて表示されます。

Pending_online

指定した任意のノード上で Pending_online 状態のリソースグループがすべて表示されます。

Pending_online_blocked

指定した任意のノード上で Pending_online_blocked 状態のリソースグループがすべて表示されます。

Unmanaged

指定した任意のノード上で Unmanaged 状態の指定したリソースグループがすべて表示されます。

-S
--scalable

スケーラブルリソースグループを作成するか、または Maximum_primaries および Desired_primaries プロパティーを更新します。

このオプションは create および add-nod サブコマドとだけ使用できます。

create サブコマンドと使用された場合、このオプションはフェイルオーバーリソースグループではなくスケーラブルリソースグループを作成します。また、このオプションは Maximum_primaries および Desired_primaries の両方のプロパティーを、結果となる Nodelist プロパティー内のノードの数に設定します。

このオプションを add-node サブコマンドと使用できるのは、リソースグループがすでにスケーラブルな場合だけです。add-node サブコマンドとともに使用した場合、このオプションは Maximum_primaries および Desired_primaries の両方のプロパティーを、結果となる Nodelist プロパティー内のノードの数に更新します。

また、RG_mode、Maximum_primaries、および Desired_primaries プロパティーを -p オプションで設定することもできます。

-t *resourcetype*[,…]
--type=*resourcetype*[,…]
--type *resourcetype*[,…]

1 つのリソースタイプまたはリソースタイプのリストを指定します。

このオプションは、list、show、および status サブコマンドでのみ使用できます。このオプションは、これらのコマンドからの出力を制限します。リソースタイプリストに含まれているタイプのリソースを 1 つ以上含むリソースグループだけが出力されます。

[*prefix* としてリソースタイプを指定しています。]*type*[:*RT-version*]. たとえば、nfs リソースタイプは、SUNW.nfs:3.2、SUNW.nfs、または nfs として表される可能性があります。*RT-version* を含める必要があるのは、そのクラスタで登録されているリソースタイプの複数のバージョンが存在する場合だけです。*prefix* を含めない場合は、SUNW が使用されます。

-T *seconds*
--time=*seconds*
--time *seconds*

あるノードからリソースグループを退避したあと、そのノードにリソースグループがスイッチバックしないようにする時間を秒数で指定します。

このオプションを一緒に指定できるのは、evacuate サブコマンドだけです。*seconds* には、0 から 65535 までの整数値を指定する必要があります。値を指定しない場合、デフォルトでは、60 秒が使用されます。

退避が完了したあと 60 秒間または指定した秒数退避ノードになっていると、、リソースグループはフェイルオーバーできなかったり、自動的にオンラインになったりします。

ただし、switch または online サブコマンドを使用してリソースグループをオンライン、または退避されたノードリブートに切り替えると、退避タイマーはただちに期限切れになり、自動フェイルオーバーが再度許可されます。

-T オプションは、退避の完了後 *T* 秒間、退避したノードの RGM によってリソースグループがオンラインにならないことを指定します。-n オプション付きで switch または online サブコマンドを使用することで退避されたノードにリソースグループを切り替えることによって、-T タイマーをオーバーライドできます。そのような切り替えが完了すると、-T タイマーはただちにそのノードに対して期限切れになります。ただし、-n フラグのない online または remaster などのスイッチオーバーコマンドは、-T タイマーを引き続き認識し、任意のリソースグループが退避されたノードに切り替わるのを避けます。

-u

+ オペランドを使用する場合、このオプションは、リソースグループが中断されたリソース上でコマンドが機能するように指定します。

+ オペランドを指定するとき -u オプションを指定しないと、コマンドは停止されたすべてのリソースグループを無視します。-u オプションは、+ オペランドが add-node、manage、offline、online、quiesce、remaster、remove-node、restart、set、switch、または unamanage サブコマンドで指定されている場合に有効です。

+ オペランドを add-node、manage、offline、online、quiesce、remaster、remove-node、restart、set、switch、または unamanage サブコマンドで使用した場合は、-u オプションも指定しないかぎり、このコマンドは中断されたすべてのリソースグループを無視します。

-v
--verbose

詳細情報を標準出力 (stdout) で表示します。

-V
--version

コマンドのバージョンを表示します。

このオプションをほかのオプション、サブコマンド、またはオペランドと一緒に指定する場合、これらはすべて無視されます。コマンドのバージョンだけが表示されます。ほかの処理は行われません。

-Z {*zoneclustername* | global | all}
--zoneclustername={*zoneclustername* | global | all}
--zoneclustername {*zoneclustername* | global | all}

リソースグループが存在し、操作する 1 つまたは複数のクラスタを指定します。

このオプションは、export サブコマンドを除くすべてのサブコマンドでサポートされています。

このオプションを指定する場合は、次のいずれかの引数も指定する必要があります。

zoneclustername このオプションを使用するコマンドが、*zoneclustername* という名前のゾーンクラスタでのみ指定されたすべてのリソースグループで作動するように指定します。

`global` このオプションを使用するコマンドが、広域クラスタでのみ指定されたすべてのリソースグループで作動するように指定します。

`all` 広域クラスタでこの引数を使用すると、その引数とともに使用するコマンドが、広域クラスタおよびすべてのゾーンクラスタを含め、すべてのクラスタで指定されたすべてのリソースグループで作動するように指定します。

ゾーンクラスタでこの引数を使用すると、その引数とともに使用するコマンドがそのゾーンクラスタでのみ指定されたすべてのリソースグループで作動するように指定します。

次のオペランドがサポートされています。

resourcegroup 管理するリソースグループの名前。

`+` すべてのリソースグループ。

このコマンドセットにあるすべてのコマンドの終了ステータスコードの完全なリストについては、19 ページの`Intro`(1CL) のマニュアルページを参照してください。返される終了コードも1047 ページの`scha_calls`(3HA) のマニュアルページで説明されているリターンコードに準拠しています。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (`CL_NOERR`) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

`0 CL_NOERR`

エラーなし

実行したコマンドは正常に終了しました。

`1 CL_ENOMEM`

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

35 CL_EIO

I/O エラー

物理的な入出力エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

38 CL_EBUSY

オブジェクトはビジーです

アクティブなクラスタノードへの最後のクラスタインターコネクトパスからケーブルを取り外そうとしました。または、参照を削除していないクラスタ構成からノードを削除しようとしました。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

例 147 新しいフェイルオーバーリソースグループの作成

次の例の 1 番目のコマンドは、フェイルオーバーリソースグループ rg1 と rg2 を作成します。2 番目のコマンドは、構成ファイル cluster-1.xml に含まれているリソースをこれらのリソースグループに追加します。

```
# clresourcegroup create rg1 rg2
# clresource create -g rg1,rg2 -i /net/server/export/cluster-1.xml +
```

次の 2 つの例はどちらも、グローバルクラスタノードからゾーンクラスタ ZC 内にフェイルオーバーリソースグループ rg1 および rg2 を作成します。

```
# clresourcegroup create -Z ZC rg1 rg2
# clresourcegroup create ZC:rg1 ZC:rg2
```

例 148 すべてのリソースグループのオンライン化

次のコマンドは、すべてのリソースを有効かつモニターされている状態にし、すべてのリソースグループをオンラインにします。

```
# clresourcegroup online -eM +
```

例 149 Nodelist プロパティーへのノードの追加

次のコマンドは、ノード phys-schost-4 をすべてのリソースグループの Nodelist プロパティーに追加します。

```
# clresourcegroup set -p Nodelist+=phys-schost-4 +
```

例 150 すべてのリソースグループのノードからの退避

次のコマンドは、ノード phys-schost-3 からすべてのリソースグループを退避します。

```
# clresourcegroup evacuate -n phys-schost-3 +
```

例 151 すべてのノード上でのリソースグループのオフライン化

次のコマンドは、すべてのノード上でリソースグループ rg1をオフラインにします。

```
# clresourcegroup offline rg1
```

例 152 リソースグループ、マネージャーの構成全体のリフレッシュ

次の例の 1 番目のコマンドは、すべてのリソースおよびリソースグループを、それらが有効でオンラインの場合でも削除します。2 番目のコマンドは、すべてのリソースタイプの登録を解除します。3 番目のコマンドは、構成ファイル `cluster-1.xml` に含まれているリソースを作成します。また、3 番目のコマンドは、リソースのリソースタイプを登録し、そのリソースタイプが依存するすべてのリソースグループを作成します。

```
# clresourcegroup delete --force +
# clresourcetype unregister +
# clresource -i /net/server/export/cluster-1.xml -d +
```

例 153 すべてのリソースグループの一覧表示

次のコマンドはすべてのリソースグループを一覧表示します。

```
# clresourcegroup list
rg1
rg2
```

例 154 すべてのリソースグループとそのリソースの一覧表示

次のコマンドは、すべてのリソースグループをそのリソースとともに一覧表示します。`rg3` にはリソースはありません。

```
# clresourcegroup list -v
Resource Group Resource
-------------- --------
rg1            rs-2
rg1            rs-3
rg1            rs-4
rg1            rs-5
rg2            rs-1
rg3            -
```

例 155 特定のリソースを含むすべてのリソースグループの一覧表示

次のコマンドは、Oracle Solaris Cluster HA for NFS のリソースを含むすべてのグループを一覧表示します。

```
# clresource list -t nfs
rg1
```

例 156 リソースグループのスイッチオーバーによる Start_failed リソース状態のクリア

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

Stop_failed リソース状態とは異なり、Start_failed リソース状態は、ユーザーまたは Oracle Solaris Cluster ソフトウェアがリソースグループに対してアクションを実行することを妨げ*ません*。reset サブコマンドを実行して Start_failed リソース状態を解除する必要はありません。該当リソースを再起動するコマンドを実行するだけで済みます。

次のコマンドは、resource-grp-2 リソースグループ内のリソース上で発生した Start_failed リソース状態を解除します。このコマンドは、リソースグループを schost-2 ノードに切り替えることで、この状態を解除します。

```
# clresourcegroup switch -n schost-2 resource-grp-2
```

例 157 リソースグループの再起動による Start_failed リソース状態のクリア

次のコマンドは、resource-grp-2 リソースグループ内のリソース上で発生した Start_failed リソース状態を解除します。このコマンドは、元々リソースグループをホストしていた schost-1 ノード上でリソースグループを再起動することで、この状態を解除します。

```
# clresourcegroup restart resource-grp-2
```

例 158 load_factors プロパティーを設定する

次のコマンドは、2 つのリソースグループの負荷係数を設定します。

```
# clresourcegroup set -p load_factors=factor1@50,factor2@1 rg1 rg2
```

次のコマンドは、グローバルクラスタから、ゾーンクラスタ内の 2 つのリソースグループの負荷係数を設定します。

```
# clresourcegroup set -Z ZC load_factors=factor1@50,factor2@1 rg1 rg2
```

例 159 リソースグループの priority プロパティーを設定する

次のコマンドは、リソースグループの優先順位を設定します。

```
# clresourcegroup set -p priority=600 rg1
```

rg1 リソースグループは、ノード割り当てに関して、優先順位の低いリソースグループよりも優先されます。rg1 は、強い制限値を超えているノードにおいて、優先順位の低いほかのリソースグループを横取りできます。rg1 の優先順位が別のリソースグループの優先順位と比べて 100 以上高い場合、弱い制限値を超えているノードにおいて、そのリソースグループを横取りできます。`priority` のデフォルト値は *500* です。

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	`ha-cluster/system/core`
インタフェースの安定性	発展中

273 ページの`clresource`(1CL), 333 ページの`clresourcetype`(1CL), 551 ページの`cluster`(1CL), 19 ページの`Intro`(1CL), Unresolved link to " su1M", 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5", Unresolved link to " rbac5", 1319 ページの`rg_properties`(5), 1447 ページの`clconfiguration`(5CL)

スーパーユーザーはこのコマンドのすべての形式を実行できます。

すべてのユーザーがこのコマンドに `-?` (ヘルプ) オプションまたは `-V` (バージョン) オプションを指定して実行できます。

`clresourcegroup` コマンドをほかのサブコマンドとともに実行する場合、スーパーユーザー以外のユーザーは RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
`add-node`	`solaris.cluster.modify`
`create`	`solaris.cluster.modify`
`delete`	`solaris.cluster.modify`
`evacuate`	`solaris.cluster.admin`
`export`	`solaris.cluster.read`
`list`	`solaris.cluster.read`
`manage`	`solaris.cluster.admin`

サブコマンド	RBAC の承認
offline	solaris.cluster.admin
online	solaris.cluster.admin
quiesce	solaris.cluster.admin
remaster	solaris.cluster.admin
remove-node	solaris.cluster.modify
restart	solaris.cluster.admin
resume	solaris.cluster.admin
set	solaris.cluster.modify
show	solaris.cluster.read
status	solaris.cluster.read
suspend	solaris.cluster.admin
switch	solaris.cluster.admin
unmanage	solaris.cluster.admin

名前

clresourcetype, clrt — Oracle Solaris Cluster データサービスのリソースタイプの管理

```
/usr/cluster/bin/clresourcetype [subcommand -?]

/usr/cluster/bin/clresourcetype subcommand -v [options]
     [resourcetype]...

/usr/cluster/bin/clresourcetype add-node  -n node[,…]
     [-Z {zoneclustername | global}] {+ | resourcetype}

/usr/cluster/bin/clresourcetype export [-o {- | configfile}]
     [+ | resourcetype]

/usr/cluster/bin/clresourcetype list [ -n node[,…]]
     [-Z {zoneclustername[,...] | global | all}] [+ | resourcetype...]

/usr/cluster/bin/clresourcetype list-props [-p [name,...]] [-Z
     {zoneclustername[,...] | global | all}] [+ | resourcetype...]

/usr/cluster/bin/clresourcetype register [-i
     {- | clconfiguration}] [ {[-n node
     [,…]] | -N}] [-f rtrfile] [-p [name [+ | -]=value,]...]
     [-Z {zoneclustername | global}] {+ | resourcetype...}

/usr/cluster/bin/clresourcetype remove-node -n node
     [,…] [-Z {zoneclustername | global}]
     {+ | resourcetype...}

/usr/cluster/bin/clresourcetype set [-n node
     [,…] | -N] [-p [name [+| -]=value,...]  [-Z
     {zoneclustername | global}]{+ | resourcetype...}

/usr/cluster/bin/clresourcetype show [-n node[,…]]
     [-Z {zoneclustername[,...] | global | all}] [+ | resourcetype...]

/usr/cluster/bin/clresourcetype unregister [-Z {zoneclustername |
     global}] {+ | resourcetype...}
```

clresourcetype コマンドは、Oracle Solaris Cluster データサービスのリソースタイプを管理します。clrt コマンドは clresourcetype コマンドの短い形式です。clresourcetype コマンドと clrt コマンドは同じものです。どちらの形式のコマンドも使用できます。

管理しやすいように、グローバルクラスタノードからこのコマンドを使用してください。

clresourcetype コマンドは、ゾーンクラスタで、export を除くすべてのサブコマンドを指定して使用することができます。

export 以外のすべてのサブコマンドで -Z オプションを使用して、操作を制限する特定のゾーンクラスタの名前を指定することもできます。また、リソースタイプ名 (*zoneclustername* : *resourcetype*) にゾーンクラスタ名を付けて、操作を特定のゾーンクラスタに制限することもできます。

グローバルクラスタノードからすべてのゾーンクラスタ情報にアクセスできますが、特定のゾーンクラスタは他のゾーンクラスタを認識しません。特定のゾーンクラスタに操作を制限できない場合、使用するサブコマンドは現在のクラスタでのみ機能します。

このコマンドの一般的な形式は次のとおりです。

clresourcetype [*subcommand*] [*options*] [*operands*]

options に -? または -V オプションを指定する場合だけは、*subcommand* を省略できます。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式は、このマニュアルページの「オプション」セクションのオプションの説明で紹介されています。

サポートされるサブコマンドには次のものがあります。

add-node

コマンドのオペランドとして指定されているリソースタイプのノードリストに指定されたノードを追加します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタノードから add-node コマンドを使用している間、-Z オプションを使用してゾーンクラスタの名前を指定できます。

このサブコマンドは、すべてのリソースタイプを指定するオペランドとして正符号 (+) を受け付けます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。

remove-node サブコマンドの説明も参照してください。

export

1447 ページのclconfiguration(5CL) のマニュアルページで規定されている形式で、クラスタリソースタイプ構成をエクスポートします。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.read RBAC の承認が必要です。

このサブコマンドは、大域ゾーンだけで使用できます。

list

コマンドのオペランドとして指定されているリソースタイプのリストを表示します。デフォルトでは、クラスタ内に登録されているすべてのリソースタイプが表示されます。このサブコマン

ドは、クラスタ内に登録されているすべてのリソースタイプを指定するオペランドとしてプラス記号 (+) を受け付けます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタノードで、このサブコマンドはグローバルクラスタノードで登録されたソースのみを表示します。グローバルクラスタからゾーンクラスタに登録されたリソースタイプを表示するために、-Z オプションを使用してゾーンクラスタを指定できます。

-n *nodelist* オプションを指定すると、*nodelist* 内のノードで使用するために登録されているリソースタイプだけが表示されます。

-v オプションを指定すると、リスト内の各リソースタイプのノードリストも表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.read RBAC の承認が必要です。

`list-props`

指定されたリソースタイプのプロパティーを表示します。このサブコマンドは、クラスタ内に登録されているすべてのリソースタイプを指定するオペランドとしてプラス記号 (+) を受け付けます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタで登録されたリソースタイプのプロパティーを表示するには、-Z オプションを使用してゾーンクラスタを指定できます。

-p オプションは、表示されるプロパティーのセットを限定します。

-v オプションを指定すると、各プロパティーの説明も表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.read RBAC の承認が必要です。

`register`

コマンドのオペランドとして指定されているリソースタイプを登録します。リソースタイプは、そのタイプのリソースを作成する前に登録してください。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタでリソースタイプを登録するには、-Z オプションを使用してゾーンクラスタ名を指定します。

各リソースタイプを定義するデータサービスは、リソースタイプが使用可能になる各ノードにインストールしてください。データサービスがクラスタノードのサブセットでのみインストールされる場合は、-n *nodelist* オプションを使用してノードのサブセットを指定します。リソースタイプがクラスタ内のすべてのノードで使用可能な場合は、-N オプションを指定します。-N オプションを使用する場合は、将来クラスタに追加される可能性があるどのノードでもリソースタイプを使用できます。-N オプションと -n *nodelist* オプションの両方を省略することは、-N オプションを指定することと同じです。プロパティー名を明示的に指定するには、-p Installed_nodes=*nodelist* オプションを使用します。

クラスタに登録されているリソースタイプについての情報は、リソースタイプを定義するリソースタイプ登録 (RTR) ファイルから取得されます。一般的に、RTR ファイルの場所と名前は次に示す規約に従います。

- RTR ファイルは、通常 `/opt/cluster/lib/rgm/rtreg` ディレクトリ内にあります。
- RTR ファイルの名前は、通常リソースタイプの名前に一致します。

Oracle が提供するすべての RTR ファイルの場所とファイル名は、次に示す規約に従っています。たとえば、`SUNW.nfs` リソースタイプを定義する RTR ファイルは `/opt/cluster/lib/rgm/rtreg/SUNW.nfs` ファイルに含まれています。

RTR ファイルがこれらの規約に従わない場合は、`-f` *rtrfile* オプションを指定してください。

これらの規則は、ゾーンクラスタから登録されたリソースタイプに対しても適用されます。ゾーンクラスタに対してリソースを登録するとき、RTR ファイルをゾーンクラスタ *zonepath* 内に常駐させる必要があります。ゾーンクラスタ *zonepath* 境界の外部で RTR ファイルを登録できません。*Global_zone* プロパティーのあるソースタイプを登録している間ゾーンクラスタに対して `TRUE` に設定するには、RTR ファイルを `/opt/cluster/lib/rgm/rtreg` または `/usr/cluster/lib/rgm/rtreg` ディレクトリのグローバルクラスタノード内部に常駐させる必要があります。これらの場所の外部の任意の場所を指定すると、リソースタイプを登録できません。

注意 - 信頼できる既知のソースであるリソースタイプを除いて、`Global_zone` プロパティーに `TRUE` が設定されているリソースタイプは登録しないでください。このプロパティーに `TRUE` を設定したリソースタイプは、ゾーン分離をすり抜け、脅威をもたらします。

このサブコマンドは、まだ登録されていないすべてのリソースタイプを指定するオペランドとしてプラス記号 (+) を受け付けます。使用可能なリソースタイプのリスト全体は、次のように決定されます。

- `-i` *clconfiguration* オプションを指定すると、*clconfiguration* は使用可能なリソースタイプの完全なリストを定義します。
- `-i` オプションを指定しない場合、使用可能なリソースタイプの完全なリストには Oracle が提供するリソースタイプのみが含まれます。これらのリソースタイプもまた、ノードリスト内のすべてのノードにインストールする必要があります。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。

`unregister` サブコマンドの説明も参照してください。

`remove-node`

オペランドリスト内のリソースタイプが登録されるノードのリストからノードを削除します。このサブコマンドは、クラスタ内に登録されているすべてのリソースタイプを指定するオペランドとしてプラス記号 (+) を受け付けます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタのあるリソースタイプを削除するには、-Z オプションを使用してゾーンクラスタの名前を指定します。

このサブコマンドは、クラスタ内のすべてのノードではなく一部のノードに対してすでに登録されているリソースタイプでのみ使用できます。結果として、次に示す状況でこのサブコマンドを使用するとエラーが発生します。

- オペランドのリスト内にあるリソースタイプがクラスタ内のすべてのノードに対してすでに登録されている。クラスタ内のすべてのノードのリソースタイプの登録については、-N オプションの説明を参照してください。
- オペランドのリスト内のリソースタイプの Installed_nodes プロパティーが、クラスタ内のノードのサブセットをまだ指定していない。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

add-node サブコマンドの説明も参照してください。

set

コマドのオペランドとして指定されているリソースタイプのプロパティーを設定します。このサブコマンドは、クラスタ内に登録されているすべてのリソースタイプを指定するオペランドとしてプラス記号 (+) を受け付けます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

広域クラスタからゾーンクラスタにリソースタイプのプロパティーを設定するには、-Z オプションを使用してゾーンクラスタの名前を指定します。

1335 ページのrt_properties(5) のマニュアルページで Tunable Any Time として指定されているリソースタイププロパティーのみを設定できます。

- Installed_Nodes のプロパティーは、-p オプションを指定せずに、-n *nodelist* オプションを指定することによって変更できます。または、-p Installed_Nodes= *nodelist* オプションを指定して、プロパティ名を明示的に指定できます。
- Tunable Any Time として指定されている他のすべてのプロパティーに関しては、-p *property* = *value* オプションを使用して、プロパティー名を明示的に指定します。

リソースタイプが使用できるノードのリストを制限するには、-n *nodelist* オプションを指定します。逆に、リソースタイプがクラスタ内のすべてのノードで使用可能であることを指定するには、-N オプションを指定します。-N オプションを使用する場合は、将来クラスタに追加される可能性があるどのノードでもリソースタイプを使用できます。-n オプションまたは -N オプションを指定してください。どちらのオプションも省略すると、このサブコマンドは構成情報を一切変更しません。

show

クラスタ内に登録されているリソースタイプについての情報を表示します。デフォルトでは、登録されているすべてのリソースタイプに対して次の情報が表示されます。

- 各リソースタイプに関連付けられているプロパティーのリスト
- これらのプロパティーを定義するパラメータ

-n *nodelist* オプションを指定すると、*nodelist* 内のノードで使用するために登録されているリソースタイプだけが表示されます。

-v オプションを指定すると、次の情報もリソースタイプごとに表示されます。

- リソースタイプに対して定義されているメソッド
- 各メソッドのタイムアウトパラメータ

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタに登録されたリソースタイプを表示するために、-z オプションを使用してゾーンクラスタの名前を指定できます。

このサブコマンドは、クラスタ内に登録されているすべてのリソースタイプを指定するオペランドとしてプラス記号 (+) を受け付けます。オペランドが指定されていない場合、クラスタに登録されているすべてのリソースタイプについての情報が表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。

`unregister`

コマンドのオペランドとして指定されているリソースタイプの登録を解除します。このサブコマンドは、そのタイプのインスタンスが存在しないすべての登録リソースタイプを指定するオペランドとしてプラス記号 (+) を受け付けます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

option.グローバルクラスタからゾーンクラスタのあるリソースタイプを登録解除するには、-z オプションを使用してゾーンクラスタの名前を指定します。

リソースタイプを定義するデータサービスをアンインストールする前にリソースタイプの登録を解除してください。

特定のリソースタイプのリソースが存在する場合、そのリソースタイプの登録は解除できません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.admin` RBAC の承認が必要です。

`register` サブコマンドの説明も参照してください。

次のオプションがサポートされています。

`-?`
`--help`

ヘルプ情報を表示します。このオプションを使用する場合、ほかの処理は実行されません。

このオプションを指定するとき、サブコマンドは指定してもしなくてもかまいません。

サブコマンドなしでこのオプションを指定すると、このコマンドのサブコマンドのリストが表示されます。

サブコマンド付きでこのオプションを指定すると、サブコマンドの使用方法が表示されます。

`-f` *rtrfile*|*rtrfiledir*
`--rtrfile=`*rtrfile*|*rtrfiledir*
`--rtrfile` *rtrfile*|*rtrfiledir*

リソースタイプの登録で使用するために、RTR ファイルのフルパスまたは RTR ファイルを含むディレクトリを指定します。このオプションは `register` サブコマンドとだけ指定できます。

ファイルを指定すると、1 つのリソースタイプだけを登録できます。

このオプションを指定する必要があるのは、使用している RTR ファイルが次に示す規約に従っていない場合だけです。

- RTR ファイルは、通常 `/opt/cluster/lib/rgm/rtreg` ディレクトリ内にあります。
- RTR ファイルの名前は、通常リソースタイプの名前に一致します。

Oracle が提供するすべての RTR ファイルの場所とファイル名は、次に示す規約に従っています。たとえば、`SUNW.nfs` リソースタイプを定義する RTR ファイルは `/opt/cluster/lib/rgm/rtreg/SUNW.nfs` ファイルに含まれています。

`-i` オプションを指定すると、構成情報で指定されているどのリソースタイプに対する構成情報でも `resourcetypeRTRFile` 要素を指定できます。`resourcetypeRTRFile` 要素は、リソースタイプの登録に使用される RTR ファイルを指定します。ただし、`export` サブコマンドは、生成される構成情報に `resourcetypeRTRFile` 要素を含みません。`resourcetypeRTRFile` 要素の詳細は、1447 ページの`clconfiguration`(5CL) のマニュアルページを参照してください。

`-i` {- | *clconfiguration*}
`--input=`{- | *clconfiguration*}
`--input` {- | *clconfiguration*}

リソースタイプの登録または登録されているリソースタイプのノードリストの変更に使用される構成情報を指定します。この情報は、1447 ページの`clconfiguration`(5CL) のマニュアルページに定義されている形式に準拠している必要があります。この情報はファイルに含まれている場合と、標準入力 `stdin` を通して指定される場合があります。標準入力を指定するには、ファイル名の代わりに `-` を指定します。

コマンドのオペランドとして指定されているリソースタイプだけが、このオプションによって影響されます。コマンドで指定されているオプションは、`clconfiguration` ファイル内に設定されているあらゆるオプションをオーバーライドします。構成パラメータが `clconfiguration` ファイル内にない場合は、不足しているパラメータをコマンド行で指定してください。

-N
--allnodes

オペランドのリスト内のリソースタイプがクラスタ内のすべてのノードで使用可能であることを指定します。-N オプションは、将来クラスタに追加される可能性がある任意のノードでもこれらのリソースタイプを使用可能にします。オプションは、Installed_nodes プロパティーをクリアすることによって、この結果を取得します。

-N オプションを指定すると、同じコマンドで -n オプションは指定できません。

-N オプションは、register サブコマンドまたは set サブコマンドとだけ指定できます。

-n *node*[,…]
--node=*node*[,…]
--node *node*[,…]

ターゲットのグローバルクラスタまたはゾーンクラスタに、ノードまたはノードのリストを指定します。各ノードはノード名またはノード ID で指定できます。

-Z オプションが指定されている場合は、-n オプションで、グローバルクラスタホスト名ではなくゾーンクラスタホスト名のみを指定できます。-Z オプションが指定されていない場合は、-n オプションでグローバルクラスタホスト名のみを指定できます。

-N オプションを指定すると、同じコマンドで -n オプションは指定できません。

このオプションとともに指定できるサブコマンドは、次のとおりです。

add-node

指定されたノードをリソースタイプが登録されているノードのリストに追加します。

list

指定されたノード上で使用するために登録されているリソースタイプだけを表示します。

register

指定されたノード上で使用する際にのみリソースタイプを登録します。-n オプションを省略すると、register サブコマンドはすべてのノード上で使用されるリソースタイプを登録します。このサブコマンドは、将来クラスタに追加される任意のノードのリソースタイプも登録します。

remove-node

指定されたノードをリソースタイプが登録されているノードのリストから削除します。

set

指定されたノードでのみリソースタイプを使用可能にします。

show

指定されたノード上で使用するために登録されているリソースタイプについての情報だけを表示します。

-o {- | *clconfiguration*}
--output={- | *clconfiguration*}
--output {- | *clconfiguration*}

リソースタイプについての構成情報が書き込まれる場所を指定します。この場所はファイルの場合と標準出力 stdout の場合があります。標準出力を指定するには、ファイル名の代わりに - を指定します。標準出力を指定すると、コマンドに対するほかのすべての標準出力が抑制されます。このオプションは、export サブコマンドの場合にだけ指定できます。

構成情報は、コマンドに対するオペランドとして指定されるリソースタイプに対してのみ記述されます。この情報は、1447 ページのclconfiguration(5CL) のマニュアルページに定義されている形式で書き込まれます。

-p *name=value*
-p *name+=array-values*
-p *name-=array-values*
--property=*name=value*
--property=*name+=array-values*
--property=*name-=array-values*
--property *name=value*
--property *name+=array-values*
--property *name-=array-values*

コマンドのオペランドとして指定されるリソースタイプのプロパティーの値を設定します。

このオプションとともに使用する演算子は、次のとおりです。

=	プロパティーに、指定した値を設定します。
+=	1 つまたは複数の値を文字列配列値に追加します。この演算子は、たとえば、Installed_nodes のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。
-=	1 つまたは複数の値が、文字列配列値から削除されます。この演算子は、たとえば、Installed_nodes のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。

オプション -p Installed_nodes+=nodeC,nodeD を set サブコマンドで使用することは、オプション -n nodeC,nodeD を add-node サブコマンドで使用することと同じです。

-p *name*[,…]
--property=*name*[,…]
--property *name*[,…]

list-props サブコマンドのプロパティーのリストを指定します。

-V
--version

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンドやオペランドなどのオプションは無視されます。-V オプションは、コマンドのバージョンだけを表示します。その他の処理は行いません。

-v
--verbose

詳細メッセージを標準出力 (stdout) に表示します。

このオプションは、このコマンドの任意の形式とともに指定できます。

-v オプションと -o - オプションを同時に指定してはいけません。-v オプ ションは無視されます。-o - オプションは、ほかのすべての標準出力を抑制します。

-Z {*zoneclustername* | global | all}
--zoneclustername={*zoneclustername* | global | all}
--zoneclustername {*zoneclustername* | global | all}

リソースタイプが登録されている場合、操作する必要がある 1 つまたは複数のクラスタを指定します。

このオプションは、export サブコマンド以外のすべてのサブコマンドでサポートされています。

このオプションを指定する場合は、次のいずれかの引数も指定する必要があります。

zoneclustername
: このオプションを使用するコマンドが、*zoneclustername* という名前のゾーンクラスタでのみ指定されたすべてのリソースタイプで機能するように指定します。

global
: このオプションを使用するコマンドが、グローバルクラスタでのみ指定されたすべてのリソースタイプで機能するように指定します。

all
: グローバルクラスタでこの引数を使用する場合、それを使用するコマンドがグローバルクラスタとすべてのゾーンクラスタを含め、すべてのクラスタで指定されたすべてのリソースタイプで機能するように指定します。

  ゾーンクラスタでこの引数を使用する場合、この引数を使用するコマンドがそのゾーンクラスタでのみ指定されたすべてのリソースタイプで機能するように指定します。

次のオペランドだけがサポートされています。

resourcetype
: 管理対象となる 1 つまたは複数のリソースタイプを指定します。サブコマンドが複数のリソースタイプを受け入れる場合は、プラス記号 (+) を使用してすべてのリソースタイプを指定できます。

  リソースタイプ名の形式については、Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド の有効な RGM 名"を参照してください。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

次の終了コードが返されます。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

18 CL_EINTERNAL

内部エラーが発生しました

内部エラーは、ソフトウェアの欠陥またはその他の欠陥を示しています。

35 CL_EIO

I/O エラー

物理的な入出力エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。

- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

37 CL_EOP

操作が許可されていません

サポートされていない構成に対する操作を実行しようとしたか、サポートされていない操作を実行しました。

41 CL_ETYPE

無効なタイプです

-t または -p オプションで指定したタイプは存在しません。

これらの終了値は、1047 ページのscha_calls(3HA) のマニュアルページで説明されているリターンコードと互換性があります。

例 160 リソースタイプの登録

この例では、データサービスがすべてのノードにインストールされ、まだ登録されていないすべてのリソースタイプを登録する方法を示します。このコマンドは簡略モードで実行されます。

```
# clresourcetype register +
```

例 161 選択されたノード上でのリソースタイプの登録

この例では、データサービスがノード phys-schost-1 およびノード phys-schost-2 にインストールされており、かつまだ登録されていないすべてのリソースタイプを登録します。リソースはこれらのノードでのみ使用可能になります。この例では、コマンドがエラーを返すことはありません。このコマンドは詳細モードで実行されます。

```
# clresourcetype register -v -n phys-schost-1,phys-schost-2 +
```

次のコマンドは、そのデータサービスがゾーンクラスタ ZC のゾーンクラスタのゾーンクラスタノード zc-host-1 と zc-host-2 にインストールされたすべてのリソースタイプおよび登録されていないリソースタイプを登録します。リソースは、これらのゾーンクラスタノードでのみ使用可能です。

```
#.clresourcetype register -n zc-host-1,zc-host-2 -Z ZC +
```

例 162 単一のリソースタイプの登録

この例では、SUNW.nfs:3.2 リソースタイプを登録する方法を示します。このリソースタイプのデータサービスは、すべてのクラスタノードにインストールされます。

```
# clresourcetype register nfs:3.2
```

例 163 リソースタイプの一覧表示

この例では、登録されているすべてのリソースタイプの名前だけを一覧表示する方法を示します。

```
# clresourcetype list
SUNW.LogicalHostname
SUNW.SharedAddress
SUNW.nfs
SUNW.apache
```

例 164 リソースタイプとリソースタイプのノードリストの一覧表示

この例では、登録されているすべてのリソースタイプをそれらのノードリストとともに一覧表示する方法を示します。

```
# clresourcetype list -v

Resource Type         Node List
-------------         ---------
SUNW.LogicalHostname  <all>
SUNW.SharedAddress    <all>
SUNW.nfs              phys-schost-1,phys-schost-2,phys-schost-3
SUNW.apache           phys-schost-1,phys-schost-2,phys-schost-3
```

グローバルクラスタノードから次のコマンドを実行するとき、コマンドはゾーンクラスタ ZC に登録されたすべてのリソースタイプを一覧します。

```
#.clresourcetype list -Z ZC
SUNW.nfs
SUNW.apache
```

例 165 指定されたノード上でのリソースタイプの一覧表示

この例では、 phys-schost-4 上で登録されているすべてのリソースタイプを一覧表示する方法を示します。

```
# clrt list -n phys-schost-4
SUNW.LogicalHostname
SUNW.SharedAddress
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core

属性タイプ	属性値
インタフェースの安定性	発展中

19 ページの`Intro`(1CL), 251 ページの`clreslogicalhostname`(1CL), 273 ページの`clresource`(1CL), 305 ページの`clresourcegroup`(1CL), 347 ページの`clressharedaddress`(1CL), 551 ページの`cluster`(1CL), 1047 ページの`scha_calls`(3HA), 1447 ページの`clconfiguration`(5CL), 1287 ページの`r_properties`(5), Unresolved link to " attributes5", Unresolved link to " rbac5"

Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド のリソースグループプロパティー"

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のオプションを指定してこのコマンドを実行できます。

- `-?` オプション
- `-V` オプション

サブコマンドを指定してこのコマンドを実行する場合、スーパーユーザー以外のユーザーは RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
`add-node`	`solaris.cluster.modify`
`export`	`solaris.cluster.read`
`list`	`solaris.cluster.read`
`list-props`	`solaris.cluster.read`
`set`	`solaris.cluster.modify`
`register`	`solaris.cluster.modify`
`remove-node`	`solaris.cluster.modify`
`show`	`solaris.cluster.read`
`unregister`	`solaris.cluster.admin`

名前

clressharedaddress, clrssa — 共有アドレスの Oracle Solaris Cluster リソースの管理

```
/usr/cluster/bin/clressharedaddress [subcommand] -?

/usr/cluster/bin/clressharedaddress -V

/usr/cluster/bin/clressharedaddress [subcommand [options]] -v
     [saresource]...

/usr/cluster/bin/clressharedaddress create -g resourcegroup [-h
     lhost[,…]] [-N netif@node[,…]] [-X node[,…]]
     [-p name=value] [-Z {zoneclustername | global}] [-d] saresource

/usr/cluster/bin/clressharedaddress create -i
     {- | clconfiguration} [-a] [-g resourcegroup[,…]] [-X
     node[,…]] [-p name=value] [-d] {+ | saresource...}

/usr/cluster/bin/clressharedaddress delete [-g resourcegroup[,…]]
     [-Z {zoneclustername | global}] [-F] {+ | saresource...}

/usr/cluster/bin/clressharedaddress disable [-g resourcegroup[,…]]
     [-R] [-n node[,…]] [-Z
     {zoneclustername | global}] {+ | saresource...}

/usr/cluster/bin/clressharedaddress enable [-g resourcegroup[,…]]
     [-R] [ -n node[,…]] [-Z
     {zoneclustername | global}]{+ | saresource...}

/usr/cluster/bin/clressharedaddress export [-o {- | configfile}]
     [+ | saresource...]

/usr/cluster/bin/clressharedaddress list [-s state[,…]]
     [-g resourcegroup[,…]] [-Z {zoneclustername
     [,...] | global | all}] [+ | saresource...]

/usr/cluster/bin/clressharedaddress list-props [-l listtype] [-p
     name[,…]] [-Z {zoneclustername [,...] | global | all}]
     [+ | lhresource...]

/usr/cluster/bin/clressharedaddress monitor [-g resourcegroup[,…]]
     [-Z {zoneclustername | global}] {+ | saresource...}

/usr/cluster/bin/clressharedaddress reset [-f errorflag] [-g
     resourcegroup[,…]] [-Z {zoneclustername | global}]
     {+ | saresource...}

/usr/cluster/bin/clressharedaddress set [-i {- | clconfiguration}]
     [-g resourcegroup[,…]] [-X node[,…]] [-p
```

```
     name[+|-]=value] [-Z {zoneclustername | global}] {+ | saresource...}

/usr/cluster/bin/clressharedaddress show [-g resourcegroup[,…]]
     [-p name[,…]] [-Z {zoneclustername [,...] | global | all}]
     [+ | saresource...]

/usr/cluster/bin/clressharedaddress status [-s state[,…]]
     [ -n node[,…]] [-g resourcegroup[,…]]
     [-Z {zoneclustername [,...] | global | all}] [+ | saresource...]

/usr/cluster/bin/clressharedaddress unmonitor [-g resourcegroup[,…]]
     [-Z {zoneclustername | global}] {+ | saresource...}
```

clressharedaddress コマンドは、Oracle Solaris Cluster 共有アドレスのリソースを管理します。clrssa コマンドは clressharedaddress コマンドの短い形式です。clressharedaddress コマンドと clrssa コマンドは同じものです。どちらの形式のコマンドも使用できます。

273 ページのclresource(1CL) コマンドを使用して、共有アドレスのリソースを管理することもできます。

clressharedaddress コマンドの一部のサブコマンドは、リソース構成を変更します。これらのサブコマンドは、グローバルクラスタまたはゾーンクラスタから使用できます。リソース構成を変更するサブコマンドは、次のとおりです。

- disable
- enable
- monitor
- reset
- set
- unmonitor

clressharedaddress コマンドの一部のサブコマンドは、リソースに関する情報のみを取得します。

- export
- list
- list-props
- show
- status

このコマンドからの予想不能な結果を避けるには、コマンドのすべての書式をグローバルクラスタノードから実行してください。

コマンドの一般的な形式は次のとおりです。

`clressharedaddress` [*subcommand*] [*options*] [*operands*]

subcommand を省略できるのは、*options* がオプション `-?` または `-V` を指定している場合だけです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式は、このマニュアルページの「オプション」セクションのオプションの説明で紹介されています。

ゾーンクラスタでの操作

`clressharedaddress` コマンドは、ゾーンクラスタで、`export` を除くすべてのサブコマンドを指定して使用することができます。

`export` 以外のすべてのサブコマンドで `-Z` オプションを使用して、操作を制限する特定のゾーンクラスタの名前を指定することもできます。また、共有アドレスリソース (*zoneclustername* : *saresource*) にゾーンクラスタ名を付けて、操作を特定のゾーンクラスタに制限することもできます。

グローバルクラスタノードからすべてのゾーンクラスタ情報にアクセスできますが、特定のゾーンクラスタは他のゾーンクラスタを認識しません。特定のゾーンクラスタに操作を制限できない場合、使用するサブコマンドは現在のクラスタでのみ機能します。

サポートされるサブコマンドには次のものがあります。

`create`

コマンドに対するオペランドとして共有アドレス指定されたリソースを作成します。

`create` を `-i` オプションと使用して構成ファイルを指定した場合、サブコマンドはプラス記号 (+) をオペランドとして受け付けます。`+` オペランドを使用すると、構成ファイル内に存在しないすべてのリソースが作成されます。

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

デフォルトでは、リソースはモニタリング対象となり、有効な状態で作成されます。ただし、リソースがオンライン状態になり、モニターされるのは、リソースのリソースグループがオンラインになったあとだけです。無効な状態でリソースを作成するには、`-d` オプションを指定します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタに共有アドレスリソースを作成するために、-z オプションを使用してゾーンクラスタの名前を指定できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。

delete サブコマンドの説明も参照してください。

delete

コマンドに対するオペランドとして指定された共有アドレスリソースを削除します。このサブコマンドに対しオペランドとしてプラス記号 (+) を指定すると、すべてのリソースが削除されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタに共有アドレスリソースを作成するために、-z オプションを使用してゾーンクラスタの名前を指定できます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、削除するリソースを限定することができます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを削除します。

- デフォルトでは、リソースは次の条件が満たされる場合に *のみ* 削除されます。
- リソースが無効な状態である。
- リソースに対するすべての依存性が削除されている。
- 指定したすべてのリソースを確実に削除するには、-F オプションを指定します。-F オプションの効果は、次のとおりです。
- 指定したすべてのリソース (無効になっていないリソースも含む) が削除されます。
- 指定したすべてのリソースが、他のリソースのリソース依存性設定から削除されます。

リソースは、コマンド行でリソースを指定した順序とは無関係に、リソース間の依存性を満たすのに必要な順序に従って削除されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

create サブコマンドの説明も参照してください。

disable

コマンドに対するオペランドとして指定された共有アドレスリソースを無効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースが無効になります。

-g オプションを指定すると、オペランドのリストをフィルタリングし、無効にするリソースを限定することができます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを無効にします。

必要なリソース依存性をすべて確実に満たすには、-R オプションを指定します。-R オプションは、コマンドに対するオペランドとして指定したリソースに依存しているリソース (コマンドに対するオペランドとして指定しなかったリソースも含まれる) をすべて無効にします。-g オプションと -t オプションは、リソース依存性を満たすためだけに無効化されるリソースには適用されません。

リソースは、コマンド行でリソースを指定した順序とは無関係に、リソース間の依存性を満たすのに必要な順序に従って無効化されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタに登録された共有アドレスリソースを無効にするには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

enable サブコマンドの説明も参照してください。

enable

コマンドに対するオペランドとして指定された共有アドレスリソースを有効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースが有効になります。

-g オプションを指定すると、オペランドのリストをフィルタリングし、有効にするリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを有効にします。

必要なリソース依存性をすべて確実に満たすには、-R オプションを指定します。-R オプションは、コマンドに対するオペランドとして指定したリソースに依存しているリソース (コマンドに対するオペランドとして指定しなかったリソースも含まれる) をすべて有効にします。-g オプションは、リソース依存性を満たすためだけに有効化されるリソースには適用されません。

リソースは、コマンド行でリソースを指定した順序とは無関係に、リソース間の依存性を満たすのに必要な順序に従って有効化されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタに登録された共有アドレスリソースを有効にするには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

disable サブコマンドの説明も参照してください。

export

1447 ページのclconfiguration(5CL) のマニュアルページで規定されている形式で、共有アドレスリソース構成をエクスポートします。

このサブコマンドは、大域ゾーンだけで使用できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

`list`

マンドに対するオペランドとして指定した共有アドレスリソースのリストを表示します。デフォルトでは、すべてのリソースが表示されます。

`-g` オプションを指定すると、オペランドのリストをフィルタリングし、表示するリソースを限定できます。`-g` オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを表示します。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、指定したリソースグループ内のすべてのリソースまたは指定したリソースタイプのインスタンスであるすべてのリソースを指定できます。オペランドが指定されていない場合、指定されているリソースグループ内のすべてのリソースまたは指定されているリソースタイプのインスタンスであるすべてのリソースが表示されます。

`-v` オプションを指定すると、該当するリソースグループおよびリスト内の各リソースのリソースタイプも表示されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタに登録された共有アドレスリソースを表示するには、`-Z` オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

`list-props`

コマンドに対するオペランドとして指定した共有アドレスリソースのプロパティーのリストを表示します。デフォルトでは、すべてのリソースの拡張プロパティーが表示されます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、プロパティーが表示されるリソースを限定できます。

`-g` *resourcegrouplist*
: *resourcegrouplist* 内のリソースグループのメンバーであるオペランドのリストの共有アドレスリソースのプロパティーだけを表示します。

`-l` オプションでは、表示するリソースプロパティーのタイプを指定します。

`-l all`
: 標準プロパティーと拡張プロパティーを表示するように指定します。

`-l extension`
: 拡張プロパティーだけを表示するように指定します。デフォルトでは、拡張プロパティーだけが表示されます。

`-l standard`
: 標準プロパティーだけを表示するように指定します。

`-l` オプションを指定しない場合、`-p` オプションまたは `-y` オプションを使用して標準プロパティーを明示的に指定しないかぎり、拡張プロパティーだけが表示されます。

`-p` オプションは、表示するリソースプロパティーのセットを制限します。`-p` オプションを指定すると、*namelist* で指定したプロパティーだけが表示されます。*namelist* では、標準プロパティーと拡張プロパティーを指定できます。

-v オプションを指定すると、各プロパティーの説明も表示されます。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、指定したリソースグループ内のすべてのリソースまたは指定したリソースタイプのインスタンスであるすべてのリソースを指定できます。オペランドを指定しないと、指定したリソースグループ内のすべてのリソースのプロパティーまたは指定したリソースタイプのインスタンスであるすべてのリソースのプロパティーが表示されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタの共有アドレスリソースのプロパティーのリストを表示するには、-z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

`monitor`

コマンドに対するオペランドとして指定した共有アドレスリソースのモニタリングを有効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースに対してモニタリングが有効になります。

-g オプションを指定すると、オペランドのリストをフィルタリングし、モニター対象のリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけをモニターします。

リソースは、モニタリングが有効になっている場合、次の条件が満たされているときだけモニタリングされます。

- リソースが有効な状態にある。
- 該当リソースが含まれるリソースグループが、1 つ以上のクラスタノード上でオンライン状態にある。

注記 - リソースに対するモニタリングを有効にする場合、該当リソースを有効にする必要は*ありません*。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタのリソースをモニターするには、-z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.admin` RBAC の承認が必要です。

`unmonitor` サブコマンドの説明も参照してください。

`reset`

コマンドに対するオペランドとして指定した共有アドレスリソースに関連付けられているエラーフラグをクリアします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースのエラーフラグがクリアされます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、リセットするリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけをリセットします。

デフォルトでは、reset サブコマンドはエラーフラグ STOP_FAILED をクリアします。クリアするエラーフラグを明示的に指定するには、-f オプションを使用します。-f オプションが受け付けるエラーフラグは、STOP_FAILED だけです。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタの共有アドレスリソースをリセットするには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

set

コマンドのオペランドとして指定されている共有アドレスリソースの指定プロパティーを変更します。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースの指定したプロパティーが変更されます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、変更するリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを変更します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタに共有アドレスリソースのプロパティーを設定するには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

show

コマンドに対するオペランドとして指定した共有アドレスリソースの構成を表示します。デフォルトでは、すべてのリソースの構成が表示されます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、構成が表示されるリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけの構成を表示します。

-p オプションは、表示するリソースプロパティーのセットを制限します。-p オプションを指定すると、*namelist* で指定したプロパティーだけが表示されます。*namelist* では、標準プロパティーと拡張プロパティーを指定できます。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、指定したリソースグループ内のすべてのリソースまたは指定したリソースタイプのインスタンスであるすべてのリソースを指定できます。オペランドを指定しないと、指定したリソースグループ内のすべてのリソースの構成または指定したリソースタイプのインスタンスであるすべてのリソースの構成が表示されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタに共有アドレスリソースの構成を表示するには、-z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

status

コマンドに対するオペランドとして指定したリソースのステータスを表示します。デフォルトでは、すべてのリソースのステータスが表示されます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、ステータスが表示されるリソースを限定できます。

-g *resourcegrouplist* — *resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけのステータスを表示します。

-n *nodelist* — *nodelist* 内のノード上でホストされている、オペランドのリスト内のリソースについてのみステータスを表示します。

-s *statelist* — *statelist* 内の状態にある、オペランドのリスト内のリソースだけのステータスを表示します。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、指定したリソースグループ内のすべてのリソースまたは指定したリソースタイプのインスタンスであるすべてのリソースを指定できます。オペランドを指定しないと、指定したリソースグループ内のすべてのリソースのステータスまたは指定したリソースタイプのインスタンスであるすべてのリソースのステータスが表示されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタに共有アドレスリソースのステータスを表示するには、-z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

unmonitor

コマンドに対するオペランドとして指定した共有アドレスリソースのモニタリングを無効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースに対するモニタリングが無効になります。

無効になっているリソースのモニタリングを無効にしても、リソースは影響を受けません。リソースとそのモニターは、すでにオフライン状態です。

注記 - リソースのモニタリングを無効にしても、リソースは無効に*なりません*。ただし、リソースを無効にする場合、モニタリングを無効にする必要はありません。無効なリソースとそのモニターは、オフライン状態が維持されます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、モニタリングを無効にするリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースのモニタリングを無効にします。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタへの共有アドレスリソースのモニタリングをオフにするには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

disable サブコマンドおよび monitor サブコマンドの説明も参照してください。

次のオプションがサポートされています。

-?
--help

ヘルプ情報を表示します。このオプションを使用する場合、ほかの処理は実行されません。

このオプションを指定するとき、サブコマンドは指定してもしなくてもかまいません。

サブコマンドなしでこのオプションを指定すると、このコマンドのサブコマンドのリストが表示されます。

サブコマンド付きでこのオプションを指定すると、サブコマンドの使用方法が表示されます。

特定のサブコマンドを指定した場合のこのオプションの効果は、次のようになります。

create	-g オプションとともに指定された場合、指定されたリソースグループのすべてのリソースプロパティーのヘルプ情報を表示します。
set	コマンドに対するオペランドとして指定したリソースのプロパティーに関する情報を表示します。

-a
--automatic

クラスタ構成情報からリソースが作成される場合、次の処理も自動的に実行します。

- リソースタイプの登録
- リソースグループの作成
- オペランドのリスト内で指定されているリソースの依存先リソースの作成

クラスタ構成情報には、次の処理をすべて実行するのに必要な十分な情報が含まれている必要があります。

- リソースタイプの登録を有効にする
- リソースグループの作成を有効にする

■ 作成されるリソースの有効化

このオプションを指定できるのは、create サブコマンドの場合だけです。このオプションを指定する場合は、-i オプションも指定し、構成ファイルを指定します。

-d
--disable

リソースの作成時にリソースを無効にします。このオプションを指定できるのは、create サブコマンドの場合だけです。デフォルトでは、リソースは作成されたあと、有効な状態になります。

リソースは、有効化しても、オンライン状態になるとは限りません。リソースは、リソースのリソースグループが 1 つ以上のノードでオンライン状態になったあとでのみオンライン状態になります。

-f *errorflag*
--flag *errorflag*

reset サブコマンドによってクリアするエラーフラグを明示的に指定します。このオプションを指定できるのは、reset サブコマンドの場合だけです。デフォルト時、reset サブコマンドはエラーフラグ STOP_FAILED をクリアします。

-f オプションが受け付けるエラーフラグは、STOP_FAILED だけです。

-F
--force

無効状態でないリソースの削除が、強制的に実行されます。このオプションは、delete サブコマンドの場合にだけ指定できます。

-g *resourcegroup*[,…]
--resourcegroup *resourcegroup*[,…]

1 つのリソースグループまたはリソースグループのリストを指定します。

create 以外のサブコマンドの場合、コマンドは -g オプションで指定したリソースグループのメンバーである、オペランドのリスト内のリソースにだけ作用します。

特定のサブコマンドを指定した場合のこのオプションの効果は、次のようになります。

create	指定したリソースグループ内でリソースを作成するように指定します。-create サブコマンドを指定して g を使用する場合、リソースグループは 1 つしか指定することができません。

-h *lhost*[,…]
--logicalhost *lhost*[,…]

ホスト名リストを指定します。複数の論理ホストを新しい -SharedAddress リソースに関連付ける必要がある場合や論理ホストがリソース自体と同じ名前を持っていない場合は、h オプションを使用します。SharedAddress リソースの HostnameList 内のすべての論理

ホストは同じサブネット上に置いてください。HostnameList プロパティーを指定しない場合、HostnameList は SharedAddress リソースと同じになります。

SharedAddress リソースの論理ホスト名は同じサブネット上に置いてください。

-HostnameList プロパティーを p とともに設定する代わりに、-h を使用することは可能です。ただし、-h を使用して、同じコマドで明示的に HostnameList を設定できません。

-h は、create サブコマンドの場合にだけ使用できます。

-i {- | *clconfiguration*}
--input {- | *clconfiguration*}

共有アドレスリソースの作成または変更に使用する構成情報を指定します。この情報は、1447 ページのclconfiguration(5CL) のマニュアルページに定義されている形式に準拠している必要があります。この情報は、ファイルに含めることも、標準入力を介して指定することもできます。標準入力を指定するには、ファイル名の代わりに - を指定します。

コマンドに対するオペランドとして指定したリソースだけが、作成または変更されます。コマンドで指定したオプションは、構成情報で設定されているオプションより優先されます。構成パラメータは、構成情報内で設定されていない場合、コマンド行で指定します。

特定のサブコマンドを指定した場合のこのオプションの効果は、次のようになります。

create	-a オプションと指定された場合は、必要なすべてのリソースタイプを登録し、必要なすべてのリソースグループを作成します。登録および構成に必要なすべての情報を指定します。その他の構成データはすべて無視します。

-l *listtype*
--listtype *listtype*

list-props サブコマンドによって表示するリソースプロパティーのタイプを指定します。このオプションは、list-props サブコマンドの場合にだけ指定できます。

listtype に対して、次のリストから値を 1 つ指定します。

all	標準プロパティーと拡張プロパティーを表示するように指定します。
extension	拡張プロパティーだけを表示するように指定します。デフォルトでは、拡張プロパティーだけが表示されます。
standard	標準プロパティーだけを表示するように指定します。

-l オプションを指定しないと、-p オプションを使って標準プロパティーを明示的に指定しないかぎり、拡張プロパティーしか表示されません。

-n *node*[,…]
--node *node*[,…]

ターゲットのグローバルクラスタまたはゾーンクラスタに、ノードまたはノードのリストを指定します。各ノードは、ノード名またはノード ID として指定できます。

-Z オプションが指定されている場合は、-n オプションで、グローバルクラスタホスト名ではなくゾーンクラスタホスト名のみを指定できます。-Z オプションが指定されていない場合は、-n オプションでグローバルクラスタホスト名のみを指定できます。

このオプションとともに指定できるサブコマンドは、次のとおりです。

`disable`	指定したノード上でホストされている、オペランドのリスト内のリソースのみを無効にします。
`enable`	指定したノード上でホストされている、オペランドのリスト内のリソースのみを有効にします。
`status`	指定したノード上でホストされている、オペランドのリスト内のリソースについてのみステータスを報告します。

`-N` *netif@node*[,…]
`--netiflist` *netif@node*[,…]

リソースプロパティーを指定します。-N オプションを使用すると、プロパティーの p オプションを使用せずに -NetIfList プロパティーを設定できます。-N を指定しないと、clressharedaddress コマンドは使用可能な IPMP グループまたはパブリックアダプタと、HostnameList プロパティーと関連付けられているサブネットに基づいて、NetIfList プロパティーを自動的に設定するよう試みます。

NetIfList プロパティーは、次の形式で指定できます。*ipmpgroup@node*[,…]。ただし、-N は両方の *ipmpgroup@node*[,…] および *publicNIC@node*[,…] を受け入れます。-N を使用しなかったり、*publicNIC@node* とともに使用したりすると、clressharedaddress コマンドは、必要な IPMP グループを作成しようとします。システムは、標準 Oracle Solaris インタフェースを使用して複数のアダプタを含めるようにあとで変更されるデフォルトのセットで、1 つ以上の単一アダプタ IPMP グループのセットを作成します。

-p を指定して NetIfList プロパティーを直接設定する代わりに -N を指定することができます。ただし、-N を使用し、同じコマンド内で NetIfList を明示的に設定できません。

-N は、create サブコマンドの場合にだけ使用できます。

`-o` {- | *clconfiguration*}
`--output` {- | *clconfiguration*}

リソース構成情報の書き込み先を指定します。この書き込み先は、ファイルでも、標準出力でもかまいません。標準出力を指定するには、ファイル名の代わりに - を指定します。標準出力を指定すると、該当コマンドにおける他のすべての標準出力は抑制されます。このオプションは、export サブコマンドの場合にだけ指定できます。

構成情報は、コマンドに対するオペランドとして指定したリソースの場合だけ書き込まれます。この情報は、1447 ページのclconfiguration(5CL) のマニュアルページに定義されている形式で書き込まれます。

-p *name=value*
-p *name+=array-values*
-p *name-=array-values*
--property *name=value*
--property *name+=array-values*
--property *name-=array-values*

リソースの標準プロパティーと拡張プロパティーを設定します。このオプションは、create サブコマンドおよび set サブコマンドの場合にだけ指定できます。

標準プロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

リソースタイプの拡張プロパティーについては、リソースタイプに関するドキュメントを参照してください。

このオプションとともに使用する演算子は、次のとおりです。

=	プロパティーに、指定した値を設定します。この演算子は、create サブコマンドおよび set サブコマンドで使用できます。
+=	1 つまたは複数の値を文字列配列値に追加します。この演算子は、set サブコマンドでのみ使用できます。この演算子は、文字列配列値に対してだけ指定できます。
-=	1 つまたは複数の値が、文字列配列値から削除されます。この演算子は、set サブコマンドでのみ使用できます。この演算子は、文字列配列値に対してだけ指定できます。

ノード単位のプロパティーがクラスタノードのサブセット上でのみ設定される場合は、中括弧内のノードのリストを次のようにプロパティー名に付け加えることで、プロパティーが設定されるノードを指定します。

name{*nodelist*}

nodelist は、ノード名またはノード ID をコンマで区切ったリストです。ノード単位のプロパティーの詳細は、1335 ページのrt_properties(5) のマニュアルページを参照してください。

-p *name*[,…]
--property *name*[,…]

list-props サブコマンドおよび show サブコマンドのプロパティーのリストを指定します。

このオプションは、リソースの標準プロパティーおよび拡張プロパティーに対して使用できます。

標準プロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

リソースタイプの拡張プロパティーについては、リソースタイプに関するドキュメントを参照してください。

このオプションを指定しなかった場合、list-props サブコマンドおよび show サブコマンドは、-v オプションも指定されているかどうかに基づいて、すべてまたはほとんどのリソースプロパティーを一覧表示します。

-R
--recursive

必要な依存性がすべて満たされるように、リソースの有効化または無効化を再帰的に実行します。このオプションは、disable サブコマンドおよび enable サブコマンドの場合にだけ指定できます。

このオプションをこれらのサブコマンドとともに指定した場合の効果は、次のとおりです。

disable	コマンドに対するオペランドとして指定したリソースに依存しているリソース (コマンドに対するオペランドとして指定しなかったリソースも含まれる) をすべて無効にします。
enable	コマンドに対するオペランドとして指定したリソースの依存先リソース (コマンドに対するオペランドとして指定しなかったリソースも含まれる) がすべて有効になります。

-s *state*[,…]
--state *state*[,…]

list サブコマンドおよび status サブコマンドの状態のリストを指定します。

このオプションは出力を制限し、ノードリスト内の 1 つまたは複数のノード上で指定されている状態の 1 つにあるリソースだけが含まれるようにします。

可能な状態は、次のとおりです。

- degraded
- detached
- faulted
- monitor_failed
- not_online - online または online_not_monitored 以外のすべてのステータスを指定します
- offline
- online
- online_not_monitored
- start_failed
- stop_failed
- unknown

- `unmonitored`
- `wait`

`-V`
`--version`

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンドやオペランドなどのオプションは無視されます。`-V` オプションは、コマンドのバージョンを表示するだけです。その他の処理は行いません。

`-v`
`--verbose`

詳細なメッセージを標準出力に表示します。

このオプションは、このコマンドの任意の形式とともに指定できます。

`-v` オプションと `-o -` オプションを同時に指定してはいけません。`-v` オプ ションは無視されます。`-o -` オプションは、ほかのすべての標準出力を抑制します。

`-X` *node*[,…]
`--auxnode` *node*[,…]

`AuxNodeList SharedAddress` リソースプロパティーを設定します。

`AuxNodeList` リスト内のノードは、共有アドレスリソースに関連付けられている論理ホストのセットをホストできます。ただし、フェイルオーバー時にプライマリノードの役割を果たすことはできません。

`-Z` {*zoneclustername* | global | all}
`--zoneclustername=`{*zoneclustername* | global | all}
`--zoneclustername` {*zoneclustername* | global | all}

クラスタ、またはリソースが存在するクラスタや処理するクラスタを指定します。

このオプションは、`export` サブコマンド以外のすべてのサブコマンドでサポートされています。

このオプションを指定する場合は、次のいずれかの引数も指定する必要があります。

zoneclustername	このオプションを使用するコマンドが、*zoneclustername* という名前のゾーンクラスタでのみ指定されたすべてのリソースで機能するように指定します。
`global`	このオプションを使用するコマンドが、広域クラスタでのみ指定されたすべてのリソースで機能するように指定します。
`all`	広域クラスタでこの引数を使用する場合、それを使用するコマンドが広域クラスタとすべてのゾーンクラスタを含め、すべてのクラスタで指定されたすべてのリソースで機能するように指定します。

この引数をゾーンクラスタで使用すると、このオプションを使用するコマンドが、ゾーンクラスタ内だけで指定されたすべてのリソースに対して機能するように指定されます。

次のオペランドがサポートされています。

resource Oracle Solaris Cluster のリソース名をオペランドとして受け付けるように指定します。サブコマンドで複数のリソースを指定できる場合は、プラス記号 (+) を使用すると、すべてのリソースを指定できます。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

9 CL_ESTATE

オブジェクトの状態が不正です

その時点で変更できない、または常時変更できないプロパティー、リソースグループ、またはその他のオブジェクトを変更しようとしました。

10 CL_EMETHOD

リソースのメソッドが失敗しました

リソースのメソッドが失敗しました。次のいずれかの理由のために、メソッドは失敗しました。

- validate メソッドは、リソースを作成しようとしたときに、あるいは、リソースのプロパティーを変更しようとしたときに失敗しました。
- validate 以外のメソッドは、リソースを有効、無効、または削除しようとしたときに失敗しました。

15 CL_EPROP

無効なプロパティーです

-p、-y、または -x オプションで指定したプロパティーまたは値が存在しないか、許可されていません。

35 CL_EIO

I/O エラー

物理的な入出力エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

これらの終了値は、1047 ページのscha_calls(3HA) のマニュアルページで説明されているリターンコードと互換性があります。

例 166 共有アドレスリソースの作成

このコマンドは、sharedhost1 という名前のリソースを rg-failover という名前のリソースグループ内に作成します。リソースは有効な状態で作成され、モニタリングも有効になっています。

```
# clressharedaddress create -g rg-failover sharedhost1
```

次の 2 つのコマンドはどちらも、ゾーンクラスタ ZC 内に sharedhost1 という名前のリソースを作成します。これらのコマンドは、グローバルクラスタノードで、またはゾーンクラスタ ZC の内部で実行できます。

```
# clressharedaddress create -g rg-failover -Z ZC sharedhost1

# clressharedaddress create -g rg-failover ZC:sharedhost1
```

例 167 異なる論理ホスト名を持つ共有アドレスリソースの作成

このコマンドは、rs-sharedhost1 という名前のリソースを rg-failover という名前のリソースグループ内に作成します。

論理ホスト名はリソース名と同じではありませんが、論理ホストの名前と IP アドレスは同じままです。

```
# clressharedaddress create -g rg-failover \
-h sharedhost1 rs-sharedhost1
```

例 168 共有アドレスリソースの IPMP グループの指定

このコマンドは、sharedhost1 リソースの IPMP グループを設定します。

```
# clressharedaddress create -g rg-failover \
-N ipmp0@black,ipmp0@white sharedhost1
```

例 169 共有アドレスリソースの削除

このコマンドは、sharedhost1 という名前のリソースを削除します。

```
# clressharedaddress delete sharedhost1
```

例 170 共有アドレスリソースの一覧表示

このコマンドは、すべての共有アドレスリソースを一覧表示します。

```
# clressharedaddress list
sharedhost1
sharedhost2
```

例 171 共有アドレスリソース、リソースグループ、およびリソースタイプの一覧表示

このコマンドは、すべての共有アドレスリソースをそれらのリソースグループおよびリソースタイプと合わせて一覧表示します。

```
# clressharedaddress list -v
Resources   Resource Groups Resource Types
---------   -------------- --------------
sharedhost1 rg-failover-1   SUNW.SharedAddress
sharedhost2 rg-failover-2   SUNW.SharedAddress
```

例 172 共有アドレスリソースの拡張プロパティーの一覧表示

このコマンドは、すべての共有アドレスリソースの拡張プロパティーを一覧表示します。

```
# clressharedaddress list-props -v
Properties       Descriptions
----------       ------------
NetIfList        List of IPMP groups on each node
AuxNodeList      List of nodes on which this resource is available
HostnameList     List of hostnames this resource manages
CheckNameService Name service check flag
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページの`Intro`(1CL), 551 ページの`cluster`(1CL), 273 ページの`clresource`(1CL), 251 ページの`clreslogicalhostname`(1CL), 305 ページの`clresourcegroup`(1CL), 333 ページの`clresourcetype`(1CL), 1047 ページの`scha_calls`(3HA), 1447 ページの`clconfiguration`(5CL), Unresolved link to " rbac5", 1287 ページの`r_properties`(5)

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のオプションを指定してこのコマンドを実行できます。

- `-?` オプション
- `-V` オプション

サブコマンドを指定してこのコマンドを実行する場合、スーパーユーザー以外のユーザーは RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
create	solaris.cluster.modify
delete	solaris.cluster.modify
disable	solaris.cluster.admin
enable	solaris.cluster.admin
export	solaris.cluster.read
list	solaris.cluster.read
list-props	solaris.cluster.read
monitor	solaris.cluster.admin
reset	solaris.cluster.admin
set	solaris.cluster.modify
show	solaris.cluster.read
status	solaris.cluster.read
unmonitor	solaris.cluster.admin

名前

clresourcegroup, clrg — Oracle Solaris Cluster データサービスのリソースグループの管理

```
/usr/cluster/bin/clresourcegroup -V

/usr/cluster/bin/clresourcegroup [subcommand] -?

/usr/cluster/bin/clresourcegroup subcommand [options] -v [resourcegroup …]

/usr/cluster/bin/clresourcegroup add-node -n node[,...] [-S] [-Z {zoneclustername | global}]
 {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup create [-S] [-n node[,...]] [-p name=value] […] [-Z
 {zoneclustername | global}] {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup create -i {- | clconfigfile} [-S] [-n node [,...]] [-
p name=value] […] {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup delete [-F] [-Z {zoneclustername | global}] {+
 | resourcegroup...}

/usr/cluster/bin/clresourcegroup evacuate -n node[,...] [-T seconds] [-Z {zoneclustername
 | global}] {+}

/usr/cluster/bin/clresourcegroup export [-o {- | configfile}] [+ | resourcegroup...]

/usr/cluster/bin/clresourcegroup list [-n node[,...]] [-r resource[,...]] [-s state[,...]] [-
t resourcetype[,...]] [ [-Z {zoneclustername[,...] | global | all}] [+ | resourcegroup...]

/usr/cluster/bin/clresourcegroup manage [-Z {zoneclustername | global}] {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup offline [-n node [,...]] ] [-Z {zoneclustername | global}] {+
 | resourcegroup...}

/usr/cluster/bin/clresourcegroup online [-e] [-m] [-M] [-n node [,...]] [-Z {zoneclustername
 | global}] {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup quiesce [-k] [-Z {zoneclustername | global}] {+
 | resourcegroup...}

/usr/cluster/bin/clresourcegroup remaster [-Z {zoneclustername |
     global}] {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup remove-node -n node
     [,...]  [-Z {zoneclustername | global}]
     {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup restart [-n node[,...]]
     [-Z zoneclustername |global}] {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup resume [-Z {zoneclustername |
     global}] {+ | resourcegroup...}
```

```
/usr/cluster/bin/clresourcegroup set [-n node[,...]] -p name[+|-]=value […]
     [-Z {zoneclustername | global}] {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup show [-n node[,...]]
     [-p name[,...]] [-r resource[,...]] [-t resourcetype[,...]]
     [-Z {zoneclustername[,...] | global | all}]
     [+ | resourcegroup...]

/usr/cluster/bin/clresourcegroup status [-n node[,...]]
     [-r resource [,]...] [-s state [,]...] [-t resourcetype
     [,]...] [-Z {zoneclustername[,...] | global | all}]
     [+ | resourcegroup...]

/usr/cluster/bin/clresourcegroup suspend [-k] [-Z
     {zoneclustername | global}] {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup switch -n node[,...]
     [-e] [-m] [-M] [-Z {zoneclustername | global}]
     {+ | resourcegroup...}

/usr/cluster/bin/clresourcegroup unmanage [-Z {zoneclustername |
     global}] {+ | resourcegroup...}
```

このコマンドは、Oracle Solaris Cluster データサービスのリソースグループを管理します。

subcommand を省略できるのは、*options* が `-?` オプションまたは `-V` オプションの場合のみです。

このコマンドの各オプションには長形式と短形式があります。各オプションの両方の形式は、オプション に説明とともに記載されています。

`clrg` コマンドは、`clresourcegroup` コマンドの短い形式です。

`list`、`show`、および `status` 以外のサブコマンドでは、1 つ以上のオペランドが必要です。ただし、多くのサブコマンドはプラス記号のオペランド (+) を受け付けます。このオペランドは、サブコマンドを*すべての*適用できるオブジェクトに適用します。

このコマンドは、ゾーンクラスタでいくつかの形式で使用できます。このコマンドの有効な使用方法の詳細については、個々のサブコマンドの説明を参照してください。管理しやすいように、広域クラスタノードからこのコマンドを使用してください。

リソースとリソースグループ

リソースのステータス、リソースグループのステータス、リソースのステータスは、すべてノード単位で管理されます。たとえば、リソースは、各クラスタノード上で固有のステータス、固有のステータスを持ちます。

注記 - Offline や Start_failed などの状態の名前は、大文字と小文字が区別*されません*。状態の名前を指定する際には、大文字と小文字を任意に組み合わせることができます。

リソース状態は、そのリソース上でどのメソッドが呼び出されたかにのみ基づいて、各ノード上の Resource Group Manager (RGM) によって設定されます。たとえば、指定のノード上でリソースに対する STOP メソッドを正しく実行した場合、そのリソースのノード上での状態は Offline になります。STOP メソッドが 0 以外またはタイムアウトで終了した場合、そのリソースの状態は Stop_failed になります。

可能性のあるリソース状態には、Online、Offline、Start_failed、Stop_failed、Monitor_failed、Online_not_monitored、Starting、および Stopping があります。

可能性のあるリソースグループの状態には、Unmanaged、Online、Offline、Pending_online、Pending_offline、Error_stop_failed、Online_faulted、および Pending_online_blocked があります。

RGM は、リソースのステータスだけでなく、リソース自体が API を使って設定するリソースのステータスも維持します。Status Message のフィールドは、実際には、ステータスキーワードとステータスメッセージからなります。ステータスメッセージは、ステータスキーワードのあとに出力される任意のテキスト文字列で、リソースによって任意に設定されます。

リソースステータスの値には、次のものがあります。

Degraded	リソースはオンラインですが、何らかの理由でパフォーマンスまたは可用性が低下しています。
Faulted	リソースの機能を妨げるエラーが検出されました。
Offline	リソースはオフラインです。
Online	リソースはオンラインでサービスを提供します。
Unknown	現在のステータスは不明または遷移中です。

ゾーンクラスタでこのコマンドを使用する

clresourcegroup コマンドは、ゾーンクラスタで、export を除くすべてのサブコマンドを指定して使用することができます。

export 以外のすべてのサブコマンドで -Z オプションを使用して、操作を制限する特定のゾーンクラスタの名前を指定することもできます。また、リソースグループ (*zoneclustername*

:*resourcegroup*) にゾーンクラスタ名を付けて、操作を特定のゾーンクラスタに制限することもできます。

グローバルクラスタノードからすべてのゾーンクラスタ情報にアクセスできますが、特定のゾーンクラスタは他のゾーンクラスタを認識しません。特定のゾーンクラスタに操作を制限できない場合、使用するサブコマンドは現在のクラスタでのみ機能します。

ゾーンクラスタのリソースグループと別のゾーンクラスタまたはグローバルクラスタのリソースグループ間のアフィニティーを指定することができます。次のコマンドで別のゾーンクラスタのリソースグループ間のアフィニティーを指定することができます。

```
# clresourcegroup set -p RG_affinities={+|++|-|--}
target-zc:target-rg
source-zc:source-rg
```

アフィニティータイプは次のいずれかとなります。

- + (弱い肯定的)
- ++ (強い肯定的)
- +++ (フェイルオーバー委託付きの強い肯定的)
- - (弱い否定的)
- -- (強い否定的)

たとえば、ゾーンクラスタ ZC1 内のリソースグループ RG1 とゾーンクラスタ ZC2 内のリソースグループ RG2 の間に強い肯定的なアフィニティー (++) を指定する必要がある場合は、次のコマンドを使用します。

```
# clresourcegroup set -p RG_affinities=++ZC2:RG2 ZC1:RG1
```

ゾーンクラスタ ZC1 内のリソースグループ RG1 とゾーンクラスタ ZC2 内のリソースグループ RG2 の間にフェイルオーバー委託付きの強い肯定的なアフィニティー (+++) を指定するには、次のコマンドを使用します。

```
# clresourcegroup set -p RG_affinities=+++ZC2:RG2 ZC1:RG1
```

ゾーンクラスタ ZC1 内のリソースグループ RG1 とグローバルクラスタ内のリソースグループ RG2 の間に強い否定的なアフィニティー (--) を指定するには、次のコマンドを使用します。

```
# clresourcegroup set -p RG_affinities=--global:RG2 ZC1:RG1
```

リソースグループはクラスタノードまたはゾーンに自動的に分散されます。詳細は、1319 ページのrg_properties(5) のマニュアルページにある Load_factors、Priority、および Preemption_mode のエントリを参照してください。

サポートされるサブコマンドには次のものがあります。

add-node

ノードをリソースグループの Nodelist プロパティーの最後に追加します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

リスト内のノードとゾーンの順序は、それらのノードまたはゾーンでリソースグループがオンラインにされる優先順位を指定します。Nodelist プロパティー内の別の位置にノードを追加するには、set サブコマンドを使用します。

広域クラスタノードから特定のゾーンクラスタのリソースグループに対してノードを追加するために、-Z オプションを使用してゾーンクラスタの名前を指定できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

create

新しいリソースグループを作成します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

広域クラスタノードから特定のゾーンクラスタにリソースグループを作成するために、-Z オプションを使用してゾーンクラスタの名前を指定できます。

-i オプションで構成ファイルを指定する場合は、プラス記号のオペランド (+) を指定できます。このオペランドは、存在していないすべてのリソースをそのファイル内に作成することを指定します。

新しいリソースグループの Nodelist プロパティーを設定するには、次に示すオプションのいずれかを指定します。

- -n *node*
- -p Nodelist=-*node-*][,…]
- -i *clconfigfile*

リスト内のノードの順序は、それらのノードでリソースグループをオンラインにする優先順位を指定します。作成時にノードリストを指定しない場合、クラスタ内で構成されているすべてのノードが Nodelist プロパティーに設定されます。順序は任意です。

デフォルトでは、リソースグループは Failover が設定された RG_mode プロパティーで作成されます。ただし、-S オプションまたは -p RG_mode=Scalable オプションを使用するか、Maximum_primaries に 1 より大きい値を指定することで、スケーラブルリソースグループを作成できます。リソースグループの RG_mode プロパティーを設定できるのは、このグループを作成するときだけです。

リソースグループは、作成時には常に管理されない状態に置かれます。ただし、manage サブコマンドを実行したり、online または switch サブコマンドを -M> オプションとともに実行したりすると、RGM の状態は管理状態に変わります。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

delete

リソースグループを削除します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

広域クラスタノードから特定のゾーンクラスタのリソースグループを削除するために、-Z オプションを使用してゾーンクラスタの名前を指定できます。

このサブコマンドとともに正符号のオペランド (+) を指定すると、すべてのリソースグループを削除できます。

リソースグループがリソースを含む場合は、-F オプションを指定しないかぎり、そのリソースグループを削除できません。-F オプションを指定した場合は、各グループ内のすべてのリソースおよびそのグループが削除されます。すべての依存関係とアフィニティーも削除されます。

このサブコマンドは、リソースとリソースグループの依存関係を反映する順序で複数のリソースグループを削除します。コマンド行でリソースグループを指定する順序は関係ありません。

次の clresourcegroup delete コマンドはいくつかのステップで実行されます。

- 複数のリソースグループを同時に削除する場合
- リソースグループを -F オプションで削除する場合

たとえばノードで問題が発見された場合などのように、このいずれかのフォームのコマンドが割り込まれたときには、いくつかのリソースグループが無効な構成のまま残される場合があります。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

evacuate

-n オプションで指定したノード上のすべてのリソースグループをオフラインにします。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタノードから evacuate コマンドを実行すると、このサブコマンドはグローバルクラスタまたはゾーンクラスタのすべてのリソースグループを退避します。ゾーンクラスタで、このサブコマンドは指定したゾーンクラスタのリソースグループのみを退避します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを退避する場合は、-Z オプションを使用してゾーンクラスタの名前を指定できます。

リソースグループは、リソースとリソースグループの依存関係を反映する順序でオフラインにされます。

このサブコマンドの -T オプションを使用して、リソースグループが切り替わらないようにする秒数を指定できます。値を指定しない場合、デフォルトでは、60 秒が使用されます。

退避が完了したあと 60 秒間または指定した秒数の間、リソースグループは退避中ノードでフェイルオーバーできなかったり自動的にオンラインになれなくなったりします。

ただし、switch または online サブコマンドを使用して、リソースグループをオンラインまたは退避ノードリブートに切り替えると、退避タイマーはただちに期限切れになり、自動フェイルオーバーが再度許可されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`export`

リソースグループの構成情報をファイルまたは標準出力 (stdout) に書き込みます。

このサブコマンドは、大域ゾーンだけで使用できます。

この構成情報の形式は、1447 ページのclconfiguration(5CL) のマニュアルページで説明されています。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`list`

指定したリソースグループの修飾子オプションでフィルタリングされたリストを表示します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

-r *resource* を使用すると、リソースを含むリソースグループのみを含めることができます。-t *resourcetype* を使用すると、*resourcetype* 内のリソースタイプを含むリソースグループのみを含めることができます。-n *node* を使用すると、1 つ以上のノード内でオンラインであるリソースグループのみを含めることができます。

-s *state* を指定すると、指定した状態のグループのみが一覧表示されます。

オペランドを指定しない場合や正符号のオペランド (+) を指定した場合は、指定した任意の修飾子オプションでフィルタリングされたすべてのリソースグループが一覧表示されます。

詳細オプション -v を指定すると、リソースグループがオンラインかオフラインにかかわらず、そのステータスが表示されます。クラスタ内の 1 つのノードだけでオンラインの場合でも、リソースグループはオンラインとして一覧表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`manage`

指定したリソースグループを管理状態にします。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾーンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを管理するには、-Z オプションを使用してゾーンクラスタの名前を指定できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

offline

指定したリソースグループをオフライン状態にします。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾーンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを退避するには、-Z オプションを使用してゾーンクラスタの名前を指定できます。

-n オプションを指定すると、リソースグループは、指定したノードでのみオフラインになります。

-n オプションを指定しないと、リソースグループはすべてのノード上でオフラインになります。

リソースグループを offline サブコマンドでオフラインにした場合、リソースグループの Offline 状態はノードのリブートで解除されます。つまり、ノードを終了またはクラスタに接続すると、以前にオフラインにしたリソースグループが一部のノード上でオンラインになる可能性があります。すべてのリソースを無効にしていても、リソースグループはオンラインになります。

同様に、任意の RG_dependencies または強い RG_affinities を示すリソースグループは、別のリソースグループに切り替わると、自動的のオンラインになります。

リソースグループが自動的にオンラインになるのを防止するには、suspend サブコマンドを使用してリソースグループの自動復旧処理を中断します。自動復旧処理を再開するには、resume サブコマンドを使用します。

リソースグループは、リソースとリソースグループの依存関係を反映する順序でオフラインにされます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

online

指定したリソースグループをオンライン状態にします。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾー

ンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを退避するには、-Z オプションを使用してゾーンクラスタの名前を指定できます。

リソースグループをオンラインにするノードのリストを指定するには、-n オプションを使用します。-n オプションを指定しない場合、このサブコマンドは、リソースグループを現在のプライマリノードからオフラインにせずに、リソースグループを最優先ノードでオンラインにします。リソースグループごとのオンラインノードの総数は、Desired_primaries および Maximum_primaries プロパティーによって制限されます。ノードの優先順序は、Nodelist、RG_affinities、および Load_factors プロパティーによって決定されます。これらのプロパティーの詳細は、1319 ページのrg_properties(5) のマニュアルページを参照してください。

コマンド行で複数のリソースグループのオペランドが指定され、-n オプションが指定されていない場合、それらのリソースグループのオペランドには Priority プロパティーによって決定される順序でプライマリノードが割り当てられます。つまり、優先順位のもっとも高いリソースグループが最初にノードの割り当てを受けます。プライマリノードが割り当てられたあと、リソースの依存関係またはリソースグループの依存関係によって制約されている場合を除き、すべてのリソースグループオペランドが並列でオンラインになります。コマンド行でリソースグループを指定する順序は関係ありません。Priority プロパティーの詳細は、1319 ページのrg_properties(5) のマニュアルページを参照してください。

優先順位の低いリソースグループは、最優先ノードに割り当てられなかったり、負荷制限を超えている場合は優先順位の高いリソースグループによって強制的にオフラインにされたりすることがあります。詳細は、185 ページのclnode(1CL) のマニュアルページの loadlimit サブコマンドを参照してください。

switch サブコマンドと異なり、このサブコマンドは Nodelist プロパティーで一覧表示されているどのノードも Offline 状態にしようとしません。

このサブコマンドとともに -e オプションを指定すると、オンラインにされるリソースグループのセット内のすべてのリソースが有効になります。

-m オプションを指定すると、オンラインにされるリソースグループのセット内のすべてのリソースのモニタリングを有効にできます。ただし、最初に有効にされ、MONITOR_START メソッドと関連付けられている場合以外、リソースが実際にモニターされることはありません。

-M オプションを指定して、オンラインにされるすべてのリソースグループを管理状態に置くように指示することもできます。-M オプションが指定されていない場合、このサブコマンドは、管理されていないリソースグループには効果を持ちません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

quiesce

指定されたリソースグループを休止状態にします。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾーンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを退避するには、-Z オプションを使用してゾーンクラスタの名前を指定できます。

このコマンドは、START または STOP メソッドが失敗した場合に、あるノードから別のノードにリソースグループが連続して切り替わるのを阻止します。また、停止メソッドが失敗したとき、リソースの Failover_mode プロパティーが HARD に設定されている場合に通常実行されるノードのリブートも阻止します。その場合、リソースは代わりに STOP_FAILED 状態に移動します。

影響されるリソースグループ内のリソースの代わりに実行されているメソッドを終了するには、-k オプションを使用します。-k オプションを指定しないと、終了するか構成されているタイムアウトを超えるまで継続的にメソッドを実行できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

remaster

指定したリソースグループを、現在のプライマリノードから最優先ノードに切り替えます。各リソースグループのオンラインノードの合計数は、 Desired_primaries および Maximum_primaries プロパティーによって制限されます。ノードの優先順序は、Nodelist、RG_affinities、および Load_factors プロパティーによって決定されます。詳細は、185 ページのclnode(1CL) および 1319 ページのrg_properties(5) のマニュアルページを参照してください。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾーンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを退避するには、-Z オプションを使用してゾーンクラスタの名前を指定できます。

online サブコマンドと異なり、このサブコマンドでは、リソースグループを現在のマスターからオフラインに切り替えて、より優先されるマスター上でオンラインにすることができます。

コマンド行で複数のリソースグループオペランドを指定すると、Priority プロパティーで決定されている順序でリソースグループオペランドがプライマリノードに割り当てられます。優先順位がもっとも高いリソースグループに最初にノードが割り当てられます。コマンド行でリソースグループを指定する順序は関係ありません。詳細は、1319 ページのrg_properties(5) のマニュアルページを参照してください。

優先順位の低いリソースグループは、最優先ノードに割り当てられなかったり、負荷制限を超えている場合は優先順位の高いリソースグループによって強制的にオフラインにされたりすることがあります。詳細は、185 ページのclnode(1CL) のマニュアルページの loadlimit サブコマンドを参照してください。

このサブコマンドは、管理されていないリソースグループには影響を与えません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

remove-node

リソースグループの Nodelist プロパティーからノードを削除します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドは、グローバルクラスタノードまたはゾーンクラスタから使用できます。グローバルクラスタノードからゾーンクラスタ内のリソースグループのノードを削除するには、-z オプションを使用してゾーンクラスタの名前を指定できます。

ノードを削除すると、remove-node は Maximum_primaries または Desired_primaries プロパティーの値を Nodelist プロパティー内の新しいノード数にリセットする場合があります。remove-node が Maximum_primaries または Desired_primaries プロパティーの値をリセットするのは、いずれかの値が Nodelist プロパティー内の新しいノード数を超える場合のみです。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

restart

現在リソースグループをホストしているプライマリノードの同じセット上でリソースグループをオフラインにしてからオンラインに戻します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾーンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを退避するには、-z オプションを使用してゾーンクラスタの名前を指定できます。

-n オプションを指定すると、リソースグループは、指定するノードのリストにある現在のマスターでのみ再起動されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

resume

suspend サブコマンドによって以前中断されていた指定のリソースグループで自動復旧処理を再開します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾーンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定

のゾーンクラスタのリソースグループを退避するには、-Z オプションを使用してゾーンクラスタの名前を指定できます。

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

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.admin` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`set`

指定するリソースグループに関連付けられているプロパティーを変更します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾーンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを退避するには、-Z オプションを使用してゾーンクラスタの名前を指定できます。

`Nodelist` プロパティーは、`-p Nodelist=`*node* で、または便宜的に `-n` *node* で変更できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`show`

指定したリソースグループの修飾子オプションでフィルタリングされた構成レポートを生成します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

-r*resource* を使用すると、リソースを含むリソースグループのみを含めることができます。-t*resourcetype* を使用すると、*resourcetype* 内のリソースタイプを含むリソースグループのみを含めることができます。`-n` *node* を使用すると、1 つ以上のノード内でオンラインであるリソースグループのみを含めることができます。広域クラスタから -Z オプションを使用すると、指定されたゾーンクラスタでオンラインになるこれらのリソースグループのみを含めることができます。

-p オプションを使用すると、すべてのリソースグループプロパティーではなく、選択されたリソースグループプロパティーのセットを表示できます。

オペランドを指定しない場合や正符号のオペランド (+) を指定した場合は、指定した任意の修飾子オプションでフィルタリングされたすべてのリソースグループが一覧表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`status`

指定したリソースグループの修飾子オプションでフィルタリングされたステータスレポートを生成します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

ゾーンクラスタでこのコマンドを使用すると、このサブコマンドはゾーンクラスタのリソースグループにのみ適用されます。

`-r` *resource* を使用すると、リソースを含むリソースグループのみを含めることができます。`-t` *resourcetype* を使用すると、*resourcetype* 内のリソースタイプを含むリソースグループのみを含めることができます。`-n` *node* を使用すると、1 つ以上のノード内でオンラインであるリソースグループのみを含めることができます。`-Z` オプションを使用すると、広域クラスタノードからゾーンクラスタを指定して、指定されたゾーンクラスタでオンラインになるこれらのリソースグループのみを含めることができます。

`-s` *state* を指定すると、指定した状態のグループのみが一覧表示されます。

注記 - `status` サブコマンドでは、`-n` オプションまたは `-s` オプションのいずれかを指定できます。ただし、両方のオプションを同時に `status` サブコマンドとともに指定できません。

オペランドを指定しない場合や正符号のオペランド (+) を指定した場合は、指定した任意の修飾子オプションでフィルタリングされたすべてのリソースグループが一覧表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`suspend`

指定されたリソースグループ上の自動復旧アクションを中断し、そのリソースグループを休止します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾーンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを退避するには、`-Z` オプションを使用してゾーンクラスタの名前を指定できます。

自動復旧を再開するコマンドを明示的に実行するまで、*中断されたリソースグループが*自動的に再開またはフェイルオーバーされることはありません。オンラインかオフラインかにかかわらず、中断されたデータサービスは現在の状態のままです。リソースグループが一時停止されている間、`clresourcegroup`(1CL) または `clresource`(1CL) コマンドを `switch`、`online`、`offline`、`disable`、`enable` などのサブコマンドとともに使用すると、リ

ソースグループまたはそのリソースを特定のノード上の別の状態に手動で切り替えることができます。アプリケーションプロセスの強制終了やアプリケーション固有のコマンドの実行など、リソースに対して直接に操作するのではなく、clresourcegroup(1CL) または clresource(1CL) コマンドを使用してください。これにより、クラスタのフレームワークがリソースおよびリソースグループの現在のステータスを正確に把握することができ、resume サブコマンドが実行されたときに可用性が正しく復元されます。

リソースグループの自動回復を中断する必要があるのは、クラスタの問題を調査して修正する場合、あるいは、リソースグループサービスに保守を実行する場合などです。

また、-k オプションを指定すると、影響されるリソースグループ内のリソースの代わりに実行されているメソッドを即座に終了できます。-k オプションを使用することで、リソースグループの休止を早めることができます。-k オプションを指定しない場合、メソッドは終了するか、または構成されているタイムアウトを超えるまで実行を継続することを許可されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

switch

指定したリソースグループをマスターするノードまたはノードのセットを変更します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、ゾーンクラスタのリソースグループでのみ正常に動作します。

リソースグループをオンラインにするノードのリストを指定するには、-n オプションを使用します。-Z オプションを使用すると、広域クラスタノードからゾーンクラスタを指定して、指定されたゾーンクラスタのリソースグループのリストのみを含めることができます。

リソースグループがまだオンラインでない場合は、-n オプションで指定されているノードのセット上でオンラインになります。ただし、これらのグループが新しいノード上でオンラインになる前に、-n オプションで指定されていないノード上でオンラインのグループがオフラインになります。

このサブコマンドとともに -e を指定すると、オンラインにされるリソースグループのセット内のすべてのリソースが有効になります。

-m を指定すると、オンラインにされるリソースグループのセット内のすべてのリソースのモニタリングを有効にできます。ただし、最初に有効にされ、MONITOR_START メソッドと関連付けられている場合以外、リソースが実際にモニターされることはありません。

-M オプションを指定して、オンラインにされるすべてのリソースグループを管理状態に置くように指示することもできます。-M オプションが指定されていない場合、このサブコマンドは、管理されていないリソースグループには効果を持ちません。

リソースグループは、リソースとリソースグループの依存関係を反映する順序でオンラインにされます。コマンド行でグループを指定する順序は関係ありません。

優先順位の低いリソースグループは、指定されたノードに切り替えられなかったり、負荷制限を超えている場合は優先順位の高いリソースグループによって強制的にオフラインに

されたりすることがあります。詳細は、185 ページのclnode(1CL) のマニュアルページの loadlimit サブコマンドを参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

unmanage

指定したリソースグループを管理されない状態にします。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドをグローバルクラスタノードで使用する場合、このサブコマンドは、任意のリソースグループに対して機能します。ゾーンクラスタでこのサブコマンドを使用すると、同じゾーンクラスタのリソースグループでのみ正常に動作します。グローバルクラスタノードから特定のゾーンクラスタのリソースグループを退避するには、-Z オプションを使用してゾーンクラスタの名前を指定できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

次のオプションがサポートされています。

注記 - このセクションでは、各オプションの短い形式と長い形式の両方が示されています。

-?
--help

ヘルプ情報を表示します。

このオプションは*サブコマンド*付きでもサブコマンドなしでも指定できます。

このオプションを *subcommand* なしで指定すると、使用可能なすべてのサブコマンドのリストが表示されます。

このオプションを *subcommand* とともに指定すると、*subcommand* の使用法が表示されます。

このオプションを create または set サブコマンドで指定した場合は、すべてのリソースグループプロパティーのヘルプ情報が表示されます。

このオプションをほかのオプション、サブコマンド、またはオペランドと一緒に指定する場合、これらはすべて無視されます。ほかの処理は行われません。

-e
--enable

グループがオンラインにされる時点でリソースグループ内のすべてのリソースを有効にします。

このオプションは switch および online サブコマンドとだけ使用できます。

-F
--force

リソースが有効またはオンラインであっても、リソースグループおよびそのすべてのリソースを強制的に削除します。また、このオプションは、ほかのリソースおよびほかのリソースグループ内のすべての依存関係プロパティー設定またはアフィニティープロパティー設定からもリソースとリソースグループの両方を削除します。

-F オプションを delete サブコマンドとともに使用する場合は、注意が必要です。依存関係やアフィニティーが設定されている場合などは、強制削除により、削除されたリソースグループを参照するその他のリソースグループが変更される場合があります。強制削除後、依存リソースが無効またはエラー状態のまま残されることもあります。このような現象が発生した場合は、影響を受ける依存リソースを再構成または再起動する必要がある場合があります。

-i {- | *clconfigfile*}
--input={- | *clconfigfile*}
--input {- | *clconfigfile*}

clconfigfile ファイル内にある構成情報を使用することを指定します。1447 ページのclconfiguration(5CL) のマニュアルページを参照してください。

標準入力 stdin を通して構成情報を提供するには、このオプションとともにダッシュ - を指定します。

ほかのオプションを指定した場合は、*clconfigfile* 内のオプションや情報より指定したオプションが優先されます。

指定したリソースグループだけがこのオプションの影響を受けます。

-k
--kill

指定したリソースグループ内のリソースの代わりに実行されている RGM のリソースメソッドを終了します。

このオプションは、quiesce および suspend サブコマンドで使用できます。-k オプションを指定しないと、終了するか構成されているタイムアウトを超えるまで継続的にメソッドを実行できます。

-m
--monitor

リソースグループがオンラインにされる時点でリソースグループ内のすべてのリソースのモニタリングを有効にします。

ただし、最初に有効にされ、MONITOR_START メソッドと関連付けられている場合以外、リソースが実際にモニターされることはありません。

このオプションは switch および online サブコマンドとだけ使用できます。

-M
--manage

サブコマンドによってオンラインにされるすべてのリソースグループが管理状態となることを指定します。

-n *node*[,…]]
--node=*node*[,…]]
--node *node*[,…]]

ターゲットのグローバルクラスタまたはゾーンクラスタに、ノードまたはノードのリストを指定します。-Z オプションが指定されている場合は、-n オプションで、グローバルクラスタホスト名ではなくゾーンクラスタホスト名のみを指定できます。-Z オプションが指定されていない場合は、-n オプションでグローバルクラスタホスト名のみを指定できます。

node には、ノードの名前または識別子を指定できます。

list、show、および status サブコマンドとともに使用されると、このオプションは出力を限定します。ノードリスト内の 1 つ以上のノードで現在オンラインのリソースグループのみが対象となります。

このオプションを create、add-node、remove-node、および set サブコマンドで指定することは、Nodelist プロパティーを設定することと同じです。Nodelist プロパティー内のノードの順序は、それらのノードでグループがオンラインになる順序を指定します。create サブコマンドでノードリストを指定しない場合は、クラスタ内のすべてのノードが Nodelist プロパティーに設定されます。順序は任意です。

switch および online サブコマンドとともに使用するときは、このオプションによりリソースグループをオンラインにするノードを指定します。

evacuate および offline サブコマンドとともに使用するときは、このオプションによりリソースグループをオフラインにするノードを指定します。

restart サブコマンドとともに使用するときは、このオプションによりリソースグループを再起動するノードを指定します。リソースグループは、指定されたリストにある現在のマスターで再起動されます。

-o {- | *clconfigfile*}
--output={- | *clconfigfile*}
--output {- | *clconfigfile*}

リソースグループの構成情報をファイルまたは標準出力 stdout に書き込みます。この構成情報の形式は、1447 ページのclconfiguration(5CL) のマニュアルページで説明されています。

このオプションにファイル名を指定する場合、このオプションは新しいファイルを作成します。次に、ノード構成情報はそのファイルに格納されます。- をこのオプションとともに指定すると、構成情報は標準出力 (stdout) に送信されます。このコマンドのほかの標準出力はすべて抑制されます。

このオプションを一緒に指定できるのは、export サブコマンドだけです。

-p *name*
--property=*name*
--property *name*

リソースグループのプロパティーのリストを指定します。

このオプションは、show サブコマンドとともに使用します。

create または set サブコマンドで設定または変更できるプロパティーについては、-p*name=value* オプションの説明を参照してください。

このオプションを指定しないと、show サブコマンドはほとんどのリソースグループプロパティーを一覧表示します。このオプションを指定せず、-verbose オプションを show サブコマンドで指定した場合、このサブコマンドはすべてのリソースグループプロパティーを一覧表示します。

指定できるリソースグループプロパティーは、Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド のリソースグループプロパティー"に記載されています。

-p *name=value*
-p *name+=array-values*
-p *name=array-values*
--property=*name=value*
--property=*name+=array-values*
--property=*name-=array-values*
--property *name=value*
--property *name+=array-values*
--property *name-=array-values*

リソースグループプロパティーの値を設定または変更します。

このオプションは create および set サブコマンドとだけ使用できます。

show サブコマンドで情報を表示できるプロパティーについては、-p *name* オプションの説明を参照してください。

-p は複数のインスタンスが許可されます。

このオプションとともに使用する演算子は、次のとおりです。

=
: プロパティーに、指定した値を設定します。create および set サブコマンドがこの演算子を受け付けます。

+=
: 1 つ以上の値をプロパティー値のリストに追加します。この演算子は、set サブコマンドでのみ使用できます。この演算子は、たとえば、Nodelist のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。

–=
: 1 つ以上の値をプロパティー値のリストから削除します。この演算子は、set サブコマンドでのみ使用できます。この演算子は、たとえば、Nodelist のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。

-r *resource*[,…]
--resource=*resource*[,…]
--resource *resource*[,…]

リソースまたはリソースのリストを指定します。

このオプションは、list、show、および status サブコマンドでのみ使用できます。このオプションは、これらのコマンドからの出力を制限します。リソースリスト内に 1 つ以上のリソースを含むリソースグループだけが出力されます。

-s *state*[,…]
--state=*state*[,…]
--state *state*[,…]

リソースグループの状態またはリソースグループの状態のリストを指定します。

このオプションを一緒に指定できるのは、evacuate サブコマンドだけです。このオプションは、指定した任意のノード上で指定した状態にあるリソースグループのみが表示されるように、出力を制限します。このオプションとともに次に示す引数を 1 つ以上指定できます。引数は、状態を表します。

Error_stop_failed

指定した任意のノード上で Error_stop_failed 状態のリソースグループがすべて表示されます。

Not_online

指定した任意のノード上で online 以外の状態の指定したリソースグループがすべて表示されます。

Offline

指定したすべてのノードで Offline 状態の場合にのみ、指定したリソースグループが表示されます。

Online

指定した任意のノード上で Online 状態の指定したリソースグループがすべて表示されます。

Online_faulted

指定した任意のノード上で Online_faulted 状態のリソースグループがすべて表示されます。

Pending_offline

指定した任意のノード上で Pending_offline 状態のリソースグループがすべて表示されます。

Pending_online

指定した任意のノード上で Pending_online 状態のリソースグループがすべて表示されます。

`Pending_online_blocked`

指定した任意のノード上で `Pending_online_blocked` 状態のリソースグループがすべて表示されます。

`Unmanaged`

指定した任意のノード上で `Unmanaged` 状態の指定したリソースグループがすべて表示されます。

`-S`
`--scalable`

スケーラブルリソースグループを作成するか、または `Maximum_primaries` および `Desired_primaries` プロパティーを更新します。

このオプションは `create` および `add-nod` サブコマドとだけ使用できます。

`create` サブコマンドと使用された場合、このオプションはフェイルオーバーリソースグループではなくスケーラブルリソースグループを作成します。また、このオプションは `Maximum_primaries` および `Desired_primaries` の両方のプロパティーを、結果となる `Nodelist` プロパティー内のノードの数に設定します。

このオプションを `add-node` サブコマンドと使用できるのは、リソースグループがすでにスケーラブルな場合だけです。`add-node` サブコマンドとともに使用した場合、このオプションは `Maximum_primaries` および `Desired_primaries` の両方のプロパティーを、結果となる `Nodelist` プロパティー内のノードの数に更新します。

また、`RG_mode`、`Maximum_primaries`、および `Desired_primaries` プロパティーを `-p` オプションで設定することもできます。

`-t` *resourcetype*[,…]
`--type=`*resourcetype*[,…]
`--type` *resourcetype*[,…]

1 つのリソースタイプまたはリソースタイプのリストを指定します。

このオプションは、`list`、`show`、および `status` サブコマンドでのみ使用できます。このオプションは、これらのコマンドからの出力を制限します。リソースタイプリストに含まれているタイプのリソースを 1 つ以上含むリソースグループだけが出力されます。

[*prefix* としてリソースタイプを指定しています。]*type*[:*RT-version*]. たとえば、`nfs` リソースタイプは、`SUNW.nfs:3.2`、`SUNW.nfs`、または `nfs` として表される可能性があります。*RT-version* を含める必要があるのは、そのクラスタで登録されているリソースタイプの複数のバージョンが存在する場合だけです。*prefix* を含めない場合は、`SUNW` が使用されます。

`-T` *seconds*
`--time=`*seconds*
`--time` *seconds*

あるノードからリソースグループを退避したあと、そのノードにリソースグループがスイッチバックしないようにする時間を秒数で指定します。

このオプションを一緒に指定できるのは、evacuate サブコマンドだけです。*seconds* には、0 から 65535 までの整数値を指定する必要があります。値を指定しない場合、デフォルトでは、60 秒が使用されます。

退避が完了したあと 60 秒間または指定した秒数退避ノードになっていると、、リソースグループはフェイルオーバーできなかったり、自動的にオンラインになったりします。

ただし、switch または online サブコマンドを使用してリソースグループをオンライン、または退避されたノードリブートに切り替えると、退避タイマーはただちに期限切れになり、自動フェイルオーバーが再度許可されます。

-T オプションは、退避の完了後 *T* 秒間、退避したノードの RGM によってリソースグループがオンラインにならないことを指定します。-n オプション付きで switch または online サブコマンドを使用することで退避されたノードにリソースグループを切り替えることによって、-T タイマーをオーバーライドできます。そのような切り替えが完了すると、-T タイマーはただちにそのノードに対して期限切れになります。ただし、-n フラグのない online または remaster などのスイッチオーバーコマンドは、-T タイマーを引き続き認識し、任意のリソースグループが退避されたノードに切り替わるのを避けます。

-u

+ オペランドを使用する場合、このオプションは、リソースグループが中断されたリソース上でコマンドが機能するように指定します。

+ オペランドを指定するとき -u オプションを指定しないと、コマンドは停止されたすべてのリソースグループを無視します。-u オプションは、+ オペランドが add-node、manage、offline、online、quiesce、remaster、remove-node、restart、set、switch、または unamanage サブコマンドで指定されている場合に有効です。

+ オペランドを add-node、manage、offline、online、quiesce、remaster、remove-node、restart、set、switch、または unamanage サブコマンドで使用した場合は、-u オプションも指定しないかぎり、このコマンドは中断されたすべてのリソースグループを無視します。

-v
--verbose

詳細情報を標準出力 (stdout) で表示します。

-V
--version

コマンドのバージョンを表示します。

このオプションをほかのオプション、サブコマンド、またはオペランドと一緒に指定する場合、これらはすべて無視されます。コマンドのバージョンだけが表示されます。ほかの処理は行われません。

-Z {*zoneclustername* | global | all}
--zoneclustername={*zoneclustername* | global | all}
--zoneclustername {*zoneclustername* | global | all}

リソースグループが存在し、操作する 1 つまたは複数のクラスタを指定します。

このオプションは、export サブコマンドを除くすべてのサブコマンドでサポートされています。

このオプションを指定する場合は、次のいずれかの引数も指定する必要があります。

zoneclustername このオプションを使用するコマンドが、*zoneclustername* という名前のゾーンクラスタでのみ指定されたすべてのリソースグループで作動するように指定します。

`global` このオプションを使用するコマンドが、広域クラスタでのみ指定されたすべてのリソースグループで作動するように指定します。

`all` 広域クラスタでこの引数を使用すると、その引数とともに使用するコマンドが、広域クラスタおよびすべてのゾーンクラスタを含め、すべてのクラスタで指定されたすべてのリソースグループで作動するように指定します。

ゾーンクラスタでこの引数を使用すると、その引数とともに使用するコマンドがそのゾーンクラスタでのみ指定されたすべてのリソースグループで作動するように指定します。

次のオペランドがサポートされています。

resourcegroup 管理するリソースグループの名前。

`+` すべてのリソースグループ。

このコマンドセットにあるすべてのコマンドの終了ステータスコードの完全なリストについては、19 ページの`Intro`(1CL) のマニュアルページを参照してください。返される終了コードも1047 ページの`scha_calls`(3HA) のマニュアルページで説明されているリターンコードに準拠しています。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (`CL_NOERR`) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

`0 CL_NOERR`

エラーなし

実行したコマンドは正常に終了しました。

`1 CL_ENOMEM`

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

35 CL_EIO

I/O エラー

物理的な入出力エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

38 CL_EBUSY

オブジェクトはビジーです

アクティブなクラスタノードへの最後のクラスタインターコネクトパスからケーブルを取り外そうとしました。または、参照を削除していないクラスタ構成からノードを削除しようとしました。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

例 173 新しいフェイルオーバーリソースグループの作成

次の例の 1 番目のコマンドは、フェイルオーバーリソースグループ rg1 と rg2 を作成します。2 番目のコマンドは、構成ファイル cluster-1.xml に含まれているリソースをこれらのリソースグループに追加します。

```
# clresourcegroup create rg1 rg2
# clresource create -g rg1,rg2 -i /net/server/export/cluster-1.xml +
```

次の 2 つの例はどちらも、グローバルクラスタノードからゾーンクラスタ ZC 内にフェイルオーバーリソースグループ rg1 および rg2 を作成します。

```
# clresourcegroup create -Z ZC rg1 rg2
# clresourcegroup create ZC:rg1 ZC:rg2
```

例 174 すべてのリソースグループのオンライン化

次のコマンドは、すべてのリソースを有効かつモニターされている状態にし、すべてのリソースグループをオンラインにします。

```
# clresourcegroup online -eM +
```

例 175 Nodelist プロパティーへのノードの追加

次のコマンドは、ノード phys-schost-4 をすべてのリソースグループの Nodelist プロパティーに追加します。

```
# clresourcegroup set -p Nodelist+=phys-schost-4 +
```

例 176 すべてのリソースグループのノードからの退避

次のコマンドは、ノード phys-schost-3 からすべてのリソースグループを退避します。

```
# clresourcegroup evacuate -n phys-schost-3 +
```

例 177 すべてのノード上でのリソースグループのオフライン化

次のコマンドは、すべてのノード上でリソースグループ rg1をオフラインにします。

```
# clresourcegroup offline rg1
```

例 178 リソースグループ、マネージャーの構成全体のリフレッシュ

次の例の 1 番目のコマンドは、すべてのリソースおよびリソースグループを、それらが有効でオンラインの場合でも削除します。2 番目のコマンドは、すべてのリソースタイプの登録を解除します。3 番目のコマンドは、構成ファイル `cluster-1.xml` に含まれているリソースを作成します。また、3 番目のコマンドは、リソースのリソースタイプを登録し、そのリソースタイプが依存するすべてのリソースグループを作成します。

```
# clresourcegroup delete --force +
# clresourcetype unregister +
# clresource -i /net/server/export/cluster-1.xml -d +
```

例 179 すべてのリソースグループの一覧表示

次のコマンドはすべてのリソースグループを一覧表示します。

```
# clresourcegroup list
rg1
rg2
```

例 180 すべてのリソースグループとそのリソースの一覧表示

次のコマンドは、すべてのリソースグループをそのリソースとともに一覧表示します。`rg3` にはリソースはありません。

```
# clresourcegroup list -v
Resource Group Resource
-------------- --------
rg1            rs-2
rg1            rs-3
rg1            rs-4
rg1            rs-5
rg2            rs-1
rg3            -
```

例 181 特定のリソースを含むすべてのリソースグループの一覧表示

次のコマンドは、Oracle Solaris Cluster HA for NFS のリソースを含むすべてのグループを一覧表示します。

```
# clresource list -t nfs
rg1
```

例 182 リソースグループのスイッチオーバーによる Start_failed リソース状態のクリア

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

Stop_failed リソース状態とは異なり、Start_failed リソース状態は、ユーザーまたは Oracle Solaris Cluster ソフトウェアがリソースグループに対してアクションを実行することを妨げ*ません*。reset サブコマンドを実行して Start_failed リソース状態を解除する必要はありません。該当リソースを再起動するコマンドを実行するだけで済みます。

次のコマンドは、resource-grp-2 リソースグループ内のリソース上で発生した Start_failed リソース状態を解除します。このコマンドは、リソースグループを schost-2 ノードに切り替えることで、この状態を解除します。

```
# clresourcegroup switch -n schost-2 resource-grp-2
```

例 183 リソースグループの再起動による Start_failed リソース状態のクリア

次のコマンドは、resource-grp-2 リソースグループ内のリソース上で発生した Start_failed リソース状態を解除します。このコマンドは、元々リソースグループをホストしていた schost-1 ノード上でリソースグループを再起動することで、この状態を解除します。

```
# clresourcegroup restart resource-grp-2
```

例 184 load_factors プロパティーを設定する

次のコマンドは、2 つのリソースグループの負荷係数を設定します。

```
# clresourcegroup set -p load_factors=factor1@50,factor2@1 rg1 rg2
```

次のコマンドは、グローバルクラスタから、ゾーンクラスタ内の 2 つのリソースグループの負荷係数を設定します。

```
# clresourcegroup set -Z ZC load_factors=factor1@50,factor2@1 rg1 rg2
```

例 185 リソースグループの priority プロパティーを設定する

次のコマンドは、リソースグループの優先順位を設定します。

```
# clresourcegroup set -p priority=600 rg1
```

rg1 リソースグループは、ノード割り当てに関して、優先順位の低いリソースグループよりも優先されます。rg1 は、強い制限値を超えているノードにおいて、優先順位の低いほかのリソースグループを横取りできます。rg1 の優先順位が別のリソースグループの優先順位と比べて 100 以上高い場合、弱い制限値を超えているノードにおいて、そのリソースグループを横取りできます。`priority` のデフォルト値は *500* です。

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	`ha-cluster/system/core`
インタフェースの安定性	発展中

273 ページの`clresource`(1CL), 333 ページの`clresourcetype`(1CL), 551 ページの`cluster`(1CL), 19 ページの`Intro`(1CL), Unresolved link to " su1M", 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5", Unresolved link to " rbac5", 1319 ページの`rg_properties`(5), 1447 ページの`clconfiguration`(5CL)

スーパーユーザーはこのコマンドのすべての形式を実行できます。

すべてのユーザーがこのコマンドに `-?` (ヘルプ) オプションまたは `-V` (バージョン) オプションを指定して実行できます。

`clresourcegroup` コマンドをほかのサブコマンドとともに実行する場合、スーパーユーザー以外のユーザーは RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
`add-node`	`solaris.cluster.modify`
`create`	`solaris.cluster.modify`
`delete`	`solaris.cluster.modify`
`evacuate`	`solaris.cluster.admin`
`export`	`solaris.cluster.read`
`list`	`solaris.cluster.read`
`manage`	`solaris.cluster.admin`

サブコマンド	RBAC の承認
offline	solaris.cluster.admin
online	solaris.cluster.admin
quiesce	solaris.cluster.admin
remaster	solaris.cluster.admin
remove-node	solaris.cluster.modify
restart	solaris.cluster.admin
resume	solaris.cluster.admin
set	solaris.cluster.modify
show	solaris.cluster.read
status	solaris.cluster.read
suspend	solaris.cluster.admin
switch	solaris.cluster.admin
unmanage	solaris.cluster.admin

名前

clresource, clrs — Oracle Solaris Cluster データサービスのリソースの管理

```
/usr/cluster/bin/clresource subcommand [-?]

/usr/cluster/bin/clresource -V

/usr/cluster/bin/clresource subcommand [options] -v [resource]...

/usr/cluster/bin/clresource clear [-f errorflag] [-g
     [resourcegroup,]...] [-t [resourcetype,]...] -n node
     [,…]] [-Z {zoneclustername | global}]
     {+ | resource...}

/usr/cluster/bin/clresource create -g resourcegroup -t
     resourcetype [-d] [-p "property-name[{node-specifier,...}]"=
     value] [-x "extension-property[{node-specifier,...}]"=value] [-y
     standard-property=value] [-Z {zoneclustername | global}]
     resource

/usr/cluster/bin/clresource create -i {- | clconfiguration} -t
     resourcetype [-a] [-d] [-g [resourcegroup,]...] [-p "
     property-name[{node-specifier,...}]"=value] [-x "
     extension-property[{node-specifier,...}]"=value] [-y
     standard-property=value] {+ | resource...}

/usr/cluster/bin/clresource delete [-F] [-g [resourcegroup,]...]
     [-t [resourcetype,...]] [-Z {zoneclustername | global}]
     {+ | resource...}

/usr/cluster/bin/clresource disable [-r] [-g [resourcegroup,]...]
     [-t [resourcetype,]...] [-n node[,…]]
     [-Z {zoneclustername | global}] {+ | resource...}

/usr/cluster/bin/clresource enable [-r] [-g [resourcegroup,]...]
     [-t [resourcetype,]...] [ -n node[,…]]
     [-Z {zoneclustername | global}] {+ | resource...}

/usr/cluster/bin/clresource export [-o {- | configfile}]
     [+ | resource...]

/usr/cluster/bin/clresource list [-g [resourcegroup,]...] [-t
     [resourcetype,]...] [ -n node[,…]] [-Z
     {zoneclustername [,...] | global | all}] [+ | resource...]

/usr/cluster/bin/clresource list-props [-l listtype] [-g
     [resourcegroup,]...] [-p "property-name[{node-specifier,…}]",...]
     [-t [resourcetype,]...] [-x "extension-property[{node-specifier,…}]"...]
     [-y "standard-property[{node-specifier,…}]",...] [-Z
```

```
     {zoneclustername [,...] | global | all}] [+ | resource...]

/usr/cluster/bin/clresource monitor [-g [resourcegroup,]...] [-t
     [resourcetype,]...] [ -n node[,…]] [-Z
     {zoneclustername | global}] {+ | resource...}

/usr/cluster/bin/clresource set [-g [resourcegroup,]...] [-p "
     property-name[{node-specifier,…}]"=value] [-t
     [resourcetype,]...] [-x "extension-property[{node-specifier,…}]"=
     value] [-y standard-property [+= | -=]value] [-Z
     {zoneclustername | global}] {+ | resource...}

/usr/cluster/bin/clresource show [-g [resourcegroup,]...] [-p
     property-name[{node-specifier,…}]",...] [-t [resourcetype,]...]
     [-x "extension-property[{node-specifier,…}]",...] [-y "
     standard-property[{node-specifier,…}]",...] [-Z
     {zoneclustername [,...] | global | all}] [+ | resource...]

/usr/cluster/bin/clresource status [-g [resourcegroup,]...] [-s
     [state,]...] [-t [resourcetype,]...] [ -n node[,…]]
     [-Z {zoneclustername [,...] | global | all}] [+ | resource...]

/usr/cluster/bin/clresource unmonitor [-g [resourcegroup,]...]
     [-t [resourcetype,]...] [ -n node[,…]]
     [-Z {zoneclustername | global}] {+ | resource...}
```

clresource コマンドは、Oracle Solaris Cluster データサービスのリソースを管理します。clrs コマンドは、clresource コマンドの短縮形式です。clresource コマンドと clrs コマンドは同じものです。どちらの形式のコマンドも使用できます。

このコマンドの一般的な形式は次のとおりです。

```
clresource [subcommand] [options] [operands]
```

options に -? または -V オプションを指定する場合だけは、*subcommand* を省略できます。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式は、このマニュアルページの「オプション」セクションのオプションの説明で紹介されています。

ゾーンクラスタでこのコマンドを使用する

clresource コマンドは、ゾーンクラスタで、export を除くすべてのサブコマンドを指定して使用することができます。

export 以外のすべてのサブコマンドで -Z オプションを使用して、操作を制限する特定のゾーンクラスタの名前を指定することもできます。また、リソース名 (*zoneclustername* : *resource*) にゾーンクラスタ名を付けて、操作を特定のゾーンクラスタに制限することもできます。

グローバルクラスタノードからすべてのゾーンクラスタ情報にアクセスできますが、特定のゾーンクラスタは他のゾーンクラスタを認識しません。特定のゾーンクラスタに操作を制限できない場合、使用するサブコマンドは現在のクラスタでのみ機能します。

ゾーンクラスタのリソースは、別のゾーンクラスタのリソース、またはグローバルクラスタのリソースに依存性を持ちます。また、グローバルクラスタからのリソースは、そのグローバルクラスタのゾーンクラスタのリソースに依存性を持ちます。このインタークラスタ依存性は、グローバルクラスタより設定できます。

インタークラスタ依存性は、次のコマンドで指定できます。

```
# clresource set -p resource_dependencies=target-
zc
:target-rs  source-zc:
source-rs
```

たとえば、ゾーンクラスタ ZC1 内のリソース R1 からゾーンクラスタ ZC2 内のリソース R2 への依存性を指定する必要がある場合は、次のコマンドを使用します。

```
# clresource set -p resource_dependencies=ZC2:R2 ZC1:R1
```

グローバルクラスタリソース R2 にゾーンクラスタ ZC1 リソース R1 の依存性を指定する必要がある場合、次のコマンドを使用してください。

```
# clresource set -p resource_dependencies=global:R2 ZC1:R1
```

既存のリソース依存性（強、弱、リスタート、オフラインリスタート）がサポートされています。

リソースのステータスとステータス

リソースのステータスとリソースのステータスは、ノード単位で維持されます。リソースは、各クラスタノード上で固有のステータス、固有のステータスを持ちます。

Resource Group Manager(RGM) は、リソースに対して起動されたメソッドに基づき、各ノード上でリソースの状態を設定します。たとえば、指定のノード上でリソースに対する STOP メソッドを正しく実行した場合、そのリソースのノード上での状態は OFFLINE になります。STOP メソッドが 0 以外またはタイムアウトで終了した場合、そのリソースの状態は Stop_failed になります。

リソースの可能な状態は、次のとおりです。

- Online
- Offline
- Start_failed

- Stop_failed
- Monitor_failed
- Online_not_monitored
- Starting
- Stopping
- Not_online

注記 - Offline や Start_failed などの状態の名前は、大文字と小文字が区別*されません*。状態の名前を指定する際には、大文字と小文字を任意に組み合わせることができます。

RGM は、リソースのステータスだけでなく、リソース自体が API を使って設定するリソースのステータスも維持します。Status Message のフィールドは、実際には、ステータスキーワードとステータスメッセージからなります。ステータスメッセージは、ステータスキーワードのあとに出力される任意のテキスト文字列で、リソースによって任意に設定されます。

リソースステータスの値には、次のものがあります。

DEGRADED	リソースはオンラインですが、何らかの理由でパフォーマンスまたは可用性が低下しています。
FAULTED	リソースの機能を妨げるエラーが検出されました。
OFFLINE	リソースはオフラインです。
ONLINE	リソースはオンラインでサービスを提供します。
UNKNOWN	現在のステータスは不明または遷移中です。

サポートされるサブコマンドには次のものがあります。

clear

コマンドに対するオペランドとして指定したリソースに関連付けられているエラーフラグをクリアします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースのエラーフラグがクリアされます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、エラーフラグがクリアされるリソースを限定できます。

-g *resourcegroup*	*resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけをクリアします。

-n *node*

指定した 1 つまたは複数のノード上のリソースをクリアします。-n オプションを指定しないと、すべてのノード上のリソースがクリアされます。

-t *resourcetype*

resourcetype 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースだけをクリアします。

-Z {*zoneclustername* | global}

特定のクラスタまたは指定したクラスタ内のリソースだけをクリアします。グローバルクラスタからゾーンクラスタ内のリソースをクリアするには、-Z オプションを使用してゾーンクラスタを指定します。

デフォルトでは、clear サブコマンドは STOP_FAILED エラーフラグをクリアします。クリアするエラーフラグを明示的に指定するには、-f オプションを使用します。-f オプションが受け付けるエラーフラグは、STOP_FAILED だけです。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

create

コマンドに対するオペランドとして指定されたリソースを作成します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから特定のゾーンクラスタ内のリソースを作成するには、-Z オプションを使用してゾーンクラスタの名前を指定します。

create を -i オプションとともに使用して構成ファイルを指定した場合、このサブコマンドはプラス記号 (+) をオペランドとして受け付けます。+ オペランドを使用すると、存在しないすべてのリソースが構成ファイル内に作成されます。

デフォルトでは、リソースはモニタリング対象となり、有効な状態で作成されます。ただし、リソースは、そのリソースのリソースグループがオンライン状態になったあとでのみ、オンライン状態になり、モニター対象となります。無効な状態でリソースを作成するには、-d オプションを指定します。

リソースの作成時にプロパティー値を設定するには、次のオプションを使用します。

-p *property-name*= *value*

標準プロパティーまたは拡張プロパティーを、それらの名前が一意であるかぎり、設定します。

-x *extension-property*= *value*

拡張プロパティーを設定します。

-y *standard-property*= *value*

標準プロパティーを設定します。

node-specifier は、-p および -x オプションに対する任意の修飾子です。リソースが作成されると、指定した 1 つまたは複数のノード上にあるプロパティー*のみ*が設定されることを示します。指定したプロパティーは、クラスタ内のほかのノード上では設定されません。*node-specifier* を含めない場合、指定したプロパティーは、クラスタ内のすべてのノード上で設定されます。*node-specifier* の構文例を次に示します。

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

中括弧 ({}) は、指定したプロパティーをノード `phys-schost-1` でのみ設定することを示します。ほとんどのシェルの場合、中括弧は引用符で囲みます。

次の構文を使用すると、2 つのノード上でプロパティーを設定できます。

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

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` 役割に基づくアクセス制御 (RBAC) の承認が必要です。

`delete` サブコマンドの説明も参照してください。

`delete`

コマンドに対するオペランドとして指定されたリソースを削除します。このサブコマンドでプラス記号 (`+`) をオペランドとして指定すると、すべてのリソースが削除されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

このサブコマンドは、リソース間の依存性を満たすのに必要な順序で複数のリソースを削除します。このサブコマンドは、コマンド行でリソースを指定する順序を無視します。

同時に複数のリソースを削除する場合、このコマンドは、いくつかのステップに分けて実行されます。たとえばノードで問題が発見された場合などのように、コマンドが割り込まれたときには、いくつかのリソースが無効な構成のまま残される場合があります。問題を修正して、リソースの削除を終了するには、同じコマンドを正常なノードで再実行してください。

次のオプションを指定すると、オペランドのリストをフィルタリングし、削除するリソースを限定できます。

`-g` *resourcegroup*	*resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを削除します。
`-t` *resourcetype*	*resourcetype* 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースだけを削除します。
`-Z` {*zoneclustername* | `global`}	特定のクラスタまたは指定したクラスタ内のリソースだけを削除します。グローバルクラスタからゾーンクラスタ内のリソースを削除するには、`-Z` オプションを使用してゾーンクラスタを指定します。

デフォルトでは、リソースは次の条件が満たされる場合に *のみ* 削除されます。

- リソースが無効な状態である。
- リソースに対するすべての依存性が削除されている。

指定したリソースの削除を強制的に実行するには、`-F` オプションを指定します。このオプションは、次のような影響があるので、注意して使用してください。

- 指定したすべてのリソース (無効になっていないリソースも含む) が削除されます。
- 指定したすべてのリソースが、他のリソースのリソース依存性設定から削除されます。

このような影響により、クラスタ内でサービスが消失する場合があります。削除されていない依存リソースも、無効な状態またはエラー状態で残される場合があります。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。

`create` サブコマンドの説明も参照してください。

`disable`

コマンドに対するオペランドとして指定されたリソースを無効にします。このサブコマンドでプラス記号 (`+`) をオペランドとして指定すると、すべてのリソースが無効になります。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、無効にするリソースを限定できます。

`-g` *resourcegroup*	*resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを無効にします。
`-n` *node*	`-n` *node* を使用すると、1 つ以上のノードでリソースを無効化できます。
`-t` *resourcetype*	*resourcetype* 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースだけを無効にします。
`-Z` {*zoneclustername* | `global`}	特定のクラスタまたは指定したクラスタ内のリソースだけを無効にします。グローバルクラスタからゾーンクラスタ内のリソースを削除するには、`-Z` オプションを使用してゾーンクラスタを指定します。

`-r` オプションは、コマンドに対するオペランドとして指定したリソースに依存しているすべてのリソースを無効にします。これらのリソースは、コマンドに対するオペランドとして指定しなかった場合でも無効になります。`-g` オプションと `-t` オプションは、リソース依存性を満たすためだけに無効化されるリソースには適用されません。

このサブコマンドは、リソースのモニタリングステータスに影響を与えません。リソースは、有効な状態のときにモニターされていると、無効化されたあともモニターされます。リソースは、あとで再度有効な状態にしても、モニターされます。

このサブコマンドは、リソース間の依存性を満たすのに必要な順序でリソースを無効にします。このサブコマンドでは、コマンド行でリソースが指定された順序は無視されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.admin` RBAC の承認が必要です。

`enable` サブコマンドの説明も参照してください。

`enable`

コマンドに対するオペランドとして指定されたリソースを有効にします。このサブコマンドでプラス記号 (`+`) をオペランドとして指定すると、すべてのリソースが有効になります。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、有効にするリソースを限定できます。

`-g` *resourcegroup*	*resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを有効にします。
`-n` *node*	`-n` *node* を使用すると、1 つ以上のノードでリソースを有効化できます。
`-t` *resourcetype*	*resourcetype* 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースだけを有効にします。
`-Z` {*zoneclustername* | `global`}	特定のクラスタまたは指定したクラスタ内のリソースだけを有効にします。グローバルクラスタからゾーンクラスタ内のリソースを有効にするには、`-Z` オプションを使用してゾーンクラスタを指定します。

必要なリソース依存性をすべて確実に満たすには、`-r` オプションを指定します。`-r` オプションは、コマンドに対するオペランドとして指定したリソースが依存しているすべてのリソースを有効にします。これらのリソースは、コマンドに対するオペランドとして指定しなかった場合でも有効になります。`-g` オプションおよび `-t` オプションは、リソース依存性を満たすためだけに有効化されるリソースには適用されません。

リソースは、リソース間の依存性を満たすのに必要な順序で有効化されます。このサブコマンドでは、コマンド行でリソースが指定された順序は無視されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.admin` RBAC の承認が必要です。

`disable` サブコマンドの説明も参照してください。

`export`

1447 ページの`clconfiguration`(5CL) のマニュアルページで規定されている形式で、クラスタリソース構成をエクスポートします。

このサブコマンドは、大域ゾーンだけで使用できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。

`list`

コマンドに対するオペランドとして指定したリソースのリストを表示します。デフォルトでは、すべてのリソースが表示されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、表示するリソースを限定できます。

`-g` *resourcegroup*	*resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを表示します。

-n *node*
: -n *node* を使用すると、1 つ以上のノードでオンライン状態のリソースのみを一覧表示できます。

-t *resourcetype*
: *resourcetype* 内のリソースタイプのインスタンスであるリソースだけを表示します。

-Z {*zoneclustername* | `global` | `all`}
: 特定のクラスタまたは指定したクラスタ内のリソースだけを表示します。グローバルクラスタからゾーンクラスタ内のリソースを表示するには、-Z オプションを使用してゾーンクラスタを指定します。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソース構成が表示されます。-g オプションまたは -t オプションを指定すると、表示する情報を特定のリソースグループまたはリソースタイプに制限することができます。オペランドが指定されていない場合、指定されているリソースグループ内のすべてのリソースまたは指定されているリソースタイプのインスタンスであるすべてのリソースが表示されます。

-v オプションを指定すると、リスト内の各リソースのリソースグループおよびリソースタイプも表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。

`list-props`

コマンドに対するオペランドとして指定したリソースのプロパティーのリストを表示します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、プロパティーが表示されるリソースを限定できます。

-g *resourcegroup*
: *resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースのプロパティーだけを表示します。

-t *resourcetype*
: *resourcetype* 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースのプロパティーだけを表示します。

-l オプションでは、表示するリソースプロパティーのタイプを指定します。

`-l all`
: 標準プロパティーと拡張プロパティーを表示するように指定します。

`-l extension`
: 拡張プロパティーだけを表示するように指定します。デフォルトでは、拡張プロパティーだけが表示されます。

`-l standard`
: 標準プロパティーだけを表示するように指定します。

-l オプションを指定しないと、拡張プロパティーだけが表示されます。標準プロパティーを表示するには、-p オプションまたは -y オプションを使用して、プロパティーを明示的に指定します。

次のオプションを指定すると、表示するリソースプロパティーのセットを限定できます。

-p *property-name*
: *property-name* で指定したプロパティーだけ表示します。*property-name* では、標準プロパティーと拡張プロパティーを指定できます。

-x *extension-property*
: *extension-property* で指定した 1 つまたは複数のノード上の拡張プロパティーだけを表示します。

-y *standard-property*
: *standard-property* で指定した標準プロパティーだけを表示します。

node-specifier は、-p、-x、および -y オプションに対する任意の修飾子です。指定した 1 つまたは複数のノード上にあるプロパティー*のみ*が表示されることを示します。指定したプロパティーは、クラスタ内のほかのノード上では表示されません。*node-specifier* を含めない場合、指定したプロパティーは、クラスタ内のすべてのノード上で表示されます。*node-specifier* の構文例を次に示します。

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

中括弧 ({}) は、指定したプロパティーをノード `phys-schost-1` でのみ表示するよう指定します。ほとんどのシェルの場合、中括弧は引用符で囲みます。

次の構文を使用すると、2 つのノード上でプロパティーを表示できます。

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

-v オプションを指定すると、各プロパティーの説明も表示されます。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースプロパティーが表示されます。オペランドを指定しないと、指定したリソースグループ内のすべてのリソースのプロパティーまたは指定したリソースタイプのインスタンスであるすべてのリソースのプロパティーが表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。

-Z {*zoneclustername* | `global` | `all`}
: 特定のクラスタまたは指定したクラスタ内のリソースのプロパティーを一覧表示します。グローバルクラスタからゾーンクラスタ内のリソースを一覧表示するには、-Z オプションを使用してゾーンクラスタを指定します。

`monitor`
: コマンドに対するオペランドとして指定したリソースのモニタリングを有効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースに対してモニタリングが有効になります。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、モニタリングを有効にするリソースを限定できます。

-g *resourcegroup*
: *resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースのモニタリングだけを有効にします。

-n *node*
: 1 つ以上のノードでオンライン状態のリソースのモニタリングのみを有効にします。

-t *resourcetype*
: *resourcetype* 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースのモニタリングだけを有効にします。

-Z {*zoneclustername* | `global`}
: 特定のクラスタまたは指定したクラスタ内のリソースのモニタリングだけを有効にします。グローバルクラスタからゾーンクラスタ内のリソースの監視を有効にするには、`-Z` オプションを使用してゾーンクラスタを指定します。

リソースは、モニタリングが有効になっている場合、次の条件が満たされているときだけモニタリングされます。

- リソースが有効な状態にある。
- 該当リソースが含まれるリソースグループが、1 つ以上のクラスタノード上でオンライン状態にある。

注記 - リソースに対するモニタリングを有効にする場合、該当リソースを有効にする必要は*ありません*。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.admin` RBAC の承認が必要です。

`unmonitor` サブコマンドの説明も参照してください。

`set`

コマンドに対するオペランドとして指定したリソースの、指定したプロパティーを設定します。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースの指定したプロパティーが変更されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、プロパティーが変更の対象となるリソースを限定できます。

-g *resourcegroup*
: *resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースのプロパティーだけを変更します。

-t *resourcetype*
: *resourcetype* 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースのプロパティーだけを変更します。

プロパティー値は、次のオプションを使用して設定します。

-p *property-name*= *value*
: 標準プロパティーまたは拡張プロパティーを、それらの名前が一意であるかぎり、設定します。

-x *extension-property*= *value*
: 拡張プロパティーを設定します。

-y *standard-property*= *value*
: 標準プロパティーを設定します。

node-specifier は、ノード単位の拡張プロパティーを更新するための -p および -x オプションに対する*オプションの*修飾子です。指定した 1 つまたは複数のノード上で*のみ*プロパティーが設定されることを示します。指定したプロパティーは、クラスタ内のほかのノード上では設定されません。*node-specifier* を含めない場合、指定したプロパティーは、クラスタ内のすべてのノード上で設定されます。*node-specifier* の構文例を次に示します:

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

中括弧 ({}) は、指定したプロパティーをノード `phys-schost-1` でのみ設定することを示します。ほとんどのシェルの場合、中括弧は引用符で囲みます。

次の構文を使用すると、2 つのノード上でプロパティーを設定できます。

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

-Z {*zoneclustername* | `global`}
: 特定のクラスタまたは指定したクラスタ内のリソースのプロパティーだけを設定します。グローバルクラスタからゾーンクラスタ内のリソースのプロパティーを設定するには、-Z オプションを使用してゾーンクラスタを指定します。

  スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。

`show`
: コマンドに対するオペランドとして指定したリソースの構成を表示します。デフォルトでは、すべてのリソースの構成が表示されます。

  このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

  次のオプションを指定すると、オペランドのリストをフィルタリングし、構成が表示されるリソースを限定できます。

  -g *resourcegroup*
  : *resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースの構成だけを表示します。

  -n *node*
  : -n *node* を使用すると、1 つ以上のノードでオンライン状態のリソースについてのみ構成を表示できます。

  -t *resourcetype*
  : *resourcetype* 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースの構成だけを表示します。

-Z {*zoneclustername* | `global` | `all`}

特定のクラスタまたは指定したクラスタ内のリソースだけを表示します。グローバルクラスタからゾーンクラスタ内のリソースを表示するには、-Z オプションを使用してゾーンクラスタを指定します。

次のオプションを指定すると、表示するリソースプロパティーのセットを限定できます。

-p *property-name*
: *property-name* で指定したプロパティーだけ表示します。*property-name* では、標準プロパティーと拡張プロパティーを指定できます。

-x *extension-property*
: *extension-property* で指定した 1 つまたは複数のノード上の拡張プロパティーだけを表示します。

-y *standard-property*
: *standard-property* で指定した標準プロパティーだけを表示します。

node-specifier は、-p、-x、および -y オプションに対する任意の修飾子です。指定した 1 つまたは複数のノード上にあるプロパティー*のみ*が表示されることを示します。指定したプロパティーは、クラスタ内のほかのノード上では表示されません。*node-specifier* を含めない場合、指定したプロパティーは、クラスタ内のすべてのノード上で表示されます。*node-specifier* の構文例を次に示します。

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

中括弧 ({}) は、指定したプロパティーをノード `phys-schost-1` でのみ表示するよう指定します。ほとんどのシェルの場合、中括弧は引用符で囲みます。

次の構文を使用すると、2 つのノード上でプロパティーを表示できます。

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

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソース構成が表示されます。-g オプションまたは -t オプションを指定すると、表示する情報を特定のリソースグループまたはリソースタイプに制限することができます。オペランドを指定しないと、指定したすべてのリソースの構成が表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。

`status`

コマンドに対するオペランドとして指定したリソースのステータスを表示します。デフォルトでは、すべてのリソースのステータスが表示されます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、ステータスが表示されるリソースを限定できます。

-g *resourcegroup*
: *resourcegroup* 内のリソースグループのメンバーである、オペランドのリスト内のリソースのステータスだけを表示します。

-n *node*

-n *node* を使用すると、1 つ以上のノードでオンライン状態のリソースについてのみステータスを表示できます。-n オプションおよび -s オプションは、同時に指定できません。

-s *state*

state のステータス内にある、オペランドのリスト内のリソースのステータスだけを表示します。-n オプションおよび -s オプションは、同時に指定できません。

-t *resourcetype*

resourcetype 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースのステータスだけを表示します。

-Z {*zoneclustername* | `global` | `all`}

特定のクラスタまたは指定したクラスタ内のリソースのステータスを表示します。グローバルクラスタからゾーンクラスタ内のリソースのステータスを表示するには、-Z オプションを使用してゾーンクラスタを指定します。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースのステータスが表示されます。-g オプションまたは -t オプションを指定すると、表示する情報を特定のリソースグループまたはリソースタイプに制限することができます。オペランドを指定しないと、指定したすべてのリソースのステータスが表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。

`unmonitor`

コマンドに対するオペランドとして指定したリソースのモニタリングを無効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースに対するモニタリングが無効になります。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

無効になっているリソースのモニタリングを無効にしても、リソースは影響を受けません。リソースとそのモニターは、すでにオフライン状態です。

注記 - リソースのモニタリングを無効にしても、リソースは無効に*なりません*。ただし、リソースを無効にする場合、モニタリングを無効にする必要はありません。無効なリソースとそのモニターは、オフライン状態が維持されます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、モニタリングを無効にするリソースを限定できます。

-g *resourcegroup*

resourcegroup 内のリソースグループのメンバーである、オペランドのリスト内のリソースのモニタリングだけを無効にします。

-n *node*

1 つ以上のノードでオンライン状態のリソースのモニタリングのみを無効にします。

-t *resourcetype*
: *resourcetype* 内のリソースタイプのインスタンスである、オペランドのリスト内のリソースのモニタリングだけを無効にします。

-Z {*zoneclustername* | `global`}
: 特定のクラスタまたは指定したクラスタ内のリソースのモニタリングだけを無効にします。グローバルクラスタからゾーンクラスタ内のリソースのモニタリングを無効にするには、`-Z` オプションを使用してゾーンクラスタを指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.admin` RBAC の承認が必要です。

`disable` サブコマンドおよび `monitor` サブコマンドの説明も参照してください。

次のオプションがサポートされています。

`-?`
`--help`
: ヘルプ情報を表示します。このオプションを使用する場合、ほかの処理は実行されません。

このオプションを指定するとき、サブコマンドは指定してもしなくてもかまいません。

サブコマンドなしでこのオプションを指定すると、このコマンドのサブコマンドのリストが表示されます。

サブコマンド付きでこのオプションを指定すると、サブコマンドの使用方法が表示されます。

`-a`
`--automatic`
: リソースがクラスタ構成ファイル (1447 ページの`clconfiguration`(5CL)) から作成される場合は、次の追加の操作を自動的に実行します。

- リソースタイプの登録
- リソースグループの作成
- オペランドのリスト内で指定されているリソースの依存先リソースの作成

クラスタ構成情報には、次の処理をすべて実行するのに必要な十分な情報が含まれている必要があります。

- リソースタイプの登録を有効にする
- リソースグループの作成を有効にする
- リソースの作成を有効にする

このオプションを指定できるのは、`create` サブコマンドの場合だけです。このオプションを指定する場合は、`-i` オプションも指定し、構成ファイルを指定します。

-d
--disable

リソースの作成時にリソースを無効にします。このオプションを指定できるのは、create サブコマンドの場合だけです。デフォルトでは、リソースは作成されたあと、有効な状態になります。

リソースは、有効化しても、オンライン状態になるとは限りません。リソースは、リソースのリソースグループが 1 つ以上のノードでオンラインになってはじめてオンラインになります。

-f *errorflag*
--flag=*errorflag*
--flag *errorflag*

clear サブコマンドによってクリアするエラーフラグを明示的に指定します。このオプションは、clear サブコマンドの場合にだけ指定できます。デフォルトでは、clear サブコマンドは STOP_FAILED エラーフラグをクリアします。

-f オプションが受け付けるエラーフラグは、STOP_FAILED だけです。

-F
--force

無効状態でないリソースの削除が、強制的に実行されます。このオプションは、delete サブコマンドの場合にだけ指定できます。

このオプションは、次のような影響があるので、注意して使用してください。

- 指定したすべてのリソース (無効になっていないリソースも含む) が削除されます。
- 指定したすべてのリソースが、他のリソースのリソース依存性設定から削除されます。

このような影響により、クラスタ内でサービスが消失する場合があります。削除されていない依存リソースも、無効な状態またはエラー状態で残される場合があります。

-g *resourcegroup*[,…]
--resourcegroup=*resourcegroup*[,…]
--resourcegroup *resourcegroup*[,…]

1 つのリソースグループまたはリソースグループのリストを指定します。

create 以外のサブコマンドの場合、このコマンドは指定されたリソースグループのメンバーである、オペランドのリスト内のリソースに対してのみ作用します。-g オプションを使用して、リソースグループを指定します。

-g オプションと create サブコマンドを同時に指定すると、clresource は指定されたリソースグループ内に該当するリソースを作成します。このオプションを使用する場合は、リソースグループを 1 つしか指定することができません。

-i {- | *clconfiguration*}
--input={- | *clconfiguration*}
--input {- | *clconfiguration*}

リソースの作成または変更に使用する構成情報を指定します。この情報は、1447 ページのclconfiguration(5CL) のマニュアルページに定義されている形式に

準拠している必要があります。この情報は、ファイルに含めることも、標準入力を使って指定することもできます。標準入力を指定するには、ファイル名の代わりに - を指定します。

コマンドに対するオペランドとして指定したリソースだけが、作成または変更されます。コマンドで指定したオプションは、構成情報で設定されているオプションより優先されます。構成パラメータは、構成情報内で設定されていない場合、コマンド行で指定します。

-i オプションと create サブコマンドを同時に指定すると、clresource は必要なすべてのリソースタイプを登録し、必要なリソースグループをすべて作成します。登録および構成に必要なすべての情報を指定します。その他の構成データはすべて無視します。

-l *listtype*
--listtype=*listtype*
--listtype *listtype*

list-props サブコマンドによって表示するリソースプロパティーのタイプを指定します。このオプションは、list-props サブコマンドの場合にだけ指定できます。

listtype に対して、次のリストから値を 1 つ指定します。

all	標準プロパティーと拡張プロパティーを表示するように指定します。
extension	拡張プロパティーだけを表示するよう指定します。デフォルトでは、拡張プロパティーだけが表示されます。
standard	標準プロパティーだけを表示するように指定します。

-l オプションを指定しないと、拡張プロパティーだけが表示されます。標準プロパティーを表示するには、-p オプションまたは -y オプションを使用して、プロパティーを明示的に指定します。

-n *node*[,…]
--node=*node*[,…]
--node *node*[,…]

ターゲットのグローバルクラスタまたはゾーンクラスタに、ノードまたはノードのリストを指定します。各ノードはノード名またはノード ID で指定できます。

-Z オプションが指定されている場合は、-n オプションで、グローバルクラスタホスト名ではなくゾーンクラスタホスト名のみを指定できます。-Z オプションが指定されていない場合は、-n オプションでグローバルクラスタホスト名のみを指定できます。

このオプションとともに指定できるサブコマンドは、次のとおりです。

disable	指定したノード上でホストされている、オペランドのリスト内のリソースのみを無効にします。
enable	指定したノード上でホストされている、オペランドのリスト内のリソースのみを有効にします。
list	指定したノード上でホストされている、オペランドのリスト内のリソースのみのリストを表示します。

`monitor`
: 指定したノード上でホストされている、オペランドのリスト内のリソースのみをモニターします。

`show`
: 指定したノード上でホストされている、オペランドのリスト内のリソースについてのみ構成情報を表示します。

`status`
: 指定したノード上でホストされている、オペランドのリスト内のリソースについてのみステータスを報告します。

`unmonitor`
: 指定したノード上でホストされている、オペランドのリスト内のリソースについてのみモニターを解除します。

`-o` {- | *clconfiguration*}
`--output=`{- | *clconfiguration*}
`--output` {- | *clconfiguration*}

リソース構成情報の書き込み先を指定します。この書き込み先は、ファイルでも、標準出力でもかまいません。標準出力を指定するには、ファイル名の代わりに、ダッシュ (-) を指定します。標準出力を指定すると、コマンドに対するほかのすべての標準出力が抑制されます。このオプションは、`export` サブコマンドの場合にだけ指定できます。

構成情報は、コマンドに対するオペランドとして指定したリソースの場合だけ書き込まれます。この情報は、1447 ページの`clconfiguration`(5CL) のマニュアルページに定義されている形式で書き込まれます。

`-p` *property-name=value*
`-p` *property-name+=array-values*
`-p` *property-name-=array-values*
`--property=`*property-name=value*
`--property=`*property-name+=array-values*
`--property=`*property-name-=array-values*
`--property` *property-name=value*
`--property` *property-name+=array-values*
`--property` *property-name-=array-values*

コマンドに対するオペランドとして指定したリソースに対し、プロパティーの値を設定します。このオプションの代入形式を指定できるのは、`create` サブコマンドおよび `set` サブコマンドだけです。

`-p` オプションを使用すると、任意の標準プロパティーまたは拡張プロパティーを指定できます。あるリソースタイプの拡張プロパティーが同じリソースタイプの標準プロパティーと同じ名前を持っている場合、`-p` オプションを使用するとエラーが返されます。そのような場合、拡張プロパティーを指定するには `-x` オプション、標準プロパティーを指定するには `-y` オプションを使用します。

標準プロパティーについては、1287 ページの`r_properties`(5) のマニュアルページを参照してください。

リソースタイプの拡張プロパティーについては、リソースタイプに関するドキュメントを参照してください。

このオプションとともに使用する演算子は、次のとおりです。

=　　プロパティーに、指定した値を設定します。この演算子は、create サブコマンドおよび set サブコマンドで使用できます。

+=　　1 つまたは複数の値を文字列配列値に追加します。この演算子は、set サブコマンドでのみ使用できます。この演算子は、たとえば、Resource_dependencies のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。

-=　　1 つまたは複数の値を文字列配列値から削除します。この演算子は、set サブコマンドでのみ使用できます。この演算子は、たとえば、Resource_dependencies のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。

クラスタノードのサブセットに対してノード単位の拡張プロパティーを設定するには、プロパティーの設定時にノードを指定します。次のように、中括弧で囲まれたノードのリストをプロパティー名のあとに付加します。

name{*node*}

node は、ノード名またはノード ID をコンマで区切ったリストです。ノード単位の拡張プロパティーの詳細は、1335 ページのrt_properties(5) のマニュアルページを参照してください。

ノード単位のリソース依存関係をクラスタノードのサブセットに設定するには、各ノード単位の依存関係を次の形式で指定します:

```
myres1@node1,myres2@node2,myres3@node3
```

gds-rs リソースの場合は、次のコマンドで、ノード ptrancos1 上のリソース trancos-3-rs およびノード ptrancos2 上のリソース trancos-4-rs に対する依存関係を設定します。

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

phys-schost-1# clresource show -p resource_dependencies gds-rs
=== Resources ===
Resource: gds-rs
Resource_dependencies: trancos-3-rs@ptrancos1 trancos-4-rs@ptrancos2
```

ローカルノードスコープでリソースの依存関係を設定するには、LOCAL_NODE　修飾子を次の形式で指定します:

```
myres1{LOCAL_NODE},myres2{LOCAL_NODE}
```

gds-rs リソースの場合は、次のコマンドで、リソース trancos-3-rs に対するローカルノードの依存関係を設定します。

```
# clresource set -p resource_dependencies=trancos-3-rs{LOCAL_NODE} gds-rs

phys-schost-1# clresource show -p resource_dependencies gds-rs
=== Resources ===
```

Resource: gds-rs
Resource_dependencies: trancos-3-rs{LOCAL_NODE}

ノード単位のリソース依存関係および依存関係のスコープ修飾子の詳細は、1287 ページの`r_properties`(5) のマニュアルページを参照してください。

`-p` *property-name*[,…]
`--property=`*property-name*[,…]
`--property` *property-name*[,…]

`list-props` サブコマンドおよび `show` サブコマンドのプロパティーのリストを指定します。

`-p` オプションを使用すると、任意の標準プロパティーまたは拡張プロパティーを指定できます。あるリソースタイプの拡張プロパティーが同じリソースタイプの標準プロパティーと同じ名前を持っている場合、`-p` オプションを使用するとエラーが返されます。そのような場合、拡張プロパティーを指定するには `-x` オプション、標準プロパティーを指定するには `-y` オプションを使用します。

標準プロパティーについては、1287 ページの`r_properties`(5) のマニュアルページを参照してください。

リソースタイプの拡張プロパティーについては、リソースタイプに関するドキュメントを参照してください。

このオプションを指定しなかった場合、`list-props` サブコマンドおよび `show` サブコマンドは、`-v` オプションも指定されているかどうかに基づいて、すべてまたはほとんどのリソースプロパティーを一覧表示します。

`-r`
`--recursive`

必要な依存性がすべて満たされるように、リソースの有効化または無効化を再帰的に実行します。このオプションは、`disable` サブコマンドおよび `enable` サブコマンドの場合にだけ指定できます。

このオプションをこれらのサブコマンドとともに指定した場合の効果は、次のとおりです。

`disable`	コマンドに対するオペランドとして指定したリソースに依存しているリソースを、すべて無効にします。これらのリソースは、コマンドに対するオペランドとして指定しなかった場合でも無効になります。
`enable`	コマンドに対するオペランドとして指定したリソースが依存しているリソースを、すべて有効にします。これらのリソースは、コマンドに対するオペランドとして指定しなかった場合でも有効になります。

`-s` *state*[,…]
`--state=`*state*[,…]
`--state` *state*[,…]

`list` サブコマンドおよび `status` サブコマンドの状態のリストを指定します。

このオプションは出力を制限し、ノードリスト内の 1 つまたは複数のノード上で指定されている状態の 1 つにあるリソースだけが含まれるようにします。

可能な状態は、次のとおりです。

- `Online`
- `Offline`
- `Start_failed`
- `Stop_failed`
- `Monitor_failed`
- `Online_not_monitored`
- `Starting`
- `Stopping`
- `Not_online`

注記 - `Offline` や `Start_failed` などの状態の名前は、大文字と小文字が区別*されません*。状態の名前を指定する際には、大文字と小文字を任意に組み合わせることができます。

`-t` *resourcetype*[,…]
`--type=`*resourcetype*[,…]
`--type` *resourcetype*[,…]

1 つのリソースタイプまたはリソースタイプのリストを指定します。

`create` を除くこのオプションを使用できるすべてのサブコマンドの場合、コマンドは、次の両方の条件を満たすリソースに対してだけ作用します。

- リソースが、オペランドのリスト内にある。
- リソースが、`-t` オプションで指定するリソースタイプのインスタンスである。

`-t` オプションと `clresource create` を同時に指定すると、指定したタイプのリソースが作成されます。指定できるリソースタイプは、1 つだけです。

リソースタイプ名の形式については、Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド の有効な RGM 名"を参照してください。

`-u`

`+` オペランドを使用する場合、このオプションは、リソースグループが中断されたリソース上でコマンドが機能するように指定します。

`+` オペランドを指定した状態で `-u` オプションを指定しないと、コマンドはリソースグループが中断されたすべてのリソースを無視します。`-u` オプションは、`+` オペランドが `clear`、`disable`、`enable`、`monitor`、`set`、または `unmonitor` コマンドとともに指定されている場合に有効となります。

`+` オペランドが `clear`、`disable`、`enable`、`monitor`、`set`、または`unmonitor` サブコマンドとともに使用されている場合、`-u` オプションも指定しないかぎり、コマンドはリソースグループが中断されたすべてのリソースを無視します。

-V
--version

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンドやオペランドなどのオプションは無視されます。-V オプションは、コマンドのバージョンだけを表示します。その他の処理は行いません。

-v
--verbose

詳細メッセージを標準出力に表示します。

このオプションは、このコマンドの任意の形式とともに指定できます。

-v オプションと -o - オプションを同時に指定してはいけません。-v オプ ションは無視されます。-o - オプションは、ほかのすべての標準出力を抑制します。

-x *extension-property=value*
-x *extension-property+=array-value*
-x *extension-property-=array-value*
--extension-property=*extension-property=value*
--extension-property=*extension-property+=array-value*
--extension-property=*extension-property-=array-value*
--extension-property *extension-property=value*
--extension-property *extension-property+=array-value*
--extension-property *extension-property-=array-value*

コマンドに対するオペランドとして指定したリソースの拡張プロパティーの値を、設定または変更します。

一般に、-p オプションを使用すると、任意の標準プロパティーまたは拡張プロパティーを指定できます。あるリソースタイプの拡張プロパティーが同じリソースタイプの標準プロパティーと同じ名前を持っている場合、-p オプションを使用するとエラーが返されます。そのような場合、拡張プロパティーを指定するには -x オプション、標準プロパティーを指定するには -y オプションを使用します。

このオプションの代入形式を指定できるのは、create サブコマンドおよび set サブコマンドだけです。

リソースタイプの拡張プロパティーについては、リソースタイプに関するドキュメントを参照してください。

このオプションとともに使用する演算子は、次のとおりです。

=
プロパティーに、指定した値を設定します。この演算子は、create サブコマンドおよび set サブコマンドで使用できます。

+=
1 つまたは複数の値を文字列配列値に追加します。この演算子は、set サブコマンドでのみ使用できます。この演算子は、たとえば、Resource_dependencies のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。

-=

1 つまたは複数の値が、文字列配列値から削除されます。この演算子は、set サブコマンドでのみ使用できます。この演算子は、たとえば、Resource_dependencies のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。

クラスタノードのサブセットに対してノード単位の拡張プロパティーを設定するには、プロパティーの設定時にノードを指定します。次のように、中括弧で囲まれたノードのリストをプロパティー名のあとに付加します。

name{*node*}

node は、ノード名またはノード ID をコンマで区切ったリストです。ノード単位のプロパティーの詳細は、1335 ページのrt_properties(5) のマニュアルページを参照してください。

-x *extension-property*[,…]
--extension-property=*extension-property*[…]
--extension-property *extension-property*[,…]

list-props サブコマンドおよび show サブコマンドの拡張プロパティーのリストを指定します。

リソースタイプの拡張プロパティーについては、リソースタイプに関するドキュメントを参照してください。

-p オプションを使用すると、任意の標準プロパティーまたは拡張プロパティーを指定できます。あるリソースタイプの拡張プロパティーが同じリソースタイプの標準プロパティーと同じ名前を持っている場合、-p オプションを使用するとエラーが返されます。そのような場合、拡張プロパティーを指定するには -x オプション、標準プロパティーを指定するには -y オプションを使用します。

このオプションを指定しないと、list-props サブコマンドおよび show サブコマンドは、-v オプションも指定されているかどうかに基づいて、すべてまたはほとんどのリソースプロパティーを一覧表示します。

-y *standard-property=value*
-y *standard-property+=array-value*
-y *standard-property-=array-value*
--standard-property=*standard-property=value*
--standard-property=*standard-property+=array-value*
--standard-property=*standard-property-=array-value*
--standard-property *standard-property=value*
--standard-property *standard-property+=array-value*
--standard-property *standard-property-=array-value*

コマンドに対するオペランドとして指定したリソースの標準プロパティーの値を、設定または変更します。

-p オプションを使用すると、任意の標準プロパティーまたは拡張プロパティーを指定できます。あるリソースタイプの拡張プロパティーが同じリソースタイプの標準プロパティーと同じ名前を持っている場合、-p オプションを使用するとエラーが返されます。そのような場合、拡張プロパティーを指定するには -x オプション、標準プロパティーを指定するには -y オプションを使用します。

このオプションの代入形式を指定できるのは、create サブコマンドおよび set サブコマンドだけです。

標準プロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

このオプションとともに使用する演算子は、次のとおりです。

=
: プロパティーに、指定した値を設定します。この演算子は、create サブコマンドおよび set サブコマンドで使用できます。

+=
: 1 つまたは複数の値を文字列配列値に追加します。この演算子は、set サブコマンドでのみ使用できます。この演算子は、たとえば、Resource_dependencies のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。

-=
: 1 つまたは複数の値が、文字列配列値から削除されます。この演算子は、set サブコマンドでのみ使用できます。この演算子は、たとえば、Resource_dependencies のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。

-y *standard-property*[,…]
--standard-property=*standard-property*[,...]
--standard-property *standard-property*[,...]

list-props サブコマンドおよび show サブコマンドの標準プロパティーのリストを指定します。

標準プロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

-p オプションを使用すると、任意の標準プロパティーまたは拡張プロパティーを指定できます。あるリソースタイプの拡張プロパティーが同じリソースタイプの標準プロパティーと同じ名前を持っている場合、-p オプションを使用するとエラーが返されます。そのような場合、拡張プロパティーを指定するには -x オプション、標準プロパティーを指定するには -y オプションを使用します。

このオプションを指定しないと、list-props サブコマンドおよび show サブコマンドは、-v オプションも指定されているかどうかに基づいて、すべてまたはほとんどのリソースプロパティーを一覧表示します。

-Z {*zoneclustername* | global | all}
--zoneclustername={*zoneclustername* | global | all}
--zoneclustername {*zoneclustername* | global | all}

クラスタ、またはリソースが存在するクラスタや処理するクラスタを指定します。

このオプションは、export サブコマンド以外のすべてのサブコマンドでサポートされています。

このオプションを指定する場合は、次のいずれかの引数も指定する必要があります。

zoneclustername このオプションを使用するコマンドが、*zoneclustername* という名前のゾーンクラスタでのみ指定されたすべてのリソースで機能するように指定します。

`global` このオプションを使用するコマンドが、広域クラスタでのみ指定されたすべてのリソースで機能するように指定します。

`all` 広域クラスタでこの引数を使用する場合、それを使用するコマンドが広域クラスタとすべてのゾーンクラスタを含め、すべてのクラスタで指定されたすべてのリソースで機能するように指定します。

この引数をゾーンクラスタで使用すると、このオプションを使用するコマンドが、ゾーンクラスタ内だけで指定されたすべてのリソースに対して機能するように指定されます。

次のオペランドだけがサポートされています。

resource 管理対象のリソース (1 つまたは複数) を指定します。サブコマンドで複数のリソースを指定できる場合は、プラス記号 (+) を使用すると、すべてのリソースを指定できます。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (`CL_NOERR`) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

次の終了コードが返されます。

`0 CL_NOERR`

エラーなし

実行したコマンドは正常に終了しました。

`1 CL_ENOMEM`

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

`3 CL_EINVAL`

無効な引数

コマンドを間違って入力したか、`-i` オプションで指定したクラスタ構成情報の構文が間違っていました。

`6 CL_EACCESS`

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

9 CL_ESTATE

オブジェクトの状態が不正です

その時点で変更できない、または常時変更できないプロパティー、リソースグループ、またはその他のオブジェクトを変更しようとしました。

10 CL_EMETHOD

リソースのメソッドが失敗しました

リソースのメソッドが失敗しました。次のいずれかの理由のために、メソッドは失敗しました。

- `validate` メソッドは、リソースを作成しようとしたときに、あるいは、リソースのプロパティーを変更しようとしたときに失敗しました。
- `validate` 以外のメソッドは、リソースを有効、無効、または削除しようとしたときに失敗しました。

15 CL_EPROP

無効なプロパティーです

-p、-y、または -x オプションで指定したプロパティーまたは値が存在しないか、許可されていません。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

41 CL_ETYPE

無効なタイプです

-t または -p オプションで指定したタイプは存在しません。

これらの終了値は、1047 ページのscha_calls(3HA) のマニュアルページで説明されているリターンコードと互換性があります。

例 186 リソースの作成

この例では、rg-failover という名前のリソースグループ内で rs-nfs という名前のリソースを作成します。リソースは、リソースタイプ SUNW.nfs のインスタンスです。リソースは作成後に有効な状態になり、リソースモニタリングも有効になります。

```
# clresource create -g rg-failover -t SUNW.nfs rs-nfs
```

次の 2 つのコマンドのいずれかで、rg-failover という名前のリソースグループにあるゾーンクラスタ ZC 内に、rs-nfs という名前のリソースを作成します。これらのコマンドは、グローバルクラスタノードから、またはゾーンクラスタ ZC 内で実行できます。コマンドをゾーンクラスタから実行する場合、任意でゾーンクラスタ名を持つリソースのスコープを明示的に定義できます。

```
# clresource create -g rg-failover -t SUNW.nfs -Z ZC rs-nfs
# clresource create -g rg-failover -t SUNW.nfs ZC:rs-nfs
```

例 187 リソースモニタリングの有効化

この例では、rs-nfs という名前のリソースに対するモニタリングを有効にします。

```
# clresource monitor rs-nfs
```

リソースのモニタリングは、いったん有効になると、clresource unmonitor コマンドを使って明示的に無効にするまで、有効な状態のままとなります。リソースの無効化および有効化は、リソースがモニターされるかどうかに影響を与えません。

例 188 リソースの有効化

この例では、リソースグループ rg-failover および rg-failover2 の中のすべてのリソースを有効にします。

```
# clresource enable -g rg-failover,rg-failover2 +
```

このコマンドは、リソースがモニターされるかどうかに影響を与えません。

例 189 リソースプロパティーの設定

この例では、リソースタイプ SUNW.nfs の全インスタンスの r_description プロパティーを HA-NFS res に設定します。

```
# clresource set -t SUNW.nfs -p r_description="HA-NFS res" +
```

例 190 ノード単位リソースプロパティーの設定

この例では、リソース rs-oracle のノード単位プロパティー oracle_sid に、次のように別のノード上の別の値を設定します。

- ノード phys-schost-1 およびノード phys-schost-2 上で、このプロパティーには myora1 が設定されます。
- ノード phys-schost-3 上の場合、このプロパティーには myora2 が設定されます。

この例の場合、中括弧文字は、使用されるシェルに対して特別な意味があります。そのため、ノードリストが付加される各プロパティー名は、二重引用符で囲んであります。

```
# clresource set -p "oracle_sid{phys-schost-1,phys-schost-2}"=myora1 \
-p "oracle_sid{phys-schost-3}"=myora2 rs-oracle
```

例 191 ノード単位のリソース依存関係の設定

この例では、gds-rs のノード単位のリソース依存関係を設定して、それが 2 つの異なる論理ホストリソースに依存するようにします。

```
# clresource set -p resource_dependencies=node-3-rs@pnode1,node-4-rs@pnode2 gds-rs

# clresource show -p resource_dependencies gds-rs
Resource: gds-rs
Standard Properties:
Resource_dependencies: node-3-rs@pnode1,node-4-rs@pnode2
```

例 192 文字列配列プロパティーへの値の追加

この例では、値 rs-oracle を、リソース rs-myapp の文字列配列プロパティー resource_dependencies に追加します。この文字列配列プロパティー内の既存の値は、変更されません。

```
# clresource set -p resource_dependencies+=rs-oracle rs-myapp

# clresource show -p resource_dependencies rs-myapp
Resource: rs-myapp
Standard Properties:
Resource_dependencies: rs-nfs rs-oracle
```

例 193 リソースの削除

この例では、rs-nfs という名前のリソースを削除します。

```
# clresource delete rs-nfs
```

例 194 クラスタ構成全体の更新

この例では、次の順序で処理を実行することにより、クラスタ構成全体を更新します。

1. クラスタ内のリソースグループをすべてオフラインにしたあと、すべてのリソースを削除してから、すべてのリソースグループを削除します。
2. すべてのリソースタイプの登録解除
3. 構成ファイル /net/server/export/mycluster.xml の中で指定されているすべてのリソースの作成、そのリソースタイプの登録、および必要なすべてのリソースグループの作成

```
# clresourcegroup delete --force +
# clresourcetype unregister +
# clresource -i /net/server/export/mycluster.xml -a +
```

例 195 リソースの一覧表示

この例では、すべてのリソースを一覧表示します。

```
# clresource list
logicalhost1
rs-nfs-1
rs-nfs-2
logicalhost2
rs-apache-1
```

例 196 リソースおよびグループとタイプの一覧表示

この例では、すべてのリソースおよびそのリソースグループとリソースタイプを一覧表示します。

```
# clresource list -v

Resource Name      Resource Group    Resource Type
-------------      --------------    -------------
logicalhost1       rg-failover-1     SUNW.LogicalHostname
rs-nfs-1           rg-failover-1     SUNW.nfs
logicalhost2       rg-failover-2     SUNW.LogicalHostname
rs-nfs-2           rg-failover-2     SUNW.nfs
rs-apache-1        rg-failover-1     SUNW.apache
```

例 197 特定のタイプのリソースの一覧表示

この例では、リソースタイプ nfs のすべてのインスタンスを一覧表示します。

```
# clresource list -t nfs
rs-nfs-1
rs-nfs-2
```

例 198 リソースタイプの拡張プロパティーと説明の一覧表示

この例では、リソースタイプ nfs の拡張プロパティーと各拡張プロパティーの説明を一覧表示します。

```
# clresource list-props -t nfs -v
Properties              Descriptions
----------              ------------
Monitor_retry_count     Number of PMF restarts allowed for the fault monitor
Monitor_retry_interval  Time window (minutes) for fault monitor restarts
Rpcbind_nullrpc_timeout Timeout(seconds) to use when probing rpcbind
Nfsd_nullrpc_timeout    Timeout(seconds) to use when probing nfsd
Mountd_nullrpc_timeout  Timeout(seconds) to use when probing mountd
Statd_nullrpc_timeout   Timeout(seconds) to use when probing statd
Lockd_nullrpc_timeout   Timeout(seconds) to use when probing lockd
Rpcbind_nullrpc_reboot  Boolean to indicate if we should reboot system when
                          null rpc call on rpcbind fails
Nfsd_nullrpc_restart    Boolean to indicate if we should restart nfsd when
                          null rpc call fails
Mountd_nullrpc_restart  Boolean to indicate if we should restart mountd when
                          null rpc call fails
```

Line breaks in the Descriptions column are added to enhance the readability of this example. Actual output from the command does not contain these line breaks.

例 199 リソースの無効化および有効化によるリソース状態 Start_failed のクリア

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

Stop_failed リソース状態とは異なり、Start_failed リソース状態は、Oracle Solaris Cluster ソフトウェアがリソースグループ上で操作を実行することを*妨げません*。リソース状態

Start_failed は、*command* clear コマンドを発行しなくてもクリアすることができます。該当リソースを再起動するコマンドを実行するだけで済みます。

次のコマンドでは、リソース resource-1 上で発生したリソース状態 Start_failed を、リソースをいったん無効化したあと再度有効化することにより、クリアします。

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

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページのIntro(1CL), 251 ページのclreslogicalhostname(1CL), 305 ページのclresourcegroup(1CL), 333 ページのclresourcetype(1CL), 347 ページのclressharedaddress(1CL), 551 ページのcluster(1CL), 1047 ページのscha_calls(3HA), 1447 ページのclconfiguration(5CL), Unresolved link to " attributes5", 1287 ページのr_properties(5), Unresolved link to " rbac5"

Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド "

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のオプションを指定してこのコマンドを実行できます。

- -? オプション
- -V オプション

サブコマンドを指定してこのコマンドを実行する場合、スーパーユーザー以外のユーザーは RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
create	solaris.cluster.modify
delete	solaris.cluster.modify

サブコマンド	RBAC の承認
disable	solaris.cluster.admin
enable	solaris.cluster.admin
export	solaris.cluster.read
list	solaris.cluster.read
list-props	solaris.cluster.read
set	solaris.cluster.modify
monitor	solaris.cluster.admin
clear	solaris.cluster.admin
show	solaris.cluster.read
status	solaris.cluster.read
unmonitor	solaris.cluster.admin

名前

clreslogicalhostname, clrslh — Oracle Solaris Cluster 論理ホスト名用リソースの管理

```
/usr/cluster/bin/clreslogicalhostname [subcommand] -?

/usr/cluster/bin/clreslogicalhostname -V

/usr/cluster/bin/clreslogicalhostname [subcommand [options]] -v
     [lhresource]...

/usr/cluster/bin/clreslogicalhostname create -g resourcegroup
     [-h lhost[,…]] [-N netif@node[,…]] [-p name=value]
     [-Z {zoneclustername | global}] [-d] lhresource

/usr/cluster/bin/clreslogicalhostname create -i
     {- | clconfiguration} [-a] [-g resourcegroup[,…]] [-p
     name=value] [-d] {+ | lhresource...}

/usr/cluster/bin/clreslogicalhostname delete [-g resourcegroup[,…]]
     [-Z {zoneclustername | global}] [-F] {+ | lhresource...}

/usr/cluster/bin/clreslogicalhostname disable [-g resourcegroup[,…]]
     [-R] [-n node[,…]] [-Z {zoneclustername | global}]
     {+ | lhresource...}

/usr/cluster/bin/clreslogicalhostname enable [-g resourcegroup[,…]]
     [-R] [-n node[,…]] [-Z {zoneclustername | global}]
     {+ | lhresource...}

/usr/cluster/bin/clreslogicalhostname export [-o {- | configfile}]
     [+ | lhresource...]

/usr/cluster/bin/clreslogicalhostname list [-s state[,…]]
     [-g resourcegroup[,…]] [-Z {zoneclustername
     [,...] | global | all}] [+ | lhresource...]

/usr/cluster/bin/clreslogicalhostname list-props [-l listtype]
     [-p name[,…]] [-Z {zoneclustername [,...] | global | all}]
     [+ | lhresource...]

/usr/cluster/bin/clreslogicalhostname monitor [-g resourcegroup[,…]]
     [-Z zoneclustername | all | global] {+ | lhresource...}

/usr/cluster/bin/clreslogicalhostname reset [-f errorflag] [-g
     resourcegroup[,…]] [-Z {zoneclustername | global}]
     {+ | lhresource...}

/usr/cluster/bin/clreslogicalhostname set [-i
     {- | clconfiguration}] [-g resourcegroup[,…]] [-p name
     [+|-]=value] [-Z {zoneclustername}] {+ | lhresource...}

/usr/cluster/bin/clreslogicalhostname show [-g resourcegroup[,…]]
     [-p name[,…]] [-Z {zoneclustername [,...] | global | all}]
```

```
     [+ | lhresource...]

/usr/cluster/bin/clreslogicalhostname status [-s state[,…]]
     [-n node[,…]] [-g resourcegroup[,…]] [-Z
     {zoneclustername [,...] | global | all}] [+ | lhresource...]

/usr/cluster/bin/clreslogicalhostname unmonitor [-g
     resourcegroup[,…]] [-Z {zoneclustername | global}]
     {+ | lhresource...}
```

clreslogicalhostname コマンドは、Oracle Solaris Cluster 論理ホスト名用のリソースを管理します。clrslh コマンドは、clreslogicalhostname コマンドの短い形式です。clreslogicalhostname コマンドと clrslh コマンドは同じものです。どちらの形式のコマンドも使用できます。

clreslogicalhostname コマンドには、論理ホスト名リソースを作成するための便利なオプションが組み込まれています。clreslogicalhostname コマンドでは、Solaris IP マルチパス (IPMP) グループの自動作成もサポートされています。

clreslogicalhostname コマンドの一部のサブコマンドは、リソース構成を変更します:

- disable
- enable
- monitor
- reset
- set
- unmonitor

clreslogicalhostname コマンドの一部のサブコマンドは、リソースに関する情報だけを取得します。これらのサブコマンドは、グローバルクラスタまたはゾーンクラスタから使用できます: 次のコマンドは、リソースに関する情報のみを取得します:

- export
- list
- list-props
- show
- status

このコマンドからの予想不能な結果を避けるには、コマンドのすべての書式をグローバルクラスタノードから実行してください。

このコマンドの一般的な形式は次のとおりです。

`clreslogicalhostname` [*subcommand*] [*options*] [*operands*]

subcommand を省略できるのは、*options* で `-?`、`-o`、`-V`、または `-v` オプションが指定されている場合のみです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式については、このマニュアルページの「オプション」セクションを参照してください。

ゾーンクラスタでの操作

`clreslogicalhostname` コマンドは、ゾーンクラスタで、`export` を除くすべてのサブコマンドを指定して使用することができます。

`export` 以外のすべてのサブコマンドで `-Z` オプションを使用して、操作を制限する特定のゾーンクラスタの名前を指定することもできます。また、論理ホスト名リソース (*zoneclustername* : *lhresource*) にゾーンクラスタ名を付けて、操作を特定のゾーンクラスタに制限することもできます。

グローバルクラスタノードからすべてのゾーンクラスタ情報にアクセスできますが、特定のゾーンクラスタは他のゾーンクラスタを認識しません。特定のゾーンクラスタに操作を制限できない場合、使用するサブコマンドは現在のクラスタでのみ機能します。

サポートされるサブコマンドには次のものがあります。

`create`

コマンドに対するオペランドとして指定された論理ホスト名リソースを作成します。

`create` を `-i` オプションと使用して構成ファイルを指定した場合、サブコマンドはプラス記号 (+) をオペランドとして受け付けます。`+` オペランドを使用すると、構成ファイル内に存在しないすべてのリソースが作成されます。

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

デフォルトでは、リソースはモニタリング対象となり、有効な状態で作成されます。ただし、リソースがオンライン状態になり、モニターされるのは、リソースのリソースグループがオンラインになったあとだけです。無効な状態でリソースを作成するには、`-d` オプションを指定します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから、ゾーンクラスタ内の論理ホスト名リソースを作成するには、`-Z` オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。

delete サブコマンドの説明も参照してください。

delete

コマンドに対するオペランドとして指定された論理ホスト名リソースを削除します。このサブコマンドに対しオペランドとしてプラス記号 (+) を指定すると、すべてのリソースが削除されます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、削除するリソースを限定することができます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを削除します。

- デフォルトでは、リソースは次の条件が満たされる場合に *のみ* 削除されます。
- リソースが無効なとき
- リソースに対するすべての依存性が削除されているとき
- 指定したすべてのリソースを確実に削除するには、-F オプションを指定します。-F オプションの効果は、次のとおりです。
- 指定したすべてのリソース (無効になっていないリソースも含む) が削除されます。
- 指定したすべてのリソースが、他のリソースのリソース依存性設定から削除されます。

リソースは、コマンド行でリソースを指定した順序とは無関係に、リソース間の依存性を満たすのに必要な順序に従って削除されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから、ゾーンクラスタ内の論理ホスト名リソースを削除するには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

create サブコマンドの説明も参照してください。

disable

コマンドに対するオペランドとして指定された論理ホスト名リソースを無効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースが無効になります。

-g オプションを指定すると、オペランドのリストをフィルタリングし、無効にするリソースを限定することができます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを無効にします。

必要なリソース依存性をすべて確実に満たすには、-R オプションを指定します。-R オプションは、コマンドに対するオペランドとして指定したリソースに依存しているリソース (コマンドに対するオペランドとして指定しなかったリソースも含まれる) をすべて無効にします。-g オプションと -t オプションは、リソースの依存関係を満たすためだけに無効化されるリソースには適用されません。

リソースは、コマンド行でリソースを指定した順序とは無関係に、リソース間の依存性を満たすのに必要な順序に従って無効化されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから、ゾーンクラスタ内で登録された論理ホスト名リソースを無効にするには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

enable サブコマンドの説明も参照してください。

enable

コマンドに対するオペランドとして指定された論理ホスト名リソースを有効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースが有効になります。

-g オプションを指定すると、オペランドのリストをフィルタリングし、有効にするリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを有効にします。

必要なリソース依存性をすべて確実に満たすには、-R オプションを指定します。-R オプションは、コマンドに対するオペランドとして指定したリソースに依存しているリソース (コマンドに対するオペランドとして指定しなかったリソースも含まれる) をすべて有効にします。-g オプションは、リソースの依存関係を満たすためだけに有効化されるリソースには適用されません。

リソースは、コマンド行でリソースを指定した順序とは無関係に、リソース間の依存性を満たすのに必要な順序に従って有効化されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから、ゾーンクラスタ内で登録された論理ホスト名リソースを有効にするには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

disable サブコマンドの説明も参照してください。

export

1447 ページのclconfiguration(5CL) のマニュアルページで規定されている形式で、論理ホスト名リソース構成をエクスポートします。

このサブコマンドは、大域ゾーンだけで使用できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

list

コマンドに対するオペランドとして指定された論理ホスト名リソースのリストを表示します。デフォルトでは、すべてのリソースが表示されます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、表示するリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを表示します。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、指定したリソースグループ内のすべてのリソースまたは指定したリソースタイプのインスタンスであるすべてのリソースを指定できます。オペランドが指定されていない場合、指定されているリソースグループ内のすべてのリソースまたは指定されているリソースタイプのインスタンスであるすべてのリソースが表示されます。

-v オプションを指定すると、該当するリソースグループおよびリスト内の各リソースのリソースタイプも表示されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから、ゾーンクラスタ内で登録された論理ホスト名リソースを表示するには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

`list-props`

コマンドに対するオペランドとして指定された論理ホスト名リソースのプロパティーのリストを表示します。デフォルトでは、すべてのリソースの拡張プロパティーが表示されます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、プロパティーが表示されるリソースを限定できます。

-g *resourcegrouplist*
: *resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内の論理ホスト名リソースのみのプロパティーを表示します。

-l オプションでは、表示するリソースプロパティーのタイプを指定します。

`-l all`
: 標準プロパティーと拡張プロパティーを表示するように指定します。

`-l extension`
: 拡張プロパティーだけを表示するように指定します。デフォルトでは、拡張プロパティーだけが表示されます。

`-l standard`
: 標準プロパティーだけを表示するように指定します。

-l オプションを指定しない場合、-p オプションまたは -y オプションを使用して標準プロパティーを明示的に指定しないかぎり、拡張プロパティーだけが表示されます。

-p オプションは、表示するリソースプロパティーのセットを制限します。-p オプションを指定すると、*namelist* で指定したプロパティーだけが表示されます。*namelist* では、標準プロパティーと拡張プロパティーを指定できます。

-v オプションを指定すると、各プロパティーの説明も表示されます。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、指定したリソースグループ内のすべてのリソースまたは指定したリソースタイプのインスタンスであるすべてのリソー

スを指定できます。オペランドを指定しないと、指定したリソースグループ内のすべてのリソースのプロパティーまたは指定したリソースタイプのインスタンスであるすべてのリソースのプロパティーが表示されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

ゾーンクラスタの論理ホスト名リソースのプロパティーのリストを表示するには、-z オプションを使用してゾーンクラスタ名を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

`monitor`

コマンドに対するオペランドとして指定された論理ホスト名リソースのモニタリングを有効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースに対してモニタリングが有効になります。

-g オプションを指定すると、オペランドのリストをフィルタリングし、モニター対象のリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけをモニターします。

- リソースは、モニタリングが有効になっている場合、次の条件が満たされているときだけモニタリングされます。
- リソースが有効な状態にある。
- 該当リソースが含まれるリソースグループが、1 つ以上のクラスタノード上でオンライン状態にある。

注記 - リソースに対するモニタリングを有効にしても、そのリソースは有効*になりません*。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタのリソースをモニターするには、-z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.admin` RBAC の承認が必要です。

`unmonitor` サブコマンドの説明も参照してください。

`reset`

コマンドに対するオペランドとして指定された論理ホスト名リソースに関連付けられているエラーフラグをクリアします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースのエラーフラグがクリアされます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、リセットするリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけをリセットします。

デフォルトでは、reset サブコマンドによって STOP_FAILED エラーフラグがクリアーされます。クリアするエラーフラグを明示的に指定するには、-f オプションを使用します。-f オプションが受け付けるエラーフラグは、STOP_FAILED だけです。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから、ゾーンクラスタ内の論理ホスト名リソースをリセットするには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

set

コマンドに対するオペランドとして指定された論理ホスト名リソースの指定プロパティーを変更します。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースの指定したプロパティーが変更されます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、変更するリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを変更します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから、ゾーンクラスタ内の論理ホスト名リソースのプロパティーを設定するには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

show

コマンドに対するオペランドとして指定された論理ホスト名リソースの構成を表示します。デフォルトでは、すべてのリソースの構成が表示されます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、構成が表示されるリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースのみの構成を表示します。

-p オプションは、表示するリソースプロパティーのセットを制限します。-p オプションを指定すると、*namelist* で指定したプロパティーだけが表示されます。*namelist* では、標準プロパティーと拡張プロパティーを指定できます。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、指定したリソースグループ内のすべてのリソースまたは指定したリソースタイプのインスタンスであるすべてのリソースを指定できます。オペランドを指定しないと、指定したリソースグループ内のすべてのリソースの構成または指定したリソースタイプのインスタンスであるすべてのリソースの構成が表示されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから、ゾーンクラスタ内の論理ホスト名リソースの構成を表示するには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

`status`

コマンドに対するオペランドとして指定された論理ホスト名リソースのステータスを表示します。デフォルトでは、すべてのリソースのステータスが表示されます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、ステータスが表示されるリソースを限定できます。

`-g` *resourcegrouplist* — *resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースのみのステータスを表示します。

`-n` *nodelist* — *nodelist* 内のノード上でホストされている、オペランドのリスト内のリソースについてのみステータスを表示します。

`-s` *statelist* — *statelist* 内の状態にある、オペランドのリスト内のリソースだけのステータスを表示します。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、指定したリソースグループ内のすべてのリソースまたは指定したリソースタイプのインスタンスであるすべてのリソースを指定できます。オペランドを指定しないと、指定したリソースグループ内のすべてのリソースのステータスまたは指定したリソースタイプのインスタンスであるすべてのリソースのステータスが表示されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタから、ゾーンクラスタ内の論理ホスト名リソースのステータスを表示するには、`-Z` オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

`unmonitor`

コマンドに対するオペランドとして指定された論理ホスト名リソースのモニタリングを無効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースに対するモニタリングが無効になります。

無効になっているリソースのモニタリングを無効にしても、リソースは影響を受けません。リソースとそのモニターは、すでにオフライン状態です。

注記 - リソースに対するモニタリングを無効にしても、そのリソースは無効に*なりません*。ただし、リソースを無効にする場合、モニタリングを無効にする必要はありません。無効なリソースとそのモニターは、オフライン状態が維持されます。

`-g` オプションを指定すると、オペランドのリストをフィルタリングし、モニタリングを無効にするリソースを限定できます。`-g` オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースのモニタリングを無効にします。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタへの論理ホスト名リソースのモニタリングをオフにするには、-z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

disable サブコマンドおよび monitor サブコマンドの説明も参照してください。

次のオプションがサポートされています。

-?
--help

ヘルプ情報を表示します。このオプションを使用する場合、ほかの処理は実行されません。

このオプションを指定するとき、サブコマンドは指定してもしなくてもかまいません。

サブコマンドなしでこのオプションを指定すると、このコマンドのサブコマンドのリストが表示されます。

サブコマンド付きでこのオプションを指定すると、サブコマンドの使用方法が表示されます。

特定のサブコマンドを指定した場合のこのオプションの効果は、次のようになります。

create

-g オプションとともに指定された場合、指定されたリソースグループのすべてのリソースプロパティーのヘルプ情報を表示します。

set

コマンドに対するオペランドとして指定したリソースのプロパティーに関する情報を表示します。

-a
--automatic

クラスタ構成情報からリソースが作成される場合、次の処理も自動的に実行します。

- リソースタイプの登録
- リソースグループの作成
- オペランドのリスト内で指定されているリソースの依存先リソースの作成
- クラスタ構成情報には、次の処理をすべて実行するのに必要な十分な情報が含まれている必要があります。
- リソースタイプの登録を有効にする
- リソースグループの作成を有効にする
- リソースの作成を有効にする

このオプションを指定できるのは、create サブコマンドの場合だけです。このオプションを指定する場合は、-i オプションも指定し、構成ファイルを指定します。

-d
--disable

リソースの作成時にリソースを無効にします。このオプションを指定できるのは、create サブコマンドの場合だけです。デフォルトでは、リソースは作成されたあと、有効な状態になります。

リソースは、有効化しても、オンライン状態になるとは限りません。リソースは、リソースのリソースグループが 1 つ以上のノードでオンライン状態になったあとでのみオンライン状態になります。

-f *errorflag*
--flag *errorflag*

reset サブコマンドによってクリアするエラーフラグを明示的に指定します。このオプションを指定できるのは、reset サブコマンドの場合だけです。デフォルトでは、reset サブコマンドはエラーフラグ STOP_FAILED をクリアします。

-f オプションが受け付けるエラーフラグは、STOP_FAILED だけです。

-F
--force

無効状態でないリソースの削除が、強制的に実行されます。このオプションは、delete サブコマンドの場合にだけ指定できます。

-g *resourcegroup*[,…]
--resourcegroup *resourcegroup*[,…]

1 つのリソースグループまたはリソースグループのリストを指定します。

create 以外のサブコマンドの場合、コマンドは -g オプションで指定したリソースグループのメンバーである、オペランドのリスト内のリソースにだけ作用します。

特定のサブコマンドを指定した場合のこのオプションの効果は、次のようになります。

create	指定したリソースグループ内でリソースを作成するように指定します。-create サブコマンドを指定して g を使用する場合、リソースグループは 1 つしか指定することができません。

-h *lhost*[,…]
--logicalhost *lhost*[,…]

このリソースが表す論理ホスト名のリストを指定します。複数の論理ホスト名を新しい論理ホスト名リソースと関連付ける場合または論理ホスト名がリソース自体と同じ名前でない場合は、-h オプションを使用します。リスト内のすべての論理ホスト名は、同じサブネット上に置きます。-h オプションを指定しない場合、リソースはリソース名と同じ名前の 1 台の論理ホスト名を表します。

-h は、-p とともに HostnameList プロパティーを設定する代わりに使用できます。ただし、-h を使用し、同じコマンド内で HostnameList を明示的に設定することはできません。

-h は、create サブコマンドの場合にだけ使用できます。

注記 - ゾーンクラスタの場合、すべての論理ホスト名または対応する IP アドレスを、ゾーンクラスタ構成内のグローバルスコープのネットプロパティーで指定する必要があります。指定しない場合、リソースグループの作成に失敗します。

グローバルスコープのネットプロパティーについての詳細は、613 ページの`clzonecluster`(1CL) のマニュアルページを参照してください。

`-i` {- | *clconfiguration*}
`--input` {- | *clconfiguration*}

論理ホスト名リソースの作成または変更に使用する構成情報を指定します。この情報は、1447 ページの`clconfiguration`(5CL) のマニュアルページに定義されている形式に準拠している必要があります。この情報は、ファイルに含めることも、標準入力を使って指定することもできます。標準入力を指定するには、ファイル名の代わりに - を指定します。

コマンドに対するオペランドとして指定したリソースだけが、作成または変更されます。コマンドで指定したオプションは、構成情報で設定されているオプションより優先されます。構成パラメータは、構成情報内で設定されていない場合、コマンド行で指定します。

特定のサブコマンドを指定した場合のこのオプションの効果は、次のようになります。

`create`	`-a` オプションと指定された場合は、必要なすべてのリソースタイプを登録し、必要なすべてのリソースグループを作成します。登録および構成に必要なすべての情報を指定します。その他の構成データはすべて無視します。

`-l` *listtype*
`--listtype` *listtype*

`list-props` サブコマンドによって表示するリソースプロパティーのタイプを指定します。このオプションは、`list-props` サブコマンドの場合にだけ指定できます。

listtype に対して、次のリストから値を 1 つ指定する必要があります。

`all`	標準プロパティーと拡張プロパティーを表示するように指定します。
`extension`	拡張プロパティーだけを表示するように指定します。デフォルトでは、拡張プロパティーだけが表示されます。
`standard`	標準プロパティーだけを表示するように指定します。

`-l` オプションを指定しないと、`-p` オプションを使って標準プロパティーを明示的に指定しないかぎり、拡張プロパティーしか表示されません。

`-n` *node*[,…]
`--node` *node*[,…]

ターゲットのグローバルクラスタまたはゾーンクラスタに、ノードまたはノードのリストを指定します。各ノードは、ノード名またはノード ID として指定できます。`-Z` オプションが指定されて

いる場合は、-n オプションで、グローバルクラスタホスト名ではなくゾーンクラスタホスト名のみを指定できます。-Z オプションが指定されていない場合は、-n オプションでグローバルクラスタホスト名のみを指定できます。

このオプションとともに指定できるサブコマンドは、次のとおりです。

`disable`	指定したノード上でホストされている、オペランドのリスト内のリソースのみを無効にします。
`enable`	指定したノード上でホストされている、オペランドのリスト内のリソースのみを有効にします。
`status`	指定したノード上でホストされている、オペランドのリスト内のリソースについてのみステータスを報告します。

`-N` *netif@node*[,…]
`--netiflist` *netif@node*[,…]

リソースプロパティーを指定します。-N オプションを使用すると、プロパティーの p オプションを使用せずに -NetIfList プロパティーを設定できます。-N を指定しない場合、clreslogicalhostname コマンドは、利用できる IPMP グループまたはパブリックアダプタと、HostnameList プロパティーと関連付けられているサブネットに基づいて、NetIfList プロパティーを設定しようと試みます。

NetIfList プロパティーは、*ipmpgroup@node*[,…]、または *publicNIC@node*[,…] の形式で指定できます。-N を使用しない場合、またはそれを *publicNIC@node* とともに使用する場合、clreslogicalhostname コマンドは必要な IPMP グループを作成しようと試みます。システムは基本デフォルトで単一アダプタの IPMP グループを作成します。このグループは、あとで標準 Solaris IPMP インタフェースを使用して変更できます。IPMP グループは、グローバルクラスタノードの場合のみ、自動的に作成されます。

-N は、NetIfList プロパティーを -p とともに直接設定する代わりに使用できます。ただし、同じコマンド内で -N を使用して、NetIfList を明示的に設定できません。

-N は、create サブコマンドの場合にだけ使用できます。

`-o` {- | *clconfiguration*}
`--output` {- | *clconfiguration*}

リソース構成情報の書き込み先を指定します。この書き込み先は、ファイルでも、標準出力でもかまいません。標準出力を指定するには、ファイル名の代わりに - を指定します。標準出力を指定すると、コマンドに対するほかのすべての標準出力が抑制されます。このオプションは、export サブコマンドの場合にだけ指定できます。

構成情報は、コマンドに対するオペランドとして指定したリソースの場合だけ書き込まれます。この情報は、1447 ページのclconfiguration(5CL) のマニュアルページに定義されている形式で書き込まれます。

-p *name=value*
-p *name+=array-values*
-p *name-=array-values*
--property *name=value*
--property *name+=value-values*
--property *name-=value-values*

リソースの標準プロパティーと拡張プロパティーを設定します。このオプションは、create サブコマンドおよび set サブコマンドの場合にだけ指定できます。

標準プロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

リソースタイプの拡張プロパティーについては、リソースタイプに関するドキュメントを参照してください。

このオプションとともに使用する演算子は、次のとおりです。

=	プロパティーに、指定した値を設定します。create サブコマンドおよび set サブコマンドがこの演算子を受け付けます。
+=	1 つまたは複数の値を文字列配列値に追加します。この演算子は、set サブコマンドでのみ使用できます。この演算子は、文字列配列値に対してだけ指定できます。
-=	1 つまたは複数の値が、文字列配列値から削除されます。この演算子は、set サブコマンドでのみ使用できます。この演算子は、文字列配列値に対してだけ指定できます。

ノード単位のプロパティーをクラスタノードのサブセット上でのみ設定する場合は、中括弧で囲まれたノードのリストを次のようにプロパティー名の後に付加し、プロパティーが設定の対象となるノードを指定します。

name{*nodelist*}

nodelist は、ノード名またはノード ID をコンマで区切ったリストです。ノード単位のプロパティーの詳細は、1335 ページのrt_properties(5) のマニュアルページを参照してください。

-p *name*[,…]
--property *name*[,…]

list-props サブコマンドおよび show サブコマンドのプロパティーのリストを指定します。

このオプションは、リソースの標準プロパティーおよび拡張プロパティーに対して使用できます。

標準プロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

リソースタイプの拡張プロパティーについては、リソースタイプに関するドキュメントを参照してください。

このオプションを指定しない場合、list-props サブコマンドおよび show サブコマンドは、-v オプションも指定されているかどうかに応じて、すべてまたはほとんどのリソースプロパティーを一覧表示します。

-R
--recursive

必要な依存性がすべて満たされるように、リソースの有効化または無効化を再帰的に実行します。このオプションは、disable サブコマンドおよび enable サブコマンドでのみ指定できます。

このオプションをこれらのサブコマンドとともに指定した場合の効果は、次のとおりです。

disable	コマンドに対するオペランドとして指定したリソースに依存しているリソース (コマンドに対するオペランドとして指定しなかったリソースも含まれる) をすべて無効にします。
enable	コマンドに対するオペランドとして指定したリソースの依存先リソース (コマンドに対するオペランドとして指定しなかったリソースも含まれる) がすべて有効になります。

-s *state*[,…]
--state *state*[,…]

list サブコマンドおよび status サブコマンドの状態のリストを指定します。

このオプションは出力を制限し、ノードリスト内の 1 つまたは複数のノード上で指定されている状態の 1 つにあるリソースだけが含まれるようにします。

可能な状態は、次のとおりです。

- degraded
- detached
- faulted
- monitor_failed
- not_online - online または online_not_monitored 以外のすべてのステータスを指定します
- offline
- online
- online_not_monitored
- start_failed
- stop_failed
- unknown
- unmonitored

- `wait`

`-V`
`--version`

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンドやオペランドなどのオプションは無視されます。`-V` オプションは、コマンドのバージョンを表示するだけです。その他の処理は行いません。

`-v`
`--verbose`

詳細なメッセージを標準出力に表示します。

このオプションは、このコマンドの任意の形式とともに指定できます。

`-v` オプションと `-o -` オプションを同時に指定してはいけません。`-v` オプ ションは無視されます。`-o-` オプションは、その他のすべての標準出力を抑制します。

`-Z` {*zoneclustername* | global | all}
`--zoneclustername=`{*zoneclustername* | global | all}
`--zoneclustername` {*zoneclustername* | global | all}

クラスタ、またはリソースが存在するクラスタや処理するクラスタを指定します。

このオプションは、`export` サブコマンドを除くすべてのサブコマンドでサポートされています。

このオプションを指定する場合は、次のいずれかの引数も指定する必要があります。

zoneclustername	このオプションを使用しているコマンドが、*zoneclustername* という名前のゾーンクラスタ内のみの指定されたすべてのリソースに対して動作することを指定します。
`global`	このオプションを使用するコマンドが、広域クラスタでのみ指定されたすべてのリソースで機能するように指定します。
`all`	広域クラスタでこの引数を使用する場合、それを使用するコマンドが広域クラスタとすべてのゾーンクラスタを含め、すべてのクラスタで指定されたすべてのリソースで機能するように指定します。 この引数をゾーンクラスタで使用すると、このオプションを使用するコマンドが、ゾーンクラスタ内だけで指定されたすべてのリソースに対して機能するように指定されます。

次のオペランドを指定できます。

resource	Oracle Solaris Cluster のリソース名をオペランドとして受け付けるように指定します。サブコマンドで複数のリソースを指定できる場合は、プラス記号 (+) を使用すると、すべての論理ホスト名リソースを指定できます。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

9 CL_ESTATE

オブジェクトの状態が不正です

その時点で変更できない、または常時変更できないプロパティー、リソースグループ、またはその他のオブジェクトを変更しようとしました。

10 CL_EMETHOD

リソースのメソッドが失敗しました

リソースのメソッドが失敗しました。次のいずれかの理由のために、メソッドは失敗しました。

- validate メソッドは、リソースを作成しようとしたときに、あるいは、リソースのプロパティーを変更しようとしたときに失敗しました。
- validate 以外のメソッドは、リソースを有効、無効、または削除しようとしたときに失敗しました。

15 CL_EPROP

無効なプロパティーです

-p、-y、または -x オプションで指定したプロパティーまたは値が存在しないか、許可されていません。

35 CL_EIO

I/O エラー

物理的な入出力エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

これらの終了値は、1047 ページのscha_calls(3HA) のマニュアルページで説明されているリターンコードと互換性があります。

例 200 論理ホスト名リソースの作成

このコマンドは、rg-failover という名前のリソースグループ内に logicalhost1 という名前のリソースを作成します。リソースは有効な状態で作成され、モニタリングも有効になっています。

```
# clreslogicalhostname create -g rg-failover logicalhost1
```

次の 2 つのコマンドはどちらも、ゾーンクラスタ ZC 内に logicalhost1 という名前のリソースを作成します。これらのコマンドは、グローバルクラスタノードまたはゾーンクラスタ ZC から実行できます。コマンドをゾーンクラスタから実行する場合、任意でゾーンクラスタ名を持つリソースのスコープを明示的に定義できます。

```
# clreslogicalhostname create -g rg-failover -Z ZC logicalhost1

# clreslogicalhostname create -g rg-failover ZC:logicalhost1
```

例 201 異なる論理ホスト名を持つ論理ホスト名リソースの作成

このコマンドは、rg-failover という名前のリソースグループ内に、rs-logicalhost1 という名前のリソースを作成します。

論理ホスト名はリソース名と同じではありませんが、論理ホストの名前と IP アドレスは同じままです。

```
# clreslogicalhostname create -g rg-failover \
-h logicalhost1 rs-logicalhost1
```

例 202 論理ホスト名リソースの IPMP グループの指定

このコマンドは、logicalhost1 リソースの IPMP グループを設定します。

```
# clreslogicalhostname create -g rg-failover \
-N ipmp0@black,ipmp0@white logicalhost1
```

例 203 論理ホスト名リソースの削除

このコマンドは、logicalhost1 という名前のリソースを削除します。

```
# clreslogicalhostname delete logicalhost1
```

例 204 論理ホスト名リソースの一覧表示

このコマンドは、すべての論理ホスト名リソースを一覧表示します。

```
# clreslogicalhostname list
logicalhost1
logicalhost2
```

例 205 論理ホスト名リソース、リソースグループ、およびリソースタイプの一覧表示

このコマンドは、すべての論理ホスト名リソースをそれらのリソースグループおよびリソースタイプと合わせて一覧表示します。

```
# clreslogicalhostname list -v
Resources    Resource Groups Resource Types
---------    --------------- --------------
logicalhost1 rg-failover-1   SUNW.LogicalHostname
logicalhost2 rg-failover-2   SUNW.LogicalHostname
```

例 206 論理ホスト名リソースの拡張プロパティーの一覧表示

このコマンドは、すべての論理ホスト名リソースの拡張プロパティーを一覧表示します。

```
# clreslogicalhostname list-props -v
Properties       Descriptions
----------       ------------
NetIfList        List of IPMP groups on each node
HostnameList     List of hostnames this resource manages
CheckNameService Name service check flag
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページの`Intro`(1CL), 551 ページの`cluster`(1CL), 273 ページの`clresource`(1CL), 347 ページの`clressharedaddress`(1CL), 305 ページの`clresourcegroup`(1CL), 333 ページの`clresourcetype`(1CL), 1047 ページの`scha_calls`(3HA), 1447 ページの`clconfiguration`(5CL), Unresolved link to " rbac5", 1287 ページの`r_properties`(5)

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のオプションを指定してこのコマンドを実行できます。

- `-?` オプション
- `-V` オプション

サブコマンドを指定してこのコマンドを実行する場合、スーパーユーザー以外のユーザーは RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
`create`	`solaris.cluster.modify`
`delete`	`solaris.cluster.modify`
`disable`	`solaris.cluster.admin`
`enable`	`solaris.cluster.admin`
`export`	`solaris.cluster.read`
`list`	`solaris.cluster.read`

サブコマンド	RBAC の承認
list-props	solaris.cluster.read
monitor	solaris.cluster.admin
reset	solaris.cluster.admin
set	solaris.cluster.modify
show	solaris.cluster.read
status	solaris.cluster.read
unmonitor	solaris.cluster.admin

名前

clressharedaddress, clrssa — 共有アドレスの Oracle Solaris Cluster リソースの管理

```
/usr/cluster/bin/clressharedaddress [subcommand] -?

/usr/cluster/bin/clressharedaddress -V

/usr/cluster/bin/clressharedaddress [subcommand [options]] -v
     [saresource]...

/usr/cluster/bin/clressharedaddress create -g resourcegroup [-h
     lhost[,…]] [-N netif@node[,…]] [-X node[,…]]
     [-p name=value] [-Z {zoneclustername | global}] [-d] saresource

/usr/cluster/bin/clressharedaddress create -i
     {- | clconfiguration} [-a] [-g resourcegroup[,…]] [-X
     node[,…]] [-p name=value] [-d] {+ | saresource...}

/usr/cluster/bin/clressharedaddress delete [-g resourcegroup[,…]]
     [-Z {zoneclustername | global}] [-F] {+ | saresource...}

/usr/cluster/bin/clressharedaddress disable [-g resourcegroup[,…]]
     [-R] [-n node[,…]] [-Z
     {zoneclustername | global}] {+ | saresource...}

/usr/cluster/bin/clressharedaddress enable [-g resourcegroup[,…]]
     [-R] [ -n node[,…]] [-Z
     {zoneclustername | global}]{+ | saresource...}

/usr/cluster/bin/clressharedaddress export [-o {- | configfile}]
     [+ | saresource...]

/usr/cluster/bin/clressharedaddress list [-s state[,…]]
     [-g resourcegroup[,…]] [-Z {zoneclustername
     [,...] | global | all}] [+ | saresource...]

/usr/cluster/bin/clressharedaddress list-props [-l listtype] [-p
     name[,…]] [-Z {zoneclustername [,...] | global | all}]
     [+ | lhresource...]

/usr/cluster/bin/clressharedaddress monitor [-g resourcegroup[,…]]
     [-Z {zoneclustername | global}] {+ | saresource...}

/usr/cluster/bin/clressharedaddress reset [-f errorflag] [-g
     resourcegroup[,…]] [-Z {zoneclustername | global}]
     {+ | saresource...}

/usr/cluster/bin/clressharedaddress set [-i {- | clconfiguration}]
     [-g resourcegroup[,…]] [-X node[,…]] [-p
```

```
     name[+|-]=value] [-Z {zoneclustername | global}] {+ | saresource...}

/usr/cluster/bin/clressharedaddress show [-g resourcegroup[,…]]
     [-p name[,…]] [-Z {zoneclustername [,...] | global | all}]
     [+ | saresource...]

/usr/cluster/bin/clressharedaddress status [-s state[,…]]
     [ -n node[,…]] [-g resourcegroup[,…]]
     [-Z {zoneclustername [,...] | global | all}] [+ | saresource...]

/usr/cluster/bin/clressharedaddress unmonitor [-g resourcegroup[,…]]
     [-Z {zoneclustername | global}] {+ | saresource...}
```

clressharedaddress コマンドは、Oracle Solaris Cluster 共有アドレスのリソースを管理します。clrssa コマンドは clressharedaddress コマンドの短い形式です。clressharedaddress コマンドと clrssa コマンドは同じものです。どちらの形式のコマンドも使用できます。

273 ページのclresource(1CL) コマンドを使用して、共有アドレスのリソースを管理することもできます。

clressharedaddress コマンドの一部のサブコマンドは、リソース構成を変更します。これらのサブコマンドは、グローバルクラスタまたはゾーンクラスタから使用できます。リソース構成を変更するサブコマンドは、次のとおりです。

- disable
- enable
- monitor
- reset
- set
- unmonitor

clressharedaddress コマンドの一部のサブコマンドは、リソースに関する情報のみを取得します。

- export
- list
- list-props
- show
- status

このコマンドからの予想不能な結果を避けるには、コマンドのすべての書式をグローバルクラスタノードから実行してください。

コマンドの一般的な形式は次のとおりです。

`clressharedaddress` [*subcommand*] [*options*] [*operands*]

subcommand を省略できるのは、*options* がオプション `-?` または `-V` を指定している場合だけです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式は、このマニュアルページの「オプション」セクションのオプションの説明で紹介されています。

ゾーンクラスタでの操作

`clressharedaddress` コマンドは、ゾーンクラスタで、`export` を除くすべてのサブコマンドを指定して使用することができます。

`export` 以外のすべてのサブコマンドで `-Z` オプションを使用して、操作を制限する特定のゾーンクラスタの名前を指定することもできます。また、共有アドレスリソース (*zoneclustername* : *saresource*) にゾーンクラスタ名を付けて、操作を特定のゾーンクラスタに制限することもできます。

グローバルクラスタノードからすべてのゾーンクラスタ情報にアクセスできますが、特定のゾーンクラスタは他のゾーンクラスタを認識しません。特定のゾーンクラスタに操作を制限できない場合、使用するサブコマンドは現在のクラスタでのみ機能します。

サポートされるサブコマンドには次のものがあります。

`create`

コマンドに対するオペランドとして共有アドレス指定されたリソースを作成します。

`create` を `-i` オプションと使用して構成ファイルを指定した場合、サブコマンドはプラス記号 (+) をオペランドとして受け付けます。`+` オペランドを使用すると、構成ファイル内に存在しないすべてのリソースが作成されます。

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

デフォルトでは、リソースはモニタリング対象となり、有効な状態で作成されます。ただし、リソースがオンライン状態になり、モニターされるのは、リソースのリソースグループがオンラインになったあとだけです。無効な状態でリソースを作成するには、`-d` オプションを指定します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタに共有アドレスリソースを作成するために、-z オプションを使用してゾーンクラスタの名前を指定できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。

delete サブコマンドの説明も参照してください。

delete

コマンドに対するオペランドとして指定された共有アドレスリソースを削除します。このサブコマンドに対しオペランドとしてプラス記号 (+) を指定すると、すべてのリソースが削除されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタに共有アドレスリソースを作成するために、-z オプションを使用してゾーンクラスタの名前を指定できます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、削除するリソースを限定することができます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを削除します。

- デフォルトでは、リソースは次の条件が満たされる場合に *のみ* 削除されます。
- リソースが無効な状態である。
- リソースに対するすべての依存性が削除されている。
- 指定したすべてのリソースを確実に削除するには、-F オプションを指定します。-F オプションの効果は、次のとおりです。
- 指定したすべてのリソース (無効になっていないリソースも含む) が削除されます。
- 指定したすべてのリソースが、他のリソースのリソース依存性設定から削除されます。

リソースは、コマンド行でリソースを指定した順序とは無関係に、リソース間の依存性を満たすのに必要な順序に従って削除されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

create サブコマンドの説明も参照してください。

disable

コマンドに対するオペランドとして指定された共有アドレスリソースを無効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースが無効になります。

-g オプションを指定すると、オペランドのリストをフィルタリングし、無効にするリソースを限定することができます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを無効にします。

必要なリソース依存性をすべて確実に満たすには、-R オプションを指定します。-R オプションは、コマンドに対するオペランドとして指定したリソースに依存しているリソース (コマンドに対するオペランドとして指定しなかったリソースも含まれる) をすべて無効にします。-g オプションと -t オプションは、リソース依存性を満たすためだけに無効化されるリソースには適用されません。

リソースは、コマンド行でリソースを指定した順序とは無関係に、リソース間の依存性を満たすのに必要な順序に従って無効化されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタに登録された共有アドレスリソースを無効にするには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

enable サブコマンドの説明も参照してください。

enable

コマンドに対するオペランドとして指定された共有アドレスリソースを有効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースが有効になります。

-g オプションを指定すると、オペランドのリストをフィルタリングし、有効にするリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを有効にします。

必要なリソース依存性をすべて確実に満たすには、-R オプションを指定します。-R オプションは、コマンドに対するオペランドとして指定したリソースに依存しているリソース (コマンドに対するオペランドとして指定しなかったリソースも含まれる) をすべて有効にします。-g オプションは、リソース依存性を満たすためだけに有効化されるリソースには適用されません。

リソースは、コマンド行でリソースを指定した順序とは無関係に、リソース間の依存性を満たすのに必要な順序に従って有効化されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタに登録された共有アドレスリソースを有効にするには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

disable サブコマンドの説明も参照してください。

export

1447 ページのclconfiguration(5CL) のマニュアルページで規定されている形式で、共有アドレスリソース構成をエクスポートします。

このサブコマンドは、大域ゾーンだけで使用できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。

`list`

マンドに対するオペランドとして指定した共有アドレスリソースのリストを表示します。デフォルトでは、すべてのリソースが表示されます。

`-g` オプションを指定すると、オペランドのリストをフィルタリングし、表示するリソースを限定できます。`-g` オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを表示します。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、指定したリソースグループ内のすべてのリソースまたは指定したリソースタイプのインスタンスであるすべてのリソースを指定できます。オペランドが指定されていない場合、指定されているリソースグループ内のすべてのリソースまたは指定されているリソースタイプのインスタンスであるすべてのリソースが表示されます。

`-v` オプションを指定すると、該当するリソースグループおよびリスト内の各リソースのリソースタイプも表示されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタに登録された共有アドレスリソースを表示するには、`-Z` オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

`list-props`

コマンドに対するオペランドとして指定した共有アドレスリソースのプロパティーのリストを表示します。デフォルトでは、すべてのリソースの拡張プロパティーが表示されます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、プロパティーが表示されるリソースを限定できます。

`-g` *resourcegrouplist*
: *resourcegrouplist* 内のリソースグループのメンバーであるオペランドのリストの共有アドレスリソースのプロパティーだけを表示します。

`-l` オプションでは、表示するリソースプロパティーのタイプを指定します。

`-l all`
: 標準プロパティーと拡張プロパティーを表示するように指定します。

`-l extension`
: 拡張プロパティーだけを表示するように指定します。デフォルトでは、拡張プロパティーだけが表示されます。

`-l standard`
: 標準プロパティーだけを表示するように指定します。

`-l` オプションを指定しない場合、`-p` オプションまたは `-y` オプションを使用して標準プロパティーを明示的に指定しないかぎり、拡張プロパティーだけが表示されます。

`-p` オプションは、表示するリソースプロパティーのセットを制限します。`-p` オプションを指定すると、*namelist* で指定したプロパティーだけが表示されます。*namelist* では、標準プロパティーと拡張プロパティーを指定できます。

-v オプションを指定すると、各プロパティーの説明も表示されます。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、指定したリソースグループ内のすべてのリソースまたは指定したリソースタイプのインスタンスであるすべてのリソースを指定できます。オペランドを指定しないと、指定したリソースグループ内のすべてのリソースのプロパティーまたは指定したリソースタイプのインスタンスであるすべてのリソースのプロパティーが表示されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタの共有アドレスリソースのプロパティーのリストを表示するには、-z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

`monitor`

コマンドに対するオペランドとして指定した共有アドレスリソースのモニタリングを有効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースに対してモニタリングが有効になります。

-g オプションを指定すると、オペランドのリストをフィルタリングし、モニター対象のリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけをモニターします。

リソースは、モニタリングが有効になっている場合、次の条件が満たされているときだけモニタリングされます。

- リソースが有効な状態にある。
- 該当リソースが含まれるリソースグループが、1 つ以上のクラスタノード上でオンライン状態にある。

注記 - リソースに対するモニタリングを有効にする場合、該当リソースを有効にする必要は*ありません*。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタのリソースをモニターするには、-z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.admin` RBAC の承認が必要です。

`unmonitor` サブコマンドの説明も参照してください。

`reset`

コマンドに対するオペランドとして指定した共有アドレスリソースに関連付けられているエラーフラグをクリアします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースのエラーフラグがクリアされます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、リセットするリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけをリセットします。

デフォルトでは、reset サブコマンドはエラーフラグ STOP_FAILED をクリアします。クリアするエラーフラグを明示的に指定するには、-f オプションを使用します。-f オプションが受け付けるエラーフラグは、STOP_FAILED だけです。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタの共有アドレスリソースをリセットするには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

set

コマンドのオペランドとして指定されている共有アドレスリソースの指定プロパティーを変更します。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースの指定したプロパティーが変更されます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、変更するリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけを変更します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタに共有アドレスリソースのプロパティーを設定するには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

show

コマンドに対するオペランドとして指定した共有アドレスリソースの構成を表示します。デフォルトでは、すべてのリソースの構成が表示されます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、構成が表示されるリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけの構成を表示します。

-p オプションは、表示するリソースプロパティーのセットを制限します。-p オプションを指定すると、*namelist* で指定したプロパティーだけが表示されます。*namelist* では、標準プロパティーと拡張プロパティーを指定できます。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、指定したリソースグループ内のすべてのリソースまたは指定したリソースタイプのインスタンスであるすべてのリソースを指定できます。オペランドを指定しないと、指定したリソースグループ内のすべてのリソースの構成または指定したリソースタイプのインスタンスであるすべてのリソースの構成が表示されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタに共有アドレスリソースの構成を表示するには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

`status`

コマンドに対するオペランドとして指定したリソースのステータスを表示します。デフォルトでは、すべてのリソースのステータスが表示されます。

次のオプションを指定すると、オペランドのリストをフィルタリングし、ステータスが表示されるリソースを限定できます。

-g *resourcegrouplist* — *resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースだけのステータスを表示します。

-n *nodelist* — *nodelist* 内のノード上でホストされている、オペランドのリスト内のリソースについてのみステータスを表示します。

-s *statelist* — *statelist* 内の状態にある、オペランドのリスト内のリソースだけのステータスを表示します。

このサブコマンドでプラス記号 (+) をオペランドとして指定すると、指定したリソースグループ内のすべてのリソースまたは指定したリソースタイプのインスタンスであるすべてのリソースを指定できます。オペランドを指定しないと、指定したリソースグループ内のすべてのリソースのステータスまたは指定したリソースタイプのインスタンスであるすべてのリソースのステータスが表示されます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタに共有アドレスリソースのステータスを表示するには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。

`unmonitor`

コマンドに対するオペランドとして指定した共有アドレスリソースのモニタリングを無効にします。このサブコマンドでプラス記号 (+) をオペランドとして指定すると、すべてのリソースに対するモニタリングが無効になります。

無効になっているリソースのモニタリングを無効にしても、リソースは影響を受けません。リソースとそのモニターは、すでにオフライン状態です。

注記 - リソースのモニタリングを無効にしても、リソースは無効に*なりません*。ただし、リソースを無効にする場合、モニタリングを無効にする必要はありません。無効なリソースとそのモニターは、オフライン状態が維持されます。

-g オプションを指定すると、オペランドのリストをフィルタリングし、モニタリングを無効にするリソースを限定できます。-g オプションは、*resourcegrouplist* 内のリソースグループのメンバーである、オペランドのリスト内のリソースのモニタリングを無効にします。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタへの共有アドレスリソースのモニタリングをオフにするには、-Z オプションを使用してゾーンクラスタの名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。

disable サブコマンドおよび monitor サブコマンドの説明も参照してください。

次のオプションがサポートされています。

-?
--help

ヘルプ情報を表示します。このオプションを使用する場合、ほかの処理は実行されません。

このオプションを指定するとき、サブコマンドは指定してもしなくてもかまいません。

サブコマンドなしでこのオプションを指定すると、このコマンドのサブコマンドのリストが表示されます。

サブコマンド付きでこのオプションを指定すると、サブコマンドの使用方法が表示されます。

特定のサブコマンドを指定した場合のこのオプションの効果は、次のようになります。

create	-g オプションとともに指定された場合、指定されたリソースグループのすべてのリソースプロパティーのヘルプ情報を表示します。
set	コマンドに対するオペランドとして指定したリソースのプロパティーに関する情報を表示します。

-a
--automatic

クラスタ構成情報からリソースが作成される場合、次の処理も自動的に実行します。

- リソースタイプの登録
- リソースグループの作成
- オペランドのリスト内で指定されているリソースの依存先リソースの作成

クラスタ構成情報には、次の処理をすべて実行するのに必要な十分な情報が含まれている必要があります。

- リソースタイプの登録を有効にする
- リソースグループの作成を有効にする

■ 作成されるリソースの有効化

このオプションを指定できるのは、create サブコマンドの場合だけです。このオプションを指定する場合は、-i オプションも指定し、構成ファイルを指定します。

-d
--disable

リソースの作成時にリソースを無効にします。このオプションを指定できるのは、create サブコマンドの場合だけです。デフォルトでは、リソースは作成されたあと、有効な状態になります。

リソースは、有効化しても、オンライン状態になるとは限りません。リソースは、リソースのリソースグループが 1 つ以上のノードでオンライン状態になったあとでのみオンライン状態になります。

-f *errorflag*
--flag *errorflag*

reset サブコマンドによってクリアするエラーフラグを明示的に指定します。このオプションを指定できるのは、reset サブコマンドの場合だけです。デフォルト時、reset サブコマンドはエラーフラグ STOP_FAILED をクリアします。

-f オプションが受け付けるエラーフラグは、STOP_FAILED だけです。

-F
--force

無効状態でないリソースの削除が、強制的に実行されます。このオプションは、delete サブコマンドの場合にだけ指定できます。

-g *resourcegroup*[,…]
--resourcegroup *resourcegroup*[,…]

1 つのリソースグループまたはリソースグループのリストを指定します。

create 以外のサブコマンドの場合、コマンドは -g オプションで指定したリソースグループのメンバーである、オペランドのリスト内のリソースにだけ作用します。

特定のサブコマンドを指定した場合のこのオプションの効果は、次のようになります。

create	指定したリソースグループ内でリソースを作成するように指定します。-create サブコマンドを指定して g を使用する場合、リソースグループは 1 つしか指定することができません。

-h *lhost*[,…]
--logicalhost *lhost*[,…]

ホスト名リストを指定します。複数の論理ホストを新しい -SharedAddress リソースに関連付ける必要がある場合や論理ホストがリソース自体と同じ名前を持っていない場合は、h オプションを使用します。SharedAddress リソースの HostnameList 内のすべての論理

ホストは同じサブネット上に置いてください。HostnameList プロパティーを指定しない場合、HostnameList は SharedAddress リソースと同じになります。

SharedAddress リソースの論理ホスト名は同じサブネット上に置いてください。

-HostnameList プロパティーを p とともに設定する代わりに、-h を使用することは可能です。ただし、-h を使用して、同じコマドで明示的に HostnameList を設定できません。

-h は、create サブコマンドの場合にだけ使用できます。

-i {- | *clconfiguration*}
--input {- | *clconfiguration*}

共有アドレスリソースの作成または変更に使用する構成情報を指定します。この情報は、1447 ページのclconfiguration(5CL) のマニュアルページに定義されている形式に準拠している必要があります。この情報は、ファイルに含めることも、標準入力を介して指定することもできます。標準入力を指定するには、ファイル名の代わりに - を指定します。

コマンドに対するオペランドとして指定したリソースだけが、作成または変更されます。コマンドで指定したオプションは、構成情報で設定されているオプションより優先されます。構成パラメータは、構成情報内で設定されていない場合、コマンド行で指定します。

特定のサブコマンドを指定した場合のこのオプションの効果は、次のようになります。

create	-a オプションと指定された場合は、必要なすべてのリソースタイプを登録し、必要なすべてのリソースグループを作成します。登録および構成に必要なすべての情報を指定します。その他の構成データはすべて無視します。

-l *listtype*
--listtype *listtype*

list-props サブコマンドによって表示するリソースプロパティーのタイプを指定します。このオプションは、list-props サブコマンドの場合にだけ指定できます。

listtype に対して、次のリストから値を 1 つ指定します。

all	標準プロパティーと拡張プロパティーを表示するように指定します。
extension	拡張プロパティーだけを表示するように指定します。デフォルトでは、拡張プロパティーだけが表示されます。
standard	標準プロパティーだけを表示するように指定します。

-l オプションを指定しないと、-p オプションを使って標準プロパティーを明示的に指定しないかぎり、拡張プロパティーしか表示されません。

-n *node*[,…]
--node *node*[,…]

ターゲットのグローバルクラスタまたはゾーンクラスタに、ノードまたはノードのリストを指定します。各ノードは、ノード名またはノード ID として指定できます。

-Z オプションが指定されている場合は、-n オプションで、グローバルクラスタホスト名ではなくゾーンクラスタホスト名のみを指定できます。-Z オプションが指定されていない場合は、-n オプションでグローバルクラスタホスト名のみを指定できます。

このオプションとともに指定できるサブコマンドは、次のとおりです。

`disable`	指定したノード上でホストされている、オペランドのリスト内のリソースのみを無効にします。
`enable`	指定したノード上でホストされている、オペランドのリスト内のリソースのみを有効にします。
`status`	指定したノード上でホストされている、オペランドのリスト内のリソースについてのみステータスを報告します。

`-N` *netif*`@`*node*[,…]
`--netiflist` *netif*`@`*node*[,…]

リソースプロパティーを指定します。-N オプションを使用すると、プロパティーの p オプションを使用せずに -NetIfList プロパティーを設定できます。-N を指定しないと、clressharedaddress コマンドは使用可能な IPMP グループまたはパブリックアダプタと、HostnameList プロパティーと関連付けられているサブネットに基づいて、NetIfList プロパティーを自動的に設定するよう試みます。

NetIfList プロパティーは、次の形式で指定できます。*ipmpgroup*`@`*node*[,…]。ただし、-N は両方の *ipmpgroup*`@`*node*[,…] および *publicNIC*`@`*node*[,…] を受け入れます。-N を使用しなかったり、*publicNIC*`@`*node* とともに使用したりすると、clressharedaddress コマンドは、必要な IPMP グループを作成しようとします。システムは、標準 Oracle Solaris インタフェースを使用して複数のアダプタを含めるようにあとで変更されるデフォルトのセットで、1 つ以上の単一アダプタ IPMP グループのセットを作成します。

-p を指定して NetIfList プロパティーを直接設定する代わりに -N を指定することができます。ただし、-N を使用し、同じコマンド内で NetIfList を明示的に設定できません。

-N は、create サブコマンドの場合にだけ使用できます。

`-o` {- | *clconfiguration*}
`--output` {- | *clconfiguration*}

リソース構成情報の書き込み先を指定します。この書き込み先は、ファイルでも、標準出力でもかまいません。標準出力を指定するには、ファイル名の代わりに - を指定します。標準出力を指定すると、該当コマンドにおける他のすべての標準出力は抑制されます。このオプションは、export サブコマンドの場合にだけ指定できます。

構成情報は、コマンドに対するオペランドとして指定したリソースの場合だけ書き込まれます。この情報は、1447 ページのclconfiguration(5CL) のマニュアルページに定義されている形式で書き込まれます。

-p *name=value*
-p *name+=array-values*
-p *name-=array-values*
--property *name=value*
--property *name+=array-values*
--property *name-=array-values*

リソースの標準プロパティーと拡張プロパティーを設定します。このオプションは、create サブコマンドおよび set サブコマンドの場合にだけ指定できます。

標準プロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

リソースタイプの拡張プロパティーについては、リソースタイプに関するドキュメントを参照してください。

このオプションとともに使用する演算子は、次のとおりです。

=	プロパティーに、指定した値を設定します。この演算子は、create サブコマンドおよび set サブコマンドで使用できます。
+=	1 つまたは複数の値を文字列配列値に追加します。この演算子は、set サブコマンドでのみ使用できます。この演算子は、文字列配列値に対してだけ指定できます。
-=	1 つまたは複数の値が、文字列配列値から削除されます。この演算子は、set サブコマンドでのみ使用できます。この演算子は、文字列配列値に対してだけ指定できます。

ノード単位のプロパティーがクラスタノードのサブセット上でのみ設定される場合は、中括弧内のノードのリストを次のようにプロパティー名に付け加えることで、プロパティーが設定されるノードを指定します。

name{*nodelist*}

nodelist は、ノード名またはノード ID をコンマで区切ったリストです。ノード単位のプロパティーの詳細は、1335 ページのrt_properties(5) のマニュアルページを参照してください。

-p *name*[,…]
--property *name*[,…]

list-props サブコマンドおよび show サブコマンドのプロパティーのリストを指定します。

このオプションは、リソースの標準プロパティーおよび拡張プロパティーに対して使用できます。

標準プロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

リソースタイプの拡張プロパティーについては、リソースタイプに関するドキュメントを参照してください。

このオプションを指定しなかった場合、`list-props` サブコマンドおよび `show` サブコマンドは、`-v` オプションも指定されているかどうかに基づいて、すべてまたはほとんどのリソースプロパティーを一覧表示します。

`-R`
`--recursive`

必要な依存性がすべて満たされるように、リソースの有効化または無効化を再帰的に実行します。このオプションは、`disable` サブコマンドおよび `enable` サブコマンドの場合にだけ指定できます。

このオプションをこれらのサブコマンドとともに指定した場合の効果は、次のとおりです。

`disable`	コマンドに対するオペランドとして指定したリソースに依存しているリソース (コマンドに対するオペランドとして指定しなかったリソースも含まれる) をすべて無効にします。
`enable`	コマンドに対するオペランドとして指定したリソースの依存先リソース (コマンドに対するオペランドとして指定しなかったリソースも含まれる) がすべて有効になります。

`-s` *state*[,…]
`--state` *state*[,…]

`list` サブコマンドおよび `status` サブコマンドの状態のリストを指定します。

このオプションは出力を制限し、ノードリスト内の 1 つまたは複数のノード上で指定されている状態の 1 つにあるリソースだけが含まれるようにします。

可能な状態は、次のとおりです。

- `degraded`
- `detached`
- `faulted`
- `monitor_failed`
- `not_online` - `online` または `online_not_monitored` 以外のすべてのステータスを指定します
- `offline`
- `online`
- `online_not_monitored`
- `start_failed`
- `stop_failed`
- `unknown`

- `unmonitored`
- `wait`

`-V`
`--version`

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンドやオペランドなどのオプションは無視されます。`-V` オプションは、コマンドのバージョンを表示するだけです。その他の処理は行いません。

`-v`
`--verbose`

詳細なメッセージを標準出力に表示します。

このオプションは、このコマンドの任意の形式とともに指定できます。

`-v` オプションと `-o -` オプションを同時に指定してはいけません。`-v` オプ ションは無視されます。`-o -` オプションは、ほかのすべての標準出力を抑制します。

`-X` *node*[,…]
`--auxnode` *node*[,…]

`AuxNodeList SharedAddress` リソースプロパティーを設定します。

`AuxNodeList` リスト内のノードは、共有アドレスリソースに関連付けられている論理ホストのセットをホストできます。ただし、フェイルオーバー時にプライマリノードの役割を果たすことはできません。

`-Z` {*zoneclustername* | global | all}
`--zoneclustername=`{*zoneclustername* | global | all}
`--zoneclustername` {*zoneclustername* | global | all}

クラスタ、またはリソースが存在するクラスタや処理するクラスタを指定します。

このオプションは、`export` サブコマンド以外のすべてのサブコマンドでサポートされています。

このオプションを指定する場合は、次のいずれかの引数も指定する必要があります。

zoneclustername	このオプションを使用するコマンドが、*zoneclustername* という名前のゾーンクラスタでのみ指定されたすべてのリソースで機能するように指定します。
`global`	このオプションを使用するコマンドが、広域クラスタでのみ指定されたすべてのリソースで機能するように指定します。
`all`	広域クラスタでこの引数を使用する場合、それを使用するコマンドが広域クラスタとすべてのゾーンクラスタを含め、すべてのクラスタで指定されたすべてのリソースで機能するように指定します。

この引数をゾーンクラスタで使用すると、このオプションを使用するコマンドが、ゾーンクラスタ内だけで指定されたすべてのリソースに対して機能するように指定されます。

次のオペランドがサポートされています。

resource Oracle Solaris Cluster のリソース名をオペランドとして受け付けるように指定します。サブコマンドで複数のリソースを指定できる場合は、プラス記号 (+) を使用すると、すべてのリソースを指定できます。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

9 CL_ESTATE

オブジェクトの状態が不正です

その時点で変更できない、または常時変更できないプロパティー、リソースグループ、またはその他のオブジェクトを変更しようとしました。

10 CL_EMETHOD

リソースのメソッドが失敗しました

リソースのメソッドが失敗しました。次のいずれかの理由のために、メソッドは失敗しました。

- validate メソッドは、リソースを作成しようとしたときに、あるいは、リソースのプロパティーを変更しようとしたときに失敗しました。
- validate 以外のメソッドは、リソースを有効、無効、または削除しようとしたときに失敗しました。

15 CL_EPROP

無効なプロパティーです

-p、-y、または -x オプションで指定したプロパティーまたは値が存在しないか、許可されていません。

35 CL_EIO

I/O エラー

物理的な入出力エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

これらの終了値は、1047 ページのscha_calls(3HA) のマニュアルページで説明されているリターンコードと互換性があります。

例 207 共有アドレスリソースの作成

このコマンドは、sharedhost1 という名前のリソースを rg-failover という名前のリソースグループ内に作成します。リソースは有効な状態で作成され、モニタリングも有効になっています。

```
# clressharedaddress create -g rg-failover sharedhost1
```

次の 2 つのコマンドはどちらも、ゾーンクラスタ ZC 内に sharedhost1 という名前のリソースを作成します。これらのコマンドは、グローバルクラスタノードで、またはゾーンクラスタ ZC の内部で実行できます。

```
# clressharedaddress create -g rg-failover -Z ZC sharedhost1

# clressharedaddress create -g rg-failover ZC:sharedhost1
```

例 208 異なる論理ホスト名を持つ共有アドレスリソースの作成

このコマンドは、rs-sharedhost1 という名前のリソースを rg-failover という名前のリソースグループ内に作成します。

論理ホスト名はリソース名と同じではありませんが、論理ホストの名前と IP アドレスは同じままです。

```
# clressharedaddress create -g rg-failover \
-h sharedhost1 rs-sharedhost1
```

例 209 共有アドレスリソースの IPMP グループの指定

このコマンドは、sharedhost1 リソースの IPMP グループを設定します。

```
# clressharedaddress create -g rg-failover \
-N ipmp0@black,ipmp0@white sharedhost1
```

例 210 共有アドレスリソースの削除

このコマンドは、sharedhost1 という名前のリソースを削除します。

```
# clressharedaddress delete sharedhost1
```

例 211 共有アドレスリソースの一覧表示

このコマンドは、すべての共有アドレスリソースを一覧表示します。

```
# clressharedaddress list
sharedhost1
sharedhost2
```

例 212 共有アドレスリソース、リソースグループ、およびリソースタイプの一覧表示

このコマンドは、すべての共有アドレスリソースをそれらのリソースグループおよびリソースタイプと合わせて一覧表示します。

```
# clressharedaddress list -v
Resources   Resource Groups Resource Types
---------   -------------- --------------
sharedhost1 rg-failover-1   SUNW.SharedAddress
sharedhost2 rg-failover-2   SUNW.SharedAddress
```

例 213 共有アドレスリソースの拡張プロパティーの一覧表示

このコマンドは、すべての共有アドレスリソースの拡張プロパティーを一覧表示します。

```
# clressharedaddress list-props -v
Properties       Descriptions
----------       ------------
NetIfList        List of IPMP groups on each node
AuxNodeList      List of nodes on which this resource is available
HostnameList     List of hostnames this resource manages
CheckNameService Name service check flag
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページの`Intro`(1CL), 551 ページの`cluster`(1CL), 273 ページの`clresource`(1CL), 251 ページの`clreslogicalhostname`(1CL), 305 ページの`clresourcegroup`(1CL), 333 ページの`clresourcetype`(1CL), 1047 ページの`scha_calls`(3HA), 1447 ページの`clconfiguration`(5CL), Unresolved link to " rbac5", 1287 ページの`r_properties`(5)

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のオプションを指定してこのコマンドを実行できます。

- `-?` オプション
- `-V` オプション

サブコマンドを指定してこのコマンドを実行する場合、スーパーユーザー以外のユーザーは RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
create	solaris.cluster.modify
delete	solaris.cluster.modify
disable	solaris.cluster.admin
enable	solaris.cluster.admin
export	solaris.cluster.read
list	solaris.cluster.read
list-props	solaris.cluster.read
monitor	solaris.cluster.admin
reset	solaris.cluster.admin
set	solaris.cluster.modify
show	solaris.cluster.read
status	solaris.cluster.read
unmonitor	solaris.cluster.admin

名前

clresourcetype, clrt — Oracle Solaris Cluster データサービスのリソースタイプの管理

```
/usr/cluster/bin/clresourcetype [subcommand -?]

/usr/cluster/bin/clresourcetype subcommand -v [options]
     [resourcetype]...

/usr/cluster/bin/clresourcetype add-node  -n node[,…]
     [-Z {zoneclustername | global}] {+ | resourcetype}

/usr/cluster/bin/clresourcetype export [-o {- | configfile}]
     [+ | resourcetype]

/usr/cluster/bin/clresourcetype list [ -n node[,…]]
     [-Z {zoneclustername[,...] | global | all}] [+ | resourcetype...]

/usr/cluster/bin/clresourcetype list-props [-p [name,...]] [-Z
     {zoneclustername[,...] | global | all}] [+ | resourcetype...]

/usr/cluster/bin/clresourcetype register [-i
     {- | clconfiguration}] [ {[-n node
     [,…]] | -N}] [-f rtrfile] [-p [name [+ | -]=value,]...]
     [-Z {zoneclustername | global}] {+ | resourcetype...}

/usr/cluster/bin/clresourcetype remove-node -n node
     [,…] [-Z {zoneclustername | global}]
     {+ | resourcetype...}

/usr/cluster/bin/clresourcetype set [-n node
     [,…] | -N] [-p [name [+| -]=value,...]  [-Z
     {zoneclustername | global}]{+ | resourcetype...}

/usr/cluster/bin/clresourcetype show [-n node[,…]]
     [-Z {zoneclustername[,...] | global | all}] [+ | resourcetype...]

/usr/cluster/bin/clresourcetype unregister [-Z {zoneclustername |
     global}] {+ | resourcetype...}
```

clresourcetype コマンドは、Oracle Solaris Cluster データサービスのリソースタイプを管理します。clrt コマンドは clresourcetype コマンドの短い形式です。clresourcetype コマンドと clrt コマンドは同じものです。どちらの形式のコマンドも使用できます。

管理しやすいように、グローバルクラスタノードからこのコマンドを使用してください。

clresourcetype コマンドは、ゾーンクラスタで、export を除くすべてのサブコマンドを指定して使用することができます。

export 以外のすべてのサブコマンドで -Z オプションを使用して、操作を制限する特定のゾーンクラスタの名前を指定することもできます。また、リソースタイプ名 (*zoneclustername* : *resourcetype*) にゾーンクラスタ名を付けて、操作を特定のゾーンクラスタに制限することもできます。

グローバルクラスタノードからすべてのゾーンクラスタ情報にアクセスできますが、特定のゾーンクラスタは他のゾーンクラスタを認識しません。特定のゾーンクラスタに操作を制限できない場合、使用するサブコマンドは現在のクラスタでのみ機能します。

このコマンドの一般的な形式は次のとおりです。

clresourcetype [*subcommand*] [*options*] [*operands*]

options に -? または -V オプションを指定する場合だけは、*subcommand* を省略できます。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式は、このマニュアルページの「オプション」セクションのオプションの説明で紹介されています。

サポートされるサブコマンドには次のものがあります。

add-node

コマンドのオペランドとして指定されているリソースタイプのノードリストに指定されたノードを追加します。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタノードから add-node コマンドを使用している間、-Z オプションを使用してゾーンクラスタの名前を指定できます。

このサブコマンドは、すべてのリソースタイプを指定するオペランドとして正符号 (+) を受け付けます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。

remove-node サブコマンドの説明も参照してください。

export

1447 ページのclconfiguration(5CL) のマニュアルページで規定されている形式で、クラスタリソースタイプ構成をエクスポートします。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.read RBAC の承認が必要です。

このサブコマンドは、大域ゾーンだけで使用できます。

list

コマンドのオペランドとして指定されているリソースタイプのリストを表示します。デフォルトでは、クラスタ内に登録されているすべてのリソースタイプが表示されます。このサブコマン

ドは、クラスタ内に登録されているすべてのリソースタイプを指定するオペランドとしてプラス記号 (+) を受け付けます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタノードで、このサブコマンドはグローバルクラスタノードで登録されたソースのみを表示します。グローバルクラスタからゾーンクラスタに登録されたリソースタイプを表示するために、-Z オプションを使用してゾーンクラスタを指定できます。

-n *nodelist* オプションを指定すると、*nodelist* 内のノードで使用するために登録されているリソースタイプだけが表示されます。

-v オプションを指定すると、リスト内の各リソースタイプのノードリストも表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。

`list-props`

指定されたリソースタイプのプロパティーを表示します。このサブコマンドは、クラスタ内に登録されているすべてのリソースタイプを指定するオペランドとしてプラス記号 (+) を受け付けます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタで登録されたリソースタイプのプロパティーを表示するには、-Z オプションを使用してゾーンクラスタを指定できます。

-p オプションは、表示されるプロパティーのセットを限定します。

-v オプションを指定すると、各プロパティーの説明も表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。

`register`

コマンドのオペランドとして指定されているリソースタイプを登録します。リソースタイプは、そのタイプのリソースを作成する前に登録してください。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタでリソースタイプを登録するには、-Z オプションを使用してゾーンクラスタ名を指定します。

各リソースタイプを定義するデータサービスは、リソースタイプが使用可能になる各ノードにインストールしてください。データサービスがクラスタノードのサブセットでのみインストールされる場合は、-n *nodelist* オプションを使用してノードのサブセットを指定します。リソースタイプがクラスタ内のすべてのノードで使用可能な場合は、-N オプションを指定します。-N オプションを使用する場合は、将来クラスタに追加される可能性があるどのノードでもリソースタイプを使用できます。-N オプションと -n *nodelist* オプションの両方を省略することは、-N オプションを指定することと同じです。プロパティー名を明示的に指定するには、-p `Installed_nodes=`*nodelist* オプションを使用します。

クラスタに登録されているリソースタイプについての情報は、リソースタイプを定義するリソースタイプ登録 (RTR) ファイルから取得されます。一般的に、RTR ファイルの場所と名前は次に示す規約に従います。

- RTR ファイルは、通常 `/opt/cluster/lib/rgm/rtreg` ディレクトリ内にあります。
- RTR ファイルの名前は、通常リソースタイプの名前に一致します。

Oracle が提供するすべての RTR ファイルの場所とファイル名は、次に示す規約に従っています。たとえば、`SUNW.nfs` リソースタイプを定義する RTR ファイルは `/opt/cluster/lib/rgm/rtreg/SUNW.nfs` ファイルに含まれています。

RTR ファイルがこれらの規約に従わない場合は、`-f` *rtrfile* オプションを指定してください。

これらの規則は、ゾーンクラスタから登録されたリソースタイプに対しても適用されます。ゾーンクラスタに対してリソースを登録するとき、RTR ファイルをゾーンクラスタ *zonepath* 内に常駐させる必要があります。ゾーンクラスタ *zonepath* 境界の外部で RTR ファイルを登録できません。*Global_zone* プロパティーのあるソースタイプを登録している間ゾーンクラスタに対して `TRUE` に設定するには、RTR ファイルを `/opt/cluster/lib/rgm/rtreg` または `/usr/cluster/lib/rgm/rtreg` ディレクトリのグローバルクラスタノード内部に常駐させる必要があります。これらの場所の外部の任意の場所を指定すると、リソースタイプを登録できません。

注意 - 信頼できる既知のソースであるリソースタイプを除いて、`Global_zone` プロパティーに `TRUE` が設定されているリソースタイプは登録しないでください。このプロパティーに `TRUE` を設定したリソースタイプは、ゾーン分離をすり抜け、脅威をもたらします。

このサブコマンドは、まだ登録されていないすべてのリソースタイプを指定するオペランドとしてプラス記号 (+) を受け付けます。使用可能なリソースタイプのリスト全体は、次のように決定されます。

- `-i` *clconfiguration* オプションを指定すると、*clconfiguration* は使用可能なリソースタイプの完全なリストを定義します。
- `-i` オプションを指定しない場合、使用可能なリソースタイプの完全なリストには Oracle が提供するリソースタイプのみが含まれます。これらのリソースタイプもまた、ノードリスト内のすべてのノードにインストールする必要があります。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。

`unregister` サブコマンドの説明も参照してください。

`remove-node`

オペランドリスト内のリソースタイプが登録されるノードのリストからノードを削除します。このサブコマンドは、クラスタ内に登録されているすべてのリソースタイプを指定するオペランドとしてプラス記号 (+) を受け付けます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタのあるリソースタイプを削除するには、-Z オプションを使用してゾーンクラスタの名前を指定します。

このサブコマンドは、クラスタ内のすべてのノードではなく一部のノードに対してすでに登録されているリソースタイプでのみ使用できます。結果として、次に示す状況でこのサブコマンドを使用するとエラーが発生します。

- オペランドのリスト内にあるリソースタイプがクラスタ内のすべてのノードに対してすでに登録されている。クラスタ内のすべてのノードのリソースタイプの登録については、-N オプションの説明を参照してください。
- オペランドのリスト内のリソースタイプの Installed_nodes プロパティーが、クラスタ内のノードのサブセットをまだ指定していない。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。

add-node サブコマンドの説明も参照してください。

set

コマドのオペランドとして指定されているリソースタイプのプロパティーを設定します。このサブコマンドは、クラスタ内に登録されているすべてのリソースタイプを指定するオペランドとしてプラス記号 (+) を受け付けます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

広域クラスタからゾーンクラスタにリソースタイプのプロパティーを設定するには、-Z オプションを使用してゾーンクラスタの名前を指定します。

1335 ページのrt_properties(5) のマニュアルページで Tunable Any Time として指定されているリソースタイププロパティーのみを設定できます。

- Installed_Nodes のプロパティーは、-p オプションを指定せずに、-n *nodelist* オプションを指定することによって変更できます。または、-p Installed_Nodes= *nodelist* オプションを指定して、プロパティ名を明示的に指定できます。
- Tunable Any Time として指定されている他のすべてのプロパティーに関しては、-p *property* = *value* オプションを使用して、プロパティー名を明示的に指定します。

リソースタイプが使用できるノードのリストを制限するには、-n *nodelist* オプションを指定します。逆に、リソースタイプがクラスタ内のすべてのノードで使用可能であることを指定するには、-N オプションを指定します。-N オプションを使用する場合は、将来クラスタに追加される可能性があるどのノードでもリソースタイプを使用できます。-n オプションまたは -N オプションを指定してください。どちらのオプションも省略すると、このサブコマンドは構成情報を一切変更しません。

show

クラスタ内に登録されているリソースタイプについての情報を表示します。デフォルトでは、登録されているすべてのリソースタイプに対して次の情報が表示されます。

- 各リソースタイプに関連付けられているプロパティーのリスト
- これらのプロパティーを定義するパラメータ

-n *nodelist* オプションを指定すると、*nodelist* 内のノードで使用するために登録されているリソースタイプだけが表示されます。

-v オプションを指定すると、次の情報もリソースタイプごとに表示されます。

- リソースタイプに対して定義されているメソッド
- 各メソッドのタイムアウトパラメータ

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

グローバルクラスタからゾーンクラスタに登録されたリソースタイプを表示するために、-z オプションを使用してゾーンクラスタの名前を指定できます。

このサブコマンドは、クラスタ内に登録されているすべてのリソースタイプを指定するオペランドとしてプラス記号 (+) を受け付けます。オペランドが指定されていない場合、クラスタに登録されているすべてのリソースタイプについての情報が表示されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` RBAC の承認が必要です。

`unregister`

コマンドのオペランドとして指定されているリソースタイプの登録を解除します。このサブコマンドは、そのタイプのインスタンスが存在しないすべての登録リソースタイプを指定するオペランドとしてプラス記号 (+) を受け付けます。

このサブコマンドは、グローバルクラスタまたはゾーンクラスタで使用できます。

option.グローバルクラスタからゾーンクラスタのあるリソースタイプを登録解除するには、-z オプションを使用してゾーンクラスタの名前を指定します。

リソースタイプを定義するデータサービスをアンインストールする前にリソースタイプの登録を解除してください。

特定のリソースタイプのリソースが存在する場合、そのリソースタイプの登録は解除できません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.admin` RBAC の承認が必要です。

`register` サブコマンドの説明も参照してください。

次のオプションがサポートされています。

`-?`
`--help`

ヘルプ情報を表示します。このオプションを使用する場合、ほかの処理は実行されません。

このオプションを指定するとき、サブコマンドは指定してもしなくてもかまいません。

サブコマンドなしでこのオプションを指定すると、このコマンドのサブコマンドのリストが表示されます。

サブコマンド付きでこのオプションを指定すると、サブコマンドの使用方法が表示されます。

`-f` *rtrfile*|*rtrfiledir*
`--rtrfile=`*rtrfile*|*rtrfiledir*
`--rtrfile` *rtrfile*|*rtrfiledir*

リソースタイプの登録で使用するために、RTR ファイルのフルパスまたは RTR ファイルを含むディレクトリを指定します。このオプションは `register` サブコマンドとだけ指定できます。

ファイルを指定すると、1 つのリソースタイプだけを登録できます。

このオプションを指定する必要があるのは、使用している RTR ファイルが次に示す規約に従っていない場合だけです。

- RTR ファイルは、通常 `/opt/cluster/lib/rgm/rtreg` ディレクトリ内にあります。
- RTR ファイルの名前は、通常リソースタイプの名前に一致します。

Oracle が提供するすべての RTR ファイルの場所とファイル名は、次に示す規約に従っています。たとえば、`SUNW.nfs` リソースタイプを定義する RTR ファイルは `/opt/cluster/lib/rgm/rtreg/SUNW.nfs` ファイルに含まれています。

`-i` オプションを指定すると、構成情報で指定されているどのリソースタイプに対する構成情報でも `resourcetypeRTRFile` 要素を指定できます。`resourcetypeRTRFile` 要素は、リソースタイプの登録に使用される RTR ファイルを指定します。ただし、`export` サブコマンドは、生成される構成情報に `resourcetypeRTRFile` 要素を含みません。`resourcetypeRTRFile` 要素の詳細は、1447 ページの`clconfiguration`(5CL) のマニュアルページを参照してください。

`-i` {- | *clconfiguration*}
`--input=`{- | *clconfiguration*}
`--input` {- | *clconfiguration*}

リソースタイプの登録または登録されているリソースタイプのノードリストの変更に使用される構成情報を指定します。この情報は、1447 ページの`clconfiguration`(5CL) のマニュアルページに定義されている形式に準拠している必要があります。この情報はファイルに含まれている場合と、標準入力 `stdin` を通して指定される場合があります。標準入力を指定するには、ファイル名の代わりに `-` を指定します。

コマンドのオペランドとして指定されているリソースタイプだけが、このオプションによって影響されます。コマンドで指定されているオプションは、`clconfiguration` ファイル内に設定されているあらゆるオプションをオーバーライドします。構成パラメータが `clconfiguration` ファイル内にない場合は、不足しているパラメータをコマンド行で指定してください。

-N
--allnodes

オペランドのリスト内のリソースタイプがクラスタ内のすべてのノードで使用可能であることを指定します。-N オプションは、将来クラスタに追加される可能性がある任意のノードでもこれらのリソースタイプを使用可能にします。オプションは、Installed_nodes プロパティーをクリアすることによって、この結果を取得します。

-N オプションを指定すると、同じコマンドで -n オプションは指定できません。

-N オプションは、register サブコマンドまたは set サブコマンドとだけ指定できます。

-n *node*[,…]
--node=*node*[,…]
--node *node*[,…]

ターゲットのグローバルクラスタまたはゾーンクラスタに、ノードまたはノードのリストを指定します。各ノードはノード名またはノード ID で指定できます。

-Z オプションが指定されている場合は、-n オプションで、グローバルクラスタホスト名ではなくゾーンクラスタホスト名のみを指定できます。-Z オプションが指定されていない場合は、-n オプションでグローバルクラスタホスト名のみを指定できます。

-N オプションを指定すると、同じコマンドで -n オプションは指定できません。

このオプションとともに指定できるサブコマンドは、次のとおりです。

add-node

指定されたノードをリソースタイプが登録されているノードのリストに追加します。

list

指定されたノード上で使用するために登録されているリソースタイプだけを表示します。

register

指定されたノード上で使用する際にのみリソースタイプを登録します。-n オプションを省略すると、register サブコマンドはすべてのノード上で使用されるリソースタイプを登録します。このサブコマンドは、将来クラスタに追加される任意のノードのリソースタイプも登録します。

remove-node

指定されたノードをリソースタイプが登録されているノードのリストから削除します。

set

指定されたノードでのみリソースタイプを使用可能にします。

show

指定されたノード上で使用するために登録されているリソースタイプについての情報だけを表示します。

-o {- | *clconfiguration*}
--output={- | *clconfiguration*}
--output {- | *clconfiguration*}

リソースタイプについての構成情報が書き込まれる場所を指定します。この場所はファイルの場合と標準出力 stdout の場合があります。標準出力を指定するには、ファイル名の代わりに - を指定します。標準出力を指定すると、コマンドに対するほかのすべての標準出力が抑制されます。このオプションは、export サブコマンドの場合にだけ指定できます。

構成情報は、コマンドに対するオペランドとして指定されるリソースタイプに対してのみ記述されます。この情報は、1447 ページのclconfiguration(5CL) のマニュアルページに定義されている形式で書き込まれます。

-p *name=value*
-p *name+=array-values*
-p *name-=array-values*
--property=*name=value*
--property=*name+=array-values*
--property=*name-=array-values*
--property *name=value*
--property *name+=array-values*
--property *name-=array-values*

コマンドのオペランドとして指定されるリソースタイプのプロパティーの値を設定します。

このオプションとともに使用する演算子は、次のとおりです。

=	プロパティーに、指定した値を設定します。
+=	1 つまたは複数の値を文字列配列値に追加します。この演算子は、たとえば、Installed_nodes のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。
-=	1 つまたは複数の値が、文字列配列値から削除されます。この演算子は、たとえば、Installed_nodes のような文字列値のリストを受け付けるプロパティーに対してのみ指定できます。

オプション -p Installed_nodes+=nodeC,nodeD を set サブコマンドで使用することは、オプション -n nodeC,nodeD を add-node サブコマンドで使用することと同じです。

-p *name*[,…]
--property=*name*[,…]
--property *name*[,…]

list-props サブコマンドのプロパティーのリストを指定します。

-V
--version

コマンドのバージョンを表示します。

このオプションには、サブコマンドやオペランドなどのオプションは指定しないでください。サブコマンドやオペランドなどのオプションは無視されます。-V オプションは、コマンドのバージョンだけを表示します。その他の処理は行いません。

-v
--verbose

詳細メッセージを標準出力 (stdout) に表示します。

このオプションは、このコマンドの任意の形式とともに指定できます。

-v オプションと -o - オプションを同時に指定してはいけません。-v オプ ションは無視されます。-o - オプションは、ほかのすべての標準出力を抑制します。

-Z {*zoneclustername* | global | all}
--zoneclustername={*zoneclustername* | global | all}
--zoneclustername {*zoneclustername* | global | all}

リソースタイプが登録されている場合、操作する必要がある 1 つまたは複数のクラスタを指定します。

このオプションは、export サブコマンド以外のすべてのサブコマンドでサポートされています。

このオプションを指定する場合は、次のいずれかの引数も指定する必要があります。

zoneclustername
: このオプションを使用するコマンドが、*zoneclustername* という名前のゾーンクラスタでのみ指定されたすべてのリソースタイプで機能するように指定します。

global
: このオプションを使用するコマンドが、グローバルクラスタでのみ指定されたすべてのリソースタイプで機能するように指定します。

all
: グローバルクラスタでこの引数を使用する場合、それを使用するコマンドがグローバルクラスタとすべてのゾーンクラスタを含め、すべてのクラスタで指定されたすべてのリソースタイプで機能するように指定します。

  ゾーンクラスタでこの引数を使用する場合、この引数を使用するコマンドがそのゾーンクラスタでのみ指定されたすべてのリソースタイプで機能するように指定します。

次のオペランドだけがサポートされています。

resourcetype
: 管理対象となる 1 つまたは複数のリソースタイプを指定します。サブコマンドが複数のリソースタイプを受け入れる場合は、プラス記号 (+) を使用してすべてのリソースタイプを指定できます。

  リソースタイプ名の形式については、Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド の有効な RGM 名"を参照してください。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

次の終了コードが返されます。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

18 CL_EINTERNAL

内部エラーが発生しました

内部エラーは、ソフトウェアの欠陥またはその他の欠陥を示しています。

35 CL_EIO

I/O エラー

物理的な入出力エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。

- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

37 CL_EOP
: 操作が許可されていません
  サポートされていない構成に対する操作を実行しようとしたか、サポートされていない操作を実行しました。

41 CL_ETYPE
: 無効なタイプです
  -t または -p オプションで指定したタイプは存在しません。

これらの終了値は、1047 ページのscha_calls(3HA) のマニュアルページで説明されているリターンコードと互換性があります。

例 214 リソースタイプの登録

この例では、データサービスがすべてのノードにインストールされ、まだ登録されていないすべてのリソースタイプを登録する方法を示します。このコマンドは簡略モードで実行されます。

```
# clresourcetype register +
```

例 215 選択されたノード上でのリソースタイプの登録

この例では、データサービスがノード phys-schost-1 およびノード phys-schost-2 にインストールされており、かつまだ登録されていないすべてのリソースタイプを登録します。リソースはこれらのノードでのみ使用可能になります。この例では、コマンドがエラーを返すことはありません。このコマンドは詳細モードで実行されます。

```
# clresourcetype register -v -n phys-schost-1,phys-schost-2 +
```

次のコマンドは、そのデータサービスがゾーンクラスタ ZC のゾーンクラスタのゾーンクラスタノード zc-host-1 と zc-host-2 にインストールされたすべてのリソースタイプおよび登録されていないリソースタイプを登録します。リソースは、これらのゾーンクラスタノードでのみ使用可能です。

```
#.clresourcetype register -n zc-host-1,zc-host-2 -Z ZC +
```

例 216 単一のリソースタイプの登録

この例では、SUNW.nfs:3.2 リソースタイプを登録する方法を示します。このリソースタイプのデータサービスは、すべてのクラスタノードにインストールされます。

```
# clresourcetype register nfs:3.2
```

例 217 リソースタイプの一覧表示

この例では、登録されているすべてのリソースタイプの名前だけを一覧表示する方法を示します。

```
# clresourcetype list
SUNW.LogicalHostname
SUNW.SharedAddress
SUNW.nfs
SUNW.apache
```

例 218 リソースタイプとリソースタイプのノードリストの一覧表示

この例では、登録されているすべてのリソースタイプをそれらのノードリストとともに一覧表示する方法を示します。

```
# clresourcetype list -v

Resource Type         Node List
-------------         ---------
SUNW.LogicalHostname  <all>
SUNW.SharedAddress    <all>
SUNW.nfs              phys-schost-1,phys-schost-2,phys-schost-3
SUNW.apache           phys-schost-1,phys-schost-2,phys-schost-3
```

グローバルクラスタノードから次のコマンドを実行するとき、コマンドはゾーンクラスタ ZC に登録されたすべてのリソースタイプを一覧します。

```
#.clresourcetype list -Z ZC
SUNW.nfs
SUNW.apache
```

例 219 指定されたノード上でのリソースタイプの一覧表示

この例では、 phys-schost-4 上で登録されているすべてのリソースタイプを一覧表示する方法を示します。

```
# clrt list -n phys-schost-4
SUNW.LogicalHostname
SUNW.SharedAddress
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core

属性タイプ	属性値
インタフェースの安定性	発展中

19 ページの Intro(1CL), 251 ページの clreslogicalhostname(1CL), 273 ページの clresource(1CL), 305 ページの clresourcegroup(1CL), 347 ページの clressharedaddress(1CL), 551 ページの cluster(1CL), 1047 ページの scha_calls(3HA), 1447 ページの clconfiguration(5CL), 1287 ページの r_properties(5), Unresolved link to " attributes5", Unresolved link to " rbac5"

Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド のリソースグループプロパティー"

スーパーユーザーはこのコマンドのすべての形式を実行できます。

任意のユーザーは次のオプションを指定してこのコマンドを実行できます。

- -? オプション
- -V オプション

サブコマンドを指定してこのコマンドを実行する場合、スーパーユーザー以外のユーザーは RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
add-node	solaris.cluster.modify
export	solaris.cluster.read
list	solaris.cluster.read
list-props	solaris.cluster.read
set	solaris.cluster.modify
register	solaris.cluster.modify
remove-node	solaris.cluster.modify
show	solaris.cluster.read
unregister	solaris.cluster.admin

名前

clsetup — Oracle Solaris Cluster の対話型構成

```
/usr/cluster/bin/clsetup -V

/usr/cluster/bin/clsetup -?

/usr/cluster/bin/clsetup [-f logfile]
```

clsetup コマンドは、実行するときのクラスタの状態によって、次に示す構成機能を提供します。このコマンドを実行するには、スーパーユーザーである必要があります。

このコマンドに短形式はありません。

- clsetup コマンドをインストール後の作業時に実行した場合、このコマンドは、定足数デバイスの構成や installmode プロパティーのリセットなどの初期設定タスクを実行します。scinstall または cluster create コマンドのどちらかを使用してクラスタを作成したときに自動定足数構成の選択を解除した場合は、クラスタがインストールされた直後に clsetup コマンドを実行する必要があります。clsetup コマンドを実行して installmode プロパティーをリセットする前に、すべてのノードがクラスタに参加していることを確認してください。

  クラスタを作成したときに自動定足数構成を使用した場合は、クラスタのインストール後に clsetup コマンドを実行する必要はありません。自動定足数構成機能は、クラスタの installmode プロパティーもリセットします。

  この形式の clsetup コマンドは、クラスタ内のどのノードからでも実行できます。
- 通常のクラスタ動作中に clsetup コマンドを実行すると、clsetup コマンドは、クラスタ構成タスクを実行するための対話形式でメニュー選択方式のユーティリティーを提供します。このユーティリティーが管理するクラスタコンポーネントの一部を次に示します。
  - 定足数
  - リソースグループ
  - データサービス
  - クラスタインターコネクト
  - デバイスグループとボリューム
  - プライベートホスト名
  - 新規ノード

- ゾーンクラスタ
- その他のクラスタプロパティー

この形式の clsetup コマンドは、クラスタ内のどのノードからでも実行できます。

- clsetup コマンドを非クラスタモードのノードから実行すると、clsetup コマンドは、プライベート IP アドレス範囲を変更および表示するメニュー形式のユーティリティーを提供します。

  この形式の clsetup ユーティリティーを開始する前に、すべてのノードを非クラスタモードにリブートしてください。

次のオプションがサポートされています。

-?
--help
: コマンドのヘルプ情報を出力します。

-f *logfile*
--file *logfile*
: コマンドログを記録するログファイル名を指定します。このオプションが指定されている場合は、ユーザーの応答に応じて、clsetup によって生成されたほとんどのコマンドセットの実行と記録、または記録のみを行うことができます。

-V
--version
: コマンドセットのバージョンを出力します。コマンド行処理はまったく実行されず、コマンドは対話形式のメニューに入りません。

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページのIntro(1CL), 79 ページのcldevicegroup(1CL), 185 ページのclnode(1CL), 239 ページのclquorum(1CL), 251 ページのclreslogicalhostname(1CL), 305 ページのclresourcegroup(1CL), 333 ページのclresourcetype(1CL), 347 ページのclressharedaddress(1CL), 551 ページのcluster(1CL), 535 ページのcltelemetryattribute(1CL), 613 ページのclzonecluster(1CL)

Unresolved link to " Oracle Solaris Cluster ソフトウェアのインストール ", Unresolved link to " Oracle Solaris Cluster システム管理 "

名前

clsnmphost — Oracle Solaris Cluster SNMP ホストのリストの管理

```
/usr/cluster/bin/clsnmphost -V

/usr/cluster/bin/clsnmphost [subcommand] -?

/usr/cluster/bin/clsnmphostsubcommand [[options]] -v [host]

/usr/cluster/bin/clsnmphost add [-c community[,…]]
     [-n node,…] host […]

/usr/cluster/bin/clsnmphost add -i {- | clconfigfile}
     [-ccommunity[,…]] [-n node[,…]] host […]

/usr/cluster/bin/clsnmphost export [-o {- | clconfigfile}]
     [-c community[,…]] [-n node[,…]] [+ | host…]

/usr/cluster/bin/clsnmphost list [-c community[,…]]
     [-n node[,…]] [+ | host…]

/usr/cluster/bin/clsnmphost remove [-c community[,…]]
     [-n node[,…]] {+ | host…}

/usr/cluster/bin/clsnmphost show [-c community[,…]]
     [-n node[,…]] [+ | host…]
```

`clsnmphost` コマンドは、SNMP (Simple Network Management Protocol) ホストおよび SNMP イベントの通知を受信するコミュニティー名を管理します。SNMP ホストはクラスタの管理情報ベース (MIB) を使用してアクセス制御メカニズムを提供します。MIB が SNMP トラップ通知を送信すると、このコマンドで構成された SNMP ホストはトラップ通知を送信するホストを特定できます。クラスタ MIB の詳細は、499 ページの`clsnmpmib`(1CL) のマニュアルページを参照してください。

このコマンドに短形式はありません。

このコマンドの一般的な形式は次のとおりです。

`clsnmphost` [*subcommand*] [*options*] [*operands*]

subcommand を省略できるのは、*options* がオプション `-?` または `-V` を指定している場合だけです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式は、このマニュアルページの「オプション」セクションのオプションの説明で紹介されています。

詳細は、19 ページのIntro(1CL) のマニュアルページを参照してください。

このコマンドは、大域ゾーンだけで使用できます。

サブコマンド

サポートされるサブコマンドには次のものがあります。

add

クラスタの MIB のトラップ通知を受信し、MIB のテーブルにアクセスできるホストのリストに SNMPホストを追加します。

このサブコマンドは、大域ゾーンだけで使用できます。

add サブコマンドを -n オプションなしで使用すると、現在のノードだけが影響を受けます。add を -c オプションなしで使用すると、このサブコマンドは public をデフォルトのコミュニティー名として使用します。IP アドレスまたはホスト名のいずれかを使用してホストを指定します。

指定されたコミュニティー名が存在しない場合、このコマンドはそのコミュニティーを作成します。-clconfigfile から 1 つ以上のホスト構成をインポートするには、*i* オプションを使用します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

export

指定されたノードの SNMP ホスト情報をエクスポートします。

このサブコマンドは、大域ゾーンだけで使用できます。

SNMP ホスト情報をエクスポートする 1 つ以上のノードを指定するには、-n オプションを使用します。export を -n オプションなしで使用すると、このサブコマンドは現在のノードの SNMP ホスト情報だけをエクスポートします。

export サブコマンドからの出力形式の詳細は、1447 ページのclconfiguration(5CL) のマニュアルページを参照してください。デフォルトでは、すべての出力が標準出力に送信されます。出力先をファイルに変更するには、-o オプションを使用して、そのあとにファイル名を指定します。

-c オプションを使用することで、export サブコマンドからの出力を特定のコミュニティー内のホストの情報だけに制限できます。これらのホストだけに出力情報を制限するには、1 つ以上のホストをオペランドとして指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

list

指定されたノード上で構成されている SNMP ホストを一覧表示します。

このサブコマンドは、大域ゾーンだけで使用できます。

`list` サブコマンドを `-n` オプションなしで使用すると、現在のノード上の SNMP ホストだけが一覧表示されます。デフォルトでは、このサブコマンドはノード上のすべてのホストを一覧表示します。特定のホストについての情報に出力を制限するには、1 つ以上のホストをオペランドとして指定します。また、`-c` オプションを使用して、指定されたコミュニティー内のホストだけを一覧表示することもできます。

`superuser` 以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`remove`

SNMP ホストをノード構成から削除します。

このサブコマンドは、大域ゾーンだけで使用できます。

構成からホストを削除するには、ホスト名をオペランドとして指定します。`remove` サブコマンドを `-n` オプションなしで使用すると、現在のノード上の SNMP ホストだけが削除されます。すべてのホストを削除するには、正符号 (+) を使用します。1 つ以上のホストを特定のコミュニティーから削除するには、`-c` オプションを使用します。

`superuser` 以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`show`

指定されたノード上の SNMP ホスト情報を表示します。

このサブコマンドは、大域ゾーンだけで使用できます。

`show` サブコマンドを `-n` オプションなしで使用すると、現在のノード上の SNMP ホストの情報だけが表示されます。デフォルトでは、`show` サブコマンドはすべてのホストおよびそのコミュニティーの情報を表示します。コミュニティー内の特定のホストの情報に出力を限定するには、`-c` オプションを使用するか、1 つ以上のホストの名前をオペランドとして指定します。

`superuser` 以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

次のオプションがサポートされています。

`-?`
`--help`

ヘルプ情報を出力します。

このオプションはサブコマンド付きでもサブコマンドなしでも指定できます。

- このオプションをサブコマンドなしで使用すると、使用可能なサブコマンドのリストが表示されます。

- このオプションをサブコマンドとともに使用すると、サブコマンドの使用オプションが表示されます。

このオプションを使用する場合、ほかの処理は実行されません。

`-c` *community*
`--community` *community*

ホスト名とともに使用される SNMP コミュニティー名を指定します。このオプションは、サブコマンド操作の範囲を絞り込むためにほかのサブコマンドとともに使用される場合もあります。たとえば、`remove` サブコマンドとともに使用すると、`-c` オプションは特定の *community* から 1 つまたは多数のホストを削除するために使用できます。`add` サブコマンドを `-c` オプションなしで使用すると、このサブコマンドは `public` をデフォルトのコミュニティー名として使用します。

`-i` {- | *clconfigfile*}
`--input` {- | *clconfigfile*}

SNMP ホスト構成を検証または変更するために使用できる構成情報を指定します。この情報は、1447 ページの`clconfiguration`(5CL) のマニュアルページに定義されている形式に準拠している必要があります。この情報は、ファイルに含めることも、標準入力を介して指定することもできます。標準入力を指定するには、ファイル名の代わりにマイナス記号 (`-`) を指定します。

`-n` *node*[,…]
`--node[s]` *node*[,…]

ノードまたはノードリストを指定します。各ノードは、ノード名またはノード ID として指定できます。`clsnmphost` コマンドのすべての形式で、このオプションが許可されます。

`-o` {- | *clconfigfile*}
`--output` {- | *clconfigfile*}

クラスタの SNMP ホスト構成情報を、1447 ページの`clconfiguration`(5CL) のマニュアルページで定義されている形式で書き込みます。この情報は、ファイルまたは標準出力のどちらにでも書き込むことができます。

標準出力に書き込むには、ファイル名の代わりにマイナス記号 (`-`) を指定します。標準出力を指定すると、該当コマンドにおける他のすべての標準出力は抑制されます。

ファイル名を指定すると、構成はその名前の新しいファイルにコピーされます。

`-o` オプションは、`export` サブコマンドとともに使用するときのみに有効になります。`-o` オプションを指定しない場合、出力は標準出力に出力されます。

`-V`
`--version`

コマンドのバージョンを出力します。

このオプションは、サブコマンド、オペランド、またはその他のオプションと一緒に指定しないでください。指定すると、一緒に指定されたサブコマンド、オペランド、またはその他のオ

プションは無視されます。-v オプションは、コマンドのバージョンだけを表示します。その他の処理は行いません。

-v
--verbose

詳細情報を標準出力に出力します。

このオプションはどの形式のコマンドとともに指定できますが、一部のサブコマンドは拡張出力を発生させないことがあります。たとえば、export サブコマンドは、詳細オプションを指定しても拡張出力を発生させません。

次のオペランドがサポートされています。

+	すべての SNMP ホストエントリを指定します。
host	クラスタ上の SNMP MIB へのアクセスを提供されているホストの IP アドレス、IPv6 アドレス、またはホスト名を指定します。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link

to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

18 CL_EINTERNAL
: 内部エラーが発生しました
: 内部エラーは、ソフトウェアの欠陥またはその他の欠陥を示しています。

35 CL_EIO
: I/O エラー
: 物理的な入出力エラーが発生しました。

36 CL_ENOENT
: そのようなオブジェクトはありません。
: 次のいずれかの理由のために、指定したオブジェクトを見つけることができません。
- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

例 220 ホスト名の指定によるホストの追加

次のコマンドは、ホスト myhost を現在のノード private のコミュニティーの SNMP ホストリストに追加します。

```
# clsnmphost add -c private phys-schost-1
```

ホストを public 以外のコミュニティーに追加する場合は、コミュニティー名を指定してください。

例 221 ホスト IP と IPv6 アドレスの指定によるホストの追加

次のコマンドは、現在のノード上にある public コミュニティーの SNMP ホストリストにホストを追加します。コマンドの 1 番目のバージョンは、ホストの IP アドレスを指定することによってホストを追加します。コマンドの 2 番目のバージョンは、ホストの IPv6 アドレスを指定することによってホストを追加します。

```
# clsnmphost add -c public 192.168.12.12
```
or
```
# clsnmphost add -c public fe:1::5
```

例 222 ホストの削除

次のコマンドは、private コミュニティーからすべてのホストを削除します。

```
# clsnmphost remove -c private +
```

例 223 現在のノード上のホストの一覧表示

次のコマンドは、現在のノード上にあるすべてのホストを一覧表示します。

```
# clsnmphost list
phys-schost-1
192.168.12.12
```

例 224 ホストとホストのコミュニティー名の一覧表示

次のコマンドは、詳細オプション -v を使用して、現在のノード上のすべてのホストとそれらのコミュニティー名を一覧表示します。

```
# clsnmphost list -v

--- SNMP hosts on node phys-schost-1 ---

Host Name           Community
---------           ---------
phys-schost-1       private
192.168.12.12       public
```

例 225 SNMP ホスト構成の表示

次のコマンドは、ノード phys-cluster-2 上の SNMP ホストのすべての構成情報を表示します。

```
# clsnmphost show -n phys-schost-2

--- SNMP Host Configuration on phys-schost-2 ---

SNMP Host Name:                                 phys-schost-2
  Community:                                       private
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

499 ページの`clsnmpmib`(1CL), 551 ページの`cluster`(1CL), 19 ページの`Intro`(1CL), 809 ページの`sceventmib`(1M), Unresolved link to " su1M", 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5", Unresolved link to " rbac5", 1447 ページの`clconfiguration`(5CL)

スーパーユーザーはこのコマンドのすべての形式を実行できます。

すべてのユーザーがこのコマンドに `-?` (ヘルプ) オプションまたは `-V` (バージョン情報) オプションを指定して実行できます。

`clsnmphost` コマンドをほかのサブコマンドとともに実行する場合、スーパーユーザー以外のユーザーは RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
`add`	`solaris.cluster.modify`
`export`	`solaris.cluster.read`
`list`	`solaris.cluster.read`
`remove`	`solaris.cluster.modify`
`show`	`solaris.cluster.read`

名前

clsnmpmib, clmib — Oracle Solaris Cluster SNMP MIB の管理

```
/usr/cluster/bin/clsnmpmib -V

/usr/cluster/bin/clsnmpmibsubcommand [] -?

/usr/cluster/bin/clsnmpmib [subcommand] [options] -v [mib]

/usr/cluster/bin/clsnmpmib disable [-n node[,…]] {+ | mib …}

/usr/cluster/bin/clsnmpmib enable [-n node[,…]] {+ | mib …}

/usr/cluster/bin/clsnmpmib export [-n node[,…]]
     [-o {- | clconfigfile}] [+ | mib …]

/usr/cluster/bin/clsnmpmib list [-n node[,…]] [+ | mib …]

/usr/cluster/bin/clsnmpmib set [-p name=value] […]
     [-n node[,…] {+ | mib …}

/usr/cluster/bin/clsnmpmib show [-n node[,…]] [+ | mib …]
```

`clsnmpmib` コマンドは、現在のノード上にある既存の Oracle Solaris Cluster SNMP (Simple Network Management Protocol) の管理情報ベース (MIB) を管理します。MIB を管理できる SNMP ホストを作成するには、491 ページの`clsnmphost`(1CL) のマニュアルページを参照してください。SNMPv3 プロトコルを使用しているときに MIB にアクセスできる SNMP バージョン 3 (SNMPv3) ユーザーを定義するには、509 ページの`clsnmpuser`(1CL) のマニュアルページを参照してください。

このコマンドの一般的な形式は次のとおりです。

`clsnmpmib` [*subcommand*] [*options*] [*operands*]

subcommand を省略できるのは、*options* で `-?` または `-V` オプションが指定されている場合のみです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式は、OPTIONS セクションにオプションの説明とともに記載されています。

詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

Oracle Solaris Cluster の MIB

Oracle Solaris Cluster は現在、1 つの MIB (イベント MIB) をサポートしています。Oracle Solaris Cluster SNMP の イベント MIB は、SNMP マネージャーにクラスタイベントをリアルタイムで通知します。有効になっている場合、Oracle Solaris Cluster のイベント MIB は、`clsnmphost` コマンドで定義されているすべてのホストにトラップ通知を自動的に送信します。Oracle Solaris Cluster イベント MIB は、ポート 11162 にトラップ通知を送信します。SNMP ツリーはポート 11161 に表示されます。

`clsnmpmib set` サブコマンドを使用して、`min_severity` または `log_number` の値を指定できます。クラスタは多数のイベント通知を生成するので、重要度が `min_severity` 以上のイベントのみがトラップ通知として送信されます。デフォルトでは、`min_severity` 値は `NOTICE` に設定されます。`log_number` の値には、古いエントリを破棄するまでに MIB テーブルに記録するイベントの数を指定します。MIB は、トラップが送信された最新のイベントの読み取り専用テーブルを維持しています。イベントの数は、`log_number` 値によって制限されます。この情報はリブートされると消滅します。

このコマンドは、大域ゾーンだけで使用できます。

サポートされるサブコマンドには次のものがあります。

`disable`

指定されたノード上の 1 つ以上のクラスタの MIB を無効にします。

このサブコマンドは、大域ゾーンだけで使用できます。

`-n` オプションを指定しないと、現在のノード上の MIB だけが無効にされます。MIB が無効になると、MIB のテーブルにアクセスできず、MIB はトラップ通知を一切送信しません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`enable`

指定されたノード上にある 1 つ以上のクラスタの MIB を有効にします。

このサブコマンドは、大域ゾーンだけで使用できます。

`-n` オプションを指定しないと、現在のノード上の MIB だけが有効にされます。有効にされる MIB を制限するには、*mib* オペランドを使用します。

MIB を有効にすると、その機能すべてが有効になります。ただし、すべての MIB 機能が完全に機能するには、さらにいくつかの構成が必要な場合があります。たとえば、ホストが構成されていないと、MIB はトラップ通知を送信できません。SNMP ホストの構成については、491 ページの`clsnmphost`(1CL) のマニュアルページを参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`export`

クラスタの MIB の構成情報をエクスポートします。

このサブコマンドは、大域ゾーンだけで使用できます。

MIB 構成情報をエクスポートする 1 つ以上のノードを指定するには、`-n` オプションを使用します。`export` を `-n` オプションなしで使用した場合、このサブコマンドは、現在のノードの MIB 構成情報のみをエクスポートします。デフォルトでは、このサブコマンドは現在のノードのすべての MIB から構成情報をエクスポートします。出力をさらに詳細化するには、構成情報が必要な 1 つ以上の MIB の名前を指定します。

`export` サブコマンドからの出力形式の詳細は、1447 ページの`clconfiguration`(5CL) のマニュアルページを参照してください。デフォルトでは、すべての出力が標準出力に送信されます。出力先をファイルに変更するには、`-o` オプションを使用して、そのあとにファイル名を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`list`

指定されたノード上のクラスタの MIB のリストを表示します。

このサブコマンドは、大域ゾーンだけで使用できます。

一覧表示するクラスタの MIB のノードを指定するには、`-n` オプションを使用します。`list` サブコマンドを `-n` オプションなしで使用すると、このサブコマンドは現在のノード上の MIB だけを一覧表示します。一覧表示する MIB を制限するには、一覧表示する 1 つ以上の MIB の名前を指定します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`set`

指定されたノード上の 1 つ以上の MIB で使用される SNMP プロトコル、`min_severity`、または `log_number` 設定を変更します。

このサブコマンドは、大域ゾーンだけで使用できます。

デフォルトでは、このサブコマンドはノード上のすべての MIB を変更します。ノードを指定しないと、現在のノード上の MIB の SNMP プロパティーのみが変更されます。SNMP プロパティーは、`-p` オプションを使って指定する必要があります。すべての MIB では、デフォルトのプロパティー値 `protocol:SNMPv2`、`min_severity:NOTICE`、`log_number:100` を使用します。`set` サブコマンドは、*mib* オペランドを使用して MIB 名を指定しないかぎり、すべての MIB のプロトコル、`min_severity`、または `log_number` 設定を変更します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

show

指定されたノード上の MIB の情報を表示します。

このサブコマンドは、大域ゾーンだけで使用できます。

show サブコマンドは、MIB の名前、その SNMP プロトコルバージョン、min_severity 値、または log_number 値を表示します。デフォルトでは、このサブコマンドはノード上のすべての MIB の情報を表示します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

次のオプションがサポートされています。

-?
--help

ヘルプ情報を表示します。

このオプションはサブコマンド付きでもサブコマンドなしでも指定できます。

- このオプションをサブコマンドなしで使用すると、使用可能なサブコマンドのリストが表示されます。
- このオプションをサブコマンドとともに使用すると、サブコマンドの使用オプションが表示されます。

このオプションを使用すると、ほかの処理は実行されません。

-n *node*[,…]
--node[s] *node*[,…]

ノードまたはノードリストを指定します。各ノードはノード名またはノード ID で指定できます。clsnmpmib コマンドのすべての形式で、このオプションが受け付けられます。-n オプションを使用すると、処理が実行されるノードを指定できます。- n オプションがない場合、このコマンドは現在のノードを想定します。

-o {- | *clconfigfile*}
--output {- | *clconfigfile*}

クラスタの MIB 構成についての情報が書き込まれる場所を指定します。この書き込み先は、ファイルでも、標準出力でもかまいません。標準出力を指定するには、ファイル名の代わりにマイナス記号 (-) を指定します。標準出力を指定すると、コマンドに対するほかのすべての標準出力が抑制されます。-o オプションを指定しない場合、出力は標準出力に送信されます。このオプションは、export サブコマンドの場合にだけ指定できます。

構成情報は、1447 ページの`clconfiguration`(5CL) のマニュアルページで定義されている形式で書き込まれます。

`-p` *name*= *value*
`--property=`*name* =*value*
`--property` *name*=*value*

`version` MIB で使用する SNMP プロトコルのバージョンを指定します。Oracle Solaris Cluster は、SNMPv2 および SNMPv3 プロトコルバージョンをサポートしています。

`min_severity` 重要度の最小値を指定します。`min_severity` 値に等しいか、それを上回る値を持つイベントのみが MIB テーブルに記録され、それらのイベントについて、トラップが構成済みのホストに送信されます。

`log_number` 古いエントリを破棄するまでに MIB テーブルに記録するイベントの数を指定します。

`-p` *name*=*value* は複数回指定できます。

このオプションとともに次のプロパティーを設定できます。

`version`

MIB で使用する SNMP プロトコルのバージョンを指定します。*value* は次のように指定します。

- `version=SNMPv2`
- `version=snmpv2`
- `version=2`
- `version=SNMPv3`
- `version=snmpv3`
- `version=3`

`min_severity`

MIB で使用する重要度の最小値を指定します。この値は次のように指定します。

- `min_severity=NOTICE`
- `min_severity=WARNING`
- `min_severity=ERROR`
- `min_severity=CRITICAL`
- `min_severity=FATAL`

大文字と小文字のどちらの値も指定できます。

`log_number`

古いエントリを破棄するまでに MIB テーブルに記録するイベントの数を指定します。デフォルト値は 100 です。値は 100-500 の範囲である必要があります。この値は次のように指定します。

- `log_number=`*number*

`-V`
`--version`

コマンドのバージョンを表示します。

このオプションは、サブコマンド、オペランド、またはその他のオプションと一緒に指定しないでください。指定すると、一緒に指定されたサブコマンド、オペランド、またはその他のオプションは無視されます。`-V` オプションは、コマンドのバージョンだけを表示します。その他の処理は行いません。

`-v`
`--verbose`

詳細情報を標準出力に出力します。

このオプションはどの形式のコマンドととも指定できますが、一部のサブコマンドは拡張出力を発生させないことがあります。たとえば、`export` サブコマンドでは、冗長オプションを指定すると拡張出力は生成されません。

次のオペランドがサポートされています。

mib　　サブコマンドが適用される単数または複数の MIB の名前を指定します。このオペランドを指定しない場合は、デフォルトのプラス記号 (`+`) がサブコマンドで使用されます (すべての MIB を意味します)。*mib* オペランドを使用する場合は、その他のすべてのコマンド行オプションのあとにあるスペースで区切られたリスト内で MIB を指定してください。

`+`　　クラスタのすべての MIB。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (`CL_NOERR`) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

`0 CL_NOERR`

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

18 CL_EINTERNAL

内部エラーが発生しました

内部エラーは、ソフトウェアの欠陥またはその他の欠陥を示しています。

35 CL_EIO

I/O エラー

物理的な入出力エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

例 226 MIB の一覧表示

次のコマンドは、クラスタノード上にあるすべての MIB を一覧表示します。

```
# clsnmpmib list
Event
```

例 227 MIB の有効化

次のコマンドは、現在のノード上にある Event MIB を有効にします。

```
# clsnmpmib enable event
```

クラスタの MIB の名前では大文字と小文字は区別されません。

例 228 プロトコルの変更

次のコマンドは、クラスタノード phys-cluster-2 の Event MIB のプロトコルを SNMPv3 に変更します。

```
# clsnmpmib set -n phys-cluster-2 -p version=SNMPv3 Event
```

-n オプションを使用すると、ノード名の代わりにノード ID を使用できます。

例 229 構成の表示

次のコマンドは、クラスタノード phys-cluster-1 および phys-cluster-2 上の構成情報を表示します。

```
# clsnmpmib show -n phys-cluster-1,phys-cluster-2
--- SNMP MIB Configuration on myhost ---

SNMP MIB Name:                 phys-cluster-1
  State:                         Event
  Enabled:                       yes
  Protocol:                      SNMPv3
  min_severity:                  1
  log_number:                    100
SNMP MIB Name:                 phys-cluster-2
  State:                         Event
  Enabled:                       yes
  Protocol:                      SNMPv3
  min_severity:                  3
  log_number:                    250
```

例 230 Min Severity 値の変更

次のコマンドは、クラスタノード phys-cluster-2 上のイベント MIB の min_severity を WARNING に変更します。

```
# clsnmpmib set -n phys-cluster-2 -p min_severity=WARNING Event
```

-n オプションを使用すると、ノード名の代わりにノード ID を使用できます。

例 231 Log_Number 値の変更

次のコマンドは、クラスタノード phys-cluster-2 上のイベント MIB の log_number を 250 に変更します。

```
# clsnmpmib set -n phys-cluster-2 -p log_number=250 Event
```

-n オプションを使用すると、ノード名の代わりにノード ID を使用できます。

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

/usr/cluster/lib/mib/sun-cluster-event-mib.mib

Oracle Solaris Cluster SNMP イベント MIB 定義ファイル

491 ページのclsnmphost(1CL), 509 ページのclsnmpuser(1CL), 19 ページのIntro(1CL), 551 ページのcluster(1CL), 809 ページのsceventmib(1M), 1047 ページのscha_calls(3HA), Unresolved link to " attributes5", Unresolved link to " rbac5", 1447 ページのclconfiguration(5CL)

スーパーユーザーはこのコマンドのすべての形式を実行できます。

すべてのユーザーがこのコマンドに -? (ヘルプ) オプションまたは -V (バージョン) オプションを指定して実行できます。

clsnmpmib コマンドをほかのサブコマンドとともに実行する場合、スーパーユーザー以外のユーザーは RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
disable	solaris.cluster.modify
enable	solaris.cluster.modify
export	solaris.cluster.read

サブコマンド	RBAC の承認
list	solaris.cluster.read
set	solaris.cluster.modify
show	solaris.cluster.read

名前

clsnmpuser — Oracle Solaris Cluster SNMP ユーザーの管理

```
/usr/cluster/bin/clsnmpuser -V

/usr/cluster/bin/clsnmpuser [subcommand] -?

/usr/cluster/bin/clsnmpuser [subcommand] [options] -v [operand]

/usr/cluster/bin/clsnmpuser create -i {- | clconfigfile}
     [-a authentication] -f passwdfile [-n node[,…]] {+ | user …}

/usr/cluster/bin/clsnmpuser delete [-a authentication] [-n node[,…]]
     {+ | user …}

/usr/cluster/bin/clsnmpuser export [-o {- | clconfigfile}]
     [-a authentication] [-n node[,…]] [{+ | user …}]

/usr/cluster/bin/clsnmpuser list [-a authentication] [-n node[,…]]
     {-d | + | user …}

/usr/cluster/bin/clsnmpuser set [-a authentication] [-n node[,…]]
     {+ | user …}

/usr/cluster/bin/clsnmpuser set-default {-l seclevel [,…]}
     {+ | user …}

/usr/cluster/bin/clsnmpuser show [-a authentication] [-n node[,…]]
     [-d | + | user …]
```

`clsnmpuser` コマンドは、クラスタの管理情報ベース (MIB) 上の制御メカニズムを管理できる SNMP (Simple Network Management Protocol) ユーザーの役割を管理します。クラスタ MIB の詳細は、499 ページの`clsnmpmib`(1CL) のマニュアルページを参照してください。クラスタが SNMP バージョン 3 (SNMPv3) を使用するように構成されている MIB を含む場合は、SNMP ユーザーを定義します。SNMP ユーザーは Solaris OS ユーザーと同じユーザーではありません。SNMP ユーザーは既存の OS ユーザーと同じユーザー名を持つ必要はありません。

このコマンドに短形式はありません。

このコマンドの一般的な形式は次のとおりです。

`clsnmpuser` [*subcommand*] [*options*] [*operands*]

subcommand を省略できるのは、*options* がオプション `-?` または `-V` を指定している場合だけです。

このコマンドの各オプションには、長い形式と短い形式があります。各オプションの両方の形式は、OPTIONS セクションにオプションの説明とともに記載されています。

詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

このコマンドは、大域ゾーンだけで使用できます。

サポートされるサブコマンドには次のものがあります。

`create`

ユーザーを作成し、指定されたノードの SNMP ユーザー構成に追加します。

このサブコマンドは、大域ゾーンだけで使用できます。

SNMP ユーザー を作成するクラスタノードを指定するには、このサブコマンドとともに `-n` オプションを使用します。`-n` オプションを指定しない場合は、ユーザーが作成され、現在のノード上の SNMP 構成にのみ追加されます。

`clconfiguration` ファイル内に構成されているすべてのユーザーを作成および追加するには、`-i` オプションと `-n` オプションを使用します。

作成している SNMP ユーザーに認証タイプを割り当てるには、`-a` オプションを指定します。

SNMP ユーザーのパスワードは、`-f` オプションを指定することで含められます。`-i` オプションを使用している場合は、`-f` オプションが必要です。

`-i` オプションを指定すると、1447 ページの`clconfiguration`(5CL) ファイルの構成情報が使用されます。`-i` オプションを指定すると、正符号の + オペランドまたはユーザーのリストも指定できます。

スーパーユーザー以外のユーザーの場合、このコマンドを使用するためには `solaris.cluster.modify` 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`delete`

SNMPv3 ユーザーを指定されたノードから削除します。

このサブコマンドは、大域ゾーンだけで使用できます。

`delete` サブコマンドを使用してユーザー名だけを指定すると、このサブコマンドは、そのユーザーのすべてのインスタンスを削除します。認証タイプ別にユーザーを削除するには、`-a` オプションを使用します。`-n` オプションを使用しないと、ユーザーは現在のノードからのみ削除されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`export`

SNMP ユーザー情報を指定されたノードからエクスポートします。

このサブコマンドは、大域ゾーンだけで使用できます。

-n オプションを使用しないと、SNMP ユーザー情報は現在のノードからのみエクスポートされます。export サブコマンドからの出力の形式については、1447 ページのclconfiguration(5CL) のマニュアルページを参照してください。デフォルトでは、すべての出力が標準出力に送信されます。出力先をファイルに変更するには、-o オプションを使用して、そのあとにファイル名を指定します。

-a オプションを使用して、特定の認証タイプのユーザーだけの出力を提供することもできます。1 人以上のユーザーをオペランドとして指定すると、出力は指定したユーザーについての情報に制限されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

list

指定されたノードで構成されている SNMPv3 ユーザーのリストを出力します。

このサブコマンドは、大域ゾーンだけで使用できます。

デフォルトでは、list サブコマンドは指定されたノード上のすべての SNMPv3ユーザーを表示します。デフォルト SNMP ユーザーだけを表示するには、-d オプションをオペランドなしで指定します。指定した認証タイプに出力を制限するには、-a オプションを使用します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

set

指定されたノード上のユーザーの構成を変更します。

このサブコマンドは、大域ゾーンだけで使用できます。

-n オプションを指定しないと、現在のノード上でのみユーザーの構成が変更されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

set-default

デフォルトの SNMP ユーザーの名前と MIB がトラップ通知を送信するときに使用されるセキュリティーレベルを指定します。

このサブコマンドは、大域ゾーンだけで使用できます。

セキュリティーレベルを指定するには、-l オプションを使用します。

MIB が SNMPv3 を使用するように構成されている場合は、具体的なユーザー名とトラップを認証するセキュリティーレベルを指定します。構成が複数のユーザーを含む場合は、MIB がトラップ通知を送信するときに使用するデフォルトユーザーを指定します。

構成が 1 人のユーザーしか含まない場合は、そのユーザーが自動的にデフォルトの SNMP ユーザーになります。デフォルトの SNMP ユーザーが削除されると、存在する場合は別の既存のユーザーがデフォルトになります。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

show

指定されたノード上のユーザーについての情報を出力します。

このサブコマンドは、大域ゾーンだけで使用できます。

デフォルトでは、show サブコマンドはそのノード上のすべてのユーザーについての情報を表示します。デフォルトの SNMP ユーザーについての情報だけを表示するには、オペランドを指定せずに -d オプションを指定してください。特定の認証タイプに出力を制限するには、-a オプションを使用します。-n オプションを使用しないと、このコマンドは現在のノードのユーザー情報だけを表示します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

次のオプションがサポートされています。

-?
--help

ヘルプ情報を出力します。

このオプションはサブコマンド付きでもサブコマンドなしでも指定できます。

- このオプションをサブコマンドなしで使用すると、使用可能なサブコマンドのリストが表示されます。
- このオプションをサブコマンドとともに使用すると、サブコマンドの使用オプションが表示されます。

このオプションを使用する場合、ほかの処理は実行されません。

-a *authentication*
--authentication *authentication*

ユーザーの承認に使用する認証プロトコルを指定します。認証プロトコルの値は SHA または MD5 です。

-d
--default

MIB がトラップ通知を送信するときに使用されるデフォルトの SNMP ユーザーを指定します。

`-f` *passwdfile*
`--file` *passwdfile*

1 つ以上の SNMP ユーザーパスワードを含むファイルを指定します。新しいユーザーを作成する際にこのオプションを指定しないと、コマンドはパスワードを求めるプロンプトを表示します。このオプションは、`create` サブコマンドとともに使用するときにのみ有効になります。

ユーザーパスワードは次の形式で別々の行で指定します。

user:*password*

パスワードには次に示す文字または空白文字を含めることはできません。

- `;` (セミコロン)
- `:` (コロン)
- `\` (バックスラッシュ)
- `\n` (改行)

`-i` {- | *clconfigfile*}
`--input` {- | *clconfigfile*}

SNMP ホスト構成を検証または変更するために使用される構成情報を指定します。この情報は、1447 ページの`clconfiguration`(5CL) のマニュアルページに定義されている形式に準拠している必要があります。この情報は、ファイルに含めることも、標準入力を介して指定することもできます。標準入力を指定するには、ファイル名の代わりにマイナス記号 (-) を指定します。

`-l` *seclevel*
`--securitylevel` *seclevel*

ユーザーのセキュリティーレベルを指定します。次に示す値のいずれかを *seclevel* に指定してください。

- `noAuthNoPriv`
- `AuthNoPriv`
- `authPriv`

SNMP セキュリティーレベルの詳細は、Unresolved link to " snmpcmd1" のマニュアルページを参照してください。

`-n` *node*[,…]
`--node[s]` *node*[,…]

ノードまたはノードリストを指定します。各ノードはノード名またはノード ID で指定できます。

このコマンドのすべての形式は、このオプションを受け付けます。

-o {- | *clconfigfile*}
--output {- | *clconfigfile*}

クラスタの SNMP ホスト構成情報を、1447 ページの`clconfiguration`(5CL) のマニュアルページで説明されている形式で書き込みます。この情報は、ファイルまたは標準出力のどちらにでも書き込むことができます。

標準出力に書き込むには、ファイル名の代わりにマイナス記号 (-) を指定します。標準出力を指定すると、該当コマンドにおける他のすべての標準出力は抑制されます。

-V
--version

コマンドのバージョンを出力します。

このオプションは、サブコマンド、オペランド、またはその他のオプションと一緒に指定しないでください。指定すると、一緒に指定されたサブコマンド、オペランド、またはその他のオプションは無視されます。-V オプションは、コマンドのバージョンだけを表示します。その他の処理は行いません。

-v
--verbose

詳細メッセージと情報を出力します。

このオプションはどの形式のコマンドとともに指定できますが、一部のサブコマンドは拡張出力を発生させないことがあります。たとえば、`export` サブコマンドは、詳細オプションを指定しても拡張出力を発生させません。

次のオペランドがサポートされています。

+	すべての SNMP ユーザーを指定します。
user	SNMP ユーザー名を指定します。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (`CL_NOERR`) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

18 CL_EINTERNAL

内部エラーが発生しました

内部エラーは、ソフトウェアの欠陥またはその他の欠陥を示しています。

35 CL_EIO

I/O エラー

物理的な入出力エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

例 232 SNMPv3 ユーザーの作成

次のコマンドは、新しいユーザー newuser1 を作成し、現在のノード上の構成にそのユーザーを追加します。認証タイプは SHA です。

```
# clsnmpuser create -a SHA newuser1
Enter password for user 'newuser1':
```

この例では、作成されるユーザーのパスワードを入力する必要があります。プロセスを自動化するには、-f オプションを使用します。

例 233 ユーザーの一覧表示

次のコマンドは、認証タイプが MD5 のすべてのユーザーを一覧表示します。

```
# clsnmpuser list -a MD5 +
user1
mySNMPusername
```

正符号 (+) はデフォルトなので、指定してもしなくてもかまいません。

例 234 ユーザーの表示

次のコマンドは、現在のノード上のすべてのユーザーのユーザー情報を表示します。

```
# clsnmpuser show

--- SNMP User Configuration on phys-schost-1 ---

SNMP User Name:                                 newuser1
  Authentication Protocol:                        SHA
  Default User:                                   Yes
  Default Security Level:                         authPriv
```

例 235 ユーザーの認証プロトコルとステータスの変更

次のコマンドは、ユーザー newuser1 の認証プロトコルとデフォルトのユーザーのステータスを変更します。

```
# clsnmpuser set -a MD5 newuser1
```

例 236 SNMP ユーザーの削除

次のコマンドは、すべての SNMP ユーザーを削除します。

```
# clsnmpuser delete +
```

この例では、すべてのユーザーを指定するために正符号 (+) が使用されます。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

491 ページの`clsnmphost`(1CL), 499 ページの`clsnmpmib`(1CL), 551 ページの`cluster`(1CL), 19 ページの`Intro`(1CL), 809 ページの`sceventmib`(1M), Unresolved link to " snmpcmd1", Unresolved link to " su1M", 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5", Unresolved link to " rbac5", 1447 ページの`clconfiguration`(5CL)

スーパーユーザーはこのコマンドのすべての形式を実行できます。

すべてのユーザーがこのコマンドに `-?` (ヘルプ) オプションまたは `-V` (バージョン情報) オプションを指定して実行できます。

`clsnmpmib` コマンドをほかのサブコマンドとともに実行する場合、スーパーユーザー以外のユーザーは RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
`create`	`solaris.cluster.modify`
`delete`	`solaris.cluster.modify`
`export`	`solaris.cluster.read`
`list`	`solaris.cluster.read`
`set`	`solaris.cluster.modify`
`set-default`	`solaris.cluster.modify`
`show`	`solaris.cluster.read`

名前

cltelemetryattribute, clta — システムリソースモニタリングの構成

```
/usr/cluster/bin/cltelemetryattribute -V

/usr/cluster/bin/cltelemetryattribute [subcommand] -?

/usr/cluster/bin/cltelemetryattribute subcommand [options] -v
     [telemetry-attribute …]

/usr/cluster/bin/cltelemetryattribute disable [-i
     {- | clconfigfile}] [-t object-type] {+ | telemetry-attribute ...}

/usr/cluster/bin/cltelemetryattribute enable [-i
     {- | clconfigfile}] [-t object-type] {+ | telemetry-attribute ...}

/usr/cluster/bin/cltelemetryattribute export [-o
     {- | clconfigfile}] [-t object-type[,…]]
     [+ | telemetry-attribute ...]

/usr/cluster/bin/cltelemetryattribute list [-t object-type[,…]]
     [+ | telemetry-attribute ...]

/usr/cluster/bin/cltelemetryattribute print [-b object-instance[,…]]
     [-a] [-d period] [-u] [-n node[,…]] [-t object-type[,…]]
     [+ | telemetry-attribute ...]

/usr/cluster/bin/cltelemetryattribute set-threshold -b
     object-instance [-n node] {-p name=value} [-p name=value]
     […] -t object-type telemetry-attribute

/usr/cluster/bin/cltelemetryattribute show [-b object-instance[,…]]
     [-n node[,…]] [-t object-type[,…]]
     [+ | telemetry-attribute ...]

/usr/cluster/bin/cltelemetryattribute status -b object-instance
     [-n node] {-p name} -t object-type [+ | telemetry-attribute ...]
```

このコマンドは、システムリソースのモニタリングを構成します。

システム資源の使用状況は、次に示す型をはじめとする異なる型のオブジェクト上でモニターできます。

- ディスク
- ファイルシステム
- IP アドレス
- ネットワークインタフェース
- ノード

- Solaris ゾーン
- リソースグループ

システム資源のモニター面は、テレメトリ属性と呼ばれます。

このコマンドは次のことを行います。

- テレメトリ属性を有効または無効にします。
- テレメトリ属性のしきい値を設定または変更します。
- モニターされている属性、適用されているしきい値、オブジェクトについて収集されるデータのリストを表示します。

対応するテレメトリ属性を特定することで、モニターするシステム資源使用状況の側面を選択します。あるオブジェクトでのシステム資源の使用状況をモニターするには、その型のオブジェクト上で対応するテレメトリ属性を有効にします。Oracle Solaris Cluster ソフトウェアは、クラスタ内にあるその型のすべてのオブジェクト上で指定された属性に対する使用状況データを収集します。

システム資源の場合、特定の値がクラスタの性能にとっての臨界値である場合があります。臨界値を過ぎた場合に通知を受けるため、テレメトリ属性のしきい値を設定できます。しきい値については、`set-threshold` サブコマンド、および `-p` オプションの説明を参照してください。

subcommand を省略できるのは、*options* が `-?` オプションまたは `-V` オプションの場合のみです。

このコマンドの各オプションには長形式と短形式があります。各オプションの両方の形式は、オプション に説明とともに記載されています。

`clta` コマンドは `cltelemetryattribute` コマンドの短い形式です。

システム資源のモニタリングの構成を詳細化する前にはモニタリングを初期化してください。899 ページの`sctelemetry`(1M) のマニュアルページを参照してください。

このコマンドは、大域ゾーンだけで使用できます。

サブコマンド

サポートされるサブコマンドには次のものがあります。

`disable`

指定されたオブジェクト型の指定されたテレメトリ属性を無効にします。

このサブコマンドは、大域ゾーンだけで使用できます。

Oracle Solaris Cluster ソフトウェアは、enabled 状態が設定されているシステムリソースの使用状況データを収集します。あるオブジェクト型のシステムリソースを disabled 状態に設定すると、Oracle Solaris Cluster ソフトウェアはそのオブジェクトインスタンスに対応するどのインスタンスのデータも収集しません。

また、cltelemetryattribute コマンドは、次に示す条件が両方満たされる場合、これらの属性のデータ収集を無効にします。

- 構成ファイルを -i オプションで指定する場合。
- 入力ファイルでテレメトリ属性に disabled が設定されている場合。

export サブコマンドを使用して構成ファイルを作成する場合。

テレメトリ属性に disabled を設定した場合、構成されているしきい値の設定は変更されません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

enable

指定されたオブジェクト型の指定されたテレメトリ属性のデータ収集を有効にします。

このサブコマンドは、大域ゾーンだけで使用できます。

デフォルトでは、選択されている属性が選択されているオブジェクト型に対して有効になっています。

テレメトリ属性のデータ収集を有効にするには、テレメトリ属性に enabled を設定します。

Oracle Solaris Cluster ソフトウェアは、そのオブジェクト型に対してテレメトリ属性が有効なオブジェクト型のデータだけを収集します。オブジェクト型のある属性を有効にすると、Oracle Solaris Cluster ソフトウェアは、すべてのノード上でその型のすべてのオブジェクトインスタンスの有効にされた属性についてのデータを収集します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

export

オブジェクト型およびオブジェクトインスタンスのテレメトリ属性の構成をファイルまたは標準出力 stdout にエクスポートします。

このサブコマンドは、大域ゾーンだけで使用できます。

構成には、オブジェクト型に対して属性が有効なのか無効なのかが含まれます。また、構成にはしきい値を構成するために設定するリミットも含まれる場合があります。

-o オプションを使用してファイルを指定し、構成情報をファイルに書き込みます。-o オプションを指定しない場合、cltelemetryattribute コマンドは構成情報を標準出力 (stdout) に書き込みます。

export サブコマンドはクラスタ構成データを変更しません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`list`

指定されたオブジェクト型に対して構成できるテレメトリ属性を表示します。

このサブコマンドは、大域ゾーンだけで使用できます。

詳細オプション `-v` を指定した場合、`list` サブコマンドは、属性を適用できるオブジェクトの型を表示します。

しきい値のプロパティーは、次の書式で表示されます。

`Threshold:` *severity*`,` *direction*
`,` *value*`,` *rearm*

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`print`

指定されたオブジェクトインスタンスまたはオブジェクト型に対して有効な指定されているテレメトリ属性のシステム資源の使用状況を表示します。

このサブコマンドは、大域ゾーンだけで使用できます。

出力は次に示すデータを含みます。

- 日付とタイムスタンプ
- オブジェクトインスタンス
- オブジェクト型
- テレメトリ属性
- ノード
- 値

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`set-threshold`

ノード上の指定されたオブジェクトの指定されたテレメトリ属性のしきい値の設定を変更します。

このサブコマンドは、大域ゾーンだけで使用できます。

変更されるしきい値を指定するには、`-p` オプションを使用します。また、変更するしきい値のプロパティーを指定する場合も `-p` オプションを使用します。変更できるのは、`value` および `rearm` しきい値プロパティーのみです。

指定されたしきい値のこれらのプロパティーを 1 つ以上変更してください。1 つ以上のプロパティーが構成されている場合、`status` サブコマンドを実行すると出力が表示されます。

しきい値を非アクティブにするには、次のように `value` と `rearm` に空白を指定します。

```
-y value=,rearm=
```

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`show`

オブジェクト型またはオブジェクトインスタンスのテレメトリ属性に対して構成されているプロパティーを表示します。

このサブコマンドは、大域ゾーンだけで使用できます。

これらの属性は、システム資源がオブジェクト型に対して有効かどうかを含みます。詳細オプション `-v` を指定した場合、`show` サブコマンドは、オブジェクトインスタンスに対して有効なテレメトリ属性のしきい値の設定を表示します。

しきい値のプロパティーは、次の書式で表示されます。

`Threshold:` *severity*`,` *direction*
`,` *value*`,` *rearm*

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`status`

標準出力でしきい値が構成されているオブジェクト型の現在のステータスを表示します。最低 1 つのしきい値を設定しないと、`status` コマンドを実行したときに、出力は表示されません。

このサブコマンドは、大域ゾーンだけで使用できます。

警告または重大な重要度を現在持っている有効なすべてのしきい値のステータスを表示するには、このサブコマンドを引数なしで使用します。しきい値の考えられる出力は、しきい値の現在の重要度を含みます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

次のオプションがサポートされています。

`-?`
`--help`

ヘルプ情報を表示します。

このオプションは*サブコマンド*付きでもサブコマンドなしでも指定できます。

このオプションを *subcommand* なしで指定すると、使用可能なすべてのサブコマンドのリストが表示されます。

このオプションを *subcommand* とともに指定すると、*subcommand* の使用法が表示されます。

このオプションを `set-threshold` サブコマンドとともに指定すると、すべてのリソースグループプロパティーのヘルプ情報が表示されます。

このオプションをほかのオプション、サブコマンド、またはオペランドと一緒に指定する場合、これらはすべて無視されます。ほかの処理は行われません。

`-a`
`--average`

3 時間の期間に収集されたデータの平均およびこの平均に関連する標準偏差を出力します。

`-a` オプションと `-d` オプションを一緒に指定すると、指定した *period* の間、3 時間間隔で平均化されたデータが出力されます。

`-a` オプションを指定しない場合、出力されるデータは最新のデータです。

`-b` *object-instance*
`--object-instance=`*object-instance*
`--object-instance` *object-instance*

情報を表示するまたはしきい値を設定するオブジェクトインスタンスを指定します。

オブジェクトインスタンスは常に特定の型です。たとえば、クラスタノード `phys-schost-1` は `node` 型のオブジェクトインスタンスです。Oracle Solaris Cluster ソフトウェアは、対応するテレメトリ属性がオブジェクト型に対して有効な場合だけ、オブジェクトインスタンスのシステムリソースをモニターします。

`-d` *period*
`--date-range=`*period*
`--date-range` *period*

Oracle Solaris Cluster ソフトウェアがモニタリングデータを収集する期間を指定します。

period 引数に指定する日付と時間の形式は、International Organization for Standardization (ISO) 8601 の国際日付形式に準拠する必要があります。

begin-time, *end-time*] 期間は、コンマ , で区切られた 2 つの時間の間の時間です。

begin-time+ 期間は、指定された開始時間と現在の時間の間の時間です。

end-time- 期間は、Oracle Solaris Cluster ソフトウェアが起動してデータの収集を開始する時間と指定された終了時間の間の時間です。

period の形式の例を次に示します。

```
-d 2006-04-30T18:00,2006-06-16T18:00
```

2006 年 4 月 30 日の午後 6:00 から 2006 年 6 月 16 日の午後 6:00 まで

`-d 2006-06-16+`

2006 年 6 月 16 日の夜中 12:00 以降

`-d 2006-07-31T18:00+`

2006 年 7 月 31 日の午後 6:00 以降

`-d 2006-06-16T18:00-`

Oracle Solaris Cluster ソフトウェアが起動してから 2006 年 6 月 16 日の午後 6:00 まで

`-d 2006-05-31T12:00,2006-06-16T11:59`

2006 年 5月 31 日の夜中 12:00 から 2006 年 6 月 16 日の午後 11:59 まで

このオプションは `print` サブコマンドとだけ使用できます。

`-i` {- | *clconfigfile*}
`--input=`{- | *clconfigfile*}
`--input` {- | *clconfigfile*}

clconfigfile ファイル内にある構成情報を使用して、テレメトリ属性としきい値構成を指定することを指定します。1447 ページの`clconfiguration`(5CL) のマニュアルページを参照してください。

標準入力 stdin を通して構成情報を提供するには、このオプションとともにダッシュ `-` を指定します。ほかのオプションを指定した場合は、*clconfigfile* 内のオプションや情報より指定したオプションが優先されます。

`-n` *node*
`--node=`*node*
`--node` *node*

Oracle Solaris Cluster が使用状況データを収集するノード名を指定します。名前またはノード識別子を指定できます。

`-node` 型のオブジェクトインスタンス、リソース、またはリソースグループでサブコマンドを指定する際には、`n` オプションを指定しないでください。

`-o` {- | *clconfigfile*}
`--output=`{- | *clconfigfile*}
`--output` {- | *clconfigfile*}

テレメトリ属性としきい値の構成データをファイルまたは標準出力 `stdout` に書き込みます。この構成情報の形式は、1447 ページの`clconfiguration`(5CL) のマニュアルページで説明されています。

このオプションにファイル名を指定する場合、このオプションは新しいファイルを作成します。次に、ノード構成情報はそのファイルに格納されます。`-` をこのオプションとともに指定すると、構成情報は標準出力 (`stdout`) に送信されます。このコマンドのほかの標準出力はすべて抑制されます。

このオプションを一緒に指定できるのは、`export` サブコマンドだけです。

-p *name*
--property=*name*
--property *name*

status サブコマンドのプロパティーのリストを指定します。

set-threshold サブコマンドでしきい値を設定できるプロパティーについては、-p *name*=*value* オプションの説明を参照してください。

-p *name*=*value*
--property=*name*=*value*
--property *name value*

しきい値のプロパティーを指定します。

-p *name*= *value* は複数回指定できます。

status サブコマンドで情報を表示できるプロパティーについては、-p *name* オプションの説明を参照してください。

しきい値ごとに severity および direction プロパティーを指定して、しきい値を特定してください。しきい値の設定後は、これらのプロパティーを変更できません。

各しきい値の value を設定します。各しきい値の rearm も設定できます。value および rearm プロパティーを変更するには、set-threshold サブコマンドを使用します。このオプションで指定できるプロパティーと値は次のとおりです。

severity

しきい値の重要度です。このプロパティーに設定できる値は、fatal と warning です。重要度が fatal のしきい値は、重要度が warning のしきい値より重大です。

重要度は、Oracle Solaris Cluster Manager で視覚的なアラームとして表示されます。

direction

適用されるしきい値の方向です。このプロパティーに設定できる値は、falling と rising です。direction プロパティーに falling を設定することで、fatal 重要度レベルが warning 重要度レベルより小さい値を持つことを指定します。direction プロパティーに rising を設定することで、fatal 重要度レベルが warning 重要度レベルより大きい値を持つことを指定します。

value

テレメトリ属性のしきい値に設定する値です。このしきい値を越えると、テレメトリ属性の重要度が変わります。1 つのオブジェクトの特定のテレメトリ属性には最大 4 つのしきい値を関連付けられます。

value プロパティーを設定または変更するには、set-threshold サブコマンドを使用します。

`rearm`

テレメトリ属性の重要度を解除する手段です。rearm の値を指定することで、テレメトリ属性の値が `direction` プロパティーに設定されているのと逆の方向で rearm の値を越えると、テレメトリ属性の重要度が解除されます。rearm の値を指定しないと、rearm の値は、しきい値と rearm の値に同じ値が設定されているのと同じになります。

通知の頻度は *hysteresis* の原則に従います。つまり、頻度は二重値機能により決定されます。関数が増えると 1 つの値が適用されます。関数が値と同じ場合は、もう一方の値が適用されます。

使用中のシステムに合った `rearm` および `value` の値を設定してください。オプションの `rearm` プロパティーを指定しないと、このプロパティーはデフォルトとして `value` を採用します。ただし、`rearm` プロパティーに `value` プロパティーと同じ値を設定した場合、または `rearm` に値を割り当てない場合は、モニター対象のテレメトリ属性の値が `value` に対して設定されている値以上または値以下になるたびに通知を受信します。大量の通知の受信を避けるには、`rearm` に `value` 以外の値を設定してください。

`rearm` を `set-threshold` サブコマンドとともに指定すると、`cltelemetryattribute` コマンドにより `rearm` の値が次の要件を満たすことが保証されます:

- `direction` が `rising` の場合、`value` は `rearm` 以上の値を持ちます。
- `direction` が `falling` の場合、`value` は `value` 以下の値を持ちます。

`rearm` を変更するには、`set-threshold` サブコマンドを使用します。

`-t` *object-type*
`--object-type=`*object-type*
`--object-type` *object-type*

Oracle Solaris Cluster ソフトウェアが使用状況データを収集するオブジェクトの型を指定します。すべてのオブジェクトインスタンスは特定の型です。

指定された型のオブジェクトにサブコマンドの出力を制限するには、このオプションを使用します。

システム資源をモニターできるオブジェクト型と各オブジェクト型の関連テレメトリ属性は、次のとおりです。

オブジェクトタイプ	説明	テレメトリ属性
`disk`	ディスク	`rbyte.rate`、`wbyte.rate`、`read.rate`、`write.rate`、
`filesystem`	ファイルシステム	`block.used`、`inode.used`
`ipaddr`	IP アドレス	`ipacket.rate`、`opacket.rate`
`netif`	ネットワークインタフェース	`ipacket.rate`、`opacket.rate`、`rbyte.rate`、`wbyte.rate`

オブジェクトタイプ	説明	テレメトリ属性
node	ノード	cpu.idle、cpu.iowait、cpu.used、loadavg.1mn、loadavg.5mn、loadavg.15mn、mem.used、mem.free、swap.used、swap.free
resourcegroup	リソースグループ	cpu.used、mem.used、swap.used
zone	ゾーン	cpu.idle、cpu.iowait、cpu.used、loadavg.1mn、loadavg.5mn、loadavg.15mn

モニターできるテレメトリ属性は次のとおりです。

テレメトリ属性	説明
block.used	デバイス上で使用されているブロックのパーセンテージ
cpu.idle	空き CPU の容量
cpu.iowait	入出力の完了を待っている CPU の容量
cpu.used	使用されている CPU の容量
inode.used	デバイス上で使用されている i ノードのパーセンテージ
ipacket.rate	1 秒間の着信パケット数
loadavg.1mn	最後の 1 分間に CPU を待っていたプロセスの数
loadavg.5mn	最後の 5 分間に CPU を待っていたプロセスの数
loadavg.15mn	最後の 15 分間に CPU を待っていたプロセスの数
mem.free	空きメモリの M バイト数
mem.used	使用されているメモリの M バイト数
opacket.rate	1 秒間の送信パケット数
rbyte.rate	1 秒間に読み取られる M ビット数
read.rate	秒当たりの読み取り動作数
swap.free	空きスワップメモリの M バイト数
swap.used	使用されているスワップメモリの M バイト数
wbyte.rate	1 秒間に書き込まれる M ビット数
write.rate	秒当たりの書き込み動作数

すべてのオブジェクト型に対して上記の表に一覧表示されているすべてのテレメトリ属性をモニターできません。データを収集できるオブジェクト型とオブジェクトの型ごとにモニターできるテレメトリ属性を表示するには、`list` サブコマンドを使用します。

-u
--utc

使用状況データとともに示される日時を協定世界時 (UTC) またはグリニッジ標準時 (GMT) で表示します。このオプションを指定すると、ローカルの日時へ、またはローカルの日時からの日時の変換をバイパスします。デフォルトでは、Oracle Solaris Cluster ソフトウェアはローカルの日時を表示します。

このオプションは print サブコマンドとだけ使用できます。

-v
--verbose

詳細情報を標準出力 (stdout) で表示します。

-V
--version

コマンドのバージョンを表示します。

このオプションをほかのオプション、サブコマンド、またはオペランドと一緒に指定する場合、これらはすべて無視されます。コマンドのバージョンだけが表示されます。ほかの処理は行われません。

次のオペランドがサポートされています。

telemetry-attribute
使用状況データが必要な特定のテレメトリ属性です。

Oracle Solaris Cluster ソフトウェアは、使用状況データを収集できるオブジェクトの特定の型を含んでいます。テレメトリ属性のモニタリングは、オブジェクト型ごとに有効にできます。Sun Cluster ソフトウェアは、有効な属性のデータだけを収集します。

+
すべてのテレメトリグループ。

このコマンドセットにあるすべてのコマンドの終了ステータスコードの完全なリストについては、19 ページのIntro(1CL) のマニュアルページを参照してください。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

0 CL_NOERR
エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

18 CL_EINTERNAL

内部エラーが発生しました

内部エラーは、ソフトウェアの欠陥またはその他の欠陥を示しています。

35 CL_EIO

I/O エラー

物理的な入出力エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

38 CL_EBUSY

オブジェクトはビジーです

アクティブなクラスタノードへの最後のクラスタインターコネクトパスからケーブルを取り外そうとしました。または、参照を削除していないクラスタ構成からノードを削除しようとしました。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

41 CL_ETYPE

未知のタイプ

-t または -p オプションで指定したタイプは存在しません。

例 237 オブジェクト型に対して構成されているシステム資源の表示

次のコマンドは、あるオブジェクト型、この場合はディスクに適用できるシステム資源を表示します。

```
# cltelemetryattribute list -t disk
rbyte.rate
wbyte.rate
write.rate
read.rate
```

例 238 オブジェクト型のテレメトリ属性の有効化

次のコマンドは、クラスタ内のすべてのディスク上の指定されているテレメトリ属性に対してデータ収集を有効にします。

```
# cltelemetryattribute enable -t disk rbyte.rate wbyte.rate
```

例 239 オブジェクト型のテレメトリ属性のしきい値の設定

次のコマンドは、クラスタ内のディスク d4 上のテレメトリ属性 wbyte.rate のしきい値を設定します。rearm のデフォルト値は、value の値に設定されています。結果として、ディスク d4 に書き込まれるバイト数が 100 を超えるかまたは 100 より少なくなると、Oracle Solaris Cluster ソフトウェアは fatal 通知を発行します。

```
# cltelemetryattribute set-threshold -t disk -b d4 \
-p severity=fatal,direction=rising,value=100 wbyte.rate
```

例 240 構成されているテレメトリ属性の詳細ではないリストの表示

次のコマンドは、クラスタ内のすべてのディスクに構成されているテレメトリ属性の詳細ではないリストを表示します。

```
# cltelemetryattribute show -t disk

=== Telemetry Attributes ===
```

```
Telemetry Attribute:                            read.rate
  Unit:                                            read/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            write.rate
  Unit:                                            writes/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            wbyte.rate
  Unit:                                            KBytes/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            rbyte.rate
  Unit:                                            KBytes/s
  Enabled Object Types:                            disk
```

例 241 テレメトリ属性の構成の詳細なリストの表示

次のコマンドは、クラスタ内のすべてのディスクに構成されているテレメトリ属性の詳細なリストを表示します。

```
# cltelemetryattribute show -v -t disk

=== Telemetry Attributes ===

Telemetry Attribute:                            read.rate
  Unit:                                            read/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            write.rate
  Unit:                                            writes/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            wbyte.rate
  Unit:                                            KBytes/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            rbyte.rate
  Unit:                                            KBytes/s
  Enabled Object Types:                            disk

% cltelemetryattribute show -v -t disk

=== Telemetry Attributes ===

Telemetry Attribute:                            read.rate
  Unit:                                            read/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            write.rate
  Unit:                                            writes/s
  Enabled Object Types:                            disk
```

```
  --- Object Instances of Type "disk" ---

  Object Instance:                              d4
    Thresholds:                                 <Direction, Severity, Value, Rearm>
      Threshold 1:                                 <rising, fatal, 1000, 500>

Telemetry Attribute:                            wbyte.rate
  Unit:                                            KBytes/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            rbyte.rate
  Unit:                                            KBytes/s
  Enabled Object Types:                            disk
```

例 242 テレメトリ属性のステータスの表示

次のコマンドは、クラスタ内のすべてのディスクに構成されているテレメトリ属性のステータスを表示します。

```
# cltelemetryattribute status

=== Telemetry Attributes Thresholds ===

Attribute   Obj-Instance   Obt-Type   Node      Threshold                    Status
Status
---------   ------------   --------   -------   --------------------------   -------
mem.used    phys-schost-1  node       16-v2-4   <rising, fatal, 1000, 1000>   warning
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

551 ページの`cluster`(1CL), 19 ページの`Intro`(1CL), 899 ページの`sctelemetry`(1M), Unresolved link to " su1M", Unresolved link to " attributes5", Unresolved link to " rbac5", 1431 ページの`SUNW.SCTelemetry`(5), 1447 ページの`clconfiguration`(5CL)

スーパーユーザーはこのコマンドのすべての形式を実行できます。

すべてのユーザーがこのコマンドに -? (ヘルプ) オプションまたは -V (バージョン) オプションを指定して実行できます。

cltelemetryattribute コマンドをほかのサブコマンドとともに実行する場合、スーパーユーザー以外のユーザーは RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
disable	solaris.cluster.modify
enable	solaris.cluster.modify
export	solaris.cluster.read
list	solaris.cluster.read
print	solaris.cluster.read
set-threshold	solaris.cluster.modify
show	solaris.cluster.read
status	solaris.cluster.read

名前

cltelemetryattribute, clta — システムリソースモニタリングの構成

```
/usr/cluster/bin/cltelemetryattribute -V

/usr/cluster/bin/cltelemetryattribute [subcommand] -?

/usr/cluster/bin/cltelemetryattribute subcommand [options] -v
     [telemetry-attribute …]

/usr/cluster/bin/cltelemetryattribute disable [-i
     {- | clconfigfile}] [-t object-type] {+ | telemetry-attribute ...}

/usr/cluster/bin/cltelemetryattribute enable [-i
     {- | clconfigfile}] [-t object-type] {+ | telemetry-attribute ...}

/usr/cluster/bin/cltelemetryattribute export [-o
     {- | clconfigfile}] [-t object-type[,…]]
     [+ | telemetry-attribute ...]

/usr/cluster/bin/cltelemetryattribute list [-t object-type[,…]]
     [+ | telemetry-attribute ...]

/usr/cluster/bin/cltelemetryattribute print [-b object-instance[,…]]
     [-a] [-d period] [-u] [-n node[,…]] [-t object-type[,…]]
     [+ | telemetry-attribute ...]

/usr/cluster/bin/cltelemetryattribute set-threshold -b
     object-instance [-n node] {-p name=value} [-p name=value]
     […] -t object-type telemetry-attribute

/usr/cluster/bin/cltelemetryattribute show [-b object-instance[,…]]
     [-n node[,…]] [-t object-type[,…]]
     [+ | telemetry-attribute ...]

/usr/cluster/bin/cltelemetryattribute status -b object-instance
     [-n node] {-p name} -t object-type [+ | telemetry-attribute ...]
```

このコマンドは、システムリソースのモニタリングを構成します。

システム資源の使用状況は、次に示す型をはじめとする異なる型のオブジェクト上でモニターできます。

- ディスク
- ファイルシステム
- IP アドレス
- ネットワークインタフェース
- ノード

- Solaris ゾーン
- リソースグループ

システム資源のモニター面は、テレメトリ属性と呼ばれます。

このコマンドは次のことを行います。

- テレメトリ属性を有効または無効にします。
- テレメトリ属性のしきい値を設定または変更します。
- モニターされている属性、適用されているしきい値、オブジェクトについて収集されるデータのリストを表示します。

対応するテレメトリ属性を特定することで、モニターするシステム資源使用状況の側面を選択します。あるオブジェクトでのシステム資源の使用状況をモニターするには、その型のオブジェクト上で対応するテレメトリ属性を有効にします。Oracle Solaris Cluster ソフトウェアは、クラスタ内にあるその型のすべてのオブジェクト上で指定された属性に対する使用状況データを収集します。

システム資源の場合、特定の値がクラスタの性能にとっての臨界値である場合があります。臨界値を過ぎた場合に通知を受けるため、テレメトリ属性のしきい値を設定できます。しきい値については、`set-threshold` サブコマンド、および `-p` オプションの説明を参照してください。

subcommand を省略できるのは、*options* が `-?` オプションまたは `-V` オプションの場合のみです。

このコマンドの各オプションには長形式と短形式があります。各オプションの両方の形式は、オプション に説明とともに記載されています。

`clta` コマンドは `cltelemetryattribute` コマンドの短い形式です。

システム資源のモニタリングの構成を詳細化する前にはモニタリングを初期化してください。899 ページの`sctelemetry`(1M) のマニュアルページを参照してください。

このコマンドは、大域ゾーンだけで使用できます。

サブコマンド

サポートされるサブコマンドには次のものがあります。

`disable`

指定されたオブジェクト型の指定されたテレメトリ属性を無効にします。

このサブコマンドは、大域ゾーンだけで使用できます。

Oracle Solaris Cluster ソフトウェアは、enabled 状態が設定されているシステムリソースの使用状況データを収集します。あるオブジェクト型のシステムリソースを disabled 状態に設定すると、Oracle Solaris Cluster ソフトウェアはそのオブジェクトインスタンスに対応するどのインスタンスのデータも収集しません。

また、cltelemetryattribute コマンドは、次に示す条件が両方満たされる場合、これらの属性のデータ収集を無効にします。

- 構成ファイルを -i オプションで指定する場合。
- 入力ファイルでテレメトリ属性に disabled が設定されている場合。

export サブコマンドを使用して構成ファイルを作成する場合。

テレメトリ属性に disabled を設定した場合、構成されているしきい値の設定は変更されません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

enable

指定されたオブジェクト型の指定されたテレメトリ属性のデータ収集を有効にします。

このサブコマンドは、大域ゾーンだけで使用できます。

デフォルトでは、選択されている属性が選択されているオブジェクト型に対して有効になっています。

テレメトリ属性のデータ収集を有効にするには、テレメトリ属性に enabled を設定します。

Oracle Solaris Cluster ソフトウェアは、そのオブジェクト型に対してテレメトリ属性が有効なオブジェクト型のデータだけを収集します。オブジェクト型のある属性を有効にすると、Oracle Solaris Cluster ソフトウェアは、すべてのノード上でその型のすべてのオブジェクトインスタンスの有効にされた属性についてのデータを収集します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.modify RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

export

オブジェクト型およびオブジェクトインスタンスのテレメトリ属性の構成をファイルまたは標準出力 stdout にエクスポートします。

このサブコマンドは、大域ゾーンだけで使用できます。

構成には、オブジェクト型に対して属性が有効なのか無効なのかが含まれます。また、構成にはしきい値を構成するために設定するリミットも含まれる場合があります。

-o オプションを使用してファイルを指定し、構成情報をファイルに書き込みます。-o オプションを指定しない場合、cltelemetryattribute コマンドは構成情報を標準出力 (stdout) に書き込みます。

export サブコマンドはクラスタ構成データを変更しません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`list`

指定されたオブジェクト型に対して構成できるテレメトリ属性を表示します。

このサブコマンドは、大域ゾーンだけで使用できます。

詳細オプション `-v` を指定した場合、`list` サブコマンドは、属性を適用できるオブジェクトの型を表示します。

しきい値のプロパティーは、次の書式で表示されます。

`Threshold:` *severity*`,` *direction*
`,` *value*`,` *rearm*

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`print`

指定されたオブジェクトインスタンスまたはオブジェクト型に対して有効な指定されているテレメトリ属性のシステム資源の使用状況を表示します。

このサブコマンドは、大域ゾーンだけで使用できます。

出力は次に示すデータを含みます。

- 日付とタイムスタンプ
- オブジェクトインスタンス
- オブジェクト型
- テレメトリ属性
- ノード
- 値

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`set-threshold`

ノード上の指定されたオブジェクトの指定されたテレメトリ属性のしきい値の設定を変更します。

このサブコマンドは、大域ゾーンだけで使用できます。

変更されるしきい値を指定するには、`-p` オプションを使用します。また、変更するしきい値のプロパティーを指定する場合も `-p` オプションを使用します。変更できるのは、`value` および `rearm` しきい値プロパティーのみです。

指定されたしきい値のこれらのプロパティーを 1 つ以上変更してください。1 つ以上のプロパティーが構成されている場合、`status` サブコマンドを実行すると出力が表示されます。

しきい値を非アクティブにするには、次のように `value` と `rearm` に空白を指定します。

```
-y value=,rearm=
```

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`show`

オブジェクト型またはオブジェクトインスタンスのテレメトリ属性に対して構成されているプロパティーを表示します。

このサブコマンドは、大域ゾーンだけで使用できます。

これらの属性は、システム資源がオブジェクト型に対して有効かどうかを含みます。詳細オプション `-v` を指定した場合、`show` サブコマンドは、オブジェクトインスタンスに対して有効なテレメトリ属性のしきい値の設定を表示します。

しきい値のプロパティーは、次の書式で表示されます。

`Threshold:` *severity*`,` *direction*
`,` *value*`,` *rearm*

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`status`

標準出力でしきい値が構成されているオブジェクト型の現在のステータスを表示します。最低 1 つのしきい値を設定しないと、`status` コマンドを実行したときに、出力は表示されません。

このサブコマンドは、大域ゾーンだけで使用できます。

警告または重大な重要度を現在持っている有効なすべてのしきい値のステータスを表示するには、このサブコマンドを引数なしで使用します。しきい値の考えられる出力は、しきい値の現在の重要度を含みます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

次のオプションがサポートされています。

`-?`
`--help`

ヘルプ情報を表示します。

このオプションは*サブコマンド*付きでもサブコマンドなしでも指定できます。

このオプションを *subcommand* なしで指定すると、使用可能なすべてのサブコマンドのリストが表示されます。

このオプションを *subcommand* とともに指定すると、*subcommand* の使用法が表示されます。

このオプションを `set-threshold` サブコマンドとともに指定すると、すべてのリソースグループプロパティーのヘルプ情報が表示されます。

このオプションをほかのオプション、サブコマンド、またはオペランドと一緒に指定する場合、これらはすべて無視されます。ほかの処理は行われません。

`-a`
`--average`

3 時間の期間に収集されたデータの平均およびこの平均に関連する標準偏差を出力します。

`-a` オプションと `-d` オプションを一緒に指定すると、指定した *period* の間、3 時間間隔で平均化されたデータが出力されます。

`-a` オプションを指定しない場合、出力されるデータは最新のデータです。

`-b` *object-instance*
`--object-instance=`*object-instance*
`--object-instance` *object-instance*

情報を表示するまたはしきい値を設定するオブジェクトインスタンスを指定します。

オブジェクトインスタンスは常に特定の型です。たとえば、クラスタノード `phys-schost-1` は `node` 型のオブジェクトインスタンスです。Oracle Solaris Cluster ソフトウェアは、対応するテレメトリ属性がオブジェクト型に対して有効な場合だけ、オブジェクトインスタンスのシステムリソースをモニターします。

`-d` *period*
`--date-range=`*period*
`--date-range` *period*

Oracle Solaris Cluster ソフトウェアがモニタリングデータを収集する期間を指定します。

period 引数に指定する日付と時間の形式は、International Organization for Standardization (ISO) 8601 の国際日付形式に準拠する必要があります。

begin-time, *end-time*] 期間は、コンマ , で区切られた 2 つの時間の間の時間です。

begin-time+ 期間は、指定された開始時間と現在の時間の間の時間です。

end-time- 期間は、Oracle Solaris Cluster ソフトウェアが起動してデータの収集を開始する時間と指定された終了時間の間の時間です。

period の形式の例を次に示します。

```
-d 2006-04-30T18:00,2006-06-16T18:00
```

2006 年 4 月 30 日の午後 6:00 から 2006 年 6 月 16 日の午後 6:00 まで

`-d 2006-06-16+`

2006 年 6 月 16 日の夜中 12:00 以降

`-d 2006-07-31T18:00+`

2006 年 7 月 31 日の午後 6:00 以降

`-d 2006-06-16T18:00-`

Oracle Solaris Cluster ソフトウェアが起動してから 2006 年 6 月 16 日の午後 6:00 まで

`-d 2006-05-31T12:00,2006-06-16T11:59`

2006 年 5月 31 日の夜中 12:00 から 2006 年 6 月 16 日の午後 11:59 まで

このオプションは `print` サブコマンドとだけ使用できます。

`-i` {- | *clconfigfile*}
`--input=`{- | *clconfigfile*}
`--input` {- | *clconfigfile*}

clconfigfile ファイル内にある構成情報を使用して、テレメトリ属性としきい値構成を指定することを指定します。1447 ページの`clconfiguration`(5CL) のマニュアルページを参照してください。

標準入力 stdin を通して構成情報を提供するには、このオプションとともにダッシュ `-` を指定します。ほかのオプションを指定した場合は、*clconfigfile* 内のオプションや情報より指定したオプションが優先されます。

`-n` *node*
`--node=`*node*
`--node` *node*

Oracle Solaris Cluster が使用状況データを収集するノード名を指定します。名前またはノード識別子を指定できます。

`-node` 型のオブジェクトインスタンス、リソース、またはリソースグループでサブコマンドを指定する際には、`n` オプションを指定しないでください。

`-o` {- | *clconfigfile*}
`--output=`{- | *clconfigfile*}
`--output` {- | *clconfigfile*}

テレメトリ属性としきい値の構成データをファイルまたは標準出力 `stdout` に書き込みます。この構成情報の形式は、1447 ページの`clconfiguration`(5CL) のマニュアルページで説明されています。

このオプションにファイル名を指定する場合、このオプションは新しいファイルを作成します。次に、ノード構成情報はそのファイルに格納されます。`-` をこのオプションとともに指定すると、構成情報は標準出力 (`stdout`) に送信されます。このコマンドのほかの標準出力はすべて抑制されます。

このオプションを一緒に指定できるのは、`export` サブコマンドだけです。

-p *name*
--property=*name*
--property *name*

status サブコマンドのプロパティーのリストを指定します。

set-threshold サブコマンドでしきい値を設定できるプロパティーについては、-p *name*=*value* オプションの説明を参照してください。

-p *name*=*value*
--property=*name*=*value*
--property *name value*

しきい値のプロパティーを指定します。

-p *name*= *value* は複数回指定できます。

status サブコマンドで情報を表示できるプロパティーについては、-p *name* オプションの説明を参照してください。

しきい値ごとに severity および direction プロパティーを指定して、しきい値を特定してください。しきい値の設定後は、これらのプロパティーを変更できません。

各しきい値の value を設定します。各しきい値の rearm も設定できます。value および rearm プロパティーを変更するには、set-threshold サブコマンドを使用します。このオプションで指定できるプロパティーと値は次のとおりです。

severity

しきい値の重要度です。このプロパティーに設定できる値は、fatal と warning です。重要度が fatal のしきい値は、重要度が warning のしきい値より重大です。

重要度は、Oracle Solaris Cluster Manager で視覚的なアラームとして表示されます。

direction

適用されるしきい値の方向です。このプロパティーに設定できる値は、falling と rising です。direction プロパティーに falling を設定することで、fatal 重要度レベルが warning 重要度レベルより小さい値を持つことを指定します。direction プロパティーに rising を設定することで、fatal 重要度レベルが warning 重要度レベルより大きい値を持つことを指定します。

value

テレメトリ属性のしきい値に設定する値です。このしきい値を越えると、テレメトリ属性の重要度が変わります。1 つのオブジェクトの特定のテレメトリ属性には最大 4 つのしきい値を関連付けられます。

value プロパティーを設定または変更するには、set-threshold サブコマンドを使用します。

`rearm`

テレメトリ属性の重要度を解除する手段です。rearm の値を指定することで、テレメトリ属性の値が `direction` プロパティーに設定されているのと逆の方向で rearm の値を越えると、テレメトリ属性の重要度が解除されます。rearm の値を指定しないと、rearm の値は、しきい値と rearm の値に同じ値が設定されているのと同じになります。

通知の頻度は *hysteresis* の原則に従います。つまり、頻度は二重値機能により決定されます。関数が増えると 1 つの値が適用されます。関数が値と同じ場合は、もう一方の値が適用されます。

使用中のシステムに合った `rearm` および `value` の値を設定してください。オプションの `rearm` プロパティーを指定しないと、このプロパティーはデフォルトとして `value` を採用します。ただし、`rearm` プロパティーに `value` プロパティーと同じ値を設定した場合、または `rearm` に値を割り当てない場合は、モニター対象のテレメトリ属性の値が `value` に対して設定されている値以上または値以下になるたびに通知を受信します。大量の通知の受信を避けるには、`rearm` に `value` 以外の値を設定してください。

`rearm` を `set-threshold` サブコマンドとともに指定すると、`cltelemetryattribute` コマンドにより `rearm` の値が次の要件を満たすことが保証されます:

- `direction` が `rising` の場合、`value` は `rearm` 以上の値を持ちます。
- `direction` が `falling` の場合、`value` は `value` 以下の値を持ちます。

`rearm` を変更するには、`set-threshold` サブコマンドを使用します。

`-t` *object-type*
`--object-type=`*object-type*
`--object-type` *object-type*

Oracle Solaris Cluster ソフトウェアが使用状況データを収集するオブジェクトの型を指定します。すべてのオブジェクトインスタンスは特定の型です。

指定された型のオブジェクトにサブコマンドの出力を制限するには、このオプションを使用します。

システム資源をモニターできるオブジェクト型と各オブジェクト型の関連テレメトリ属性は、次のとおりです。

オブジェクトタイプ	説明	テレメトリ属性
`disk`	ディスク	`rbyte.rate`、`wbyte.rate`、`read.rate`、`write.rate`、
`filesystem`	ファイルシステム	`block.used`、`inode.used`
`ipaddr`	IP アドレス	`ipacket.rate`、`opacket.rate`
`netif`	ネットワークインタフェース	`ipacket.rate`、`opacket.rate`、`rbyte.rate`、`wbyte.rate`

オブジェクトタイプ	説明	テレメトリ属性
node	ノード	cpu.idle、cpu.iowait、cpu.used、loadavg.1mn、loadavg.5mn、loadavg.15mn、mem.used、mem.free、swap.used、swap.free
resourcegroup	リソースグループ	cpu.used、mem.used、swap.used
zone	ゾーン	cpu.idle、cpu.iowait、cpu.used、loadavg.1mn、loadavg.5mn、loadavg.15mn

モニターできるテレメトリ属性は次のとおりです。

テレメトリ属性	説明
block.used	デバイス上で使用されているブロックのパーセンテージ
cpu.idle	空き CPU の容量
cpu.iowait	入出力の完了を待っている CPU の容量
cpu.used	使用されている CPU の容量
inode.used	デバイス上で使用されている i ノードのパーセンテージ
ipacket.rate	1 秒間の着信パケット数
loadavg.1mn	最後の 1 分間に CPU を待っていたプロセスの数
loadavg.5mn	最後の 5 分間に CPU を待っていたプロセスの数
loadavg.15mn	最後の 15 分間に CPU を待っていたプロセスの数
mem.free	空きメモリの M バイト数
mem.used	使用されているメモリの M バイト数
opacket.rate	1 秒間の送信パケット数
rbyte.rate	1 秒間に読み取られる M ビット数
read.rate	秒当たりの読み取り動作数
swap.free	空きスワップメモリの M バイト数
swap.used	使用されているスワップメモリの M バイト数
wbyte.rate	1 秒間に書き込まれる M ビット数
write.rate	秒当たりの書き込み動作数

すべてのオブジェクト型に対して上記の表に一覧表示されているすべてのテレメトリ属性をモニターできません。データを収集できるオブジェクト型とオブジェクトの型ごとにモニターできるテレメトリ属性を表示するには、`list` サブコマンドを使用します。

`-u`
`--utc`

使用状況データとともに示される日時を協定世界時 (UTC) またはグリニッジ標準時 (GMT) で表示します。このオプションを指定すると、ローカルの日時へ、またはローカルの日時からの日時の変換をバイパスします。デフォルトでは、Oracle Solaris Cluster ソフトウェアはローカルの日時を表示します。

このオプションは `print` サブコマンドとだけ使用できます。

`-v`
`--verbose`

詳細情報を標準出力 (`stdout`) で表示します。

`-V`
`--version`

コマンドのバージョンを表示します。

このオプションをほかのオプション、サブコマンド、またはオペランドと一緒に指定する場合、これらはすべて無視されます。コマンドのバージョンだけが表示されます。ほかの処理は行われません。

次のオペランドがサポートされています。

telemetry-attribute
使用状況データが必要な特定のテレメトリ属性です。

Oracle Solaris Cluster ソフトウェアは、使用状況データを収集できるオブジェクトの特定の型を含んでいます。テレメトリ属性のモニタリングは、オブジェクト型ごとに有効にできます。Sun Cluster ソフトウェアは、有効な属性のデータだけを収集します。

`+`
すべてのテレメトリグループ。

このコマンドセットにあるすべてのコマンドの終了ステータスコードの完全なリストについては、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (`CL_NOERR`) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

`0 CL_NOERR`
エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

18 CL_EINTERNAL

内部エラーが発生しました

内部エラーは、ソフトウェアの欠陥またはその他の欠陥を示しています。

35 CL_EIO

I/O エラー

物理的な入出力エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

38 CL_EBUSY

オブジェクトはビジーです

アクティブなクラスタノードへの最後のクラスタインターコネクトパスからケーブルを取り外そうとしました。または、参照を削除していないクラスタ構成からノードを削除しようとしました。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

41 CL_ETYPE

未知のタイプ

-t または -p オプションで指定したタイプは存在しません。

例 243 オブジェクト型に対して構成されているシステム資源の表示

次のコマンドは、あるオブジェクト型、この場合はディスクに適用できるシステム資源を表示します。

```
# cltelemetryattribute list -t disk
rbyte.rate
wbyte.rate
write.rate
read.rate
```

例 244 オブジェクト型のテレメトリ属性の有効化

次のコマンドは、クラスタ内のすべてのディスク上の指定されているテレメトリ属性に対してデータ収集を有効にします。

```
# cltelemetryattribute enable -t disk rbyte.rate wbyte.rate
```

例 245 オブジェクト型のテレメトリ属性のしきい値の設定

次のコマンドは、クラスタ内のディスク d4 上のテレメトリ属性 wbyte.rate のしきい値を設定します。rearm のデフォルト値は、value の値に設定されています。結果として、ディスク d4 に書き込まれるバイト数が 100 を超えるかまたは 100 より少なくなると、Oracle Solaris Cluster ソフトウェアは fatal 通知を発行します。

```
# cltelemetryattribute set-threshold -t disk -b d4 \
-p severity=fatal,direction=rising,value=100 wbyte.rate
```

例 246 構成されているテレメトリ属性の詳細ではないリストの表示

次のコマンドは、クラスタ内のすべてのディスクに構成されているテレメトリ属性の詳細ではないリストを表示します。

```
# cltelemetryattribute show -t disk

=== Telemetry Attributes ===
```

```
Telemetry Attribute:                            read.rate
  Unit:                                            read/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            write.rate
  Unit:                                            writes/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            wbyte.rate
  Unit:                                            KBytes/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            rbyte.rate
  Unit:                                            KBytes/s
  Enabled Object Types:                            disk
```

例 247 テレメトリ属性の構成の詳細なリストの表示

次のコマンドは、クラスタ内のすべてのディスクに構成されているテレメトリ属性の詳細なリストを表示します。

```
# cltelemetryattribute show -v -t disk

=== Telemetry Attributes ===

Telemetry Attribute:                            read.rate
  Unit:                                            read/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            write.rate
  Unit:                                            writes/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            wbyte.rate
  Unit:                                            KBytes/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            rbyte.rate
  Unit:                                            KBytes/s
  Enabled Object Types:                            disk

% cltelemetryattribute show -v -t disk

=== Telemetry Attributes ===

Telemetry Attribute:                            read.rate
  Unit:                                            read/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            write.rate
  Unit:                                            writes/s
  Enabled Object Types:                            disk
```

```
  --- Object Instances of Type "disk" ---

  Object Instance:                              d4
    Thresholds:                                 <Direction, Severity, Value, Rearm>
      Threshold 1:                                 <rising, fatal, 1000, 500>

Telemetry Attribute:                            wbyte.rate
  Unit:                                            KBytes/s
  Enabled Object Types:                            disk

Telemetry Attribute:                            rbyte.rate
  Unit:                                            KBytes/s
  Enabled Object Types:                            disk
```

例 248 テレメトリ属性のステータスの表示

次のコマンドは、クラスタ内のすべてのディスクに構成されているテレメトリ属性のステータスを表示します。

```
# cltelemetryattribute status

=== Telemetry Attributes Thresholds ===

Attribute   Obj-Instance   Obt-Type   Node      Threshold                    Status
Status
---------   ------------   --------   -------   --------------------------   -------
mem.used    phys-schost-1  node       16-v2-4   <rising, fatal, 1000, 1000>   warning
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

551 ページの`cluster`(1CL), 19 ページの`Intro`(1CL), 899 ページの`sctelemetry`(1M), Unresolved link to " su1M", Unresolved link to " attributes5", Unresolved link to " rbac5", 1431 ページの`SUNW.SCTelemetry`(5), 1447 ページの`clconfiguration`(5CL)

スーパーユーザーはこのコマンドのすべての形式を実行できます。

すべてのユーザーがこのコマンドに -? (ヘルプ) オプションまたは -V (バージョン) オプションを指定して実行できます。

cltelemetryattribute コマンドをほかのサブコマンドとともに実行する場合、スーパーユーザー以外のユーザーは RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
disable	solaris.cluster.modify
enable	solaris.cluster.modify
export	solaris.cluster.read
list	solaris.cluster.read
print	solaris.cluster.read
set-threshold	solaris.cluster.modify
show	solaris.cluster.read
status	solaris.cluster.read

名前

cluster — クラスタのグローバルな構成とステータスの管理

```
/usr/cluster/bin/cluster -V

/usr/cluster/bin/cluster [subcommand] -?

/usr/cluster/bin/cluster subcommand
     [options] -v [clustername …]

/usr/cluster/bin/cluster check
     [-F] [-C checkid[,…]]|-E checkid[,…]]
     [-e explorerpath[,…]] [-j jarpath[,…]]
     [-k keyword[,…]] [-n node[,…]] [-o outputdir]
     [-s severitylevel] [clustername]

/usr/cluster/bin/cluster create -i {- | clconfigfile}
     [clustername]

/usr/cluster/bin/cluster export [-o {- | configfile}]
     [-t objecttype[,…]] [clustername]

/usr/cluster/bin/cluster monitor-heartbeat [-v] [clustername]

/usr/cluster/bin/cluster list [clustername]

/usr/cluster/bin/cluster list-checks [-F] [-K]
     [-C checkid[,…]|-E checkid[,…]] [-j jar-path[,…]]
     [-o outputdir] [clustername]

/usr/cluster/bin/cluster list-cmds [clustername]

/usr/cluster/bin/cluster rename -c newclustername [clustername]

/usr/cluster/bin/cluster restore-netprops [clustername]

/usr/cluster/bin/cluster set {-p name=value} [-p name=value] […]
     [clustername]

/usr/cluster/bin/cluster set-netprops {-p name=value}
     [-p name=value] […] [clustername]

/usr/cluster/bin/cluster show [-t objecttype[,…]] [clustername]

/usr/cluster/bin/cluster show-netprops [clustername]

/usr/cluster/bin/cluster shutdown [-y] [-g graceperiod]
     [-m message] [clustername]

/usr/cluster/bin/cluster status [-t objecttype[,…]] [clustername]
```

`cluster` コマンドは、クラスタ全体の構成およびステータス情報を表示および管理します。また、このコマンドはグローバルクラスタをシャットダウンします。

次の cluster サブコマンドはゾーンクラスタ内で動作します。

- cluster show - ゾーンクラスタ、ノード、リソースグループ、リソースタイプおよびリソースプロパティを一覧表示します。
- cluster status - ゾーンクラスタコンポーネントのステータスを表示します。
- cluster shutdown - ゾーンクラスタを正常に終了します。
- cluster list - ゾーンクラスタの名前を表示します。
- cluster list-cmds - ゾーンクラスタ内でサポートされている、次のコマンドを一覧表示します:
  - clnode
  - clreslogicalhostname
  - clresource
  - clresourcegroup
  - clresourcetype
  - clressharedaddress
  - cluster

cluster コマンドとともに使用するほぼすべてのサブコマンドは、クラスタモードで動作します。これらのサブコマンドは、クラスタ内のどのノードからでも実行できます。ただし、create、set-netprops、および restore-netprops サブコマンドは例外です。これらのコマンドは非クラスタモードで実行してください。

subcommand を省略できるのは、*options* が -? オプションまたは -V オプションの場合のみです。

cluster コマンドに短形式はありません。

このコマンドの各オプションには長形式と短形式があります。各オプションの両方の形式は、オプション に説明とともに記載されています。

このコマンドは大域ゾーンで使用します。

サポートされるサブコマンドには次のものがあります。

check
: クラスタが正しく構成されているかどうかをチェックおよび報告します。

このサブコマンドは、大域ゾーンだけで使用できます。

このサブコマンドには、基本検査、対話型検査、機能検査の 3 つのモードがあります。

- 基本検査は、キーワード `-k interactive` または `-k functional` が指定されていない場合に実行されます。基本検査では、特定の構成情報を読み取り、評価して、起こり得るエラーや満たされていない要件を特定します。
- 対話型検査は、`-k interactive` オプションで指定されます。`-C`または`-E` オプションが指定されていない場合、使用可能な対話型検査がすべて実行されます。

  対話型検査は基本検査と似ていますが、検査で確認できない情報の入力をユーザーに要求します。たとえば、ファームウェアバージョンを指定するようユーザーに指示する場合があります。対話型検査によってクラスタの機能が中断されることはありません。
- 機能検査は、`-k functional -C` *checkid* オプションで指定されます。`-k functional` オプションには、`-C` オプションと機能検査の検査 ID を 1 つだけ指定する必要があります。`-E` オプションは、`-k functional` オプションに対しては有効ではありません。

  機能検査は、フェイルオーバーのトリガーやノードのパニックなど、クラスタ構成の特殊な機能または動作を行います。このような検査では、ユーザーが、フェイルオーバーをどのノードに対して行うかといった特定のクラスタ構成情報を入力したり、検査を開始または継続するかどうか確認したりする必要があります。

  一部の機能検査ではクラスタサービスの中断が必要になるので、検査の詳細説明を読み、最初にクラスタの稼働を停止するかどうか判断したうえで、機能検査を開始してください。機能検査の完全な説明を表示するには、`cluster list-checks -v -C` *checkID* コマンドを使用します。

実行中のクラスタのアクティブメンバーから実行された場合、このサブコマンドは構成検査を行います。この検査では、クラスタを正常に実行するために必要な最小要件を満たしているかを確認します。

アクティブなクラスタメンバーとして稼動していないノードから実行された場合、このサブコマンドはこのノードに対してプリインストール検査を行います。これらの検査では、クラスタでインストールの準備を行い、予測される可用性の損失を防止するために、修正の必要がある脆弱性を特定します。

構成検査を行うたびに、指定された出力ディレクトリまたはデフォルトの出力ディレクトリに一連のレポートが生成されます。各レポートには、実行した検査総数と失敗数を重要度別に示したサマリーが含まれます。

各レポートは、通常のテキスト形式と XML 形式の両方で作成されます。XML 形式の DTD は `/usr/cluster/lib/cfgchk/checkresults.dtd` ファイルに記述されています。レポートは英文のみで生成されます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` 役割に基づくアクセス制御 (RBAC) 認証が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`create`

clconfigfile ファイルに格納されている構成情報を使用して新しいクラスタを作成します。この構成情報の形式は、1447 ページの`clconfiguration`(5CL) のマニュアルページで説明されています。

このサブコマンドは、大域ゾーンだけで使用できます。

このサブコマンドは非クラスタモードで実行してください。また、このサブコマンドはクラスタの一部としてまだ構成されていないホストから実行してください。Oracle Solaris Cluster ソフトウェアは、クラスタの一部となる各ノードに事前にインストールしておいてください。

クラスタ名を指定しないと、クラスタの名前は *clconfigfile* ファイルから取り込まれます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`export`

構成情報をエクスポートします。

このサブコマンドは、大域ゾーンだけで使用できます。

`-o` オプションでファイルを指定すると、そのファイルに構成情報が書き込まれます。`-o` オプションを指定しないと、出力は標準出力 `stdout` に書き込まれます。

次のオプションは、エクスポートされる情報を制限します。

`-t` *objecttype*[,…]

指定された種類のコンポーネントの構成情報だけをエクスポートします。

`cluster` コマンドを実行するクラスタの構成情報だけをエクスポートできます。`cluster` コマンドを実行するクラスタ以外のクラスタの名前を指定すると、このサブコマンドは失敗します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`list`

クラスタの名前を表示します。

このサブコマンドは、大域ゾーンまたはゾーンクラスタで使用できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`list-checks`

検査 ID と可能な各検査の説明のリストを表示します。

このコマンドは、大域ゾーンだけで使用できます。

検査 ID の最初の文字は、検査の種類を示しています。

F　　機能検査

I　　対話型検査

M　　複数のノードに対する基本検査

S　　単一のノードに対する基本検査

-v オプションは、検査のキーワードを含む、検査の操作を詳細に表示します。その検査を実行する前にクラスタの稼動を停止するかどうか判断するためには、機能検査の詳細な説明を表示することが重要になります。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`list-cmds`

使用可能なすべての Oracle Solaris Cluster コマンドのリストを出力します。

このサブコマンドは、大域ゾーンまたはゾーンクラスタで使用できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`monitor-heartbeat`

動的再構成 (DR) 中に、クラスタノードのハートビートのタイムアウトモニタリングを手動で再有効化します。

このサブコマンドは、大域ゾーンだけで使用できます。`monitor-heartbeat` サブコマンドは、排他的 IP ゾーンクラスタではサポートされません。

CPU またはメモリーボードで DR 操作を実行すると、影響を受けるノードが応答しなくなるため、そのノードのハートビートモニタリングが、ほかのすべてのノードで中断されます。DR が完了すると、影響を受けるノードのハートビートモニタリングが自動的に再有効化されます。DR 処理が完了しない場合は、`monitor-heartbeat` サブコマンドを使用して、ハートビートモニタリングを手動で再有効化する必要が生じる場合があります。影響を受けるノードがクラスタに再結合できない場合は、クラスタメンバシップから除外されます。

ハートビートのタイムアウトモニタリングの再有効化の手順については、Unresolved link to " Oracle Solaris Cluster 4.2 Hardware Administration Manual のKernel Cage Dynamic Reconfiguration Recovery"を参照してください。DR に関する一般的な情報については、Unresolved link to " Oracle Solaris Cluster Concepts Guide のDynamic Reconfiguration Support"を参照してください。

`rename`

クラスタの名前を変更します。

このコマンドは、大域ゾーンだけで使用できます。

クラスタに新しい名前を指定するには、このサブコマンドとともに -c オプションを指定してください。

注記 - クラスタが有効な Oracle Solaris Cluster Geographic Edition パートナーシップの一部として構成されている場合は、Unresolved link to " Oracle Solaris Cluster Geographic Edition System Administration Guide のRenaming a Cluster That Is in a Partnership"を参照してください。このセクションでは、Oracle Solaris Cluster Geographic Edition パートナーシップの一部として構成されたクラスタの名前を正しく変更する方法について説明します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`restore-netprops`

クラスタのクラスタプライベートネットワーク設定をリセットします。

このサブコマンドは、大域ゾーンだけで使用できます。このサブコマンドは非クラスタモードで実行してください。

このサブコマンドは、`set-netprops` サブコマンドが失敗し、次に示す条件が存在する場合だけに使用してください。

- プライベートネットワークのプロパティーを変更しようとしている場合。
- 失敗がノード上のクラスタ構成の矛盾を示している場合。この状況では、`restore-netprops` サブコマンドを実行する必要があります。

このサブコマンドは、クラスタ内のすべてのコマンドで実行してください。このサブコマンドでクラスタ構成を修復します。また、このサブコマンドは、IP アドレス範囲の変更が失敗したことが原因で発生する矛盾を削除します。失敗した場合、構成設定を変更するために行うあらゆる試みは、機能するかどうか保証されません。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`set`

クラスタのプロパティーを変更します。

このサブコマンドは、大域ゾーンだけで使用できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`set-netprops`

プライベートネットワークのプロパティーを変更します。

このサブコマンドは、大域ゾーンだけで使用できます。

このサブコマンドは非クラスタモードで実行してください。ただし、`num_zoneclusters` プロパティーを設定している場合は、このサブコマンドをクラスタモードで実行することもできます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`show`

クラスタコンポーネントについての詳細な構成情報を表示します。

このサブコマンドは、大域ゾーンだけで使用できます。

次のオプションは、表示される情報を制限します。

`-t` *objecttype*[,…]

指定された種類のコンポーネントの構成情報だけを表示します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`show-netprops`

クラスタのプライベートネットワークのプロパティーについての情報を表示します。

このサブコマンドは、大域ゾーンだけで使用できます。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 `solaris.cluster.read` が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`shutdown`

グローバルクラスタを通常の方式でシャットダウンします。

このサブコマンドは、大域ゾーンだけで使用できます。

このサブコマンドをグローバルクラスタで発行した場合は、Oracle Solaris Cluster ソフトウェアによって、そのグローバルクラスタに関連付けられたすべてのゾーンクラスタを含むグローバルクラスタ全体がシャットダウンされます。`cluster` コマンドをゾーンクラスタで使用できません。

`cluster` コマンドを実行するクラスタ以外のクラスタの名前を指定すると、このサブコマンドは失敗します。

このサブコマンドはクラスタ内の 1 つのノードだけから実行してください。

このサブコマンドは、次のアクションを実行します。

- クラスタ内の機能中のすべてのリソースグループをオフラインにします。移行できないものあった場合、このサブコマンドは処理は続行せず、エラーメッセージを表示します。
- すべてのクラスタファイルシステムをアンマウントします。アンマウント処理できないものあった場合、このサブコマンドは処理は続行せず、エラーメッセージを表示します。
- アクティブなすべてのデバイスサービスを停止させます。1 つでもデバイスの移行ができないものあった場合、このサブコマンドは処理は続行せず、エラーメッセージを表示します。

■ クラスタ内のすべてのノードを停止させます。

このサブコマンドは、クラスタのシャットダウンを開始する前に、すべてのノード上で警告メッセージを発生させます。警告を発生させたあと、このサブコマンドは、クラスタのシャットダウンを確認するための最終メッセージを発生させます。この最終メッセージを発生させないようにするには、-y オプションを使用します。

デフォルトでは、shutdown サブコマンドは 60 秒待ってから、クラスタをシャットダウンします。-g オプションを使用すると、別の遅延時間を指定できます。

警告とともに表示されるメッセージ文字列を指定するには、-m オプションを使用します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`status`

クラスタコンポーネントのステータスを表示します。

このサブコマンドは、大域ゾーンまたはゾーンクラスタで使用できます。

オプション -t *objecttype*[,…] は、指定された種類のコンポーネントのステータス情報だけを表示します。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、RBAC の承認 solaris.cluster.read が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

次のオプションがサポートされています。

注記 - このセクションでは、各オプションの短形式と長形式の両方が示されています。

`-?`
`--help`

ヘルプ情報を表示します。

このオプションは*サブコマンド*付きでもサブコマンドなしでも指定できます。

subcommand を指定しない場合、使用可能なすべてのサブコマンドのリストが表示されます。

subcommand を指定する場合、そのサブコマンドの使用法が表示されます。

このオプションとその他のオプションを指定すると、その他のオプションは無視されます。

`-C` *checkid*[,…]
`--checkID=`*checkid*[,…]
`--checkID` *checkid*[,…]

実行する検査を指定します。指定されていない検査は実行されません。-E オプションと -C オプションの両方が指定されている場合、-C オプションは無視されます。

キーワード -k functional の場合、-C オプションは必須で、実行する *checkid* を 1 つだけ指定する必要があります。

このオプションを一緒に指定できるのは、check および list-checks サブコマンドだけです。

-c *newclustername*
--newclustername=*newclustername*
--newclustername *newclustername*

クラスタの新しい名前を指定します。

クラスタの名前を変更するには、このオプションを rename サブコマンドとともに使用します。

-E *checkid*[,…]
--excludeCheckID=*checkid*[,…]
--excludeCheckID *checkid*[,…]

除外する検査を指定します。指定された検査以外のすべての検査が実行されます。-C オプションと -E オプションの両方が指定されている場合、-C オプションは無視されます。

-E オプションは、-k functional キーワードに対しては有効ではありません。

このオプションを一緒に指定できるのは、check および list-checks サブコマンドだけです。

-e *explorerpath*[,…]
--explorer=*explorerpath*[,…]
--explorer *explorerpath*[,…]

システムの別のデータソースとして使用する Oracle Explorer または Sun Explorer アーカイブの展開済みパスを指定します。*explorerpath* の値には、該当先の絶対パスを指定する必要があります。

このオプションを一緒に指定できるのは、check サブコマンドだけです。

-F
--force

/var/cluster/logs/cluster_check/cfgchk.lck ファイルが存在する場合、それを無視してサブコマンドを強制的に実行します。check および list-checks サブコマンドがすでに実行されていないときのみこのオプションを使用します。

-g *graceperiod*
--graceperiod=*graceperiod*
--graceperiod *graceperiod*

クラスタがシャットダウンされる前の時間の長さをデフォルト設定の 60 秒から変更します。

graceperiod は秒単位で指定します。

-i {- | *clconfigfile*}
--input={- | *clconfigfile*}
--input {- | *clconfigfile*}

clconfigfile ファイル内の構成情報を使用します。1447 ページのclconfiguration(5CL) のマニュアルページを参照してください。

標準入力 (stdin) を通して構成情報を提供するには、ダッシュ (-) をこのオプションとともに指定します。

ほかのオプションを指定すると、クラスタ構成ファイル内のオプションや情報より指定したオプションが優先されます。

-j *jarpath*[,…]
--jar=*jarpath*[,…]
--jar *jarpath*[,…]

検査が記述された追加の jar ファイルのパスを指定します。*jarpath* には、絶対パスを指定する必要があります。

このオプションを一緒に指定できるのは、check および list-checks サブコマンドだけです。

-K *keyword*[,…]
--list-keywords=*keyword*
--keyword *keyword*

使用可能な検査のすべてのキーワードを一覧表示します。このオプションは他のあらゆるオプションより優先されます。

このオプションを一緒に指定できるのは、list-checks サブコマンドだけです。

-k *keyword*[,…]
--keyword=*keyword*
--keyword *keyword*

指定されたキーワードを含む検査だけを実行します。cluster list-checks -k コマンドを使用して、使用可能な検査に割り当てられたキーワードを調べます。

-k functional キーワードには、-C オプションと単一の *checkid* が必須です。複数の機能検査を同時に指定したり、同じコマンド内で他のキーワードを指定したりすることはできません。

このオプションを一緒に指定できるのは、check および list-checks サブコマンドだけです。

-m *message*
--message=*message*
--message *message*

shutdown サブコマンドを実行すると表示される警告とともに表示するメッセージの文字列を指定します。

標準的な警告メッセージは「system will be shut down in …」です。

message が複数の単語を含む場合は、単一引用符 (') または二重引用符 (") で区切ってください。`shutdown` コマンドは、シャットダウン開始の 7200、3600、1800、1200、600、300、120、60、および 30 秒前にメッセージを発生させます。

`-n` *node*[,…]
`--node=`*node*[,…]
`--node` *node*[,…]

指定されたノードまたはノードリストだけに対して検査を実行します。*node* には、値としてノード名またはノード ID 番号を指定できます。

このオプションを一緒に指定できるのは、`check` サブコマンドだけです。

`-o` {- | *clconfigfile*}
`--output=`{- | *clconfigfile*}
`--output` {- | *clconfigfile* }

クラスタ構成情報をファイルまたは標準出力 (`stdout`) に書き込みます。この構成情報の形式は、1447 ページの`clconfiguration`(5CL) のマニュアルページで説明されています。

このオプションにファイル名を指定する場合、このオプションは新しいファイルを作成します。次に、ノード構成情報はそのファイルに格納されます。- をこのオプションとともに指定すると、構成情報は標準出力 (`stdout`) に送信されます。このコマンドのほかの標準出力はすべて抑制されます。

この形式の `-o` オプションを一緒に指定できるのは、`export` サブコマンドだけです。

`-o` *outputdir*
`--output=`*outputdir*
`--output` *outputdir*

`check` サブコマンドが生成するレポートを保存するディレクトリを指定します。

この形式の `-o` オプションを一緒に指定できるのは、`check` サブコマンドおよび `list-checks` サブコマンドだけです。

出力ディレクトリ *outputdir* は、すでに存在しているか、作成できる必要があります。*outputdir* にある過去のレポートは、新しいレポートによってすべて上書きされます。

`-o` オプションを指定しない場合、デフォルトでディレクトリ `/var/cluster/logs/cluster_check/` *datestamp*`/` が *outputdir* として使用されます。

`-p` *name*=*value*
`--property=`*name*=*value*
`--property` *name*=*value*

クラスタ全体のプロパティーを変更します。

`-p` *name*= *value* は複数回指定できます。

次のプロパティーを変更するには、このオプションを `set` および `set-netprops` サブコマンドで使用します。

`concentrate_load`

Resource Group Manager (RGM) が使用可能なノードにリソースグループの負荷を分散する方法を指定します。`concentrate_load` プロパティーはグローバルクラスタでのみ設定できます。ゾーンクラスタでは、`concentrate_load` プロパティーには *FALSE* のデフォルト値があります。 値が `FALSE` に設定されている場合、RGM はリソースグループのノードリストで使用可能なすべてのノードまたはゾーンに対して、リソースグループ負荷を均等に分散しようと試みます。グローバルクラスタで値が `TRUE` に設定されている場合は、構成されている強い制限値または弱い制限値を超えずに、できるだけ少ないノードまたはゾーンにリソースグループ負荷が集約されます。デフォルト値は `FALSE` です。

リソースグループ `RG2` がリソースグループ `RG1` に対して `++` または `+++` アフィニティーを宣言している場合は、`RG2` にゼロ以外の負荷係数を設定しないようにしてください。代わりに、`RG1` と同じノードでオンラインになる `RG2` による追加の負荷を考慮して、より大きな負荷係数を `RG1` に設定します。これにより、`Concentrate_load` 機能が目的どおりに動作します。`RG2` に負荷係数を設定できますが、これらの負荷係数に強い制限値を設定しないようにし、弱い制限値のみを設定することもできます。これにより、弱い負荷制限を超えた場合であっても、`RG2` をオンラインにできます。

各ノードの強い負荷制限値と弱い負荷制限値の作成と変更には、`clnode create-loadlimit`、`clnode set-loadlimit`、および `clnode delete-loadlimit` コマンドを使用します。手順については、185 ページの`clnode`(1CL) のマニュアルページを参照してください。

`global_fencing`

すべての共有デバイスのグローバルなデフォルトのフェンシングアルゴリズムを指定します。

このプロパティーに指定できる値は、`nofencing`、`nofencing-noscrub`、`pathcount`、または `prefer3` です。

Persistent Group Reservation (PGR) キーの検索と削除が完了すると、`nofencing` 設定により、共有デバイスのフェンシングが無効になります。

`nofencing-noscrub` 設定では、先に PGR キーの検索と削除が実行されることなく、共有デバイスのフェンシングが無効になります。

`pathcount` 設定により、共有デバイスに接続されている DID パスの数でフェンシングプロトコルを決定します。3 つ以上の DID パスを使用するデバイスの場合、このプロパティーは SCSI-3 プロトコルに設定されます。

`prefer3` 設定は、すべてのデバイスのデバイスフェンシングについて SCSI-3 プロトコルを指定します。SCSI-3 プロトコルをサポートしないデバイスには、`pathcount` 設定が割り当てられます。

デフォルトでは、このプロパティーには `prefer3` が設定されています。

`heartbeat_quantum`

ハートビートを送信する頻度をミリ秒単位で定義します。

Oracle Solaris Cluster ソフトウェアは、デフォルトでは 1 秒、つまり 1,000 ミリ秒のハートビート定足数を使用します。100 ミリ秒から 10,000 ミリ秒までの値を指定します。

`heartbeat_timeout`

ピアノードからハートビートを受信しなかった場合に、対応するパスを停止中として宣言するまでの時間間隔をミリ秒単位で定義します。

Oracle Solaris Cluster ソフトウェアは、デフォルトでは 10 秒、つまり 10,000 ミリ秒のハートビートタイムアウトを使用します。2,500 ミリ秒から 60,000 ミリ秒までの値を指定します。

`set` サブコマンドを使用すると、すべてのアダプタでクラスタのグローバルなハートビートパラメータを変更できます。

Oracle Solaris Cluster ソフトウェアは、プライベートインターコネクトを介してこれらのハートビートを使用することによって、クラスタノード間の通信障害を検出します。ハートビートタイムアウトを減らすと、Oracle Solaris Cluster ソフトウェアはよりすばやく障害を検出できます。ハートビートタイムアウトの値を減らすと、障害を検出するために必要な時間が短縮されます。これにより、Oracle Solaris Cluster ソフトウェアは障害からよりすばやく回復します。より速く回復できると、クラスタの可用性が高まります。

理想的な条件下であっても、`set` サブコマンドを使用してハートビートパラメータの値を減らすと、誤ったパスタイムアウトやノードパニックが起こるおそれが常にあります。より小さい値を実際にクラスタに導入する際には、適切な負荷条件のもとでハートビートパラメータの値を必ずテストし、入念に確認する必要があります。

`heartbeat_timeout` に指定する値は常に、`heartbeat_quantum` に指定する値の 5 倍以上にします (`heartbeat_timeout` >=(5*`heartbeat_quantum`))。

`installmode`

クラスタのインストールモードの設定を指定します。`installmode` プロパティーには、`enabled` または `disabled` のいずれかを指定できます。

`installmode` プロパティーが有効な間、ノードはブート時の定足数構成のリセットを行いません。またこのモードにある間は、管理機能の多くが使用不可能になります。最初にクラスタをインストールしたときには、`installmode` プロパティーは有効になっています。

すべてのノードが最初にクラスタに参加し、共有定足数デバイスが構成に追加されたら、`installmode` プロパティーを明示的に無効にしてください。`installmode` プロパティーを無効にすると、定足数の投票数はデフォルト値に設定されます。クラスタ作成中に定足数が自動的に構成される場合、定足数が設定されたあとも `installmode` プロパティーは無効です。

`resource_security`

プログラムの実行のセキュリティーポリシーを RGM リソース別に指定します。`resource_security` で使用可能な値は、`SECURE`、`WARN`、`OVERRIDE`、または `COMPATIBILITY` です。

Start や Validate などのリソースメソッドは、常に root として実行されます。メソッドの実行可能ファイルに root 以外の所有権、あるいは group または world 書き込み権がある場合は、セキュアでない状態になりますこの場合、resource_security プロパティーが SECURE に設定されていると、リソースメソッドの実行は実行時に失敗し、エラーが返されます。resource_security がその他の設定であれば、リソースメソッドは実行を許可され、警告メッセージが表示されます。最大限のセキュリティーを確保するため、resource_security を SECURE に設定してください。

resource_security 設定では、application_user リソースプロパティーを宣言するリソースタイプの動作も変更します。application_user リソースプロパティーを宣言するリソースタイプは通常、663 ページのscha_check_app_user(1HA) インタフェースを使用して、アプリケーションプログラムの実行可能ファイルの所有権およびアクセス権の追加チェックを実行するエージェントです。詳細は、1287 ページのr_properties(5) のマニュアルページの application_user に関するセクションを参照してください。

udp_session_timeout

無効な UDP セッションを削除するまでの経過時間を秒単位で指定します。

このプロパティーはオプションであり、任意の整数を設定できます。

このプロパティーが適用されるのは、UDP サービスと、ラウンドロビン負荷分散スキームが有効になっている負荷分散ポリシー Lb_weighted のみです。

デフォルトでは、このプロパティーには 480 (8 分) が設定されています。

プライベートネットワークのプロパティー

プライベートネットワークのプロパティーは、set-netprops サブコマンドでのみ変更します。

これらのプライベートネットワーク設定は、デフォルトのプライベートネットワークアドレスがすでに使用されているアドレスと衝突する場合だけ変更してください。既存のアドレス範囲が拡張しているクラスタ構成を格納するのに十分でない場合もこれらのプライベートネットワーク設定を変更します。

ネットワークプロパティーを変更する場合は、クラスタのすべてのノードが使用可能で非クラスタモードであることが期待されます。プライベートネットワーク設定は、すべてのノードに伝えられるので、クラスタの 1 つのノードだけで変更します。

private_netaddr プロパティーを設定すると、private_netmask プロパティー、max_nodes プロパティーと max_privatenets プロパティー、またはすべてのプロパティーを設定できます。private_netmask プロパティーと max_nodes または max_privatenets プロパティーのいずれかを設定しようとすると、エラーが発生します。max_nodes または max_privatenets プロパティーは常に一緒に設定してください。

デフォルトのプライベートネットワークアドレスは 172.16.0.0 で、デフォルトのネットマスクは 255.255.240.0 です。

クラスタ構成の矛盾が原因でプロパティーの設定に失敗した場合、非クラスタモードで各ノードに対して cluster restore-netprops コマンドを実行します。

プライベートネットワークのプロパティーは次のとおりです。

max_nodes

クラスタの一部になると予想されるノードの最大数を指定します。このプロパティーは、private_netaddr および max_privatenets プロパティーと組み合わせてのみ、またオプションで private_netmask プロパティーとともに設定できます。max_nodes の最大値は 64 です。最小値は 2 です。

max_privatenets

クラスタで使用するプライベートネットワークの最大数を指定します。このプロパティーは、private_netaddr および max_nodes プロパティーと同時の場合のみ、またはオプションとして private_netmask プロパティーとだけ使用できます。max_privatenets の最大値は 128 です。最小値は 2 です。

num_zoneclusters

グローバルクラスタに対して構成しようとしているゾーンクラスタの数を指定します。Oracle Solaris Cluster ソフトウェアでは、この値、ノードの数、およびグローバルクラスタに対して指定するプライベートネットワークの数の組み合わせを使用して、プライベートネットワークネットマスクを計算します。

Oracle Solaris Cluster ソフトウェアはプライベートネットワークネットマスクを使用して、クラスタ使用のために保持するプライベートネットワーク IP アドレスの範囲を決定します。

このプロパティーはクラスタモードまたは非クラスタモードで設定できます。

このプロパティーに値を指定しないと、デフォルトで 12 に設定されます。このプロパティーを 0 に設定できます。

private_netaddr

プライベートネットワークアドレスを指定します。

private_netmask

クラスタのプライベートネットワークマスクを指定します。この場合に指定する値は、デフォルトのネットマスクの 255.255.240.0 以上にしてください。このプロパティーは、private_netaddr プロパティーと同時にのみ設定できます。

デフォルトより小さい IP アドレス範囲を割り当てる場合は、private_netmask プロパティーの代わり、またはこのプロパティーに加えて max_nodes および max_privatenets プロパティーを使用できます。

num_xip_zoneclusters

物理クラスタで構成可能な排他的 IP ゾーンクラスタの数を指定します。このコマンドでは、modify_xip_zc と呼ばれるシェルスクリプトを呼び出し、構成可能な排他的 IP ゾーンクラスタの数のエントリで clprivnet 構成ファイルを更新します。num_xip_zoneclusters プロパティーは、num_zoneclusters プロパティーのサブセットである必要があります。

このコマンドは、プライベートネットワークのプロパティーの組み合わせごとに次に示すタスクを実行します。

`-p private_netaddr=`*netaddr*

コマンドは、デフォルトのネットマスクである `255.255.240.0` をプライベートインターコネクトに割り当てます。デフォルトの IP アドレス範囲は、最大 64 のノードと 10 のプライベートネットワークを格納します。

`-p private_netaddr=`*netaddr*`,private_netmask=`*netmask*

指定されたネットマスクがデフォルトのネットマスク未満である場合、このコマンドは失敗し、エラーで終了します。

指定されたネットマスクがデフォルトのネットマスク以上の場合、このコマンドは指定されたネットマスクをプライベートインターコネクトに割り当てます。その結果の IP アドレス範囲は、最大 64 のノードと 10 のプライベートネットワークを格納します。

デフォルトより小さい IP アドレス範囲を割り当てるには、`private_netmask` プロパティーの代わり、またはこのプロパティーに加えて `max_nodes` および `max_privatenets` プロパティーを使用します。

`-p private_netaddr=`*netaddr*`,max_nodes=`*nodes*`,`
`max_privatenets=`*privatenets*`,num_xip_zoneclusters=`*xip_zoneclusters*

このコマンドは、指定された数のノードとプライベートネットワークをサポートする最小のネットマスクを計算します。このコマンドは次に、計算したネットマスクをプライベートインターコネクトに割り当てます。物理クラスタで構成可能な排他的 IP ゾーンクラスタの数も指定します。

`-p private_netaddr=`*netaddr*`,private_netmask=`*netmask*`,`
`max_nodes=`*nodes*`,max_privatenets=`*privatenets*

このコマンドは、指定された数のノードとプライベートネットワークをサポートする最小のネットマスクを計算します。

このコマンドは、計算結果と指定されたネットマスクを比較します。指定されたネットマスクが計算結果のネットマスク未満である場合、このコマンドは失敗し、エラーで終了します。指定されたネットマスクが計算結果のネットマスク以上の場合、このコマンドは指定されたネットマスクをプライベートインターコネクトに割り当てます。

`-s` *severitylevel*
`--severity=`*severitylevel*
`--severity` *severitylevel*

指定された *severitylevel* 以上の違反だけをレポートします。

このオプションを一緒に指定できるのは、`check` サブコマンドだけです。

各検査には、それぞれ割り当てられた重要度があります。失敗した検査のうち、指定した重要度より低いものはレポートから除外されます。*severity* には、次のいずれかの値を指定します。これらの値は、重要度の最も低い値から最も高い値に向けて一覧表示してあります。

`information`

`warning`

`low`

`medium`

`high`

`critical`

このオプションを指定しない場合、デフォルトでは、重要度 `information` が使用されます。重要度 `information` は、すべての重要度で失敗した検査がレポートに書き込まれることを示します。

`-t` *objecttype*[,…]
`--type=`*objecttype*[,…]
`--type` *objecttype*[,…]

`export`、`show`、および `status` サブコマンドのオブジェクト型を指定します。

このオプションは、`export`、`show`、および `status` サブコマンドの出力を指定された型のオブジェクトだけに制限するために使用します。次に示すオブジェクト型またはコンポーネント型がサポートされています。ステータスが使用できないオブジェクト型も一部存在します。

オブジェクト型/短いオブジェクト型	使用可能なステータス
access/access	いいえ
device/dev	はい
devicegroup/dg	はい
global/global	いいえ
interconnect/intr	はい
nasdevice/nas	いいえ
node/node	はい
quorum/quorum	はい
reslogicalhostname/rslh	はい
resource/rs	はい
resourcegroup/rg	はい
resourcetype/rt	いいえ
ressharedaddress/rssa	はい

-v
--verbose

詳細情報を標準出力 (stdout) で表示します。check サブコマンドと使用する場合は、実行中の詳細な進捗状況が表示されます。list-checks サブコマンドと使用する場合は、検査のより詳細な情報が表示されます。

-V
--version

コマンドのバージョンを表示します。

このオプションをほかのオプション、サブコマンド、またはオペランドと一緒に指定する場合、これらはすべて無視されます。コマンドのバージョンだけが表示されます。ほかの処理は行われません。

-y
--yes

シャットダウンの確認を促すプロンプトを発生させないようにします。クラスタはユーザーの介入なしで即座にシャットダウンされます。

次のオペランドがサポートされています。

clustername

管理対象のクラスタの名前です。

create を除くすべてのサブコマンドで、指定する *clustername* を cluster コマンドを実行するクラスタの名前と一致させてください。

create サブコマンドを使用して、新しい一意のクラスタ名を指定します。

このコマンドセットにあるすべてのコマンドの終了ステータスコードの完全なリストについては、19 ページのIntro(1CL) のマニュアルページを参照してください。返される終了コードも 1047 ページのscha_calls(3HA) のマニュアルページで説明されているリターンコードに準拠しています。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

0 CL_NOERR

エラーなし

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS

アクセス権がありません

指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

35 CL_EIO

I/O エラー

物理的な入出力エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

また、check サブコマンドは、検査レポートを格納する出力ディレクトリに cluster_check_exit_code.log という名前のテキストファイルを作成します。このサブコマンドによって CL_NOERR が終了された場合、違反があったすべての検査の中で最高の重要度を示すコードがこのファイルにレポートされます。使用される検査コードは、次のとおりです。

100	違反はレポートされませんでした。このレポートで報告される検査出力の重要度は、通常、information または warning です。
101	critical
102	high
103	medium

```
104                  low
```

例 249　　クラスタ構成情報の表示

次のコマンドは、クラスタの使用可能なすべての構成情報を表示します。

```
# cluster show
=== Cluster ===

Cluster Name:                                   schost
  clusterid:                                       0x4FA7C35F
  installmode:                                     disabled
  heartbeat_timeout:                               9999
  heartbeat_quantum:                               1000
  private_netaddr:                                 172.16.0.0
  private_netmask:                                 255.255.240.0
  max_nodes:                                       64
  max_privatenets:                                 10
  num_zoneclusters:                                12
  udp_session_timeout:                             480
  concentrate_load:                                True
  resource_security:                               SECURE
  global_fencing:                                  prefer3
  Node List:                                       phys-schost-1, phys-schost-2

  === Host Access Control ===

  Cluster name:                                 schost
    Allowed hosts:                                 None
    Authentication Protocol:                       sys

  === Cluster Nodes ===

  Node Name:                                    phys-schost-1
    Node ID:                                       1
    Enabled:                                       yes
    privatehostname:                               clusternode1-priv
    reboot_on_path_failure:                        disabled
    globalzoneshares:                              1
    defaultpsetmin:                                1
    quorum_vote:                                   1
    quorum_defaultvote:                            1
    quorum_resv_key:                               0x4FA7C35F00000001
    Transport Adapter List:                        net3, net1

  Node Name:                                    phys-schost-2
    Node ID:                                       2
    Enabled:                                       yes
    privatehostname:                               clusternode2-priv
    reboot_on_path_failure:                        disabled
    globalzoneshares:                              1
    defaultpsetmin:                                1
```

```
  quorum_vote:                                1
  quorum_defaultvote:                         1
  quorum_resv_key:                            0x4FA7C35F00000002
  Transport Adapter List:                     net3, net1

=== Transport Cables ===

Transport Cable:                            phys-schost-1:net3,switch1@1
  Endpoint1:                                   phys-schost-1:net3
  Endpoint2:                                   switch1@1
  State:                                       Enabled

Transport Cable:                            phys-schost-1:net1,switch2@1
  Endpoint1:                                   phys-schost-1:net1
  Endpoint2:                                   switch2@1
  State:                                       Enabled

Transport Cable:                            phys-schost-2:net3,switch1@2
  Endpoint1:                                   phys-schost-2:net3
  Endpoint2:                                   switch1@2
  State:                                       Enabled

Transport Cable:                            phys-schost-2:net1,switch2@2
  Endpoint1:                                   phys-schost-2:net1
  Endpoint2:                                   switch2@2
  State:                                       Enabled

=== Transport Switches ===

Transport Switch:                           switch1
  State:                                       Enabled
  Type:                                        switch
  Port Names:                                  1 2
  Port State(1):                               Enabled
  Port State(2):                               Enabled

Transport Switch:                           switch2
  State:                                       Enabled
  Type:                                        switch
  Port Names:                                  1 2
  Port State(1):                               Enabled
  Port State(2):                               Enabled

=== Quorum Devices ===

Quorum Device Name:                         d4
  Enabled:                                     yes
  Votes:                                       1
  Global Name:                                 /dev/did/rdsk/d4s2
  Type:                                        shared_disk
  Access Mode:                                 scsi3
  Hosts (enabled):                             phys-schost-1, phys-schost-2

=== Device Groups ===
```

```
=== Registered Resource Types ===

Resource Type:                                  SUNW.LogicalHostname:4
  RT_description:                                  Logical Hostname Resource Type
  RT_version:                                      4
  API_version:                                     2
  RT_basedir:                                      /usr/cluster/lib/rgm/rt/hafoip
  Single_instance:                                 False
  Proxy:                                           False
  Init_nodes:                                      All potential masters
  Installed_nodes:                                 <All>
  Failover:                                        True
  Pkglist:                                         <NULL>
  RT_system:                                       True
  Global_zone:                                     True

Resource Type:                                  SUNW.SharedAddress:2
  RT_description:                                  HA Shared Address Resource Type
  RT_version:                                      2
  API_version:                                     2
  RT_basedir:                                      /usr/cluster/lib/rgm/rt/hascip
  Single_instance:                                 False
  Proxy:                                           False
  Init_nodes:                                      <Unknown>
  Installed_nodes:                                 <All>
  Failover:                                        True
  Pkglist:                                         <NULL>
  RT_system:                                       True
  Global_zone:                                     True

=== Resource Groups and Resources ===

=== DID Device Instances ===

DID Device Name:                              /dev/did/rdsk/d1
  Full Device Path:                              phys-schost-2:/dev/rdsk/
                                                 c0t600A0B8000485B6A000058584EDCBD7Ed0
  Full Device Path:                              phys-schost-1:/dev/rdsk/
                                                 c0t600A0B8000485B6A000058584EDCBD7Ed0
  Replication:                                   none
  default_fencing:                               global

DID Device Name:                              /dev/did/rdsk/d2
  Full Device Path:                              phys-schost-2:/dev/rdsk/
                                                 c0t600A0B8000485B6A0000585A4EDCBDA4d0
  Full Device Path:                              phys-schost-1:/dev/rdsk/
                                                 c0t600A0B8000485B6A0000585A4EDCBDA4d0
  Replication:                                   none
  default_fencing:                               global

DID Device Name:                              /dev/did/rdsk/d3
  Full Device Path:                              phys-schost-2:/dev/rdsk/
                                                 c0t600A0B8000485B6A0000585C4EDCBDCAd0
```

```
  Full Device Path:                    phys-schost-1:/dev/rdsk/
                                       c0t600A0B8000485B6A0000585C4EDCBDCAd0
  Replication:                         none
  default_fencing:                     global

DID Device Name:                     /dev/did/rdsk/d4
  Full Device Path:                    phys-schost-2:/dev/rdsk/
                                       c0t600A0B8000485B6A0000585E4EDCBDF1d0
  Full Device Path:                    phys-schost-1:/dev/rdsk/
                                       c0t600A0B8000485B6A0000585E4EDCBDF1d0
  Replication:                         none
  default_fencing:                     global

DID Device Name:                     /dev/did/rdsk/d5
  Full Device Path:                    phys-schost-2:/dev/rdsk/
                                       c0t600A0B8000485B6A000058604EDCBE1Cd0
  Full Device Path:                    phys-schost-1:/dev/rdsk/
                                       c0t600A0B8000485B6A000058604EDCBE1Cd0
  Replication:                         none
  default_fencing:                     global

DID Device Name:                     /dev/did/rdsk/d6
  Full Device Path:                    phys-schost-2:/dev/rdsk/
                                       c0t600A0B8000486F08000073014EDCBED0d0
  Full Device Path:                    phys-schost-1:/dev/rdsk/
                                       c0t600A0B8000486F08000073014EDCBED0d0
  Replication:                         none
  default_fencing:                     global

DID Device Name:                     /dev/did/rdsk/d7
  Full Device Path:                    phys-schost-2:/dev/rdsk/
                                       c0t600A0B8000486F08000073034EDCBEFAd0
  Full Device Path:                    phys-schost-1:/dev/rdsk/
                                       c0t600A0B8000486F08000073034EDCBEFAd0
  Replication:                         none
  default_fencing:                     global

DID Device Name:                     /dev/did/rdsk/d8
  Full Device Path:                    phys-schost-2:/dev/rdsk/
                                       c0t600A0B8000486F08000073054EDCBF1Fd0
  Full Device Path:                    phys-schost-1:/dev/rdsk/
                                       c0t600A0B8000486F08000073054EDCBF1Fd0
  Replication:                         none
  default_fencing:                     global

DID Device Name:                     /dev/did/rdsk/d9
  Full Device Path:                    phys-schost-2:/dev/rdsk/
                                       c0t600A0B8000486F08000073074EDCBF46d0
  Full Device Path:                    phys-schost-1:/dev/rdsk/
                                       c0t600A0B8000486F08000073074EDCBF46d0
  Replication:                         none
  default_fencing:                     global

DID Device Name:                     /dev/did/rdsk/d10
```

```
  Full Device Path:                       phys-schost-2:/dev/rdsk/
                                          c0t600A0B8000486F08000073094EDCBF71d0
  Full Device Path:                       phys-schost-1:/dev/rdsk/
                                          c0t600A0B8000486F08000073094EDCBF71d0
  Replication:                            none
  default_fencing:                        global

DID Device Name:                        /dev/did/rdsk/d11
  Full Device Path:                       phys-schost-1:/dev/rdsk/c3t0d0
  Replication:                            none
  default_fencing:                        global

DID Device Name:                        /dev/did/rdsk/d12
  Full Device Path:                       phys-schost-1:/dev/rdsk/c4t0d0
  Replication:                            none
  default_fencing:                        global

DID Device Name:                        /dev/did/rdsk/d13
  Full Device Path:                       phys-schost-1:/dev/rdsk/c4t1d0
  Replication:                            none
  default_fencing:                        global

DID Device Name:                        /dev/did/rdsk/d14
  Full Device Path:                       phys-schost-2:/dev/rdsk/c3t0d0
  Replication:                            none
  default_fencing:                        global

DID Device Name:                        /dev/did/rdsk/d15
  Full Device Path:                       phys-schost-2:/dev/rdsk/c4t0d0
  Replication:                            none
  default_fencing:                        global

DID Device Name:                        /dev/did/rdsk/d16
  Full Device Path:                       phys-schost-2:/dev/rdsk/c4t1d0
  Replication:                            none
  default_fencing:                        global

=== NAS Devices ===

Nas Device:                             qualfugu
  Type:                                   sun_uss
  userid:                                 osc_agent

=== Zone Clusters ===

Zone Cluster Name:                           zc1
  zonename:                                    zc1
  zonepath:                                    /zones/zc1
  autoboot:                                    TRUE
  brand:                                       solaris10
  bootargs:                                    <NULL>
  pool:                                        <NULL>
  limitpriv:                                   <NULL>
  scheduling-class:                            <NULL>
```

```
  ip-type:                                    shared
  enable_priv_net:                            TRUE
  resource_security:                          COMPATIBILITY

  --- Solaris Resources for zc1 ---

  Resource Name:                              net
    address:                                  schost-1
    physical:                                 auto

  Resource Name:                              net
    address:                                  schost-2
    physical:                                 auto

  --- Zone Cluster Nodes for zc1 ---

  Node Name:                                  phys-schost-1
    physical-host:                            phys-schost-1
    hostname:                                 vzschost1a

    --- Solaris Resources for phys-schost-1 ---

  Node Name:                                  phys-schost-2
    physical-host:                            phys-schost-2
    hostname:                                 vzschost2a

    --- Solaris Resources for phys-schost-2 ---

Zone Cluster Name:                            zc2
  zonename:                                   zc2
  zonepath:                                   /zones/zc2
  autoboot:                                   TRUE
  brand:                                      solaris
  bootargs:                                   <NULL>
  pool:                                       <NULL>
  limitpriv:                                  <NULL>
  scheduling-class:                           <NULL>
  ip-type:                                    shared
  enable_priv_net:                            TRUE
  resource_security:                          COMPATIBILITY

  --- Solaris Resources for zc2 ---

  --- Zone Cluster Nodes for zc2 ---

  Node Name:                                  phys-schost-1
    physical-host:                            phys-schost-1
    hostname:                                 vzschost1b

    --- Solaris Resources for phys-schost-1 ---

  Node Name:                                  phys-schost-2
    physical-host:                            phys-schost-2
    hostname:                                 vzschost2b
```

```
    --- Solaris Resources for phys-schost-2 ---

  Zone Cluster Name:                          zc3
    zonename:                                    zc3
    zonepath:                                    /zones/zc3
    autoboot:                                    TRUE
    brand:                                       solaris
    bootargs:                                    <NULL>
    pool:                                        <NULL>
    limitpriv:                                   <NULL>
    scheduling-class:                            <NULL>
    ip-type:                                     shared
    enable_priv_net:                             TRUE
    resource_security:                           COMPATIBILITY

    --- Solaris Resources for zc3 ---

    --- Zone Cluster Nodes for zc3 ---

    Node Name:                                 phys-schost-2
      physical-host:                             phys-schost-2
      hostname:                                  vzschost1c

      --- Solaris Resources for phys-schost-2 ---
```

例 250 選択されたクラスタコンポーネントについての構成情報の表示

次のコマンドは、リソース、リソースタイプ、およびリソースグループについての情報を表示します。情報はそのクラスタに対してのみ表示されます。

```
# cluster show -t resource,resourcetype,resourcegroup
  Single_instance:                             False
  Proxy:                                       False
  Init_nodes:                                  <Unknown>
  Installed_nodes:                             <All>
  Failover:                                    True
  Pkglist:                                     <NULL>
  RT_system:                                   True

Resource Type:                                 SUNW.qfs
  RT_description:                              SAM-QFS Agent on SunCluster
  RT_version:                                  3.1
  API_version:                                 3
  RT_basedir:                                  /opt/SUNWsamfs/sc/bin
  Single_instance:                             False
  Proxy:                                       False
  Init_nodes:                                  All potential masters
  Installed_nodes:                             <All>
  Failover:                                    True
  Pkglist:                                     <NULL>
```

```
  RT_system:                                    False

=== Resource Groups and Resources ===

Resource Group:                                 qfs-rg
  RG_description:                               <NULL>
  RG_mode:                                      Failover
  RG_state:                                     Managed
  Failback:                                     False
  Nodelist:                                     phys-schost-2 phys-schost-1

  --- Resources for Group qfs-rg ---

  Resource:                                     qfs-res
    Type:                                       SUNW.qfs
    Type_version:                               3.1
    Group:                                      qfs-rg
    R_description:
    Resource_project_name:                      default
    Enabled{phys-schost-2}:                     True
    Enabled{phys-schost-1}:                     True
    Monitored{phys-schost-2}:                   True
    Monitored{phys-schost-1}:                   True
```

例 251 クラスタのステータスの表示

次のコマンドは、すべてのクラスタノードのステータスを表示します。

```
# cluster status -t node
=== Cluster Nodes ===

--- Node Status ---

Node Name                                       Status
---------                                       ------
phys-schost-1                                   Online
phys-schost-2                                   Online

--- Node Status ---

Node Name                                       Status
---------                                       ------
```

または、clnode コマンドを使用して、同じ情報を表示することもできます。

```
# clnode status
=== Cluster Nodes ===

--- Node Status ---

Node Name                                       Status
---------                                       ------
phys-schost-1                                   Online
```

```
phys-schost-2                                 Online
```

例 252 クラスタの作成

次のコマンドは、cluster-1 という名前のクラスタをクラスタ構成ファイル suncluster.xml から作成します。

```
# cluster create -i /suncluster.xml cluster-1
```

例 253 クラスタ名の変更

次のコマンドは、クラスタの名前を cluster-2 に変更します。

```
# cluster rename -c cluster-2
```

例 254 クラスタの installmode プロパティーの無効化

次のコマンドは、クラスタの installmode プロパティーを無効にします。

```
# cluster set -p installmode=disabled
```

例 255 プライベートネットワークの変更

次のコマンドは、クラスタのプライベートネットワーク設定を変更します。このコマンドは、プライベートネットワークアドレスに 172.10.0.0 を設定します。また、このコマンドは、指定した 8 個のノードと 4 個のプライベートネットワークをサポートするための最小プライベートネットマスク数を計算して設定し、グローバルクラスタに 8 個のゾーンクラスタを構成することを指定します。このコマンドでは、非クラスタモードの物理クラスタで構成可能な排他的 IP ゾーンクラスタの数も識別します。

```
# cluster set-netprops \
-p private_netaddr=172.10.0.0 \
-p max_nodes=8 \
-p max_privatenets=4 \
-p num_zoneclusters=8 \
-p num_xip_zoneclusters=3
```

このコマンドは、次のように非クラスタモードで指定することもできます:

```
# cluster set-netprops \
-p private_netaddr=172.10.0.0 \
-p max_nodes=8,\
-p max_privatenets=4 \
-p num_zoneclusters=8 \
-p num_xip_zoneclusters=3
```

例 256 使用可能な検査の一覧表示

次のコマンドは、クラスタで実行可能なすべての検査を 1 行にまとめて表示します。実際に使用できる検査は、リリースや更新によって異なります。

```
# cluster list-checks
 M6336822  :   (Critical)   Global filesystem /etc/vfstab entries are
not consistent across all Oracle Solaris Cluster nodes.
 S6708689  :   (Variable)   One or more Oracle Solaris Cluster resources
cannot be validated
 M6708613  :   (Critical)   vxio major numbers are not consistent across
all Oracle Solaris Cluster nodes.
 S6708255  :   (Critical)   The nsswitch.conf file 'hosts' database
entry does not have 'cluster' specified first.
 S6708479  :   (Critical)   The /etc/system rpcmod:svc_default_stksize
parameter is missing or has an incorrect value for Oracle Solaris Cluster.
 F6984121  :   (Critical)   Perform cluster shutdown
 F6984140  :   (Critical)   Induce node panic
…
```

例 257 クラスタでの基本検査の実行

次のコマンドは、詳細モードで、phys-schost-1 がクラスタメンバーとなっている schost クラスタのすべてのノードに対して、実行可能なすべての基本検査を実行します。出力は basicchks.18Nov2011.schost ファイルにリダイレクトされます。

```
phys-schost-1# cluster check -v -o basicchks.18Nov2011.schost
```

例 258 クラスタでの対話型検査の実行

次のコマンドは、キーワード vfstab を含む検査を除いて、実行可能なすべての対話型検査を実行します。検査の出力は interactive.chk.18Nov2011 というファイルに保存されます。

```
# cluster check -k interactive -E vfstab -o interactive.chk.18Nov2011 cluster-1

    User supplies information when prompted
```

例 259 クラスタでの機能検査の実行

次のコマンドは、機能検査 F6968101 の詳細説明を表示して、phys-schost-1、phys-schost-2、および phys-schost-3 がクラスタメンバーとなっているクラスタに対してこの検査を実行します。検査の出力は F6968101.failovertest.19Nov2011 というファイルに保存されます。この検査はクラスタノードのフェイルオーバーを伴うため、クラスタの稼動を停止してから検査を開始してください。

```
phys-schost-1# cluster list-checks -v -C F6968101
```

```
 initializing...
 F6968101: (Critical) Perform resource group switchover
Keywords: SolarisCluster4.x, functional
Applicability: Applicable if multi-node cluster running live.
Check Logic: Select a resource group and destination node.
Perform '/usr/cluster/bin/clresourcegroup switch' on specified
resource group either to specified node or to all nodes in succession.
Version: 1.118
Revision Date: 13/07/09

 cleaning up...
```

Take the cluster out of production

```
phys-schost-1# cluster check -k functional -C F6968101 \
-o F6968101.failovertest.19Nov2011

  initializing...
  initializing xml output...
  loading auxiliary data...
  starting check run...
     phys-schost-1, phys-schost-2, phys-schost-3:     F6968101.... starting:
 Perform resource group switchover

 ============================================================

   >>> Functional Check <<<
```

Follow onscreen directions

```
…
```

例 260 指定したノードでの限定的な検査の実行

次のコマンドは、重要度が `high` 以上のすべての検査を詳細モードで実行します。これらの検査は、`phys-schost-1` ノードだけで実行されます。

```
# cluster check -v -n phys-schost-1 -s high
 initializing...
 initializing xml output...
 loading auxiliary data...
 filtering out checks with severity less than High
 starting check run...
    phys-schost-1:     M6336822.... starting:  Global filesystem /etc/vfstab entries...
    phys-schost-1:     M6336822       not applicable
    phys-schost-1:     S6708689.... starting:  One or more Oracle Solaris Cluster...
    phys-schost-1:     S6708689       passed
…
    phys-schost-1:     S6708606       skipped: severity too low
    phys-schost-1:     S6708638       skipped: severity too low
    phys-schost-1:     S6708641.... starting:  Cluster failover/switchover might...
    phys-schost-1:     S6708641       passed
```

…

/usr/cluster/lib/cfgchk/checkresults.dtd

/var/cluster/logs/cluster_check/

/*outputdir*/cluster_check_exit_code.log

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページのIntro(1CL), Unresolved link to " init1M", Unresolved link to " su1M", 1047 ページのscha_calls(3HA), Unresolved link to " attributes5", Unresolved link to " rbac5", 1447 ページのclconfiguration(5CL)

スーパーユーザーはこのコマンドのすべての形式を実行できます。

すべてのユーザーがこのコマンドに -? (ヘルプ) オプションまたは -V (バージョン) オプションを指定して実行できます。

スーパーユーザー以外のユーザーが cluster コマンドにサブコマンドを付けて実行するには、RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
check	solaris.cluster.read
create	solaris.cluster.modify
export	solaris.cluster.read
list	solaris.cluster.read
list-checks	solaris.cluster.read
list-cmds	solaris.cluster.read
rename	solaris.cluster.modify

サブコマンド	RBAC の承認
restore-netprops	solaris.cluster.modify
set	solaris.cluster.modify
set-netprops	solaris.cluster.modify
show	solaris.cluster.read
show-netprops	solaris.cluster.read
shutdown	solaris.cluster.admin
status	solaris.cluster.read

名前

clzonecluster, clzc — ゾーンクラスタの作成と管理

```
/usr/cluster/bin/clzonecluster [subcommand] -?

/usr/cluster/bin/clzonecluster -V

/usr/cluster/bin/clzonecluster subcommand [options] -v
     [zone-cluster-name]

/usr/cluster/bin/clzonecluster apply [-n node-name[,…]] [-d]
     {+ | zone-cluster-name […]}

/usr/cluster/bin/clzonecluster boot [-n node-name[,…]] [-o]
     {+ | zone-cluster-name […]}

/usr/cluster/bin/clzonecluster clone -Z target-zone-cluster-name
     [-m method][-n node-name[,…]] {source-zone-cluster-name}

/usr/cluster/bin/clzonecluster configure [-f command-file]
     zone-cluster-name

/usr/cluster/bin/clzonecluster delete [-F] zone-cluster-name

/usr/cluster/bin/clzonecluster export [-f command-file]
     zone-cluster-name

/usr/cluster/bin/clzonecluster halt [-n node-name[,…]]
     {+ | zone-cluster-name}

/usr/cluster/bin/clzonecluster install [-c config_profile.xml]
     [-M manifest.xml] zone-cluste-rname

/usr/cluster/bin/clzonecluster install [-n node-name]
     -a absolute_path_to_archive [-x cert|ca-cert|key=file]…
     -z zone zone-cluster-name

/usr/cluster/bin/clzonecluster install [-n node-name]
     -d absolute_root_path zone-cluster-name

/usr/cluster/bin/clzonecluster install-cluster
     [-d dvd-image] [-n node-name[,…]]
     [-p patchdir=patch-dir[,patchlistfile=file-name]]
     -s software-component[,…]] [-v] zone-cluster-name

/usr/cluster/bin/clzonecluster install-cluster
     [-p patchdir=patch-dir[,patchlistfile=file-name]
     [-n node-name[,…]] [-v] zone-cluster-name

/usr/cluster/bin/clzonecluster list [+ | zone-cluster-name […]]
```

```
/usr/cluster/bin/clzonecluster move -f zone-path zone-cluster-name

/usr/cluster/bin/clzonecluster ready [-n node-name[,…]]
     {+ | zone-cluster-name […]}

/usr/cluster/bin/clzonecluster reboot [-n node-name[,…]] [-o]
     {+ | zone-cluster-name […]}

/usr/cluster/bin/clzonecluster set {-p name=value}
     [-p name=value] […] [zone-cluster-name]

/usr/cluster/bin/clzonecluster show [+ | zone-cluster-name […]]

/usr/cluster/bin/clzonecluster show-rev [-v] [-n node-name[,…]]
     [+ | zone-cluster-name …]

/usr/cluster/bin/clzonecluster status [+ | zone-cluster-name […]]

/usr/cluster/bin/clzonecluster uninstall [-F] [-n node-name
     [,…]] zone-cluster-name

/usr/cluster/bin/clzonecluster verify [-n node-name[,…]]
     {+ | zone-cluster-name […]}
```

`clzonecluster` コマンドは、Oracle Solaris Cluster 構成用のゾーンクラスタを作成し変更します。`clzc` コマンドは `clzonecluster` コマンドの短い形式です。コマンドはまったく同じです。`clzonecluster` コマンドはクラスタ対応で、管理の単一ソースをサポートします。1 つのノードからコマンドのすべての形式を実行して、単一のゾーン・クラスタノードまたはすべてノードに影響を与えることができます。

subcommand を省略できるのは、*options* が `-?` オプションまたは `-V` オプションの場合のみです。

サブコマンドは `list`、`show`、`status` サブコマンドを除き、1 つ以上のオペランドを必要とします。ただし、多くのサブコマンドはプラス記号オペランド (`+`) を受け入れて、そのサブコマンドをすべてのアプリケーションオブジェクトに適用します。`clzonecluster` コマンドはゾーンクラスタの任意のノードで実行し、そのゾーンクラスタの任意またはすべてに影響を与えることができます。

このコマンドの各オプションには長形式と短形式があります。各オプションの両方の形式は、オプション に説明とともに記載されています。

注記 - ゾーンクラスタが作成されたあとにゾーンクラスタ名を変更することはできません。

サポートされるサブコマンドには次のものがあります。

apply

構成の変更をゾーンクラスタに適用します。

apply サブコマンドは、ゾーンクラスタの永続的なライブ再構成を格納します。clzonecluster configure を実行して構成の変更を行なってから、apply サブコマンドを実行して、その変更を特定のゾーンクラスタに適用するようにしてください。apply サブコマンドは、-n オプションを使用して、再構成が適用されるノードのリストを指定します。

apply サブコマンドは、グローバルクラスタノードからのみ使用できます。

boot

ゾーンクラスタのブート。

boot サブコマンドはゾーンクラスタをブートします。boot サブコマンドは -n フラグを使用して、ノードの指定したリストのゾーンクラスタをブートします。

グローバルクラスタノードから boot サブコマンドのみを使用できます。

clone

ゾーンクラスタのクローン。

clone コマンドは、インストールされた既存のゾーンクラスタをコピーして、ゾーンクラスタをインストールします。このサブコマンドは、ゾーンクラスタのインストールに代わるものです。clone サブコマンドは、それ自身で新しいゾーンクラスタを作成しません。クローニングに使用するソースゾーンクラスタが、クローニングする前に*インストールされた*状態であることを確認します。最初に configure サブコマンドを使用して、新しいゾーンクラスタを作成する必要があります。次に、clone サブコマンドを使用して、クローニングされた構成を新しいゾーンクラスタに適用します。

グローバルクラスタノードから clone サブコマンドのみを使用できます。

configure

対話型ユーティリティーを起動して、solaris10 または labeled ブランドゾーンクラスタを構成します。

configure サブコマンドは zonecfg コマンドを使用して、指定されたそれぞれのマシンでゾーンを構成します。configure サブコマンドにより、ゾーンクラスタの各ノードに適用するプロパティーを指定できます。これらのプロパティーは、個別ゾーンの zonecfg コマンドによって確立された場合と同じ意味を持ちます。configure サブコマンドは zonecfg コマンドには分からないプロパティーの構成をサポートします。-f オプションを指定しない場合、configure サブコマンドは対話型シェルを起動します。-f オプションは、その引数としてコマンドファイルを取ります。configure サブコマンドはこのファイルを使用して、ゾーンクラスタを非対話型で作成または変更します。

また、configure サブコマンドを使用すると、統合アーカイブを使用してゾーンクラスタを構成し、復旧用のアーカイブまたはクローンアーカイブを選択することもできます。-a *archive* オプションを create サブコマンドとともに使用します。例:

```
# /usr/cluster/bin/clzc configure sczone1
sczone1: No such zone cluster configured
```

```
Use 'create' to begin configuring a new zone cluster.
clzc:sczone1> create -a archive -z archived zone
```

configure サブコマンドは、グローバルクラスタノードからのみ使用できます。詳細は、Unresolved link to " Oracle Solaris Cluster ソフトウェアのインストール "を参照してください。

solaris10 ブランドゾーンクラスタを指定するには、ゾーンクラスタの構成時にデフォルトのテンプレートを使用できます。デフォルトのテンプレートは、/etc/cluster/zone_cluster/ORCLcls10default.xml にあります。-t オプションを使用すると、デフォルトの solaris10 ゾーンクラスタテンプレート、またはクラスタにある別の既存の solaris10 ゾーンクラスタを指定できます。別の solaris10 ゾーンクラスタを指定すると、ゾーンクラスタの構成は指定したゾーンクラスタからインポートされます。verify または commit 操作が失敗しないように、sysid プロパティーに root パスワードを指定する必要もあります。次のコマンドを入力してテンプレートを適用します:

```
# /usr/cluster/bin/clzc configure sczone2
sczone2: No such zone cluster configured
Use 'create' to begin configuring a new zone cluster.
clzc:sczone2> create -t ORCLcls10default
clzc:sczone2> info
zonename: sczone2
zonepath:
autoboot: true
hostid:
brand: solaris10
```

configure コマンドの対話型と非対話型形式はどちらも、ゾーンクラスタ構成を編集するための複数のサブコマンドをサポートしています。使用可能な構成サブコマンドのリストについては、Unresolved link to " zonecfg1M" を参照してください。

configure ユーティリティーにより、ゾーンクラスタの構成を作成または変更できます。ゾーンクラスタ構成は、多くのリソースタイプおよびプロパティーで構成されます。configure ユーティリティーは *scope* のコンセプトを使用して、サブコマンドが適用される場所を決定します。configure ユーティリティーで使用されるスコープには、クラスタ、リソース、およびノード固有のリソースの 3 つのレベルがあります。デフォルトのスコープはクラスタです。次のリストは、スコープの 3 つのレベルを説明しています。

- クラスタスコープ - ゾーンクラスタ全体に影響を及ぼすプロパティー。*zoneclustername* が sczone の場合、clzonecluster コマンドの対話型シェルは次のように見えます。

  ```
  clzc:sczone>
  ```

- ノードスコープ - ノードリソーススコープ内部に入れ子になった特殊なリソーススコープ。ノードスコープ内部で設定すると、ゾーンクラスタの特定ノードに影響を与えます。たとえば、ゾーンクラスタの特定ノードにネットリソースを追加できます。clzonecluster コマンドの対話型シェルは、次のように見えます。

  ```
  clzc:sczone:node:net>
  ```

- リソーススコープ - 1 つの特定リソースに適用されるプロパティー。リソーススコーププロンプトには、付加されたリソースタイプの名前が付いています。たとえば、`clzonecluster` コマンドの対話型シェルは次のように見えます。

  ```
  clzc:sczone:net>
  ```

`delete`

特定ゾーンクラスタの削除。

このサブコマンドは、特定のゾーンクラスタのリソースグループを削除します。ワイルドカードオペランド (*) を使用するとき、`delete` コマンドはグローバルクラスタで構成されたゾーンクラスタを削除します。`delete` サブコマンドを実行する前に、ゾーンクラスタを構成状態にする必要があります。`delete` コマンドで `-F` オプションを使用した場合は、どのような状態にあるゾーンクラスタに対しても削除が試みられます。

`delete` サブコマンドはグローバルクラスタノードからのみ使用できます。

`export`

ゾーンクラスタの構成をコマンドファイルにエクスポートします。

エクスポートされた *commandfile* は、`configure` サブコマンドの入力として使用できます。必要に応じてファイルを変更し、作成する構成を反映させます。詳細は、Unresolved link to " clconfiguration(5CL)" のマニュアルページを参照してください。

`export` サブコマンドは、グローバルクラスタノードからのみ使用できます。

`halt`

ゾーンクラスタまたはゾーンクラスタの特定ノードの停止。

特定のゾーンクラスタを指定するとき、`halt` サブコマンドは特定のゾーンクラスタに対してのみ適用されます。ゾーンクラスタ全体、またはゾーンクラスタの特定ノードのみを停止できます。ゾーンクラスタを指定しない場合、`halt` サブコマンドはすべてのゾーンクラスタに適用されます。指定されたマシンですべてのゾーンクラスタを停止することもできます。

`halt` サブコマンドは `-n` オプションを使用して、特定ノードのゾーンクラスタを停止します。デフォルトで、`halt` サブコマンドはすべてのノードのすべてのゾーンクラスタを停止します。ゾーン名の代わりに `+` オペランドを指定する場合、すべてのゾーンクラスタが停止されます。

`halt` サブコマンドは、グローバルクラスタノードからのみ使用できます。

`install`

ゾーンクラスタのインストール。

このサブコマンドは、ゾーンクラスタをインストールします。

`install -M` *manifest*`.xml` オプションを使用した場合は、指定したマニフェストが、ゾーンクラスタのすべてのノード上のインストールに使用されます。マニフェストファイルには、`certificate_file`、`key_file`、パブリッシャー、任意の追加パッケージなどの、インストールのために管理者に必要な `solaris` パッケージの情報が記述されます。*manifest*`.xml`

ファイルには、ゾーンクラスタのインストールのための Oracle Solaris Cluster グループパッケージ `ha-cluster-full`、`ha-cluster-framework-full`、`ha-cluster-data-services-full`、または `ha-cluster-minimal` も指定する必要があります。Automated Installer マニフェストの詳細は、Unresolved link to " Oracle Solaris 11.2 システムのインストール のカスタム AI マニフェストの作成"を参照してください。

`-M` オプション (デフォルト) を使用しない場合は、`/usr/share/auto_install/manifest/zone_default.xml` にある自動インストーラマニフェストがインストールに使用されます。この `zone_default.xml` マニフェストが使用されている場合、発行元のゾーンクラスタノードの大域ゾーンにインストールされているすべての `ha-cluster/*` パッケージが、ゾーンクラスタのすべてのノードにインストールされます。ゾーンクラスタのインストール時にカスタムマニフェストを使用する場合、Oracle Solaris Cluster グループパッケージを指定しないと、インストールが失敗します。

インストールするすべてのゾーンクラスタノードのベースとなる大域ゾーンは、`install` サブコマンドを実行するゾーンクラスタノードの大域ゾーンにインストールされるのと同じ Oracle Solaris Cluster パッケージのセットを持っている必要があります。ゾーンクラスタのインストールは、この要件を満たさないゾーンクラスタノードでは失敗する場合があります。

`install` サブコマンドはグローバルクラスタノードからのみ使用できます。`-M` および `-c` オプションは、`solaris` ブランドゾーンクラスタのみに使用できます。

ゾーンクラスタのブランドが `solaris10` の場合は、`-a` または `-d` オプションを使用する必要があります。

`-a` archive

`solaris` または `solaris10` ブランドゾーンクラスタの統合アーカイブ、`solaris10` ブランドゾーンクラスタの `flar` アーカイブの場所、またはインストールに使用する Oracle Solaris 10 イメージアーカイブの絶対パス。サポートされているアーカイブタイプの詳細は、Unresolved link to " solaris105" のマニュアルページを参照してください。アーカイブの絶対パスは、ゾーンクラスタがインストールされるクラスタのすべての物理ノードでアクセスできるようにしてください。統合アーカイブのインストールでは、復旧用のアーカイブまたはクローンアーカイブを使用できます。

`-d` path

インストール済みの Oracle Solaris 10 システムのルートディレクトリへのパス。パスは、ゾーンクラスタがインストールされるクラスタのすべての物理ノードでアクセスできるようにしてください。

[`-x` cert|ca-cert|key=*file*]...

HTTPS 統合アーカイブの場所を使用する場合は、SSL 証明書、認証局 (CA) 証明書、および鍵ファイルを指定します。`-x` オプションは何度でも指定できます。

`-z` *zone*

統合アーカイブに複数のゾーンが含まれている場合は、構成またはインストールのソースのゾーン名を指定します。

同じアーカイブまたはインストール済みの Oracle Solaris 10 システムは、ゾーンクラスタにあるすべての `solaris10` ブランドゾーンのインストールのソースとして使用されます。インストールすると、ソースアーカイブまたはインストール済みの Oracle Solaris 10 システムのシステム識別パラメータは、ゾーンクラスタの構成時に `sysid` リソースタイプで指定したシステム識別パラメータでオーバーライドされます。

`install-cluster`

`install-cluster` サブコマンドは、Oracle Solaris 10 OS をサポートする Oracle Solaris Cluster ソフトウェアを `solaris10` ブランドゾーンクラスタノードにインストールします。インストールされているソフトウェアには、コアパッケージ、クラスタソフトウェアコンポーネント (ゾーンクラスタおよび Geographic Edition ソフトウェアでサポートされるエージェントなど)、およびパッチが含まれます。

注記 - `install-cluster` サブコマンドは、Oracle Solaris Cluster バージョン 3.3 または 3.3 5/11 ソフトウェアを `solaris10` ブランドゾーンクラスタノードにインストールすることをサポートしていません。`solaris10` ブランドゾーンクラスタのサポートされているリリースの詳細は、Unresolved link to " Oracle Solaris Cluster 4.2 リリースノート "を参照してください。

このサブコマンドを使用するのは、クラスタソフトウェアがインストールされていない Oracle Solaris 10 システムに `solaris10` ブランドソーンをインストールする場合です。このサブコマンドを使用するには、`clzonecluster install` コマンドを使用して Oracle Solaris 10 システムの Solaris OS ソフトウェアを `solaris10` ゾーンにインストールしておき、そのゾーンを `online` 状態にブートする必要があります。

クラスタコアパッケージが `solaris10` ブランドゾーンにまだインストールされていない場合は、クラスタリリース DVD ディレクトリに `-d` オプション、クラスタソフトウェアコンポーネントに `-s` オプション、パッチに `-p` オプションを指定することで、コアパッケージ、すべてのクラスタソフトウェアコンポーネント、およびパッチをすべて同時にインストールできます。クラスタソフトウェアコンポーネントおよびパッチをインストールするためのオプションはオプションです。

クラスタコアパッケージをすでにインストールしている場合でも、このサブコマンドを使用して、ゾーンクラスタでサポートされるパッチとクラスタソフトウェアコンポーネントをインストールできます。パッチ情報が指定されている場合は、ゾーンクラスタのクラスタノードを `-o` オプションで `offline-running` 状態にブートする必要があります。

`solaris10` ブランドゾーンクラスタは、共有 IP ゾーンタイプのみをサポートします (排他的 IP および共有 IP ゾーンクラスタの詳細は、Unresolved link to " Oracle Solaris Cluster ソフトウェアのインストール " を参照)。

このサブコマンドは、大域ゾーンからのみ実行できます。

`list`

構成されたゾーンクラスタの名前の表示。

このサブコマンドは、クラスタで構成されたゾーンクラスタの名前を報告します。

- グローバルクラスタノードから `list` サブコマンドを実行する場合、サブコマンドはグローバルクラスタのすべてのゾーンクラスタのリストを表示します。
- ゾーンクラスタノードから `list` サブコマンドを実行すると、サブコマンドはゾーンクラスタの名前のみを表示します。

ゾーンクラスタが構成された場所でノードのリストを表示するには、`-v` オプションを使用します。

`move`

ゾーンパスを新しいゾーンパスに移動。

このサブコマンドはゾーンパスを新しいゾーンパスに移動します。

`move` サブコマンドは、グローバルクラスタノードからのみ使用できます。

`ready`

アプリケーションに対してゾーンを準備。

このサブコマンドは、アプリケーションを実行するためのゾーンを準備します。

`ready` サブコマンドはグローバルクラスタノードからのみ使用できます。

`reboot`

ゾーンクラスタのリブート。

このサブコマンドはゾーンクラスタをリブートし、`halt` サブコマンドの実行に似ており、次に `boot` サブコマンドが続きます。詳細については、`halt` サブコマンドおよび `boot` サブコマンドを参照してください。

`reboot` サブコマンドはグローバルクラスタノードからのみ使用できます。

`set`

`-p` オプションで指定したプロパティーの値をゾーンクラスタに設定します。`set` サブコマンドは大域ゾーンまたはゾーンクラスタから使用できます。設定できるプロパティーについては、オプション セクションの `-p` の説明を参照してください。

`show`

ゾーンクラスタのプロパティーの表示。

ゾーンクラスタのプロパティーには、ゾーンクラスタ名、ブランド、IP タイプ、ノードリスト、ゾーンパス、および許可されたアドレスが含まれます。`show` サブコマンドはゾーンクラスタから実行しますが、特定のゾーンクラスタにのみ適用されます。ゾーンクラスタからこのサブコマンドを使用するとき、ゾーンパスは常に `/` です。ゾーンクラスタ名が指定される場合、このコマンドはそのゾーンクラスタにのみ適用されます。

`show` サブコマンドはグローバルクラスタノードからのみ使用できます。

`show-rev`

ゾーンクラスタの各ノードのクラスタのリリース情報を表示します。

この機能は、ゾーンクラスタにインストールされているリリースバージョンおよびパッチを一覧表示する場合に便利です。例:

```
# clzonecluster show-rev
=== Zone Clusters ===
Zone Cluster Name:   zc1
Release at vznode1a on node pnode1:  3.3u2_40u1_zc:2012-04-01
Release at vznode2a on node pnode2:  3.3u2_40u1_zc:2012-04-01
```

`show-rev` サブコマンドは、グローバルクラスタノードまたはゾーンクラスタノードから使用できます。

`status`

ゾーンクラスタノードがゾーンクラスタのメンバーであるかどうかを判断し、そのゾーンクラスタが `solaris`、`solaris10`、`labeled` ブランドのいずれであるかを表示します。

ゾーンの状態は、`Configured`、`Installed`、`Ready`、`Running`、`Shutting Down`、`Unavailable` のいずれかになります。グローバルクラスタ内のすべてのゾーンクラスタの状況は表示されているので、仮想クラスタの状態を見ることができます。

ゾーン活動を確認するには、`zoneadm` コマンドを代わりに使用します。

`status` サブコマンドはグローバルクラスタノードからのみ使用できます。

`uninstall`

ゾーンクラスタのアンインストール。

このサブコマンドは、ゾーンクラスタをアンインストールします。`uninstall` サブコマンドは `zoneadm` コマンドを使用します。

`uninstall` サブコマンドはグローバルクラスタノードからのみ使用できます。

`verify`

指定された情報の構文が正しいかどうかをチェックします。

このサブコマンドはゾーンクラスタの各ノードの `zoneadm verify` コマンドを呼び出して、各ゾーンクラスタのメンバーを安全にインストールできるようにします。詳細は、Unresolved link to " zoneadm1M" を参照してください。

`verify` サブコマンドはグローバルクラスタノードからのみ使用できます。

注記 - このセクションでは、各オプションの短い形式と長い形式の両方が示されています。

次のオプションがサポートされています。

`-?`
`--help`

ヘルプ情報を表示します。

このオプションは *subcommand* の使用に関係なく指定できます。

subcommand を指定しない場合、使用可能なすべてのサブコマンドのリストが表示されます。

subcommand を指定する場合、そのサブコマンドの使用法が表示されます。

このオプションとその他のオプションを指定すると、その他のオプションは無視されます。

`-a` *absolute_path_to_archive zoneclustername*

インストール済みの Oracle Solaris 10 システム、インストール済みの Oracle Solaris 10 ネイティブゾーン、または `solaris10` ブランドゾーンの `flash_archive`、`cpio`、`pax`、`xustar`、`zfs archive`、または `level 0 ufsdump` のパスを指定します。また、統合アーカイブの絶対パスも指定できます。詳細は、Unresolved link to " solaris105"、Unresolved link to " flash_archive4"、Unresolved link to " cpio1"、および Unresolved link to " pax1" のマニュアルページを参照してください。

`-c` *config_profile*`.xml`
`--configprofile` *config_profile*`.xml`

`solaris` ブランドゾーンクラスタの構成プロファイルテンプレートを指定します。リポジトリからのインストール後に、テンプレートはシステム構成情報をゾーンクラスタのすべてのノードに適用します。*config_profile*`.xml` が指定されていない場合は、各ノードの大域ゾーンから `zlogin -C` *zoneclustername* コマンドを実行して、各ゾーンクラスタノードを手動で構成する必要があります。すべてのプロファイルには `.xml` 拡張子が必要です。

`-c` オプションは、構成プロファイルテンプレートのゾーンクラスタノードのホスト名を置き換えます。プロファイルは、ゾーンクラスタノードのブート後に、ゾーンクラスタノードに適用されます。

`-d` *absolute_root_path*
`--dirpath` *dirpatch*

`-d` オプションを `cluster` サブコマンドとともに使用すると、インストール済みの Oracle Solaris 10 システムのルートディレクトリへのパスが指定されます。パスは、ゾーンクラスタがインストールされるクラスタのすべての物理ノードでアクセスできるようにしてください。

`-d`
`--dvd-directory` *dvd-directory*

DVD イメージディレクトリを指定します。

`-d` オプションを `install-cluster` サブコマンドとともに使用すると、`solaris10` ブランドゾーンをサポートする Oracle Solaris Cluster リリースの DVD イメージディレクトリが指定されます。DVD イメージには、コアパッケージと、ゾーンクラスタおよび Geographic Edition ソフトウェアでサポートされているほかのクラスタソフトウェアコンポーネント (エージェントなど) が含まれます。DVD ディレクトリは、コマンドを実行するノードの大域ゾーンからアクセスできるようにする必要があります。

`-d`
`--dry_run`

`apply` サブコマンドで `-d` オプションが使用されている場合、再構成は予行演習モードで実行されます。予行演習モードでは構成が変更されず、実行中のゾーンはそのままの状態に

なります。予行演習モードは、実際の再構成によって実行されるアクションを確認するために使用します。

`-f`{*commandfile* | *zonepath*}
`--file-argument` {*commandfile* | *zonepath*}

`configure` サブコマンドとともに使用するとき、`-f` オプションはコマンドファイルの引数を指定します。たとえば、`clzonecluster configure -f` *commandfile* となります。`move` サブコマンドとともに使用するとき、`-f` オプションは *zonepath* を指定します。

`-F`

`delete` および `uninstall` 操作の間、`-F` オプションを使用できます。`-F` オプションは `Are you sure you want to` *do this operation* `[y/n]?` という質問を強制的に抑制します。

`-m` *method*
`--method` *method*

ゾーンクラスタをクローニングするには、`-m` オプションを使用します。クローニングの唯一有効なメソッドは、`copy` コマンドです。`clone` サブコマンドを実行する前に、ソースゾーンクラスタを停止する必要があります。

`-M` *manifest*`.xml`
`--manifest` *manifest*`.xml`

`solaris` ブランドゾーンクラスタのすべてのノードにマニフェストを指定するには、`-M` オプションを使用します。このマニフェストは、Oracle Solaris パッケージ情報と、ゾーンクラスタのインストールのための Oracle Solaris Cluster パッケージを指定します。

`-n` *nodename*[…]
`--nodelist` *nodename*[,…]

サブコマンドのノードリストを指定します。

たとえば、`clzonecluster boot -n` *phys-schost-1, phys-schost-2 zoneclustername*.

`-o`
`--offline`

ゾーンクラスタを `offline-running` モードにブートまたはリブートします。

`offline-running` モードになるのは、ゾーンクラスタノードがゾーンクラスタメンバシップから除外されているが、Oracle Solaris ゾーン状態が実行中の場合です。ゾーンクラスタは、物理クラスタとブートモード (クラスタまたは非クラスタモード) を共有するため、オフライン状態は非クラスタモードのクラスタとは異なります。

ゾーンクラスタをオフライン実行モードにブートするには、次を入力します。

`clzonecluster boot [-n` *phys-schost-1*`,...] [-o]` *zoneclustername*

ゾーンクラスタをオフライン実行モードにリブートするには、次を入力します。

`clzonecluster reboot [-n` *phys-schost-1*`,...] [-o]` *zoneclustername*

offline-running ゾーンクラスタを online-running モードにブートするには、-o オプションを指定せずに clzonecluster reboot コマンドを実行します。

-p *name=value*
--property=*name=value*
--property *name=value*

-p オプションは、install-cluster サブコマンドおよび set サブコマンドとともに使用します。-p を指定した install-cluster サブコマンドの使用方法については、-p patchdir=*patchdir*[,patchlistfile=*patchlistfile*] の説明を参照してください。

-p オプションは set サブコマンドとともに使用して、プロパティーの値を指定します。-p *name=value* は複数回指定できます。

このオプションは set サブコマンドと一緒に使用して、次のプロパティーを変更します:

resource_security

プログラムの実行のセキュリティーポリシーを RGM リソース別に指定します。resource_security で使用可能な値は、SECURE、WARN、OVERRIDE、または COMPATIBILITY です。

Start や Validate などのリソースメソッドは、常に root として実行されます。メソッドの実行可能ファイルに root 以外の所有権、あるいは group または world 書き込み権がある場合は、セキュアでない状態になりますこの場合、resource_security プロパティーが SECURE に設定されていると、リソースメソッドの実行は実行時に失敗し、エラーが返されます。resource_security がその他の設定であれば、リソースメソッドは実行を許可され、警告メッセージが表示されます。最大限のセキュリティーを確保するため、resource_security を SECURE に設定してください。

resource_security 設定では、application_user リソースプロパティーを宣言するリソースタイプの動作も変更します。詳細は、1287 ページのr_properties(5) のマニュアルページの「application_user」セクションを参照してください。

-p patchdir=*patchdir*[,patchlistfile=*patchlistfile*]
--patch-specification=patchdir=*patchdir*[,patchlistfile=*patchlistfile*]
--patch-specification patchdir=*patchdir*[,patchlistfile=*patchlistfile*]

-p オプションで指定される patchdir および patchlistfile プロパティーは、install-cluster サブコマンドとともにのみ使用されます。コアパッケージのインストール後にパッチをインストールする場合は、パッチを適用するために、ゾーンクラスタを offline-running 状態にブートする必要があります。

-p *name= value* は複数回指定できます。

patchdir

solaris10 ブランドゾーンに適用する Oracle Solaris Cluster パッチが含まれているディレクトリを指定します。patchdir ディレクトリは必須で、ゾーンクラスタのすべてのノードで、solaris10 ブランドゾーン内部からアクセスできる必要があります。

patchlistfile

patchlistfile を指定します。patchlistfile では、インストールするパッチのリストを含むファイルを指定します。オプションの patchlistfile が指定されていない場合、コマンドは patchdir ディレクトリ内部のすべてのパッチをインストールしようとします。patchlistfile を patchdir ディレクトリに作成し、パッチ ID を 1 行に 1 つずつリストして、インストールするパッチを示すこともできます。

-s
--software-component {all | *software-component*[,...]}

DVD イメージからインストールするソフトウェアコンポーネントを指定します。

これらのコンポーネントは、コアパッケージに追加され、ゾーンクラスタまたは Geographic Edition ソフトウェアでサポートされるデータサービスにすることができます。-s all を使用すると、それ以外のコンポーネントは指定できなくなり、すべてのデータサービスと Geographic Edition ソフトウェアがインストールされます。データサービスエージェントの場合、コンポーネント名はエージェント名になります。Geographic Edition ソフトウェアの場合は、-s geo として指定します。-s を指定しない場合は、クラスタフレームワークソフトウェアのみがインストールされます。

-v
--verbose

詳細情報を標準出力 (stdout) で表示します。

-V
--version

コマンドのバージョンを表示します。

このオプションをほかのオプション、サブコマンド、またはオペランドと一緒に指定する場合、これらはすべて無視されます。コマンドのバージョンだけが表示されます。ほかの処理は行われません。

[-x cert|ca-cert|key=*file*] ...

HTTPS 統合アーカイブの場所を使用する場合は、SSL 証明書、CA 証明書、および鍵ファイルを指定します。-x オプションは何度でも指定できます。

-Z *target-zoneclustername*
--zonecluster *target-zoneclustername*

クローンするゾーンクラスタの名前。

クローンするソースゾーンクラスタの名前を使用します。このサブコマンドを使用する前に、ソースゾーンクラスタを停止する必要があります。

-z *zone*

統合アーカイブに複数のゾーンが含まれている場合は、インストールのソースのゾーン名を指定します。

リソースとプロパティー

clzonecluster コマンドは、ゾーンクラスタの複数のリソースとプロパティーをサポートします。

clzonecluster コマンドでサポートされるリソースとプロパティーを構成するには、clzonecluster コマンドを使用する必要があります。clzonecluster コマンドでサポートされていないリソースとプロパティーの構成の詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

次のサブセクション「「リソース」」および「「プロパティー」」では、clzonecluster コマンドでサポートされるリソースとプロパティーについて説明します。

リソース

次は、リソーススコープでサポートされるリソースタイプと、詳細が見つかる場所を一覧表示します。

admin

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。このリソースは、クラスタスコープとノードスコープの両方で使用できます。このリソースは、個々の Oracle Solaris ゾーンレベルに渡されます。リソースがクラスタとノードスコープの両方で指定されている場合、ノードスコープのリソース情報は、ゾーンクラスタの特定のノードの Oracle Solaris ゾーンに渡されます。

admin リソースの auths プロパティーは、次の値のいずれかに設定できます:

clone	solaris.zone.clonefrom と同等
login	solaris.zone.login と同等
manage	solaris.zone.manage と同等

capped-cpu

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。このリソースは、クラスタスコープとノードスコープの両方で使用できます。このリソースは、個々の Oracle Solaris ゾーンレベルに渡されます。リソースがクラスタとノードスコープの両方で指定されている場合、ノードスコープのリソース情報は、ゾーンクラスタの特定のノードの Oracle Solaris ゾーンに渡されます。

capped-memory

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。このリソースは、クラスタスコープとノードスコープで使用できます。このリソースは、個々の Oracle Solaris ゾーンレベルに渡されます。リソースがクラスタとノードスコープの両方で指定されている場合、ノードスコープのリソース情報は、ゾーンクラスタの特定のノードの Oracle Solaris ゾーンに渡されます。

`dataset`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。このリソースは、クラスタスコープまたはノードスコープで使用できます。データセットはクラスタとノードスコープの両方では指定できません。

クラスタスコープのリソースは、ゾーンクラスタで高可用性 ZFS ファイルシステムに使用される ZFS データセットのエクスポートに使用されます。クラスタスコープで指定されている場合、エクスポートしたデータセットは Oracle Solaris Cluster ソフトウェアによって管理され、個別の Oracle Solaris ゾーンレベルには渡されません。データセットは、ゾーンクラスタ間で共有できません。

ノードスコープのリソースは、ローカル ZFS データセットを特定のゾーンクラスタノードにエクスポートするのに使用されます。ノードスコープで指定されている場合、エクスポートしたデータセットは Oracle Solaris Cluster ソフトウェアによって管理されず、個別の Oracle Solaris ゾーンレベルに渡されます。

`dedicated-cpu`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。各ノードのゾーンクラスタ向けに設けられた CPU の定数を使用できます。

このリソースは、クラスタスコープとノードスコープで使用できます。このリソースは、個々の Oracle Solaris ゾーンレベルに渡されます。リソースがクラスタとノードスコープの両方で指定されている場合、ノードスコープのリソース情報は、ゾーンクラスタの特定のノードの Oracle Solaris ゾーンに渡されます。

`device`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。このリソースは、個々の Oracle Solaris ゾーンレベルに渡され、クラスタスコープまたはノードスコープで指定できます。ノードスコープのリソースは、ゾーンクラスタノードに固有のデバイスを追加するのに使用されます。デバイスは、1 つのゾーンクラスタに対してのみ追加できます。クラスタスコープとノードスコープの両方に同じデバイスを追加することはできません。

`fs`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。このリソースは、クラスタスコープまたはノードスコープで指定できます。`fs` リソースはクラスタとノードスコープの両方では指定できません。

クラスタスコープのリソースは、ゾーンクラスタで使用されるファイルシステムをエクスポートする場合に一般に使用されます。エクスポートされたファイルシステムは、Oracle Solaris Cluster ソフトウェアによって管理され、`cluster-control` プロパティーを `false` に設定した `lofs` ファイルシステムを除き、個別の Oracle Solaris ゾーンレベルには渡されません。`cluster-control` プロパティーの詳細は、このマニュアルページの「リソース」セクションの `fs` に関する説明を参照してください。

ノードスコープのリソースは、ローカルファイルシステムを特定のゾーンクラスタノードにエクスポートするのに使用されます。ノードスコープで指定されている場合、エクスポートしたファイルシステムは Oracle Solaris Cluster ソフトウェアによって管理されず、個別の Oracle Solaris ゾーンレベルに渡されます。

直接マウントとループバックマウントのいずれかを使用して、ファイルシステムをゾーンクラスタにエクスポートできます。直接マウントでは、指定したファイルシステムをゾーンのルート下の場所か、パスにルートゾーンが含まれるサブディレクトリにマウントすることで、ゾーンクラスタ内からファイルシステムにアクセスできます。直接マウントでは、ファイルシステムはこのゾーンクラスタに排他的に属します。ゾーンクラスタが Oracle Solaris Trusted Extensions 上で実行される場合、読み取りおよび書き込み権限の両方を付けてマウントされるファイルについては、直接マウントの使用が必須です。ゾーンクラスタは、UFS、QFS スタンドアロンファイルシステム、QFS 共有ファイルシステム、および ZFS (データセットとしてエクスポート) の直接マウントをサポートします。

ループバックマウントは、ある場所ですでにマウントされているファイルシステムを別の場所でマウントされているように見せるためのメカニズムです。ゾーンクラスタごとに 1 つのループバックマウントを使用することで、1 つのファイルシステムを複数のゾーンクラスタにエクスポートできます。これによって、1 つのファイルシステムを複数のゾーンクラスタで共有できます。管理者は、ファイルシステムを複数のゾーンクラスタで共有する前に、セキュリティー面での影響を検討する必要があります。実際のファイルシステムのマウント方法にかかわらず、ループバックマウントでは、アクセスを読み取り専用に制限することができます。

`fs: cluster-control`

`cluster-control` プロパティーが適用されるのは、クラスタスコープで指定されたループバックマウントのみです。`cluster-control` プロパティーのデフォルト値は `true` です。

プロパティー値が `true` である場合、Oracle Solaris Cluster はこのファイルシステムを管理し、ファイルシステム情報を `zonecfg` コマンドに渡しません。Oracle Solaris Cluster は、ゾーンのブート後、必要に応じてゾーンクラスタノード内のファイルシステムをマウントまたはマウント解除します。

Oracle Solaris Cluster は、QFS 共有ファイルシステム、UFS、QFS スタンドアロンファイルシステム、および UFS 上の PxFS についてループバックマウントを管理できます。

プロパティー値が `false` である場合、Oracle Solaris Cluster はファイルシステムを管理しません。クラスタソフトウェアは、このファイルシステム情報とすべての関連情報を `zonecfg` コマンドに渡し、これにより各マシンでゾーンクラスタのゾーンが作成されます。この場合、Oracle Solaris ソフトウェアはゾーンのブート時にファイルシステムをマウントします。管理者は、このオプションを UFS ファイルシステムで使用できます。

管理者はクラスタスコープでループバックマウントを指定できます。`cluster-control` プロパティー値を `false` にしてループバックマウントを構成すると、共通のローカルディレクトリ (実行可能ファイルが格納されているディレクトリなど) の読み取り専用マウントに便利です。この情報は、実際のマウントを実行する `zonecfg` コマンドに渡されます。`cluster-control` プロパティー値を `true` にしてループバックマウントを構成すると、グローバルファイルシステム (PxFS) または共有 QFS ファイルシステムをクラスタ制御下にあるゾーンクラスタで使用できるので便利です。

QFS 共有ファイルシステム、UFS、QFS スタンドアロンファイルシステム、および ZFS は最大 1 つのゾーンクラスタに構成されます。

`net`

ネットリソースの詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

論理ホストまたは共有アドレスなど、Oracle Solaris Cluste によって管理されるネットリソースは、クラスタスコープで指定されます。Oracle RAC VIP などのアプリケーションで管理されるネットリソースは、クラスタスコープで指定されます。これらのネットワークリソースは個別の Oracle Solaris ゾーンレベルには渡されません。

管理者は、指定された IP アドレスで使用するために、NIC (Network Interface Card、ネットワークインタフェースカード) を指定できます。システムは、次の 2 つの要件を満足させる NIC を自動的に選択します。

- NIC はすでに、同じサブネットに接続されています。
- NIC は、このゾーンクラスタに対して構成されています。

`node`

ノードリソースは、次の 2 つの目的を実行します。

- スコープレベルの識別。ノードスコープで指定された任意のリソースは、もっぱらこの特定ノードにのみ属します。
- ゾーンクラスタのノードの識別。管理者は、そのマシンのグローバルクラスタの大域ゾーンを識別することによって、ゾーンが実行するマシンの場所を識別します。各ゾーンクラスタノードの IP アドレスおよび NIC の指定はオプションです。管理者は、このノードに到達するためのネットワーク情報を識別する情報も指定します。

注記 - 管理者が各ゾーンクラスタノードの IP アドレスを構成しない場合、2 つのことが起こります:

1. その特定のゾーンクラスタでは、ゾーンクラスタで使用するための NAS デバイスを構成することができません。クラスタは、NAS デバイスと通信する際にはゾーンクラスタノードの IP アドレスを使用するため、IP アドレスを持たないクラスタは、NAS デバイスのフェンシングをサポートできません。
2. クラスタソフトウェアによって、NIC の論理ホスト IP アドレスが有効化されます。

`privnet`

このリソースはノードスコープで使用できます。このリソースは、ゾーンクラスタのプライベートアダプタとして使用できるデータリンクデバイスを指定します。リソースは、ゾーンクラスタに割り当てる前に、大域ゾーンで使用できるようにする必要があります。排他的 IP ゾーンクラスタが構成されている場合は、`enable_priv_net` プロパティーがデフォルトで `true` に設定され、ゾーンクラスタのノード間のプライベートネットワーク通信が有効になります。

`add node`

```
add
 privnet
set
 physical=vnic1
end
add
 privnet
set
 physical=vnic5
end
end
```

リソースプロパティー `privnet` の順序は、ゾーンクラスタのノード間のパスを形成するのに使用されます。最初のノードに指定された最初の `privnet` アダプタは、2 番目のノードに指定された最初の `privnet` パスを使用してパスを形成しようとします。`privnet` リソースの順序は、追加および削除操作のあとも保持されます。

注記 - `privnet` リソースは、複数の排他的 IP ゾーン間で共有することはできません。特定の排他的 IP ゾーンに割り当てる必要があります。

`rctl`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。このリソースは、クラスタスコープとノードスコープの両方で使用できます。このリソースは、個々の Oracle Solaris ゾーンレベルに渡されます。リソースがクラスタとノードスコープの両方で指定されている場合、ノードスコープのリソース情報は、ゾーンクラスタの特定のノードの Oracle Solaris ゾーンに渡されます。

`sysid`

Unresolved link to " sysidcfg4" のマニュアルページを参照してください。このリソースでは、`solaris10` ゾーンクラスタのすべてのゾーンでシステム識別パラメータを指定します。

プロパティー

各リソースタイプには、1 つ以上のプロパティーがあります。次のプロパティーは、クラスタでサポートされています。

(cluster)

`admin`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`allowed-address`

アダプタ上で plumb できる IP アドレスを指定します。特定の IP アドレスのみが許可されます。このオプションのプロパティーは、ノードスコープのネットリソースに使用されます。例:

```
set allowed-address=1.2.2.3/24
```

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`attr`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。ゾーンクラスタは、`cluster` に設定されたプロパティー `name`、`boolean` に設定されたプロパティー `type`、および `true` に設定されたプロパティー `value` を使用します。これらのプロパティーは、ゾーンクラスタが `create` オプションを使用して構成されるとデフォルトで設定されます。これらのプロパティーはゾーンクラスタ構成に必須であり、変更できません。

(cluster)

`autoboot`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`bootargs`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`brand`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。サポートされているブランドタイプは、solaris、solaris10、および `labeled` ブランドのみです。

(cluster)

`cpu-shares`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`device`

Unresolved link to " zonecfg1M"。

(cluster)

`enable_priv_net`

True に設定されているとき、Oracle Solaris のプライベートネットワーク通信はゾーンクラスタのノード間で有効になります。

- `ip-type` が `shared` に設定されている場合、ゾーンクラスタノード間の通信にはグローバルクラスタのプライベートネットワークが使用されます。
- `ip-type` が `exclusive` に設定されている場合、ゾーンクラスタノード間の通信には指定された `privnet` リソースが使用されます。これらのリソースが指定されていない場合

は、グローバルクラスタのプライベートネットワーク経由で仮想ネットワークインタフェース (vnic) を作成することにより、自動的にそれらが生成されます。

Oracle Solaris Cluster のゾーンクラスタノードのプライベートホスト名および IP アドレスは、システムによって自動的に生成されます。値が False に設定されている場合、プライベートネットワークは無効になります。デフォルト値は True です。

(cluster)

`ip-type`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。サポートされている値は shared および `exclusive` のみです。

(cluster)

`limitpriv`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`max-lwps`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`max-msg-ids`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`max-sem-ids`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`max-shm-ids`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`monitor_quantum`

定量値をミリ秒単位で定義します。

(cluster)

`monitor_timeout`

モニターのタイムアウトをミリ秒単位で指定します。

(cluster)

`max-shm-memory`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`pool`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`zonename`

ゾーンクラスタの名前と、ゾーンクラスタの各ゾーンの名前。

(cluster)

`zonepath`

ゾーンクラスタの各ゾーンのゾーンパス。

`admin`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

`capped-cpu`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

`capped-memory`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

`dataset`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

`dedicated-cpu`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

`device`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

`fs`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

`inherit pkg-dir`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

`net`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

node

physical-host、hostname、net を含みます。

- physical-host – このプロパティーは、ゾーンクラスタノードをホストするグローバルクラスタノードを指定します。
- hostname – このプロパティーは、物理ホストプロパティーで指定されたグローバルクラスタノードで、ゾーンクラスタノードのパブリックホスト名を指定します。
- net – このリソースは、物理ホストで指定されたグローバルクラスタノードのゾーンクラスタノードによるパブリックネットワーク通信のネットワークアドレスと物理インタフェース名を指定します。

rctl

Unresolved link to " zonecfg1M" を参照してください。

sysid

/usr/bin/sysconfig configure コマンドを使用します。Unresolved link to " sysidcfg4" を参照してください。root_password、name_service、security_policy、system_locale、timezone、terminal、および nfs4_domain を含みます。管理者は Oracle Solaris の通常の手順に従い、あとで sysidcfg config 値をノードごとに手動で変更できます。

- root_password – このプロパティーでは、ゾーンクラスタのすべてのノードで共通の root パスワードを暗号化した値を指定します。平文のパスワードは指定しないでください。/etc/shadow からの暗号化されたパスワード文字列を使用する必要があります。これは必須プロパティーです。
- name_service – このオプションのプロパティーでは、ゾーンクラスタで使用されるネームサービスを指定します。ただし、大域ゾーンの /etc/sysidcfg ファイルの設定が無効になる場合があります。このプロパティーの設定を確実に正しいものにするには、clzonecluster コマンドを使用して手動で値を入力します。
- security_policy – この値はデフォルトでなしに設定されます。
- system_locale – この値は、デフォルトで clzonecluster コマンドの環境から取得されます。
- timezone – このプロパティーでは、ゾーンクラスタで使用されるタイムゾーンを指定します。値は、デフォルトで clzonecluster コマンドの環境から取得されます。
- terminal – 値はデフォルトで xterm に設定されます。
- nfs4_domain – 値はデフォルトで dynamic に設定されます。

すべての例で、*zoneclustername* は sczone です。最初のグローバルクラスタノードは phys-schost-1 で、2 番目のノードは phys-schost-2 です。最初のゾーンクラスタノードは zc-host-1 で、2 番目は zc-host-2 です。

例 261 新しいゾーンクラスタの作成

次の例では、2 ノード solaris10 ブランドゾーンクラスタの作成方法を示します。zpool "tank" は、高可用性 ZFS ファイルシステムとして使用されるゾーンに委任されます。ゾーンクラスタで使用できるメモリの量を制限するために、メモリキャッピングが使用されます。ルートパスワードを除き、デフォルトのシステム識別値が使用されます。

```
phys-schost-1# clzonecluster configure sczone
sczone: No such zone cluster configured
Use 'create' to begin configuring a new zone cluster.
clzc:sczone> create -b
clzc:sczone> set zonepath=/zones/timuzc
clzc:sczone> set brand=solaris10
clzc:sczone> set autoboot=true
clzc:sczone> set bootargs="-m verbose"
clzc:sczone> set limitpriv="default,proc_priocntl,proc_clock_highres"

clzc:sczone> set enable_priv_net=true
clzc:sczone> set ip-type=shared
clzc:sczone> add dataset
clzc:sczone:dataset> set name=tank
clzc:sczone:dataset> end
clzc:sczone> add capped-memory
clzc:sczone:capped-memory> set physical=3G
clzc:sczone:capped-memory> end
clzc:sczone> add rctl
clzc:sczone:rctl> set name=zone.max-swap
clzc:sczone:rctl> add value (priv=privileged,limit=4294967296,action=deny)

clzc:sczone:rctl> end
clzc:sczone> add rctl
clzc:sczone:rctl> set name=zone.max-locked-memory
clzc:sczone:rctl> add value (priv=privileged,limit=3221225472,action=deny)

clzc:sczone:rctl> end
clzc:sczone> add attr
clzc:sczone:attr> set name=cluster
clzc:sczone:attr> set type=boolean
clzc:sczone:attr> set value=true
clzc:sczone:attr> end
clzc:sczone> add node
clzc:sczone:node> set physical-host=ptimu1
clzc:sczone:node> set hostname=zc-host-1
clzc:sczone:node> add net
clzc:sczone:node:net> set address=vztimu1a
```

```
clzc:sczone:node:net> set physical=sc_ipmp0
clzc:sczone:node:net> end
clzc:sczone:node> end
clzc:sczone> add node
clzc:sczone:node> set physical-host=ptimu2
clzc:sczone:node> set hostname=zc-host-2
clzc:sczone:node> add net
clzc:sczone:node:net> set address=vztimu2a
clzc:sczone:node:net> set physical=sc_ipmp0
clzc:sczone:node:net> end
clzc:sczone:node> end
clzc:sczone> add fs
clzc:sczone:fs> set dir=/opt/local
clzc:sczone:fs> set special=/usr/local
clzc:sczone:fs> set type=lofs
clzc:sczone:fs> add options [ro,nodevices]
clzc:sczone:fs> set cluster-control=false
clzc:sczone:fs> end
clzc:sczone> add sysid
clzc:sczone> set root_password=ZiitH.NOLOrRg
clzc:sczone> set name_service="NIS{domain_name=mycompany.com name_server=
     ns101c-90(10.100.10.10)}"
clzc:sczone> set nfs4_domain=dynamic
clzc:sczone> set security_policy=NONE
clzc:sczone> set system_locale=C
clzc:sczone> set terminal=xterms
clzc:sczone> set timezone=US/Pacific
clzc:sczone> end
```

create サブコマンド (前述の create -b サブコマンドではなく) を使用する場合は、デフォルトのテンプレートが使用され、attr プロパティーがすでに設定されています。

ゾーンクラスタが構成されます。次のコマンドがインストールされ、広域クラスタノードからゾーンクラスタをブートします。

```
phys-schost-1# clzonecluster install -a absolute_path_to_archive install sczone

phys-schost-1# clzonecluster boot sczone
```

例 262 統合アーカイブからのゾーンクラスタの作成

次の例では、統合アーカイブからゾーンクラスタを作成してインストールする方法を示します。統合アーカイブは、大域ゾーン、非大域ゾーン、またはゾーンクラスタノードから作成できます。統合アーカイブからのゾーンクラスタの構成およびインストールのために、クローンアーカイブと復旧用のアーカイブの両方がサポートされています。統合アーカイブをクラスタ化されていないゾーンから作成する場合は、プロパティー enable_priv_net=true を設定する必要があります。また、ゾーンのプロパティーも必要に応じて変更するようにしてください。

```
phys-schost-1# clzonecluster configure sczone
```

```
sczone: No such zone cluster configured
Use 'create' to begin configuring a new zone cluster.
clzc:sczone> create -a absolute_path_to_archive -z archived_zone_1
clzc:sczone> set zonepath=/zones/sczone

clzc:sczone> set enable_priv_net=true
clzc:sczone> set ip-type=shared

clzc:sczone> add attr
clzc:sczone:attr> set name=cluster
clzc:sczone:attr> set type=boolean
clzc:sczone:attr> set value=true
clzc:sczone:attr> end

clzc:sczone> add node
clzc:sczone:node> set physical-host=psoft1
clzc:sczone:node> set hostname=zc-host-1
clzc:sczone:node> add net
clzc:sczone:node:net> set address=vzsoft1a
clzc:sczone:node:net> set physical=sc_ipmp0
clzc:sczone:node:net> end
clzc:sczone:node> end
clzc:sczone> add node
clzc:sczone:node> set physical-host=psoft2
clzc:sczone:node> set hostname=zc-host-2
clzc:sczone:node> add net
clzc:sczone:node:net> set address=vzsoft2a
clzc:sczone:node:net> set physical=sc_ipmp0
clzc:sczone:node:net> end
clzc:sczone:node> end
```

ゾーンクラスタが構成されます。次のコマンドは、グローバルクラスタノード上の統合アーカイブからゾーンクラスタをインストールします。

```
phys-schost-1# clzonecluster install -a absolute_path_to_archive -z archived-zone sczone
```

これで、ゾーンクラスタがインストールされました。次のコマンドは、そのゾーンクラスタをブートします。

```
phys-schost-1# clzonecluster boot sczone
```

例 263 既存のゾーンクラスタの変更

次の例では、例 1 で作成されたゾーンクラスタの構成の変更方法を示しています。追加のパブリック IP アドレスが phys-schost-2 のゾーンクラスタノードに追加されます。

高可用性ファイルシステムとして使用するために、UFS ファイルシステムがゾーンクラスタにエクスポートされます。これは、UFS ファイルシステムが Oracle Solaris Volume Manager メタデバイスで作成されていることを前提としています。

```
phys-schost-1# clzonecluster configure sczone
clzc:sczone> add device
clzc:sczone:device> set match=/dev/md/1/dsk/d100
clzc:sczone:device> end
clzc:sczone> add device
clzc:sczone:device> set match=/dev/md/oraset/dsk/d100
clzc:sczone:device> end
clzc:sczone> select node physical-host=phys-schost-2
clzc:sczone:node> add net
clzc:sczone:node:net> set address=192.168.0.3/24
clzc:sczone:node:net> set physical=bge0
clzc:sczone:node:net> end
clzc:sczone:node> end
clzc:sczone> add fs
clzc:sczone:fs> set dir=/qfs/ora_home
clzc:sczone:fs> set special=oracle_home
clzc:sczone:fs> set type=samfs
clzc:sczone:fs> end
clzc:sczone> exit
```

例 264 テンプレートとして既存のゾーンクラスタを使用した新規ゾーンクラスタの作成

次の例では、テンプレートとして例 1 で作成した sczone ゾーンクラスタを使用して、sczone1 というゾーンクラスタを作成する方法を示しています。新規ゾーンクラスタの構成は、元のゾーンクラスタと同じになります。競合を避けるために、新規ゾーンクラスタの一部のプロパティーを変更する必要があります。管理者が特定リソースを指定せずにリソースタイプを削除すると、システムはそのタイプのすべてのリソースを削除します。たとえば、**remove net** を指定すると、すべてのネットリソースが削除されます。

```
phys-schost-1# clzonecluster configure sczone1
sczone1: No such zone cluster configured
Use 'create' to begin configuring a new zone cluster.

clzc:sczone1> create -t sczone
clzc:sczone1>set zonepath=/zones/sczone1

clzc:sczone1> select node physical-host=phys-schost-1
clzc:sczone1:node> set hostname=zc-host-3
clzc:sczone1:node> select net address=zc-host-1
clzc:sczone1:node:net> set address=zc-host-3
clzc:sczone1:node:net> end
clzc:sczone1:node> end
clzc:sczone1> select node physical-host=phys-schost-2
clzc:sczone1:node> set hostname=zc-host-4
clzc:sczone1:node> select net address=zc-host-2
clzc:sczone1:node:net> set address=zc-host-4
clzc:sczone1:node:net> end
clzc:sczone1:node> remove net address=192.168.0.3/24
clzc:sczone1:node> end
clzc:sczone1> remove dataset name=tank/home
clzc:sczone1> remove net
```

```
clzc:sczone1> remove device
clzc:sczone1> remove fs dir=/qfs/ora_home
clzc:sczone1> exit
```

次のオペランドがサポートされています。

zoneclustername
: ゾーンクラスタの名前。新規ゾーンクラスタの名前を指定しています。*zoneclustername* オペランドはすべてのサブコマンドに対してサポートされています。

`+`
: クラスタ内のすべてのノードです。`+` オペランドは、サブコマンドのサブセットに対してのみサポートされています。

このコマンドセットにあるすべてのコマンドの終了ステータスコードの完全なリストについては、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (`CL_NOERR`) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

`0 CL_NOERR`
: エラーなし。

  実行したコマンドは正常に終了しました。

`1 CL_ENOMEM`
: 十分なスワップ空間がありません。

  クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

`3 CL_EINVAL`
: 無効な引数。

  コマンドを間違って入力したか、`-i` オプションで指定したクラスタ構成情報の構文が間違っていました。

`18 CL_EINTERNAL`
: 内部エラーが発生しました。

`36 CL_ENOENT`
: そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

37 CL_EOP

操作が許可されていません

サポートされていない構成に対する操作を実行しようとしたか、サポートされていない操作を実行しました。

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

185 ページのclnode(1CL), 551 ページのcluster(1CL), 19 ページのIntro(1CL), 819 ページのscinstall(1M), Unresolved link to " zoneadm1M", Unresolved link to " zonecfg1M", Unresolved link to " clconfiguration(5CL)"

スーパーユーザーはこのコマンドのすべての形式を実行できます。

すべてのユーザーがこのコマンドに -? (ヘルプ) オプションまたは -V (バージョン) オプションを指定して実行できます。

スーパーユーザー以外のユーザーが clzonecluster コマンドにサブコマンドを付けて実行するには、RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
boot	solaris.cluster.admin
check	solaris.cluster.read
clone	solaris.cluster.admin
configure	solaris.cluster.admin
delete	solaris.cluster.admin

サブコマンド	RBAC の承認
export	solaris.cluster.admin
halt	solaris.cluster.admin
install	solaris.cluster.admin
list	solaris.cluster.read
monitor	solaris.cluster.modify
move	solaris.cluster.admin
ready	solaris.cluster.admin
reboot	solaris.cluster.admin
show	solaris.cluster.read
status	solaris.cluster.read
uninstall	solaris.cluster.admin
unmonitor	solaris.cluster.modify
verify	solaris.cluster.admin

名前

clzonecluster, clzc — ゾーンクラスタの作成と管理

```
/usr/cluster/bin/clzonecluster [subcommand] -?

/usr/cluster/bin/clzonecluster -V

/usr/cluster/bin/clzonecluster subcommand [options] -v
     [zone-cluster-name]

/usr/cluster/bin/clzonecluster apply [-n node-name[,…]] [-d]
     {+ | zone-cluster-name […]}

/usr/cluster/bin/clzonecluster boot [-n node-name[,…]] [-o]
     {+ | zone-cluster-name […]}

/usr/cluster/bin/clzonecluster clone -Z target-zone-cluster-name
     [-m method][-n node-name[,…]] {source-zone-cluster-name}

/usr/cluster/bin/clzonecluster configure [-f command-file]
     zone-cluster-name

/usr/cluster/bin/clzonecluster delete [-F] zone-cluster-name

/usr/cluster/bin/clzonecluster export [-f command-file]
     zone-cluster-name

/usr/cluster/bin/clzonecluster halt [-n node-name[,…]]
     {+ | zone-cluster-name}

/usr/cluster/bin/clzonecluster install [-c config_profile.xml]
     [-M manifest.xml] zone-cluste-rname

/usr/cluster/bin/clzonecluster install [-n node-name]
     -a absolute_path_to_archive [-x cert|ca-cert|key=file]…
     -z zone zone-cluster-name

/usr/cluster/bin/clzonecluster install [-n node-name]
     -d absolute_root_path zone-cluster-name

/usr/cluster/bin/clzonecluster install-cluster
     [-d dvd-image] [-n node-name[,…]]
     [-p patchdir=patch-dir[,patchlistfile=file-name]]
     -s software-component[,…]] [-v] zone-cluster-name

/usr/cluster/bin/clzonecluster install-cluster
     [-p patchdir=patch-dir[,patchlistfile=file-name]
     [-n node-name[,…]] [-v] zone-cluster-name

/usr/cluster/bin/clzonecluster list [+ | zone-cluster-name […]]
```

```
/usr/cluster/bin/clzonecluster move -f zone-path zone-cluster-name

/usr/cluster/bin/clzonecluster ready [-n node-name[,…]]
     {+ | zone-cluster-name […]}

/usr/cluster/bin/clzonecluster reboot [-n node-name[,…]] [-o]
     {+ | zone-cluster-name […]}

/usr/cluster/bin/clzonecluster set {-p name=value}
     [-p name=value] […] [zone-cluster-name]

/usr/cluster/bin/clzonecluster show [+ | zone-cluster-name […]]

/usr/cluster/bin/clzonecluster show-rev [-v] [-n node-name[,…]]
     [+ | zone-cluster-name …]

/usr/cluster/bin/clzonecluster status [+ | zone-cluster-name […]]

/usr/cluster/bin/clzonecluster uninstall [-F] [-n node-name
     [,…]] zone-cluster-name

/usr/cluster/bin/clzonecluster verify [-n node-name[,…]]
     {+ | zone-cluster-name […]}
```

clzonecluster コマンドは、Oracle Solaris Cluster 構成用のゾーンクラスタを作成し変更します。clzc コマンドは clzonecluster コマンドの短い形式です。コマンドはまったく同じです。clzonecluster コマンドはクラスタ対応で、管理の単一ソースをサポートします。1 つのノードからコマンドのすべての形式を実行して、単一のゾーン・クラスタノードまたはすべてノードに影響を与えることができます。

subcommand を省略できるのは、*options* が -? オプションまたは -V オプションの場合のみです。

サブコマンドは list、show、status サブコマンドを除き、1 つ以上のオペランドを必要とします。ただし、多くのサブコマンドはプラス記号オペランド (+) を受け入れて、そのサブコマンドをすべてのアプリケーションオブジェクトに適用します。clzonecluster コマンドはゾーンクラスタの任意のノードで実行し、そのゾーンクラスタの任意またはすべてに影響を与えることができます。

このコマンドの各オプションには長形式と短形式があります。各オプションの両方の形式は、オプション に説明とともに記載されています。

注記 - ゾーンクラスタが作成されたあとにゾーンクラスタ名を変更することはできません。

サポートされるサブコマンドには次のものがあります。

apply

構成の変更をゾーンクラスタに適用します。

apply サブコマンドは、ゾーンクラスタの永続的なライブ再構成を格納します。clzonecluster configure を実行して構成の変更を行なってから、apply サブコマンドを実行して、その変更を特定のゾーンクラスタに適用するようにしてください。apply サブコマンドは、-n オプションを使用して、再構成が適用されるノードのリストを指定します。

apply サブコマンドは、グローバルクラスタノードからのみ使用できます。

boot

ゾーンクラスタのブート。

boot サブコマンドはゾーンクラスタをブートします。boot サブコマンドは -n フラグを使用して、ノードの指定したリストのゾーンクラスタをブートします。

グローバルクラスタノードから boot サブコマンドのみを使用できます。

clone

ゾーンクラスタのクローン。

clone コマンドは、インストールされた既存のゾーンクラスタをコピーして、ゾーンクラスタをインストールします。このサブコマンドは、ゾーンクラスタのインストールに代わるものです。clone サブコマンドは、それ自身で新しいゾーンクラスタを作成しません。クローニングに使用するソースゾーンクラスタが、クローニングする前に*インストールされた*状態であることを確認します。最初に configure サブコマンドを使用して、新しいゾーンクラスタを作成する必要があります。次に、clone サブコマンドを使用して、クローニングされた構成を新しいゾーンクラスタに適用します。

グローバルクラスタノードから clone サブコマンドのみを使用できます。

configure

対話型ユーティリティーを起動して、solaris10 または labeled ブランドゾーンクラスタを構成します。

configure サブコマンドは zonecfg コマンドを使用して、指定されたそれぞれのマシンでゾーンを構成します。configure サブコマンドにより、ゾーンクラスタの各ノードに適用するプロパティーを指定できます。これらのプロパティーは、個別ゾーンの zonecfg コマンドによって確立された場合と同じ意味を持ちます。configure サブコマンドは zonecfg コマンドには分からないプロパティーの構成をサポートします。-f オプションを指定しない場合、configure サブコマンドは対話型シェルを起動します。-f オプションは、その引数としてコマンドファイルを取ります。configure サブコマンドはこのファイルを使用して、ゾーンクラスタを非対話型で作成または変更します。

また、configure サブコマンドを使用すると、統合アーカイブを使用してゾーンクラスタを構成し、復旧用のアーカイブまたはクローンアーカイブを選択することもできます。-a *archive* オプションを create サブコマンドとともに使用します。例:

```
# /usr/cluster/bin/clzc configure sczone1
sczone1: No such zone cluster configured
```

```
Use 'create' to begin configuring a new zone cluster.
clzc:sczone1> create -a archive -z archived zone
```

configure サブコマンドは、グローバルクラスタノードからのみ使用できます。詳細は、Unresolved link to " Oracle Solaris Cluster ソフトウェアのインストール "を参照してください。

solaris10 ブランドゾーンクラスタを指定するには、ゾーンクラスタの構成時にデフォルトのテンプレートを使用できます。デフォルトのテンプレートは、/etc/cluster/zone_cluster/ORCLcls10default.xml にあります。-t オプションを使用すると、デフォルトの solaris10 ゾーンクラスタテンプレート、またはクラスタにある別の既存の solaris10 ゾーンクラスタを指定できます。別の solaris10 ゾーンクラスタを指定すると、ゾーンクラスタの構成は指定したゾーンクラスタからインポートされます。verify または commit 操作が失敗しないように、sysid プロパティーに root パスワードを指定する必要もあります。次のコマンドを入力してテンプレートを適用します:

```
# /usr/cluster/bin/clzc configure sczone2
sczone2: No such zone cluster configured
Use 'create' to begin configuring a new zone cluster.
clzc:sczone2> create -t ORCLcls10default
clzc:sczone2> info
zonename: sczone2
zonepath:
autoboot: true
hostid:
brand: solaris10
```

configure コマンドの対話型と非対話型形式はどちらも、ゾーンクラスタ構成を編集するための複数のサブコマンドをサポートしています。使用可能な構成サブコマンドのリストについては、Unresolved link to " zonecfg1M" を参照してください。

configure ユーティリティーにより、ゾーンクラスタの構成を作成または変更できます。ゾーンクラスタ構成は、多くのリソースタイプおよびプロパティーで構成されます。configure ユーティリティーは *scope* のコンセプトを使用して、サブコマンドが適用される場所を決定します。configure ユーティリティーで使用されるスコープには、クラスタ、リソース、およびノード固有のリソースの 3 つのレベルがあります。デフォルトのスコープはクラスタです。次のリストは、スコープの 3 つのレベルを説明しています。

- クラスタスコープ - ゾーンクラスタ全体に影響を及ぼすプロパティー。*zoneclustername* が sczone の場合、clzonecluster コマンドの対話型シェルは次のように見えます。

  ```
  clzc:sczone>
  ```

- ノードスコープ - ノードリソーススコープ内部に入れ子になった特殊なリソーススコープ。ノードスコープ内部で設定すると、ゾーンクラスタの特定ノードに影響を与えます。たとえば、ゾーンクラスタの特定ノードにネットリソースを追加できます。clzonecluster コマンドの対話型シェルは、次のように見えます。

  ```
  clzc:sczone:node:net>
  ```

- リソーススコープ - 1 つの特定リソースに適用されるプロパティー。リソーススコーププロンプトには、付加されたリソースタイプの名前が付いています。たとえば、`clzonecluster` コマンドの対話型シェルは次のように見えます。

  ```
  clzc:sczone:net>
  ```

`delete`

特定ゾーンクラスタの削除。

このサブコマンドは、特定のゾーンクラスタのリソースグループを削除します。ワイルドカードオペランド (*) を使用するとき、`delete` コマンドはグローバルクラスタで構成されたゾーンクラスタを削除します。`delete` サブコマンドを実行する前に、ゾーンクラスタを構成状態にする必要があります。`delete` コマンドで `-F` オプションを使用した場合は、どのような状態にあるゾーンクラスタに対しても削除が試みられます。

`delete` サブコマンドはグローバルクラスタノードからのみ使用できます。

`export`

ゾーンクラスタの構成をコマンドファイルにエクスポートします。

エクスポートされた *commandfile* は、`configure` サブコマンドの入力として使用できます。必要に応じてファイルを変更し、作成する構成を反映させます。詳細は、Unresolved link to " clconfiguration(5CL)" のマニュアルページを参照してください。

`export` サブコマンドは、グローバルクラスタノードからのみ使用できます。

`halt`

ゾーンクラスタまたはゾーンクラスタの特定ノードの停止。

特定のゾーンクラスタを指定するとき、`halt` サブコマンドは特定のゾーンクラスタに対してのみ適用されます。ゾーンクラスタ全体、またはゾーンクラスタの特定ノードのみを停止できます。ゾーンクラスタを指定しない場合、`halt` サブコマンドはすべてのゾーンクラスタに適用されます。指定されたマシンですべてのゾーンクラスタを停止することもできます。

`halt` サブコマンドは `-n` オプションを使用して、特定ノードのゾーンクラスタを停止します。デフォルトで、`halt` サブコマンドはすべてのノードのすべてのゾーンクラスタを停止します。ゾーン名の代わりに `+` オペランドを指定する場合、すべてのゾーンクラスタが停止されます。

`halt` サブコマンドは、グローバルクラスタノードからのみ使用できます。

`install`

ゾーンクラスタのインストール。

このサブコマンドは、ゾーンクラスタをインストールします。

`install -M` *manifest*`.xml` オプションを使用した場合は、指定したマニフェストが、ゾーンクラスタのすべてのノード上のインストールに使用されます。マニフェストファイルには、`certificate_file`、`key_file`、パブリッシャー、任意の追加パッケージなどの、インストールのために管理者に必要な `solaris` パッケージの情報が記述されます。*manifest*`.xml`

ファイルには、ゾーンクラスタのインストールのための Oracle Solaris Cluster グループパッケージ `ha-cluster-full`、`ha-cluster-framework-full`、`ha-cluster-data-services-full`、または `ha-cluster-minimal` も指定する必要があります。Automated Installer マニフェストの詳細は、Unresolved link to " Oracle Solaris 11.2 システムのインストール のカスタム AI マニフェストの作成"を参照してください。

`-M` オプション (デフォルト) を使用しない場合は、`/usr/share/auto_install/manifest/zone_default.xml` にある自動インストーラマニフェストがインストールに使用されます。この `zone_default.xml` マニフェストが使用されている場合、発行元のゾーンクラスタノードの大域ゾーンにインストールされているすべての `ha-cluster/*` パッケージが、ゾーンクラスタのすべてのノードにインストールされます。ゾーンクラスタのインストール時にカスタムマニフェストを使用する場合、Oracle Solaris Cluster グループパッケージを指定しないと、インストールが失敗します。

インストールするすべてのゾーンクラスタノードのベースとなる大域ゾーンは、`install` サブコマンドを実行するゾーンクラスタノードの大域ゾーンにインストールされるのと同じ Oracle Solaris Cluster パッケージのセットを持っている必要があります。ゾーンクラスタのインストールは、この要件を満たさないゾーンクラスタノードでは失敗する場合があります。

`install` サブコマンドはグローバルクラスタノードからのみ使用できます。`-M` および `-c` オプションは、`solaris` ブランドゾーンクラスタのみに使用できます。

ゾーンクラスタのブランドが `solaris10` の場合は、`-a` または `-d` オプションを使用する必要があります。

`-a` archive

`solaris` または `solaris10` ブランドゾーンクラスタの統合アーカイブ、`solaris10` ブランドゾーンクラスタの `flar` アーカイブの場所、またはインストールに使用する Oracle Solaris 10 イメージアーカイブの絶対パス。サポートされているアーカイブタイプの詳細は、Unresolved link to " solaris105" のマニュアルページを参照してください。アーカイブの絶対パスは、ゾーンクラスタがインストールされるクラスタのすべての物理ノードでアクセスできるようにしてください。統合アーカイブのインストールでは、復旧用のアーカイブまたはクローンアーカイブを使用できます。

`-d` path

インストール済みの Oracle Solaris 10 システムのルートディレクトリへのパス。パスは、ゾーンクラスタがインストールされるクラスタのすべての物理ノードでアクセスできるようにしてください。

[`-x` cert|ca-cert|key=*file*]...

HTTPS 統合アーカイブの場所を使用する場合は、SSL 証明書、認証局 (CA) 証明書、および鍵ファイルを指定します。`-x` オプションは何度でも指定できます。

`-z` *zone*

統合アーカイブに複数のゾーンが含まれている場合は、構成またはインストールのソースのゾーン名を指定します。

同じアーカイブまたはインストール済みの Oracle Solaris 10 システムは、ゾーンクラスタにあるすべての `solaris10` ブランドゾーンのインストールのソースとして使用されます。インストールすると、ソースアーカイブまたはインストール済みの Oracle Solaris 10 システムのシステム識別パラメータは、ゾーンクラスタの構成時に `sysid` リソースタイプで指定したシステム識別パラメータでオーバーライドされます。

`install-cluster`

`install-cluster` サブコマンドは、Oracle Solaris 10 OS をサポートする Oracle Solaris Cluster ソフトウェアを `solaris10` ブランドゾーンクラスタノードにインストールします。インストールされているソフトウェアには、コアパッケージ、クラスタソフトウェアコンポーネント (ゾーンクラスタおよび Geographic Edition ソフトウェアでサポートされるエージェントなど)、およびパッチが含まれます。

注記 - `install-cluster` サブコマンドは、Oracle Solaris Cluster バージョン 3.3 または 3.3 5/11 ソフトウェアを `solaris10` ブランドゾーンクラスタノードにインストールすることをサポートしていません。`solaris10` ブランドゾーンクラスタのサポートされているリリースの詳細は、Unresolved link to " Oracle Solaris Cluster 4.2 リリースノート "を参照してください。

このサブコマンドを使用するのは、クラスタソフトウェアがインストールされていない Oracle Solaris 10 システムに `solaris10` ブランドソーンをインストールする場合です。このサブコマンドを使用するには、`clzonecluster install` コマンドを使用して Oracle Solaris 10 システムの Solaris OS ソフトウェアを `solaris10` ゾーンにインストールしておき、そのゾーンを `online` 状態にブートする必要があります。

クラスタコアパッケージが `solaris10` ブランドゾーンにまだインストールされていない場合は、クラスタリリース DVD ディレクトリに `-d` オプション、クラスタソフトウェアコンポーネントに `-s` オプション、パッチに `-p` オプションを指定することで、コアパッケージ、すべてのクラスタソフトウェアコンポーネント、およびパッチをすべて同時にインストールできます。クラスタソフトウェアコンポーネントおよびパッチをインストールするためのオプションはオプションです。

クラスタコアパッケージをすでにインストールしている場合でも、このサブコマンドを使用して、ゾーンクラスタでサポートされるパッチとクラスタソフトウェアコンポーネントをインストールできます。パッチ情報が指定されている場合は、ゾーンクラスタのクラスタノードを `-o` オプションで `offline-running` 状態にブートする必要があります。

`solaris10` ブランドゾーンクラスタは、共有 IP ゾーンタイプのみをサポートします (排他的 IP および共有 IP ゾーンクラスタの詳細は、Unresolved link to " Oracle Solaris Cluster ソフトウェアのインストール " を参照)。

このサブコマンドは、大域ゾーンからのみ実行できます。

`list`

構成されたゾーンクラスタの名前の表示。

このサブコマンドは、クラスタで構成されたゾーンクラスタの名前を報告します。

- グローバルクラスタノードから `list` サブコマンドを実行する場合、サブコマンドはグローバルクラスタのすべてのゾーンクラスタのリストを表示します。
- ゾーンクラスタノードから `list` サブコマンドを実行すると、サブコマンドはゾーンクラスタの名前のみを表示します。

ゾーンクラスタが構成された場所でノードのリストを表示するには、`-v` オプションを使用します。

`move`

ゾーンパスを新しいゾーンパスに移動。

このサブコマンドはゾーンパスを新しいゾーンパスに移動します。

`move` サブコマンドは、グローバルクラスタノードからのみ使用できます。

`ready`

アプリケーションに対してゾーンを準備。

このサブコマンドは、アプリケーションを実行するためのゾーンを準備します。

`ready` サブコマンドはグローバルクラスタノードからのみ使用できます。

`reboot`

ゾーンクラスタのリブート。

このサブコマンドはゾーンクラスタをリブートし、`halt` サブコマンドの実行に似ており、次に `boot` サブコマンドが続きます。詳細については、`halt` サブコマンドおよび `boot` サブコマンドを参照してください。

`reboot` サブコマンドはグローバルクラスタノードからのみ使用できます。

`set`

`-p` オプションで指定したプロパティーの値をゾーンクラスタに設定します。`set` サブコマンドは大域ゾーンまたはゾーンクラスタから使用できます。設定できるプロパティーについては、オプション セクションの `-p` の説明を参照してください。

`show`

ゾーンクラスタのプロパティーの表示。

ゾーンクラスタのプロパティーには、ゾーンクラスタ名、ブランド、IP タイプ、ノードリスト、ゾーンパス、および許可されたアドレスが含まれます。`show` サブコマンドはゾーンクラスタから実行しますが、特定のゾーンクラスタにのみ適用されます。ゾーンクラスタからこのサブコマンドを使用するとき、ゾーンパスは常に `/` です。ゾーンクラスタ名が指定される場合、このコマンドはそのゾーンクラスタにのみ適用されます。

`show` サブコマンドはグローバルクラスタノードからのみ使用できます。

`show-rev`

ゾーンクラスタの各ノードのクラスタのリリース情報を表示します。

この機能は、ゾーンクラスタにインストールされているリリースバージョンおよびパッチを一覧表示する場合に便利です。例:

```
# clzonecluster show-rev
=== Zone Clusters ===
Zone Cluster Name:   zc1
Release at vznode1a on node pnode1:  3.3u2_40u1_zc:2012-04-01
Release at vznode2a on node pnode2:  3.3u2_40u1_zc:2012-04-01
```

show-rev サブコマンドは、グローバルクラスタノードまたはゾーンクラスタノードから使用できます。

status

ゾーンクラスタノードがゾーンクラスタのメンバーであるかどうかを判断し、そのゾーンクラスタが solaris、solaris10、labeled ブランドのいずれであるかを表示します。

ゾーンの状態は、Configured、Installed、Ready、Running、Shutting Down、Unavailable のいずれかになります。グローバルクラスタ内のすべてのゾーンクラスタの状況は表示されているので、仮想クラスタの状態を見ることができます。

ゾーン活動を確認するには、zoneadm コマンドを代わりに使用します。

status サブコマンドはグローバルクラスタノードからのみ使用できます。

uninstall

ゾーンクラスタのアンインストール。

このサブコマンドは、ゾーンクラスタをアンインストールします。uninstall サブコマンドは zoneadm コマンドを使用します。

uninstall サブコマンドはグローバルクラスタノードからのみ使用できます。

verify

指定された情報の構文が正しいかどうかをチェックします。

このサブコマンドはゾーンクラスタの各ノードの zoneadm verify コマンドを呼び出して、各ゾーンクラスタのメンバーを安全にインストールできるようにします。詳細は、Unresolved link to " zoneadm1M" を参照してください。

verify サブコマンドはグローバルクラスタノードからのみ使用できます。

注記 - このセクションでは、各オプションの短い形式と長い形式の両方が示されています。

次のオプションがサポートされています。

-?
--help

ヘルプ情報を表示します。

このオプションは *subcommand* の使用に関係なく指定できます。

subcommand を指定しない場合、使用可能なすべてのサブコマンドのリストが表示されます。

subcommand を指定する場合、そのサブコマンドの使用法が表示されます。

このオプションとその他のオプションを指定すると、その他のオプションは無視されます。

`-a` *absolute_path_to_archive zoneclustername*

インストール済みの Oracle Solaris 10 システム、インストール済みの Oracle Solaris 10 ネイティブゾーン、または `solaris10` ブランドゾーンの `flash_archive`、`cpio`、`pax`、`xustar`、`zfs archive`、または `level 0 ufsdump` のパスを指定します。また、統合アーカイブの絶対パスも指定できます。詳細は、Unresolved link to " solaris105"、Unresolved link to " flash_archive4"、Unresolved link to " cpio1"、および Unresolved link to " pax1" のマニュアルページを参照してください。

`-c` *config_profile*`.xml`
`--configprofile` *config_profile*`.xml`

`solaris` ブランドゾーンクラスタの構成プロファイルテンプレートを指定します。リポジトリからのインストール後に、テンプレートはシステム構成情報をゾーンクラスタのすべてのノードに適用します。*config_profile*`.xml` が指定されていない場合は、各ノードの大域ゾーンから `zlogin -C` *zoneclustername* コマンドを実行して、各ゾーンクラスタノードを手動で構成する必要があります。すべてのプロファイルには `.xml` 拡張子が必要です。

`-c` オプションは、構成プロファイルテンプレートのゾーンクラスタノードのホスト名を置き換えます。プロファイルは、ゾーンクラスタノードのブート後に、ゾーンクラスタノードに適用されます。

`-d` *absolute_root_path*
`--dirpath` *dirpatch*

`-d` オプションを `cluster` サブコマンドとともに使用すると、インストール済みの Oracle Solaris 10 システムのルートディレクトリへのパスが指定されます。パスは、ゾーンクラスタがインストールされるクラスタのすべての物理ノードでアクセスできるようにしてください。

`-d`
`--dvd-directory` *dvd-directory*

DVD イメージディレクトリを指定します。

`-d` オプションを `install-cluster` サブコマンドとともに使用すると、`solaris10` ブランドゾーンをサポートする Oracle Solaris Cluster リリースの DVD イメージディレクトリが指定されます。DVD イメージには、コアパッケージと、ゾーンクラスタおよび Geographic Edition ソフトウェアでサポートされているほかのクラスタソフトウェアコンポーネント (エージェントなど) が含まれます。DVD ディレクトリは、コマンドを実行するノードの大域ゾーンからアクセスできるようにする必要があります。

`-d`
`--dry_run`

`apply` サブコマンドで `-d` オプションが使用されている場合、再構成は予行演習モードで実行されます。予行演習モードでは構成が変更されず、実行中のゾーンはそのままの状態に

なります。予行演習モードは、実際の再構成によって実行されるアクションを確認するために使用します。

-f{*commandfile* | *zonepath*}
--file-argument {*commandfile* | *zonepath*}

`configure` サブコマンドとともに使用するとき、`-f` オプションはコマンドファイルの引数を指定します。たとえば、`clzonecluster configure -f` *commandfile* となります。`move` サブコマンドとともに使用するとき、`-f` オプションは *zonepath* を指定します。

-F

`delete` および `uninstall` 操作の間、`-F` オプションを使用できます。`-F` オプションは `Are you sure you want to` *do this operation* `[y/n]?` という質問を強制的に抑制します。

-m *method*
--method *method*

ゾーンクラスタをクローニングするには、`-m` オプションを使用します。クローニングの唯一有効なメソッドは、`copy` コマンドです。`clone` サブコマンドを実行する前に、ソースゾーンクラスタを停止する必要があります。

-M *manifest*`.xml`
--manifest *manifest*`.xml`

`solaris` ブランドゾーンクラスタのすべてのノードにマニフェストを指定するには、`-M` オプションを使用します。このマニフェストは、Oracle Solaris パッケージ情報と、ゾーンクラスタのインストールのための Oracle Solaris Cluster パッケージを指定します。

-n *nodename*[…]
--nodelist *nodename*[,…]

サブコマンドのノードリストを指定します。

たとえば、`clzonecluster boot -n` *phys-schost-1, phys-schost-2 zoneclustername*.

-o
--offline

ゾーンクラスタを `offline-running` モードにブートまたはリブートします。

`offline-running` モードになるのは、ゾーンクラスタノードがゾーンクラスタメンバシップから除外されているが、Oracle Solaris ゾーン状態が実行中の場合です。ゾーンクラスタは、物理クラスタとブートモード (クラスタまたは非クラスタモード) を共有するため、オフライン状態は非クラスタモードのクラスタとは異なります。

ゾーンクラスタをオフライン実行モードにブートするには、次を入力します。

`clzonecluster boot [-n` *phys-schost-1*`,...] [-o]` *zoneclustername*

ゾーンクラスタをオフライン実行モードにリブートするには、次を入力します。

`clzonecluster reboot [-n` *phys-schost-1*`,...] [-o]` *zoneclustername*

offline-running ゾーンクラスタを online-running モードにブートするには、-o オプションを指定せずに clzonecluster reboot コマンドを実行します。

-p *name*=*value*
--property=*name*=*value*
--property *name*=*value*

-p オプションは、install-cluster サブコマンドおよび set サブコマンドとともに使用します。-p を指定した install-cluster サブコマンドの使用方法については、-p patchdir=*patchdir*[,patchlistfile=*patchlistfile*] の説明を参照してください。

-p オプションは set サブコマンドとともに使用して、プロパティーの値を指定します。-p *name*=*value* は複数回指定できます。

このオプションは set サブコマンドと一緒に使用して、次のプロパティーを変更します:

resource_security

プログラムの実行のセキュリティーポリシーを RGM リソース別に指定します。resource_security で使用可能な値は、SECURE、WARN、OVERRIDE、または COMPATIBILITY です。

Start や Validate などのリソースメソッドは、常に root として実行されます。メソッドの実行可能ファイルに root 以外の所有権、あるいは group または world 書き込み権がある場合は、セキュアでない状態になりますこの場合、resource_security プロパティーが SECURE に設定されていると、リソースメソッドの実行は実行時に失敗し、エラーが返されます。resource_security がその他の設定であれば、リソースメソッドは実行を許可され、警告メッセージが表示されます。最大限のセキュリティーを確保するため、resource_security を SECURE に設定してください。

resource_security 設定では、application_user リソースプロパティーを宣言するリソースタイプの動作も変更します。詳細は、1287 ページのr_properties(5) のマニュアルページの「application_user」セクションを参照してください。

-p patchdir=*patchdir*[,patchlistfile=*patchlistfile*]
--patch-specification=patchdir=*patchdir*[,patchlistfile=*patchlistfile*]
--patch-specification patchdir=*patchdir*[,patchlistfile=*patchlistfile*]

-p オプションで指定される patchdir および patchlistfile プロパティーは、install-cluster サブコマンドとともにのみ使用されます。コアパッケージのインストール後にパッチをインストールする場合は、パッチを適用するために、ゾーンクラスタを offline-running 状態にブートする必要があります。

-p *name*= *value* は複数回指定できます。

patchdir

solaris10 ブランドゾーンに適用する Oracle Solaris Cluster パッチが含まれているディレクトリを指定します。patchdir ディレクトリは必須で、ゾーンクラスタのすべてのノードで、solaris10 ブランドゾーン内部からアクセスできる必要があります。

patchlistfile

patchlistfile を指定します。patchlistfile では、インストールするパッチのリストを含むファイルを指定します。オプションの patchlistfile が指定されていない場合、コマンドは patchdir ディレクトリ内部のすべてのパッチをインストールしようとします。patchlistfile を patchdir ディレクトリに作成し、パッチ ID を 1 行に 1 つずつリストして、インストールするパッチを示すこともできます。

-s
--software-component {all | *software-component*[,...]}

DVD イメージからインストールするソフトウェアコンポーネントを指定します。

これらのコンポーネントは、コアパッケージに追加され、ゾーンクラスタまたは Geographic Edition ソフトウェアでサポートされるデータサービスにすることができます。-s all を使用すると、それ以外のコンポーネントは指定できなくなり、すべてのデータサービスと Geographic Edition ソフトウェアがインストールされます。データサービスエージェントの場合、コンポーネント名はエージェント名になります。Geographic Edition ソフトウェアの場合は、-s geo として指定します。-s を指定しない場合は、クラスタフレームワークソフトウェアのみがインストールされます。

-v
--verbose

詳細情報を標準出力 (stdout) で表示します。

-V
--version

コマンドのバージョンを表示します。

このオプションをほかのオプション、サブコマンド、またはオペランドと一緒に指定する場合、これらはすべて無視されます。コマンドのバージョンだけが表示されます。ほかの処理は行われません。

[-x cert|ca-cert|key=*file*] ...

HTTPS 統合アーカイブの場所を使用する場合は、SSL 証明書、CA 証明書、および鍵ファイルを指定します。-x オプションは何度でも指定できます。

-Z *target-zoneclustername*
--zonecluster *target-zoneclustername*

クローンするゾーンクラスタの名前。

クローンするソースゾーンクラスタの名前を使用します。このサブコマンドを使用する前に、ソースゾーンクラスタを停止する必要があります。

-z *zone*

統合アーカイブに複数のゾーンが含まれている場合は、インストールのソースのゾーン名を指定します。

リソースとプロパティー

clzonecluster コマンドは、ゾーンクラスタの複数のリソースとプロパティーをサポートします。

clzonecluster コマンドでサポートされるリソースとプロパティーを構成するには、clzonecluster コマンドを使用する必要があります。clzonecluster コマンドでサポートされていないリソースとプロパティーの構成の詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

次のサブセクション「「リソース」」および「「プロパティー」」では、clzonecluster コマンドでサポートされるリソースとプロパティーについて説明します。

リソース

次は、リソーススコープでサポートされるリソースタイプと、詳細が見つかる場所を一覧表示します。

admin

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。このリソースは、クラスタスコープとノードスコープの両方で使用できます。このリソースは、個々の Oracle Solaris ゾーンレベルに渡されます。リソースがクラスタとノードスコープの両方で指定されている場合、ノードスコープのリソース情報は、ゾーンクラスタの特定のノードの Oracle Solaris ゾーンに渡されます。

admin リソースの auths プロパティーは、次の値のいずれかに設定できます:

clone	solaris.zone.clonefrom と同等
login	solaris.zone.login と同等
manage	solaris.zone.manage と同等

capped-cpu

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。このリソースは、クラスタスコープとノードスコープの両方で使用できます。このリソースは、個々の Oracle Solaris ゾーンレベルに渡されます。リソースがクラスタとノードスコープの両方で指定されている場合、ノードスコープのリソース情報は、ゾーンクラスタの特定のノードの Oracle Solaris ゾーンに渡されます。

capped-memory

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。このリソースは、クラスタスコープとノードスコープで使用できます。このリソースは、個々の Oracle Solaris ゾーンレベルに渡されます。リソースがクラスタとノードスコープの両方で指定されている場合、ノードスコープのリソース情報は、ゾーンクラスタの特定のノードの Oracle Solaris ゾーンに渡されます。

`dataset`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。このリソースは、クラスタスコープまたはノードスコープで使用できます。データセットはクラスタとノードスコープの両方では指定できません。

クラスタスコープのリソースは、ゾーンクラスタで高可用性 ZFS ファイルシステムに使用される ZFS データセットのエクスポートに使用されます。クラスタスコープで指定されている場合、エクスポートしたデータセットは Oracle Solaris Cluster ソフトウェアによって管理され、個別の Oracle Solaris ゾーンレベルには渡されません。データセットは、ゾーンクラスタ間で共有できません。

ノードスコープのリソースは、ローカル ZFS データセットを特定のゾーンクラスタノードにエクスポートするのに使用されます。ノードスコープで指定されている場合、エクスポートしたデータセットは Oracle Solaris Cluster ソフトウェアによって管理されず、個別の Oracle Solaris ゾーンレベルに渡されます。

`dedicated-cpu`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。各ノードのゾーンクラスタ向けに設けられた CPU の定数を使用できます。

このリソースは、クラスタスコープとノードスコープで使用できます。このリソースは、個々の Oracle Solaris ゾーンレベルに渡されます。リソースがクラスタとノードスコープの両方で指定されている場合、ノードスコープのリソース情報は、ゾーンクラスタの特定のノードの Oracle Solaris ゾーンに渡されます。

`device`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。このリソースは、個々の Oracle Solaris ゾーンレベルに渡され、クラスタスコープまたはノードスコープで指定できます。ノードスコープのリソースは、ゾーンクラスタノードに固有のデバイスを追加するのに使用されます。デバイスは、1 つのゾーンクラスタに対してのみ追加できます。クラスタスコープとノードスコープの両方に同じデバイスを追加することはできません。

`fs`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。このリソースは、クラスタスコープまたはノードスコープで指定できます。`fs` リソースはクラスタとノードスコープの両方では指定できません。

クラスタスコープのリソースは、ゾーンクラスタで使用されるファイルシステムをエクスポートする場合に一般に使用されます。エクスポートされたファイルシステムは、Oracle Solaris Cluster ソフトウェアによって管理され、`cluster-control` プロパティーを `false` に設定した `lofs` ファイルシステムを除き、個別の Oracle Solaris ゾーンレベルには渡されません。`cluster-control` プロパティーの詳細は、このマニュアルページの「リソース」セクションの `fs` に関する説明を参照してください。

ノードスコープのリソースは、ローカルファイルシステムを特定のゾーンクラスタノードにエクスポートするのに使用されます。ノードスコープで指定されている場合、エクスポートしたファイルシステムは Oracle Solaris Cluster ソフトウェアによって管理されず、個別の Oracle Solaris ゾーンレベルに渡されます。

直接マウントとループバックマウントのいずれかを使用して、ファイルシステムをゾーンクラスタにエクスポートできます。直接マウントでは、指定したファイルシステムをゾーンのルート下の場所か、パスにルートゾーンが含まれるサブディレクトリにマウントすることで、ゾーンクラスタ内からファイルシステムにアクセスできます。直接マウントでは、ファイルシステムはこのゾーンクラスタに排他的に属します。ゾーンクラスタが Oracle Solaris Trusted Extensions 上で実行される場合、読み取りおよび書き込み権限の両方を付けてマウントされるファイルについては、直接マウントの使用が必須です。ゾーンクラスタは、UFS、QFS スタンドアロンファイルシステム、QFS 共有ファイルシステム、および ZFS (データセットとしてエクスポート) の直接マウントをサポートします。

ループバックマウントは、ある場所ですでにマウントされているファイルシステムを別の場所でマウントされているように見せるためのメカニズムです。ゾーンクラスタごとに 1 つのループバックマウントを使用することで、1 つのファイルシステムを複数のゾーンクラスタにエクスポートできます。これによって、1 つのファイルシステムを複数のゾーンクラスタで共有できます。管理者は、ファイルシステムを複数のゾーンクラスタで共有する前に、セキュリティー面での影響を検討する必要があります。実際のファイルシステムのマウント方法にかかわらず、ループバックマウントでは、アクセスを読み取り専用に制限することができます。

`fs: cluster-control`

`cluster-control` プロパティーが適用されるのは、クラスタスコープで指定されたループバックマウントのみです。`cluster-control` プロパティーのデフォルト値は `true` です。

プロパティー値が `true` である場合、Oracle Solaris Cluster はこのファイルシステムを管理し、ファイルシステム情報を `zonecfg` コマンドに渡しません。Oracle Solaris Cluster は、ゾーンのブート後、必要に応じてゾーンクラスタノード内のファイルシステムをマウントまたはマウント解除します。

Oracle Solaris Cluster は、QFS 共有ファイルシステム、UFS、QFS スタンドアロンファイルシステム、および UFS 上の PxFS についてループバックマウントを管理できます。

プロパティー値が `false` である場合、Oracle Solaris Cluster はファイルシステムを管理しません。クラスタソフトウェアは、このファイルシステム情報とすべての関連情報を `zonecfg` コマンドに渡し、これにより各マシンでゾーンクラスタのゾーンが作成されます。この場合、Oracle Solaris ソフトウェアはゾーンのブート時にファイルシステムをマウントします。管理者は、このオプションを UFS ファイルシステムで使用できます。

管理者はクラスタスコープでループバックマウントを指定できます。`cluster-control` プロパティー値を `false` にしてループバックマウントを構成すると、共通のローカルディレクトリ (実行可能ファイルが格納されているディレクトリなど) の読み取り専用マウントに便利です。この情報は、実際のマウントを実行する `zonecfg` コマンドに渡されます。`cluster-control` プロパティー値を `true` にしてループバックマウントを構成すると、グローバルファイルシステム (PxFS) または共有 QFS ファイルシステムをクラスタ制御下にあるゾーンクラスタで使用できるので便利です。

QFS 共有ファイルシステム、UFS、QFS スタンドアロンファイルシステム、および ZFS は最大 1 つのゾーンクラスタに構成されます。

`net`

ネットリソースの詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

論理ホストまたは共有アドレスなど、Oracle Solaris Cluste によって管理されるネットリソースは、クラスタスコープで指定されます。Oracle RAC VIP などのアプリケーションで管理されるネットリソースは、クラスタスコープで指定されます。これらのネットワークリソースは個別の Oracle Solaris ゾーンレベルには渡されません。

管理者は、指定された IP アドレスで使用するために、NIC (Network Interface Card、ネットワークインタフェースカード) を指定できます。システムは、次の 2 つの要件を満足させる NIC を自動的に選択します。

- NIC はすでに、同じサブネットに接続されています。
- NIC は、このゾーンクラスタに対して構成されています。

`node`

ノードリソースは、次の 2 つの目的を実行します。

- スコープレベルの識別。ノードスコープで指定された任意のリソースは、もっぱらこの特定ノードにのみ属します。
- ゾーンクラスタのノードの識別。管理者は、そのマシンのグローバルクラスタの大域ゾーンを識別することによって、ゾーンが実行するマシンの場所を識別します。各ゾーンクラスタノードの IP アドレスおよび NIC の指定はオプションです。管理者は、このノードに到達するためのネットワーク情報を識別する情報も指定します。

注記 - 管理者が各ゾーンクラスタノードの IP アドレスを構成しない場合、2 つのことが起こります:

1. その特定のゾーンクラスタでは、ゾーンクラスタで使用するための NAS デバイスを構成することができません。クラスタは、NAS デバイスと通信する際にはゾーンクラスタノードの IP アドレスを使用するため、IP アドレスを持たないクラスタは、NAS デバイスのフェンシングをサポートできません。
2. クラスタソフトウェアによって、NIC の論理ホスト IP アドレスが有効化されます。

`privnet`

このリソースはノードスコープで使用できます。このリソースは、ゾーンクラスタのプライベートアダプタとして使用できるデータリンクデバイスを指定します。リソースは、ゾーンクラスタに割り当てる前に、大域ゾーンで使用できるようにする必要があります。排他的 IP ゾーンクラスタが構成されている場合は、`enable_priv_net` プロパティーがデフォルトで `true` に設定され、ゾーンクラスタのノード間のプライベートネットワーク通信が有効になります。

`add node`

```
add
 privnet
set
 physical=vnic1
end
add
 privnet
set
 physical=vnic5
end
end
```

リソースプロパティー `privnet` の順序は、ゾーンクラスタのノード間のパスを形成するのに使用されます。最初のノードに指定された最初の `privnet` アダプタは、2 番目のノードに指定された最初の `privnet` パスを使用してパスを形成しようとします。`privnet` リソースの順序は、追加および削除操作のあとも保持されます。

注記 - `privnet` リソースは、複数の排他的 IP ゾーン間で共有することはできません。特定の排他的 IP ゾーンに割り当てる必要があります。

`rctl`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。このリソースは、クラスタスコープとノードスコープの両方で使用できます。このリソースは、個々の Oracle Solaris ゾーンレベルに渡されます。リソースがクラスタとノードスコープの両方で指定されている場合、ノードスコープのリソース情報は、ゾーンクラスタの特定のノードの Oracle Solaris ゾーンに渡されます。

`sysid`

Unresolved link to " sysidcfg4" のマニュアルページを参照してください。このリソースでは、`solaris10` ゾーンクラスタのすべてのゾーンでシステム識別パラメータを指定します。

プロパティー

各リソースタイプには、1 つ以上のプロパティーがあります。次のプロパティーは、クラスタでサポートされています。

(cluster)

`admin`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`allowed-address`

アダプタ上で plumb できる IP アドレスを指定します。特定の IP アドレスのみが許可されます。このオプションのプロパティーは、ノードスコープのネットリソースに使用されます。例:

```
set allowed-address=1.2.2.3/24
```

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`attr`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。ゾーンクラスタは、`cluster` に設定されたプロパティー `name`、`boolean` に設定されたプロパティー `type`、および `true` に設定されたプロパティー `value` を使用します。これらのプロパティーは、ゾーンクラスタが `create` オプションを使用して構成されるとデフォルトで設定されます。これらのプロパティーはゾーンクラスタ構成に必須であり、変更できません。

(cluster)

`autoboot`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`bootargs`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`brand`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。サポートされているブランドタイプは、solaris、solaris10、および `labeled` ブランドのみです。

(cluster)

`cpu-shares`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`device`

Unresolved link to " zonecfg1M"。

(cluster)

`enable_priv_net`

True に設定されているとき、Oracle Solaris のプライベートネットワーク通信はゾーンクラスタのノード間で有効になります。

- `ip-type` が `shared` に設定されている場合、ゾーンクラスタノード間の通信にはグローバルクラスタのプライベートネットワークが使用されます。
- `ip-type` が `exclusive` に設定されている場合、ゾーンクラスタノード間の通信には指定された `privnet` リソースが使用されます。これらのリソースが指定されていない場合

は、グローバルクラスタのプライベートネットワーク経由で仮想ネットワークインタフェース (vnic) を作成することにより、自動的にそれらが生成されます。

Oracle Solaris Cluster のゾーンクラスタノードのプライベートホスト名および IP アドレスは、システムによって自動的に生成されます。値が False に設定されている場合、プライベートネットワークは無効になります。デフォルト値は True です。

(cluster)

`ip-type`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。サポートされている値は shared および `exclusive` のみです。

(cluster)

`limitpriv`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`max-lwps`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`max-msg-ids`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`max-sem-ids`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`max-shm-ids`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`monitor_quantum`

定量値をミリ秒単位で定義します。

(cluster)

`monitor_timeout`

モニターのタイムアウトをミリ秒単位で指定します。

(cluster)

`max-shm-memory`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`pool`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

(cluster)

`zonename`

ゾーンクラスタの名前と、ゾーンクラスタの各ゾーンの名前。

(cluster)

`zonepath`

ゾーンクラスタの各ゾーンのゾーンパス。

`admin`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

`capped-cpu`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

`capped-memory`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

`dataset`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

`dedicated-cpu`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

`device`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

`fs`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

`inherit pkg-dir`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

`net`

詳細は、Unresolved link to " zonecfg1M" のマニュアルページを参照してください。

node

physical-host、hostname、net を含みます。

- physical-host – このプロパティーは、ゾーンクラスタノードをホストするグローバルクラスタノードを指定します。
- hostname – このプロパティーは、物理ホストプロパティーで指定されたグローバルクラスタノードで、ゾーンクラスタノードのパブリックホスト名を指定します。
- net – このリソースは、物理ホストで指定されたグローバルクラスタノードのゾーンクラスタノードによるパブリックネットワーク通信のネットワークアドレスと物理インタフェース名を指定します。

rctl

Unresolved link to " zonecfg1M" を参照してください。

sysid

/usr/bin/sysconfig configure コマンドを使用します。Unresolved link to " sysidcfg4" を参照してください。root_password、name_service、security_policy、system_locale、timezone、terminal、および nfs4_domain を含みます。管理者は Oracle Solaris の通常の手順に従い、あとで sysidcfg config 値をノードごとに手動で変更できます。

- root_password – このプロパティーでは、ゾーンクラスタのすべてのノードで共通の root パスワードを暗号化した値を指定します。平文のパスワードは指定しないでください。/etc/shadow からの暗号化されたパスワード文字列を使用する必要があります。これは必須プロパティーです。
- name_service – このオプションのプロパティーでは、ゾーンクラスタで使用されるネームサービスを指定します。ただし、大域ゾーンの /etc/sysidcfg ファイルの設定が無効になる場合があります。このプロパティーの設定を確実に正しいものにするには、clzonecluster コマンドを使用して手動で値を入力します。
- security_policy – この値はデフォルトでなしに設定されます。
- system_locale – この値は、デフォルトで clzonecluster コマンドの環境から取得されます。
- timezone – このプロパティーでは、ゾーンクラスタで使用されるタイムゾーンを指定します。値は、デフォルトで clzonecluster コマンドの環境から取得されます。
- terminal – 値はデフォルトで xterm に設定されます。
- nfs4_domain – 値はデフォルトで dynamic に設定されます。

このコマンドセットにあるすべてのコマンドの終了ステータスコードの完全なリストについては、19 ページのIntro(1CL) のマニュアルページを参照してください。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

0 CL_NOERR
: エラーなし
: 実行したコマンドは正常に終了しました。

1 CL_ENOMEM
: 十分なスワップ空間がありません。
: クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL
: 無効な引数
: コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

6 CL_EACCESS
: アクセス権がありません
: 指定したオブジェクトにアクセスできません。このコマンドを実行するには、スーパーユーザーまたは RBAC アクセスが必要である可能性があります。詳細は、Unresolved link to " su1M"、および Unresolved link to " rbac5" のマニュアルページを参照してください。

18 CL_EINTERNAL
: 内部エラーが発生しました
: 内部エラーは、ソフトウェアの欠陥またはその他の欠陥を示しています。

35 CL_EIO
: I/O エラー
: 物理的な入出力エラーが発生しました。

36 CL_ENOENT
: そのようなオブジェクトはありません。
: 次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

38 CL_EBUSY

オブジェクトはビジーです

アクティブなクラスタノードへの最後のクラスタインターコネクトパスからケーブルを取り外そうとしました。または、参照を削除していないクラスタ構成からノードを削除しようとしました。

39 CL_EEXIST

オブジェクトは存在します。

指定したデバイス、デバイスグループ、クラスタインターコネクトコンポーネント、ノード、クラスタ、リソース、リソースタイプ、リソースグループ、またはプライベート文字列はすでに存在します。

41 CL_ETYPE

未知のタイプ

-t または -p オプションで指定したタイプは存在しません。

すべての例で、*zoneclustername* は sczone です。最初のグローバルクラスタノードは phys-schost-1 で、2 番目のノードは phys-schost-2 です。最初のゾーンクラスタノードは zc-host-1 で、2 番目は zc-host-2 です。

例 265 新しいゾーンクラスタの作成

次の例では、2 ノード solaris10 ブランドゾーンクラスタの作成方法を示します。zpool "tank" は、高可用性 ZFS ファイルシステムとして使用されるゾーンに委任されます。ゾーンクラスタで使用できるメモリの量を制限するために、メモリキャッピングが使用されます。ルートパスワードを除き、デフォルトのシステム識別値が使用されます。

```
phys-schost-1# clzonecluster configure sczone
sczone: No such zone cluster configured
Use 'create' to begin configuring a new zone cluster.
clzc:sczone> create -b
clzc:sczone> set zonepath=/zones/timuzc
clzc:sczone> set brand=solaris10
clzc:sczone> set autoboot=true
clzc:sczone> set bootargs="-m verbose"
clzc:sczone> set limitpriv="default,proc_priocntl,proc_clock_highres"

clzc:sczone> set enable_priv_net=true
clzc:sczone> set ip-type=shared
clzc:sczone> add dataset
```

```
clzc:sczone:dataset> set name=tank
clzc:sczone:dataset> end
clzc:sczone> add capped-memory
clzc:sczone:capped-memory> set physical=3G
clzc:sczone:capped-memory> end
clzc:sczone> add rctl
clzc:sczone:rctl> set name=zone.max-swap
clzc:sczone:rctl> add value (priv=privileged,limit=4294967296,action=deny)

clzc:sczone:rctl> end
clzc:sczone> add rctl
clzc:sczone:rctl> set name=zone.max-locked-memory
clzc:sczone:rctl> add value (priv=privileged,limit=3221225472,action=deny)

clzc:sczone:rctl> end
clzc:sczone> add attr
clzc:sczone:attr> set name=cluster
clzc:sczone:attr> set type=boolean
clzc:sczone:attr> set value=true
clzc:sczone:attr> end
clzc:sczone> add node
clzc:sczone:node> set physical-host=ptimu1
clzc:sczone:node> set hostname=zc-host-1
clzc:sczone:node> add net
clzc:sczone:node:net> set address=vztimu1a
clzc:sczone:node:net> set physical=sc_ipmp0
clzc:sczone:node:net> end
clzc:sczone:node> end
clzc:sczone> add node
clzc:sczone:node> set physical-host=ptimu2
clzc:sczone:node> set hostname=zc-host-2
clzc:sczone:node> add net
clzc:sczone:node:net> set address=vztimu2a
clzc:sczone:node:net> set physical=sc_ipmp0
clzc:sczone:node:net> end
clzc:sczone:node> end
clzc:sczone> add fs
clzc:sczone:fs> set dir=/opt/local
clzc:sczone:fs> set special=/usr/local
clzc:sczone:fs> set type=lofs
clzc:sczone:fs> add options [ro,nodevices]
clzc:sczone:fs> set cluster-control=false
clzc:sczone:fs> end
clzc:sczone> add sysid
clzc:sczone> set root_password=ZiitH.NOLOrRg
clzc:sczone> set name_service="NIS{domain_name=mycompany.com name_server=
     ns101c-90(10.100.10.10)}"
clzc:sczone> set nfs4_domain=dynamic
clzc:sczone> set security_policy=NONE
clzc:sczone> set system_locale=C
clzc:sczone> set terminal=xterms
clzc:sczone> set timezone=US/Pacific
clzc:sczone> end
```

create サブコマンド (前述の create -b サブコマンドではなく) を使用する場合は、デフォルトのテンプレートが使用され、attr プロパティーがすでに設定されています。

ゾーンクラスタが構成されます。次のコマンドがインストールされ、広域クラスタノードからゾーンクラスタをブートします。

```
phys-schost-1# clzonecluster install -a absolute_path_to_archive install sczone

phys-schost-1# clzonecluster boot sczone
```

例 266 統合アーカイブからのゾーンクラスタの作成

次の例では、統合アーカイブからゾーンクラスタを作成してインストールする方法を示します。統合アーカイブは、大域ゾーン、非大域ゾーン、またはゾーンクラスタノードから作成できます。統合アーカイブからのゾーンクラスタの構成およびインストールのために、クローンアーカイブと復旧用のアーカイブの両方がサポートされています。統合アーカイブをクラスタ化されていないゾーンから作成する場合は、プロパティー enable_priv_net=true を設定する必要があります。また、ゾーンのプロパティーも必要に応じて変更するようにしてください。

```
phys-schost-1# clzonecluster configure sczone
sczone: No such zone cluster configured
Use 'create' to begin configuring a new zone cluster.
clzc:sczone> create -a absolute_path_to_archive -z archived_zone_1
clzc:sczone> set zonepath=/zones/sczone

clzc:sczone> set enable_priv_net=true
clzc:sczone> set ip-type=shared

clzc:sczone> add attr
clzc:sczone:attr> set name=cluster
clzc:sczone:attr> set type=boolean
clzc:sczone:attr> set value=true
clzc:sczone:attr> end

clzc:sczone> add node
clzc:sczone:node> set physical-host=psoft1
clzc:sczone:node> set hostname=zc-host-1
clzc:sczone:node> add net
clzc:sczone:node:net> set address=vzsoft1a
clzc:sczone:node:net> set physical=sc_ipmp0
clzc:sczone:node:net> end
clzc:sczone:node> end
clzc:sczone> add node
clzc:sczone:node> set physical-host=psoft2
clzc:sczone:node> set hostname=zc-host-2
clzc:sczone:node> add net
clzc:sczone:node:net> set address=vzsoft2a
clzc:sczone:node:net> set physical=sc_ipmp0
clzc:sczone:node:net> end
```

```
clzc:sczone:node> end
```

ゾーンクラスタが構成されます。次のコマンドは、グローバルクラスタノード上の統合アーカイブからゾーンクラスタをインストールします。

```
phys-schost-1# clzonecluster install -a absolute_path_to_archive -z archived-zone sczone
```

これで、ゾーンクラスタがインストールされました。次のコマンドは、そのゾーンクラスタをブートします。

```
phys-schost-1# clzonecluster boot sczone
```

例 267 既存のゾーンクラスタの変更

次の例では、例 1 で作成されたゾーンクラスタの構成の変更方法を示しています。追加のパブリック IP アドレスが phys-schost-2 のゾーンクラスタノードに追加されます。

高可用性ファイルシステムとして使用するために、UFS ファイルシステムがゾーンクラスタにエクスポートされます。これは、UFS ファイルシステムが Oracle Solaris Volume Manager メタデバイスで作成されていることを前提としています。

```
phys-schost-1# clzonecluster configure sczone
clzc:sczone> add device
clzc:sczone:device> set match=/dev/md/1/dsk/d100
clzc:sczone:device> end
clzc:sczone> add device
clzc:sczone:device> set match=/dev/md/oraset/dsk/d100
clzc:sczone:device> end
clzc:sczone> select node physical-host=phys-schost-2
clzc:sczone:node> add net
clzc:sczone:node:net> set address=192.168.0.3/24
clzc:sczone:node:net> set physical=bge0
clzc:sczone:node:net> end
clzc:sczone:node> end
clzc:sczone> add fs
clzc:sczone:fs> set dir=/qfs/ora_home
clzc:sczone:fs> set special=oracle_home
clzc:sczone:fs> set type=samfs
clzc:sczone:fs> end
clzc:sczone> exit
```

例 268 テンプレートとして既存のゾーンクラスタを使用した新規ゾーンクラスタの作成

次の例では、テンプレートとして例 1 で作成した sczone ゾーンクラスタを使用して、sczone1 というゾーンクラスタを作成する方法を示しています。新規ゾーンクラスタの構成は、元のゾーンクラスタと同じになります。競合を避けるために、新規ゾーンクラスタの一部のプロパティーを変更する必要があります。管理者が特定リソースを指定せずにリソースタイプを削除すると、システム

はそのタイプのすべてのリソースを削除します。たとえば、**remove net** を指定すると、すべてのネットリソースが削除されます。

```
phys-schost-1# clzonecluster configure sczone1
sczone1: No such zone cluster configured
Use 'create' to begin configuring a new zone cluster.

clzc:sczone1> create -t sczone
clzc:sczone1>set zonepath=/zones/sczone1

clzc:sczone1> select node physical-host=phys-schost-1
clzc:sczone1:node> set hostname=zc-host-3
clzc:sczone1:node> select net address=zc-host-1
clzc:sczone1:node:net> set address=zc-host-3
clzc:sczone1:node:net> end
clzc:sczone1:node> end
clzc:sczone1> select node physical-host=phys-schost-2
clzc:sczone1:node> set hostname=zc-host-4
clzc:sczone1:node> select net address=zc-host-2
clzc:sczone1:node:net> set address=zc-host-4
clzc:sczone1:node:net> end
clzc:sczone1:node> remove net address=192.168.0.3/24
clzc:sczone1:node> end
clzc:sczone1> remove dataset name=tank/home
clzc:sczone1> remove net
clzc:sczone1> remove device
clzc:sczone1> remove fs dir=/qfs/ora_home
clzc:sczone1> exit
```

次のオペランドがサポートされています。

zoneclustername
: ゾーンクラスタの名前。新規ゾーンクラスタの名前を指定しています。*zoneclustername* オペランドはすべてのサブコマンドに対してサポートされています。

+
: クラスタ内のすべてのノードです。+ オペランドは、サブコマンドのサブセットに対してのみサポートされています。

このコマンドセットにあるすべてのコマンドの終了ステータスコードの完全なリストについては、19 ページのIntro(1CL) のマニュアルページを参照してください。

指定したすべてのオペランドでコマンドが成功すると、コマンドはゼロ (CL_NOERR) を返します。あるオペランドでエラーが発生すると、コマンドはオペランドリストの次のオペランドを処理します。戻り値は常に、最初に発生したエラーを反映します。

このコマンドは、次の終了ステータスコードを返します。

0 CL_NOERR

エラーなし。

実行したコマンドは正常に終了しました。

1 CL_ENOMEM

十分なスワップ空間がありません。

クラスタノードがスワップメモリーまたはその他のオペレーティングシステムリソースを使い果たしました。

3 CL_EINVAL

無効な引数。

コマンドを間違って入力したか、-i オプションで指定したクラスタ構成情報の構文が間違っていました。

18 CL_EINTERNAL

内部エラーが発生しました。

36 CL_ENOENT

そのようなオブジェクトはありません。

次のいずれかの理由のために、指定したオブジェクトを見つけることができません。

- オブジェクトが存在しません。
- -o オプションで作成しようとした構成ファイルへのパスのディレクトリが存在しません。
- -i オプションでアクセスしようとした構成ファイルにエラーが含まれています。

37 CL_EOP

操作が許可されていません

サポートされていない構成に対する操作を実行しようとしたか、サポートされていない操作を実行しました。

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

185 ページのclnode(1CL), 551 ページのcluster(1CL), 19 ページのIntro(1CL), 819 ページのscinstall(1M), Unresolved link to " zoneadm1M", Unresolved link to " zonecfg1M", Unresolved link to " clconfiguration(5CL)"

スーパーユーザーはこのコマンドのすべての形式を実行できます。

すべてのユーザーがこのコマンドに -? (ヘルプ) オプションまたは -V (バージョン) オプションを指定して実行できます。

スーパーユーザー以外のユーザーが clzonecluster コマンドにサブコマンドを付けて実行するには、RBAC の承認が必要です。次の表を参照してください。

サブコマンド	RBAC の承認
boot	solaris.cluster.admin
check	solaris.cluster.read
clone	solaris.cluster.admin
configure	solaris.cluster.admin
delete	solaris.cluster.admin
export	solaris.cluster.admin
halt	solaris.cluster.admin
install	solaris.cluster.admin
list	solaris.cluster.read
monitor	solaris.cluster.modify
move	solaris.cluster.admin
ready	solaris.cluster.admin
reboot	solaris.cluster.admin
show	solaris.cluster.read
status	solaris.cluster.read
uninstall	solaris.cluster.admin
unmonitor	solaris.cluster.modify
verify	solaris.cluster.admin

OSC4 1ha

名前

rt_callbacks — Oracle Solaris Cluster リソースとしてサービスを管理するためのコールバックインタフェース

method-path `-R` *resource* `-T` *type* `-G` *group* `[-Z` *zonename*`]`

validate-path `[-c | -u] -R` *resource* `-T` *type* `-G` *group* `[-r` *prop*`=`*val*`] [-x` *prop*`=`*val*`] [-g` *prop*`=`*val*`] [-Z` *zonename*`] [-X` *prop*`{`*nodeid*`}=`*val*`]`

Oracle Solaris Cluster リソースタイプのコールバックインタフェースは、Resource Group Manager (RGM) がサービスをクラスタリソースとして制御するためのインタフェースを定義します。リソースタイプの実装者は、コールバックメソッドとして機能するプログラムまたはスクリプトを提供します。

method-path
: リソースタイプ登録ファイルで宣言されているプログラムへのパスです。このプログラムは、`START`、`STOP`、`INIT`、`FINI`、`BOOT`、`PRENET_START`、`POSTNET_STOP`、`MONITOR_START`、`MONITOR_STOP`、`MONITOR_CHECK`、または `UPDATE` のリソースタイプのコールバックメソッドの 1 つとして `clresourcetype` コマンドで登録されます。333 ページの`clresourcetype`(1CL) および 1251 ページの`rt_reg`(4) のマニュアルページを参照してください。

validate-path
: リソースタイプ登録ファイル内でリソースタイプの `VALIDATE` メソッドとして宣言されているプログラムへのパスです。このプログラムは `clresourcetype` コマンドで登録されます。

コールバックメソッドは指定されたオプションを渡され、クラスタ上のサービスの操作を制御する特定のアクションを実行します。

リソースタイプの開発者は、リソースタイプ登録ファイルでコールバックメソッドプログラムへのパスを宣言します。クラスタ管理者は、`clresourcetype` コマンドを使用してクラスタ構成にリソースタイプを登録します。クラスタ管理者は、登録後、登録されたリソースタイプを使用してリソースを作成できます。これらのリソースは、RGM が管理するリソースグループ内に構成されます。

RGM は、管理するリソースグループ内のリソースのコールバックメソッドを自動的に呼び出して、イベントに応答します。コールバックメソッドは、リソースによって表されるサービスに関して特定のアクションを実行すると想定されています。これらのアクションの例には、クラスタノード上でのサービスの停止と開始があります。

コールバックメソッドから返される終了ステータスコードは、コールバックメソッドの成功または失敗を RGM に通知します。メソッドが失敗またはリソース状態の障害を報告すると、RGM がアクションを実行します。この結果、クラスタ管理者は障害に気付き、適切なアクションを実行できます。

次のオプションがサポートされています。

-c

クラスタ管理者が*すべての*リソースとリソース グループのプロパティーの初期設定を検証するためのリソースを作成する際に、このメソッドを呼び出すことを指定します。

RGM は -c オプションと -u オプションのいずれかを指定しますが、同時に両方のオプションを指定することはありません。

クラスタ管理者がリソースを作成して、VALIDATE メソッドが呼び出されると、*すべての*システム定義、拡張機能、およびリソースグループのプロパティーが VALIDATE メソッドに渡されます。クラスタ管理者がリソースを更新して、VALIDATE メソッドが呼び出されると、更新されているプロパティーのみが VALIDATE メソッドに渡されます。

-g *prop* =*val*

VALIDATE メソッドに渡されるリソースグループプロパティーの値を指定します。

prop リソースグループプロパティーの名前です。

val クラスタ管理者がリソースを作成または更新した際にメソッドに渡される値です。

-G *group*

リソースを構成するリソースグループの名前を指定します。

-r *prop* =*val*

VALIDATE メソッドに渡されるシステムが定義したリソースプロパティーの値を指定します。

prop システムが定義したリソースプロパティーの名前です。

val クラスタ管理者がリソースを作成または更新した際にメソッドに渡される値です。

-R *resource*

メソッドが呼び出されるリソース名を指定します。

-T *type*

リソースが属するリソースタイプの名前を指定します。

-u

クラスタ管理者が既存のリソースまたは既存のリソースグループを更新するとメソッドが呼び出されることを指定します。

RGM は -c オプションと -u オプションのいずれかを指定しますが、同時に両方のオプションを指定することはありません。

クラスタ管理者がリソースを作成して、VALIDATE メソッドが呼び出されると、*すべての*システム定義、拡張機能、およびリソースグループのプロパティーが VALIDATE メソッドに渡されます。クラスタ管理者がリソースを更新して、VALIDATE メソッドが呼び出されると、更新されているプロパティーのみが VALIDATE メソッドに渡されます。

-x *prop* =*val*

ローカルノードのリソース拡張プロパティーの値を指定します。

prop リソース拡張プロパティー名です。拡張プロパティーは、リソースタイプの実装によって定義されます。この拡張プロパティーは、リソースタイプ登録ファイルのパラメータ表で宣言されます。

val クラスタ管理者がリソースを作成または更新した際にメソッドに渡される値です。

-X *prop*{ *nodeid*}=*val*

指定されたノードのノード単位のリソース拡張プロパティーの値を指定します。

prop リソース拡張プロパティー名です。拡張プロパティーは、リソースタイプの実装によって定義されます。この拡張プロパティーは、リソースタイプ登録ファイルのノード別のパラメータ表で宣言されます。

node 整数のノード ID。これは、ノード別のプロパティー値が設定されるノードを指定します。

val クラスタ管理者がリソースを作成または更新した際にメソッドに渡される値です。

-Z *zonename*

リソースグループが実行を構成されている非大域ゾーンの名前を指定します。

Global_zone リソースタイププロパティーに TRUE が設定されている場合、リソースを含むリソースグループが非大域ゾーンで動作している場合でも、メソッドは大域ゾーンで実行されます。このオプションは、リソースグループが実行するよう構成されている非大域ゾーンの名前を指定します。

次のいずれかの条件が満たされると、-Z オプションは渡されません。

- Global_zone プロパティーに FALSE が設定されます。
- リソースグループが大域ゾーンで実行するよう構成されている。

コールバックメソッドは、それらを呼び出す RGM によって定義されます。これらのメソッドは、クラスタリソース上で処理を実行します。また、これらのメソッドは、メソッドが成功したか失敗したかを報告する終了ステータスを返します。各コールバックメソッドについては、次のセクションで説明します。

BOOT

このメソッドは、クラスタがブートまたはリブートされたときに、ノードがクラスタに結合または再結合すると呼び出されます。このメソッドは、Init_nodes リソースタイププロパティーにより指定されたノード上で呼び出されます。INIT と同様に、このメソッドは、リソースを含むリソースグループがオンラインになったあと、クラスタに結合するノード上でリソースを初期化します。このメソッドは、管理されているソースグループ内のリソース上で呼び出されますが、管理されていないリソースグループ内のリソース上では呼び出されません。

FINI

このメソッドは、リソースを含むリソースグループが RGM の管理から外れたときに呼び出されます。このメソッドは、Init_nodes リソースタイププロパティーにより指定されたノード上で呼び出されます。このメソッドは、リソースの構成を解除し、INIT メソッドまたは BOOT メソッドにより行われたすべての永続的な設定をクリーンアップします。

INIT

このメソッドは、リソースを含むリソースグループが RGM の管理下に入ったときに呼び出されます。このメソッドは、Init_nodes リソースタイププロパティーにより指定されたノード上で呼び出されます。このメソッドは、リソースを初期化します。

MONITOR_CHECK

このメソッドは、リソースを含むリソースグループが新しいノードに再配置される前に呼び出されます。このメソッドは、障害モニターが scha_control コマンドまたは scha_control 関数の GIVEOVER() オプションを実行すると呼び出されます。683 ページのscha_control(1HA) および 1101 ページのscha_control(3HA) のマニュアルページを参照してください。

このメソッドは、リソースグループの潜在的な新しいマスターである任意のノードで呼び出されます。MONITOR_CHECK メソッドは、ノードがリソースを実行するのに十分なほど健全であるかどうかを評価します。MONITOR_CHECK メソッドは、並行して実行されるほかのメソッドと衝突しない方法で実装する必要があります。

MONITOR_CHECK メソッドが失敗した場合、コールバックが呼び出されたノードへのリソースグループの再配置は拒否されます。

MONITOR_START

このメソッドは、リソースの起動後に、リソースが起動したノード上で呼び出されます。このメソッドは、リソースのモニターを開始します。

MONITOR_START が失敗した場合、RGM によりリソースの状態が MONITOR_FAILED に設定されます。

MONITOR_STOP

このメソッドは、リソースの停止前に、リソースを実行しているノード上で呼び出されます。このメソッドは、リソースのモニターを停止します。このメソッドは、モニタリングがクラスタ管理者によって無効化されている場合も呼び出されます。

MONITOR_STOP メソッドが失敗した場合に RGM が実行するアクションは、リソースの Failover_mode プロパティーの設定に依存します。Failover_mode に HARD が設定されている場合、RGM はノードをリブートして、リソースを強制的に停止しようとします。その他の場合、RGM はリソースの状態に STOP_FAILED を設定します。

POSTNET_STOP

STOP メソッドを補助するメソッドで、関連ネットワークアドレスを停止したあとに必要な停止アクションを実行します。このメソッドは、STOP メソッドが呼び出されたノードで呼び出されます。このメソッドは、リソースグループ内のネットワークアドレスを停止したあとに、リソースの STOP メソッドが呼び出されたあと呼び出されます。ただし、このメソッドは、ネットワークアドレスが unplumb される前に呼び出されます。POSTNET_STOP メソッドは、リソースの STOP メソッドと、このリソースに依存するすべてのリソースの POSTNET_STOP メソッドのあとに呼び出されます。

POSTNET_STOP メソッドが失敗したときに RGM が実行するアクションは、リソースの Failover_mode プロパティーの設定によって異なります。Failover_mode に HARD が設定されている場合、RGM はノードを終了して、リソースを強制的に停止しようとします。その他の場合、RGM はリソースの状態に STOP_FAILED を設定します。

PRENET_START

START メソッドを補助するメソッドで、関連ネットワークアドレスを構成する前に必要な起動アクションを実行します。このメソッドは、START メソッドが呼び出されるノードで呼び出されます。このメソッドは、同じリソースグループ内のネットワークアドレスが plumb されたあと、呼び出されます。ただし、このメソッドは、アドレスが構成され、リソースの START メソッドが呼び出される前に呼び出されます。リソースの START メソッドと、このリソースに依存するすべてのリソースの PRENET_START メソッドより前に、PRENET_START メソッドが呼び出されます。

PRENET_START メソッドが失敗したときに RGM が実行するアクションは、リソースの Failover_mode プロパティーの設定によって異なります。Failover_mode に SOFT または HARD が設定されている場合、RGM は、そのリソースを含むリソースグループを別のノードに再配置しようとします。その他の場合、RGM はリソースの状態に START_FAILED を設定します。

START

このメソッドは、リソースを含むリソースグループがクラスタノード上でオンラインになったときに、そのクラスタノード上で呼び出されます。クラスタ管理者は、clresourcegroup コマンドを使用して、オンとオフの状態を切り替えることができます。START メソッドは、ノード上のリソースを有効にします。

START メソッドが失敗したときに RGM が実行するアクションは、リソースの Failover_mode プロパティーの設定によって異なります。Failover_mode に SOFT または

HARD が設定されている場合、RGM はリソースのグループを別のノードに再配置しようとします。その他の場合、RGM はリソースの状態に START_FAILED を設定します。

STOP

このメソッドは、リソースを含むリソースグループがクラスタノード上でオフラインになったときに、そのクラスタノード上で呼び出されます。クラスタ管理者は、clresourcegroup コマンドを使用して、オンとオフの状態を切り替えることができます。このメソッドは、リソースを (アクティブであれば) 停止します。

STOP メソッドが失敗したときに RGM が実行するアクションは、リソースの Failover_mode プロパティーの設定によって異なります。Failover_mode に HARD が設定されている場合、RGM はノードをリブートして、リソースを強制的に停止しようとします。その他の場合、RGM はリソースの状態に STOP_FAILED を設定します。

UPDATE

このメソッドを呼び出すと、プロパティーが変更されたことを実行中のリソースに通知することができます。UPDATE メソッドは、RGM がソースまたはそのリソースグループのプロパティーの設定に成功したあとに呼び出されます。このメソッドは、リソースがオンラインであるノード上で呼び出されます。このメソッドは、scha_resource_get および scha_resourcegroup_get コマンドを呼び出して、有効なリソースに影響を与える場合があるプロパティー値を読み取り、実行中のリソースを適宜調整できます。

VALIDATE

このメソッドは、リソースが作成された場合やリソースまたはリソースを含むリソースグループが更新された場合に呼び出されます。VALIDATE は、リソースタイプの Init_nodes プロパティーにより指定されたクラスタノードのセットに対して呼び出されます。

VALIDATE メソッドは、リソースの作成または更新が適用される前に呼び出されます。メソッドがノード上で失敗し、障害終了ステータスコードが生成されると、作成または更新は中止されます。

クラスタ管理者がリソースを作成して、VALIDATE メソッドが呼び出されると、*すべての*システム定義、拡張機能、およびリソースグループのプロパティーが VALIDATE メソッドに渡されます。クラスタ管理者がリソースを更新して、VALIDATE メソッドが呼び出されると、更新されているプロパティーのみが VALIDATE メソッドに渡されます。scha_resource_get および scha_resourcegroup_get コマンドを使用すると、更新されないリソースのプロパティーを取得できます。

リソース依存関係プロパティーは、VALIDATE コマンド行で 2 つの異なる -r オプションで渡されます。一方の -r オプションはリソース名のみを一覧表示し、ローカルノードで適用可能な依存関係を表します。もう一方の -r オプションには、{LOCAL_NODE} などの修飾子を含む依存関係の全体のリストが含まれます。プロパティー名 Resource_dependencies、Resource_dependencies_offline_restart、Resource_dependencies_restart、および Resource_dependencies_weak は、ローカルノードの修飾子を含まない依存関係名を提供します。対応するプロパティー名 Resource_dependencies_Q、Resource_dependencies_Q_offline_restart、Resource_dependencies_Q_res

および Resource_dependencies_Q_weak は、修飾子を含む依存関係の同じリストを提供します。

たとえば、次を設定します:

```
# clresource set -p Resource_dependencies=r1@node1,r2@node2,r3{local_node},r4
```

node1 で、次の引数は VALIDATE メソッドに渡されます:

```
... -r Resource_dependencies=r1,r3,r4

-r Resource_dependencies_Q=r1@node1,r2@node2,r3{local_node},r4
 ...
```

node2 で、Resource_dependencies プロパティーの値は r2、r3、r4 になり、Resource_dependencies_Q プロパティーの値はすべてのノードで同じです。同様に、プロパティー名 Resource_dependencies_Q_weak、Resource_dependencies_Q_restart、および Resource_dependencies_Q_offline_restart は、依存関係プロパティー Resource_dependencies_weak、Resource_dependencies_restart、および Resource_dependencies_offline_restart にそれぞれ対応します。

Network_resources_used プロパティーを明示的に設定しない場合は、その値は、4 つの Resource_dependencies プロパティーから派生し、これら 4 つのプロパティーに出現するすべてのネットワークアドレスリソースを含んでいます。各ノードの Network_resources_used プロパティーの派生値は、ノード単位の依存関係を反映し、ノードごとに異なる場合があります。

VALIDATE メソッドを実装すると、ユーザーが stdout または stderr に書き込むすべてのメッセージはユーザーコマンドに戻されます。このアクションは、検証障害の理由を説明したり、リソースに関係してユーザーに指示を与えるのに便利です。

Oracle Solaris Cluster リソース管理コールバックメソッドは、RGM によりスーパーユーザー権限で実行されます。これらのメソッドを実装するプログラムは、適切な実行権でインストールされます。セキュリティ上の理由から、これらのプログラムに対する書き込み権は設定しません。

コールバックメソッドの実行のために設定される環境変数は次のとおりです。

```
HOME=/
PATH=/usr/bin:/usr/cluster/bin
LD_LIBRARY_PATH=/usr/cluster/lib
```

SIGNALS

コールバックメソッド呼び出しがタイムアウト期間を超えた場合は、最初に SIGTERM シグナルがプロセスに送信されます。SIGTERM シグナルが 10 秒以内にメソッドの実行を停止できなかった場合は、プロセスに SIGKILL シグナルが送信されます。

次の終了ステータスコードが返されます。

`0` コマンドは正常に完了しました。

0 以外 エラーが発生しました。

失敗の終了ステータス値は、失敗時の RGM のアクションに影響を及ぼしません。ただし、終了ステータスはメソッドの失敗時にクラスタログに記録されます。リソースタイプを実装して、0 以外の複数の終了ステータスを定義することにより、クラスタログを介して管理者とエラー情報を交換できます。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェース 安定性	発展中

305 ページの`clresourcegroup`(1CL), 333 ページの`clresourcetype`(1CL), 675 ページの`scha_cmds`(1HA), 683 ページの`scha_control`(1HA), 691 ページの`scha_resource_get`(1HA), 703 ページの`scha_resourcegroup_get`(1HA), Unresolved link to " signal3C", Unresolved link to " stdio3C", 1047 ページの`scha_calls`(3HA), 1101 ページの`scha_control`(3HA), 1251 ページの`rt_reg`(4), Unresolved link to " attributes5"

名前

scdsbuilder — GUI バージョンの Oracle Solaris Cluster Agent Builder の起動

```
scdsbuilder
```

`scdsbuilder` コマンドでは、GUI バージョンの Oracle Solaris Cluster Agent Builder を起動できます。

Agent Builder を使用する前に、次の要件を確認してください。

- `$PATH` 変数に Java 実行時環境が含まれている。Agent Builder はバージョン 1.6 以降の Java Development Kit に依存しています。Java Development Kit が `$PATH` 変数に含まれていない場合、Agent Builder コマンド (`scdsbuilder`) がエラーメッセージを返して表示します。
- Oracle Solaris OS の「Developer System Support」ソフトウェアグループがインストールされていること。
- `cc` コンパイラが `$PATH` 変数に含まれている。Agent Builder は `$PATH` 変数で最初に現れる `cc` を使用して、リソースタイプの C バイナリコードを生成するコンパイラを識別します。`cc` が `$PATH` に含まれていない場合、Agent Builder は C コードを生成するオプションを無効にします。

注記 - Agent Builder では、標準の `cc` コンパイラ以外のコンパイラも使用できます。別のコンパイラを使用するためには、そのコンパイラ (`gcc` など) に対するシンボリックリンクを `cc` から `$PATH` 内に含まれるファイル名で指定します。あるいは、Makefile のコンパイラ指定(現在は `CC=cc`) を変更し、別のコンパイラへの完全パスを指定することもできます。たとえば、Agent Builder によって生成される Makefile で `CC=cc` を `CC=`*pathname*`/gcc` に変更します。この場合、Agent Builder を直接実行できません。代わりに、`make` や `make pkg` コマンドを使用して、データサービスコードとパッケージを生成する必要があります。

このコマンドは、次の終了ステータスコードを返します。

`0`	コマンドは正常に完了しました。
0 以外	エラーが発生しました。

install-directory `/rtconfig`	前のセッションの情報が格納されているため、この情報により、ツールの終了および再起動機能が容易に実行できます。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	発展中

cc(1)、659 ページの`scdscreate`(1HA)、655 ページの`scdsconfig`(1HA)、Unresolved link to " attributes5"

Unresolved link to " Oracle Solaris Cluster Data Services Developer's Guide "

名前

scdsconfig — リソースタイプテンプレートの構成

```
scdsconfig -s start-command [-u start-method-timeout] [-e
    validate-command] [-y validate-method-timeout] [-t stop-command]
    [-v stop-method-timeout] [-m probe-command] [-n probe-timeout]
    [-d working-directory]
```

scdsconfig コマンドは、659 ページのscdscreate(1HA) コマンドで作成したリソースタイプテンプレートを構成します。scdsconfig コマンドを使用すると、ネットワーク対応(クライアントサーバーモデル) アプリケーションと非ネットワーク対応(クライアントを持たない) アプリケーションの両方に対して、C、GDS(Generic Data Service)、または Korn シェルベースのテンプレートを作成できます。

scdsconfig コマンドは、アプリケーションを起動、停止、検証、およびプローブするアプリケーション固有のコマンドを構成します。また、scdsconfig コマンドを使用すると、start、stop、validate、および probe コマンドのタイムアウト値も設定できます。scdsconfig コマンドは、ネットワーク対応 (クライアントサーバーモデル) のアプリケーションと非ネットワーク対応 (クライアントレス) のアプリケーションを両方ともサポートします。ユーザーは scdscreate コマンドが実行されたディレクトリから scdsconfig コマンドを実行できます。-d オプションを使用すると、<command>scdscreate</command> を実行したディレクトリを指定できます。scdsconfig コマンドは、生成されたコード内の正しい場所にユーザー指定のパラメータを挿入することにより、リソースタイプテンプレートを構成します。生成されたソースコードの種類が C の場合、このコマンドはさらに、そのソースコードをコンパイルします。scdsconfig コマンドは、出力をインストール可能な Solaris パッケージにします。このコマンドは、scdscreate コマンドによって作成された $*vendor-id* $*resource-type-name*ディレクトリ下の pkg サブディレクトリにパッケージを作成します。

次のオプションがサポートされています。

-d *working-directory*

scdscreate コマンドが実行されたディレクトリを指定します。

scdscreate コマンドが実行されたディレクトリと異なるディレクトリから scdsconfig を実行する場合に、このオプションを指定します。

-e *validate-command*

アプリケーションを検証するために呼び出すコマンドの絶対パスを指定します。絶対パスを指定しない場合、アプリケーションは検証されません。アプリケーションの実行が成功した

場合、*validate-command* は `0` の終了ステータスを返します。`0` 以外の終了ステータスは、アプリケーションの実行が失敗したことを示します。この場合、過去のアプリケーションの障害履歴に応じて、次の 2 つの結果のいずれかが生じます。

- このリソースタイプのリソースは、同じノードで再起動されました。
- このリソースを含むリソースグループは、別の健全なノードにフェイルオーバーされました。

`-m` *probe-command*

ネットワーク対応アプリケーションまたは非ネットワーク対応アプリケーションの健全性を定期的にチェックするためのコマンドを指定します。直接シェルに渡すことができる完全なコマンドを指定してください。アプリケーションの実行が成功した場合、*probe-command* は `0` の終了ステータスを返します。`0` 以外の終了ステータスは、アプリケーションの実行が失敗したことを示します。この場合、過去のアプリケーションの障害履歴に応じて、次の 2 つの結果のいずれかが生じます。

- このリソースタイプのリソースは、同じノードで再起動されました。
- このリソースを含むリソースグループは、別の健全なノードにフェイルオーバーされました。

`-n` *probe-timeout*

プローブコマンドのタイムアウトを秒単位で指定します。障害の検出ミスを防ぐため、タイムアウト値は、システムの過負荷を考慮に入れて指定する必要があります。デフォルト値は 30 秒です。

`-s` *start-command*

アプリケーションを起動するコマンドを指定します。start コマンドには、直接シェルに渡すことができる完全なコマンドを指定してください。コマンド行引数を使って、ホスト名、ポート番号など、アプリケーションの起動に必要な各種構成データを指定できます。複数の独立したプロセスツリーを持つリソースタイプを作成する場合は、1 行に 1 個ずつコマンドを記述したテキストファイルを指定し、複数の異なったプロセスツリーを起動します。

`-t` *stop-command*

アプリケーションの停止コマンドを指定します。stop コマンドには、直接シェルに渡すことができる完全なコマンドを指定してください。このオプションを省略した場合、アプリケーションは、生成されたコードからのシグナルを発行することによって停止します。停止コマンドには、タイムアウト値の 80% が割り当てられます。この時間内に停止コマンドがアプリケーションを停止できない場合、`SIGKILL` には、タイムアウト値の 15% が割り当てられます。それでもアプリケーションを停止できない場合、停止メソッドはエラーを返して終了します。

`-u` *start-method-timeout*

起動コマンドのタイムアウトを秒単位で指定します。障害の検出ミスを防ぐため、タイムアウト値は、システムの過負荷を考慮に入れて指定する必要があります。デフォルト値は 300 秒です。

-v *stop-method-timeout*

停止コマンドのタイムアウトを秒単位で指定します。障害の検出ミスを防ぐため、タイムアウト値は、システムの過負荷を考慮に入れて指定する必要があります。デフォルト値は 300 秒です。

-y *validate-method-timeout*

起動コマンドのタイムアウトを秒単位で指定します。障害の検出ミスを防ぐため、タイムアウト値は、システムの過負荷を考慮に入れて指定する必要があります。デフォルト値は 300 秒です。

次の終了ステータスコードが返されます。

`0`	コマンドは正常に完了しました。
0 以外	エラーが発生しました。

working-directory `/rtconfig`	前のセッションの情報が格納されているため、ツールの終了と再起動をスムーズに行うことが可能です。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	発展中

Unresolved link to " ksh1", 653 ページの`scdsbuilder`(1HA), 659 ページの`scdscreate`(1HA), Unresolved link to " attributes5"

名前

scdscreate — Oracle Solaris Cluster リソースタイプのテンプレートの作成

```
scdscreate -V vendor-id -T resource-type-name -a [-s] [-n
     RT-version] [-d working-directory] [-k | -g]
```

scdscreate コマンドは、アプリケーションを高可用性 (HA) またはスケーラブルにするため、テンプレートを作成します。このコマンドは、ネットワーク対応 (クライアントサーバーモデル) アプリケーションと非ネットワーク対応 (クライアントを持たない) アプリケーションの両方に対して、C、GDS (Generic Data Service)、または Korn シェルベースのテンプレートを作成できます。

テンプレートを作成するには、根本的に異なる方法が 2 つあります。

GDS

scdscreate は、事前にクラスタにインストールされている単一のリソースタイプ SUNW.gds から動作する 3 つの駆動スクリプトを作成します。これらのスクリプトの名前は start *RT-Name*、stop *RT-Name*、および remove*RT-Name* で、それぞれのアプリケーションのインスタンスを起動、停止、および削除します。このモデルでは、事前にクラスタにインストールされているリソースタイプ SUNW.gds は不変です。

生成されたソースコード

scdscreate は、Oracle Solaris Cluster リソースタイプのテンプレートを作成します。指定されたアプリケーションを高可用性またはスケーラブルにするため、このテンプレートのインスタンス化は Resource Group Manager (RGM) の制御下で実行されます。

どちらのモデルでも、ネットワーク対応 (クライアントサーバーモデル) のアプリケーションと非ネットワーク対応 (クライアントレス) のアプリケーションのテンプレートを作成できます。

scdscreate は、*working-directory* の下に $*vendor-id*$*resource-type-name* という形式のサブディレクトリを作成します。このサブディレクトリには、駆動スクリプトか、そのリソースタイプのソース、バイナリ、パッケージファイルが格納されます。scdscreate はまた、構成ファイル rtconfig を作成します。この構成ファイルには、リソースタイプの構成情報を格納できます。scdscreate では、1 つのディレクトリにリソースタイプを 1 つだけ作成できます。異なるリソースタイプは異なるディレクトリに作成する必要があります。

次のオプションがサポートされています。

-a

ネットワーク対応でないリソースタイプの作成を指定します。作成されたテンプレート内のネットワーク関連コードはすべて無効になります。

-n *RT-version*

生成されたリソースタイプのバージョンを指定します。このパラメータを省略した場合で、かつ、C または Korn シェルベースのアプリケーションを作成する場合、テキスト文字列 `1.0` がデフォルトで使用されます。このパラメータを省略した場合で、かつ、GDS ベースのアプリケーションを作成する場合、GDS の `RT_version` 文字列がデフォルトで使用されます。*RT-version* は、同じベースリソースタイプの複数の登録バージョン (つまり、アップグレード) 間を区別します。

次の文字は *RT-version* に使用できません。空白、タブ、スラッシュ (`/`)、バックスラッシュ (`\`)、アスタリスク (`*`)、疑問符 (`?`)、コンマ (`,`)、セミコロン (`;`)、左大括弧 (`[`)、または右大括弧 (`]`)

-d *working-directory*

現在のディレクトリ以外のディレクトリにリソースタイプのテンプレートを作成します。このオプションを省略した場合、テンプレートは現在のディレクトリに作成されます。

-g

アプリケーションを高可用性またはスケーラブルにするため、テンプレートを GDS ベースの形式で作成します。

-k

このオプションのパラメータは、C ではなく、Korn シェルコマンド構文でソースコードを生成します。Unresolved link to " ksh1" を参照してください。

-s

リソースタイプがスケーラブルであることを示します。スケーラブルなリソースタイプのインスタンス (リソース) をフェイルオーバーリソースグループ内に構成して、スケーラブル機能を無効にすることができます。このオプションを省略した場合、フェイルオーバーリソースタイプのテンプレートが作成されます。

-T *resource-type-name*

リソースタイプの名前とバージョン。これらをベンダー ID と結合すると、作成されるリソースを一意に識別できます。

-V *vendor-id*

通常、ベンダー ID には、リソースタイプの作成元ベンダーのストックシンボルかその他の識別子を指定します。`scdscreate` は、リソースタイプの名前の始まりに、ベンダー ID とピリオド (.) を付けます。この構文によって、複数のベンダーが同じリソースタイプの名前を使用している場合でも、リソースタイプの名前は一意のままになります。

`0` コマンドは正常に完了しました。

0 以外　　エラーが発生しました。

working-directory`/rtconfig`　　以前のセッションからの情報を保持して、`scdscreate` の終了と再起動機能を利用します。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	発展中

Unresolved link to " ksh1", 653 ページの`scdsbuilder`(1HA), 655 ページの`scdsconfig`(1HA), Unresolved link to " attributes5", 1335 ページの`rt_properties`(5)

Unresolved link to " Oracle Solaris Cluster Data Services Developer's Guide "

名前

scha_check_app_user — アプリケーションユーザー名の取得と所有権およびアクセス権のチェック

```
scha_check_app_user [-R resource] [-U username] [-Z
    zoneclustername] cmd-path
```

scha_check_app_user コマンドは、Resource Group Manager (RGM) の制御下にあるリソースによって使用される構成済みのアプリケーションユーザー名を取得します。*cmd-path* で指定される実行可能ファイルの所有権およびアクセス権もチェックされます。この実行可能ファイルは通常、リソースのメソッドまたはモニターで実行されるアプリケーションプログラムになり、Unresolved link to " su1M" などのラッパーを使用して、ユーザー ID を構成済みのユーザーに設定します。リソースメソッドまたはモニターは、アプリケーションプログラムの実行前に scha_check_app_user を呼び出します。scha_check_app_user の出力に応じて、メソッドまたはモニターは、エラーを返すか、セキュリティー関連の問題が検出された場合は警告メッセージを出力します。

scha_check_app_user コマンドは、構成済みのユーザーの名前を標準出力 (ファイル記述子 1) に書き込み、セキュリティー警告またはエラーメッセージを標準エラー (ファイル記述子 2) に書き込みます。終了コードは、構成済みのセキュリティーポリシーによりコマンドの実行が許可されるかどうかを示します。終了コードが 0 の場合、呼び出し元はアプリケーションユーザーとしてコマンドの実行を試みることができます。終了コードが 0 以外の場合は、エラーが返されるため、呼び出し元でアプリケーションユーザーとしてコマンドを実行しようとしないでください。

scha_check_app_user を呼び出すスクリプトでは、コマンドの出力を使用して次を判断できます:

- コマンドを実行する必要のあるユーザー ID
- コマンドの実行を許可するかエラーをスローするか
- セキュリティーの問題が検出された場合に、ユーザーに渡すエラーまたは警告メッセージ

scha_check_app_user コマンドは、1287 ページのr_properties(5) のマニュアルページで説明されている Resource_security および Application_user プロパティーと連動します。

scha_check_app_user コマンドの動作は、Resource_security プロパティーの設定に依存します。Resource_security プロパティーは、グローバルクラスタおよび各ゾーンクラスタで値が異な

る場合があります。scha_check_app_user で使用される Resource_security の値は、コマンドが実行されるクラスタのプロパティーの値です。

scha_check_app_user コマンドは、アプリケーションプログラムが実行されるのと同じコンテキストで呼び出すように意図されています。たとえば、アプリケーションプログラムが大域ゾーンで実行される場合は、scha_check_app_user も大域ゾーンで実行するようにしてください。

通常のユースケースは次のいずれかになります:

- リソースとそのリソースグループがグローバルクラスタに構成され、scha_check_app_user プログラムがグローバルクラスタで実行されている。
- リソースとそのリソースグループがゾーンクラスタに構成され、scha_check_app_user プログラムがゾーンクラスタで実行されている。

どちらのユースケースでも、-Z *zoneclustername* オプションをコマンドに指定する必要はありません。

-Z *zoneclustername* オプションが使用されるのは、アプリケーションプログラムが大域ゾーン内で実行されるが、そのアプリケーションプログラムがゾーンクラスタのリソースに関連付けられている場合です。これは通常は必要ありませんが、Global_zone プロパティーが TRUE に設定されているリソースタイプで必要になる場合があります。詳細は、1335 ページのrt_properties(5)を参照してください。

-Z の使用や、その他のコマンドオプションとの連携の詳細は、オプション セクションを参照してください。

次のオプションがサポートされています。

-Z *zoneclustername*

リソースが構成されているクラスタを指定します。このオプションが必要になるのは、コマンドが大域ゾーンで実行されるが、ゾーンクラスタのリソースの Application_user プロパティーにアクセスする必要がある場合のみです。-Z オプションは、ゾーンクラスタ内で異なるゾーンクラスタにアクセスするために使用することはできません。

-Z オプションを省略すると、scha_check_app_user コマンドが実行されるクラスタ (グローバルクラスタとゾーンクラスタのいずれか) 内にリソースが存在するとみなされます。

scha_check_app_user コマンドが大域ゾーンで実行され、-Z および -R オプションが両方とも指定されている場合、-R で指定した *resource* は、グローバルクラスタではなく、-Z で指定したゾーンクラスタに存在します。どちらの場合も、エージェントの開発者は、リソースがゾーンクラスタで構成されている場合でも、Application_user プロパティーで指定した

ユーザー名を大域ゾーンで有効にする必要がある旨をエンドユーザーに警告するようにしてください。

scha_check_app_user コマンドが大域ゾーン内で実行され、-Z オプションが指定されている場合、*cmd-path* 引数は、-Zで指定されたゾーンではなく、大域ゾーン内のファイルのパス名を識別します。

-U *username*

このユーザー名を指定した場合は、実行可能ファイルの所有者、Application_user プロパティーの設定、または Resource_security プロパティーの設定に関係なく、アプリケーションのユーザー名になります。-U オプションを使用できるのは、呼び出し元にアプリケーションのユーザー名を判断するための独自のメカニズムがあり、呼び出し元で実行可能プログラムの所有権とアクセス権のみをチェックする場合です。呼び出し元の実際のユーザー ID が root 以外で、-U オプションで root が指定されると、エラーが発生します。

-U オプションが -Z オプションとともに使用されている場合、指定された *username* は、コマンドが実行されるゾーンで有効である必要があますが、-Z オプションで指定されている *zoneclustername* で必ずしも有効である必要はありません。

-R *resource*

RGM リソースの名前は、このコマンドの実行に関連付けられています。-U オプションも指定されていない場合、アプリケーションのユーザー名は、このリソースの Application_user プロパティーから取得されます。リソースに Application_user プロパティーが存在しないか、そのプロパティーが設定されていない場合、アプリケーションのユーザー名は実行可能ファイルの所有者になります。

-U が指定されておらず、Resource_security が COMPATIBILITY に設定されている場合は、Application_user プロパティーの設定に関係なく、アプリケーションのユーザー名は、呼び出し元のプロセスの実際のユーザー ID に設定されます。-U が指定されておらず、かつ Resource_security プロパティーが OVERRIDE に設定されている場合、Application_user プロパティー設定には関係なく、アプリケーションユーザー名は実行可能ファイルの所有者に設定されます。

-R オプションが -Z オプションとともに指定されている場合は、コマンドが実行されるゾーンで有効なユーザー名をリソースの Application_user プロパティーで指定する必要がありますが、-Z オプションで指定されている *zoneclustername* 内で必ずしも有効ある必要はありません。

cmd-path

呼び出し元のプロセスがアプリケーションユーザーとして実行する実行可能ファイルのフルパス名。-Z オプションが指定されている場合、*cmd-path* は、-Z オプションで指定されている *zoneclustername* ではなく、コマンドが実行されるゾーンと比較して評価されます。

-R も -U も指定されていない場合は、Resource_security が COMPATIBILITY に設定されていないかぎり、アプリケーションのユーザー名は実行可能ファイルの所有者になり、この場合、アプリケーションのユーザー名は呼び出しプロセスの実際のユーザー ID に設定されます。

計算されたアプリケーションユーザーが root (スーパーユーザー) であるが、呼び出し元の実際のユーザー ID が root 以外の場合、アプリケーションのユーザー名は呼び出し元の実際のユーザー ID になります。

例 269 スクリプトでの scha_check_app_user と su の使用

次の bash スクリプトは、su(1M) を使用する前に、scha_check_app_user を呼び出し、myresource という名前の RGM リソースに関連付けられている、mycommand という名前のコマンドを実行します。

```
COMMANDPATH=/opt/mypkg/bin/mycommand
RESOURCENAME=myresource
TMPFILE=$(/usr/bin/mktemp)

# Here we are redirecting the error/warning messages into
# a temp file and will write them later.
# Instead, we could just let them flow out to stderr.
APPUSER=$(/usr/cluster/bin/scha_check_app_user \
        -R $RESOURCENAME $COMMANDPATH 2>$TMPFILE)
errcode=$?

if [[ $errcode -ne 0 ]]; then
    # Security checks failed -- do not execute the program
    printf "Security checks failed on program %s:\n" $COMMANDPATH
    # Output the error messages
    /usr/bin/cat $TMPFILE
    /usr/bin/rm $TMPFILE
    exit errcode
fi

# There may still be warning messages in TMPFILE.
# Write them for the user.
/usr/bin/cat $TMPFILE
/usr/bin/rm $TMPFILE

# Application user name is in $APPUSER.
# Execute mycommand with any necessary arguments.
#
# Note that the su command might still fail, for example, if
# this script lacks the necessary privilege to execute as
# the application user.
#
# Other command wrappers such as "su -" or "pfexec" could be used
# here instead of plain "su".

su $APPUSER $COMMANDPATH arg1 arg2
```

次の終了ステータスコードが返されます。エラーコードは、1047 ページのscha_calls(3HA) で説明されています。

0 SCHA_ERR_NOERR

セキュリティーチェックに合格し、コマンドをアプリケーションユーザーで実行できます。ただし、出力が stderr に書き込まれると、アプリケーションユーザーの取得またはチェックでエラーが発生したことが示されます。このような警告メッセージは、ユーザーに戻すようにしてください。

3 SCHA_ERR_INVAL

コマンドが無効な引数で呼び出されています。この場合、アプリケーションユーザーは stdout に書き込まれません。起こり得る複数のエラーの 1 つに関する詳細なエラーメッセージが stderr に書き込まれます。

6 SCHA_ERR_ACCESS

パス引数で識別されるファイルが実行可能でないか、-U オプションで root を指定しており、呼び出し元の実際のユーザー ID が root 以外であるか、または Resource_security が SECURE であり、次のいずれかの条件が適用されます。

- 実行可能ファイルは、だれでも読み書き可能です。
- アプリケーションユーザーは root であり、実行可能ファイルはグループから書き込み可能です。

SCHA_ERR_ACCESS 終了コードはセキュリティー違反を示し、呼び出し元でコマンドを実行してはいけません。

14 SCHA_ERR_RSRC

rname 引数では、有効なリソース名を識別しません。この場合、アプリケーションユーザーは stdout に書き込まれません。エラーメッセージは stderr に書き込まれます。

16 SCHA_ERR_CHECKS

Resource_security は SECURE で、Application_user 名は有効なユーザー ID にマップされません。SCHA_ERR_CHECKS 終了コードはセキュリティー違反を示し、呼び出し元でコマンドを実行してはいけません。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

Unresolved link to " su1M", Unresolved link to " pfexec1",
551 ページのcluster(1CL), 675 ページのscha_cmds(1HA),

1047 ページの`scha_calls`(3HA), 1239 ページの`scha_strerror`(3HA), Unresolved link to " attributes5", Unresolved link to " Oracle Solaris Cluster Data Services Developer's Guide "

名前

scha_cluster_get — クラスタ情報へのアクセス

```
scha_cluster_get -O optag [-Z zoneclustername] [args]
```

scha_cluster_get コマンドでは、クラスタ情報にアクセスしたり、クラスタ情報を生成したりすることができます。クラスタ、ノード、ホスト名、リソースグループ、リソースタイプ、および状態に関する情報にアクセスできます。

このコマンドは、リソースタイプに対するコールバックメソッドのシェルスクリプト実装で使用するためのものです。リソースタイプ向けのこれらのコールバックメソッドは、クラスタの Resource Group Manager (RGM) によって制御されるサービスを表します。このコマンドは、scha_resource_get() 関数と同じ情報を提供します。

このコマンドは、675 ページのscha_cmds(1HA) のマニュアルページで説明されているように、出力を標準出力 (stdout) に別々の行で書式付き文字列として送信します。出力はシェル変数に格納できます。また、出力は awk コマンドなどのシェルコマンドで解析できます。

次のオプションがサポートされています。

-O *optag*

optag 引数に、アクセス対象の情報を指定します。*optag* によっては、情報を取得するクラスタノードを指定するのに、追加の引数が必要になる場合があります。

注記 - NODENAME_LOCAL やNODENAME_NODEID などの *optag* オプションは、大文字と小文字を区別しません。*optag* オプションを指定するときには、大文字と小文字の任意の組み合わせを使用できます。

次の *optag* 値がサポートされています。

ALL_LOADLIMITS

グローバルクラスタまたはゾーンクラスタに定義されているすべての loadlimit 名を連続した行に生成します。

ALL_NODEIDS

クラスタ内の全ノードの数値形式の識別子を複数の行に生成します。

ALL_NODENAMES

クラスタ内の全ノード名を複数の行に生成します。

ALL_PRIVATELINK_HOSTNAMES

クラスタインターコネクト上に全クラスタノードのアドレスを指定するホスト名を、複数の行に生成します。

ALL_PSTRINGS

クラスタ内のすべてのプライベート文字列の値ではなく、名前を連続した行に生成します。プライベート文字列の詳細は、217 ページのclpstring(1CL) のマニュアルページを参照してください。

ALL_RESOURCEGROUPS

クラスタ内で管理される全リソースグループの名前を複数の行に生成します。

ALL_RESOURCETYPES

クラスタに登録された全リソースタイプの名前を複数の行に生成します。

ALL_ZONES

クラスタ内のすべてのノードで、大域ゾーンを含むすべてのゾーンの nodename:zonename 文字列を連続した行に生成します。

次の条件が発生した場合に限り、非大域ゾーンがこのクエリーの出力に書き込まれます。

- クラスタがオンラインになって以降、非大域ゾーンが少なくとも 1 回ブートした。
- 非大域ゾーンが Service Management Facility (SMF) サービス /system/cluster/sc_ng_zones を正常に起動した。

SMF サービス /system/cluster/sc_ng_zones を実行しない非大域ゾーンはリソースグループをマスターできないため、出力に書き込まれません。

ALL_ZONES_NODEID

数値ノード識別子が引数として指定されているクラスタノードで、大域ゾーンを含むすべてのノードの nodename:zonename 文字列を連続した行に生成します。

次の条件が発生した場合に限り、非大域ゾーンがこのクエリーの出力に書き込まれます。

- クラスタがオンラインになって以降、非大域ゾーンが少なくとも 1 回ブートした。
- 非大域ゾーンが Service Management Facility (SMF) サービス /system/cluster/sc_ng_zones を正常に起動した。

SMF サービス /system/cluster/sc_ng_zones を実行しない非大域ゾーンはリソースグループをマスターできないため、出力に書き込まれません。

CLUSTERNAME

クラスタの名前を生成します。

HARD_LOADLIMIT

グローバルクラスタまたはゾーンクラスタのすべてのノードで特定の `limitname` に設定されている強い制限値を連続した行に生成します。負荷制限の名前文字列である、追加の旗なしの `string` 引数が必要です。

文字列配列の出力の各要素の書式は `"%s=%d"` で、左側の文字列は *nodename* または *nodename*:*zonename*、右側の整数はそのノードで指定されている制限名に対する強い負荷制限値です。強い制限値を指定しない場合、強い制限値の値として `-1` が表示されます。

LOADLIMIT_PROPS

グローバルクラスタまたはゾーンクラスタの全ノードの強い制限値と弱い制限値 (`/` で区切り) を複数行に連続して生成します。負荷制限の名前文字列である、追加の旗なしの `string` 引数が必要です。

文字列配列の出力の各要素の書式は `"%s=%d/%d"` で、左側の文字列は *nodename* または *nodename*:*zonename*、最初の整数は弱い制限値、2 番目の整数は強い制限値です。強い制限値を指定しない場合、強い制限値の値として `-1` が表示されます。弱い制限値を指定しない場合、弱い制限値の値として `0` が表示されます。

LOADLIMITS_NODE

特定のノードに設定されている負荷制限 (`/` 区切り) を連続した行に生成します。負荷制限の名前文字列である、フラグが設定されていない追加の `string` 引数が必要です。

文字列配列の出力の各要素の書式は `"%s=%d/%d"` で、文字列は指定したノードに定義されている制限名、最初の整数は弱い制限値、2 番目の整数は強い制限値です。強い制限値を指定しない場合、強い制限値の値として `-1` が表示されます。弱い制限値を指定しない場合、弱い制限値の値として `0` が表示されます。

NODEID_LOCAL

コマンドを実行するノードの数値形式の識別子を生成します。

NODEID_NODENAME

ノード名を指定して、そのノードの数値形式の識別子を生成します。フラグが設定されていない引数を追加して、クラスタノード名を指定する必要があります。

NODENAME_LOCAL

コマンドを実行するクラスタノードの名前を生成します。

NODENAME_NODEID

数値形式のノード識別子を指定して、クラスタノード名を生成します。フラグが設定されていない引数を追加して、数値形式のノード識別子を指定する必要があります。

NODESTATE_LOCAL

コマンドの実行元ノードの状態 (`UP` または `DOWN`) を生成します。

NODESTATE_NODE

指定のノードの状態 (UP または DOWN) を生成します。フラグが設定されていない引数を追加して、クラスタノード名を指定する必要があります。

PRIVATELINK_HOSTNAME_LOCAL

クラスタインターコネクト上にコマンドの実行元ノードのアドレスを指定するホスト名を生成します。

PRIVATELINK_HOSTNAME_NODE

クラスタインターコネクトで特定ノードのアドレスを指定するホスト名を生成します。フラグが設定されていない引数を追加して、クラスタノード名を指定する必要があります。

PSTRING

プライベート文字列の平文の値を生成します。プライベート文字列の名前である、フラグが設定されていない追加の引数が必要です。スーパーユーザー以外のユーザーがこのクエリータグを使用するには、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。プライベート文字列の詳細は、217 ページのclpstring(1CL) のマニュアルページを参照してください。

RESOURCE_SECURITY

resource_security クラスタプロパティーの現在の設定を生成します。

RG_FAILOVER_LOG

クラスタ上で構成されているリソースグループのフェイルオーバーおよび手動のスイッチオーバーイベントログを生成します。このオプションのあとに、ログを取得する日数が続きます。最大 7 日間のログファイルを取得できます。日数を指定しない場合は、デフォルト値の 1 日が使用されます。例272「scha_cluster コマンドを使用したイベントログの表示」を参照してください。

SOFT_LOADLIMIT

クラスタのすべてのノードで特定の limitname に設定されている弱い負荷制限値を連続した行に生成します。負荷制限の名前文字列である、追加の旗なしの string 引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d" で、左側の文字列は *nodename* または *nodename*:*zonename*、右側の整数はそのノードで指定されている制限名に対する弱い負荷制限値です。弱い制限値を指定しない場合、弱い制限値の値として 0 が表示されます。

SYSLOG_FACILITY

RGM がログメッセージで使用する Unresolved link to " syslog3C" 機能の番号を生成します。この値は 24 で、これは daemon 機能に対応しています。この値を Unresolved link to " logger1" コマンドで機能レベルとして使用することにより、クラスタログ内にメッセージを記録できます。

-Z *zoneclustername*

操作するクラスタを指定します。このオプションは、コマンドが大域ゾーンで実行されるものの、指定されたゾーンクラスタで動作する必要がある場合に使用します。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

zoneclustername クエリーが *zoneclustername* という名前のゾーンクラスタで実行されるように指定します。

-Z オプションを省略すると、コマンドが実行されるクラスタでクエリーが実行されます。

グローバルクラスタのノード状態など、ゾーン別のプロパティー値をクエリーするには、-Z オプションは使用しないでください。代わりに、クエリータグのゾーン別の書式を使用します。たとえば、NODESTATE の代わりに NODESTATE_NODE を使用し、*nodename:zonename* という形式の追加のコマンド行引数を指定します。

例 270 シェルスクリプトでの scha_cluster コマンドの使用

次のシェルスクリプトは scha_cluster コマンドを使用して、各クラスタノードが稼働しているかどうかを出力します。

```
#!/bin/sh
nodenames=`scha_cluster_get -O All_Nodenames`
for node in $nodenames
do
     state=`scha_cluster_get -O NodeState_Node $node`
     printf "State of node: %s\n exit: %d\n value: %s\n" "$node" $? "$state"
done
```

例 271 scha_cluster コマンドを使用したノードの負荷制限の表示

次のコマンドは、ノード node1 に対して定義されたすべての負荷制限を表示します。

```
# scha_cluster_get -O LOADLIMITS_NODE node1
factor1=50/100
factor2=0/4
```

例 272 scha_cluster コマンドを使用したイベントログの表示

次のコマンドは、最後の 7 日間のクラスタのフェイルオーバーおよび手動のスイッチオーバーイベントログを表示します。この出力は、*rg_name*、リソースグループのフェイルオーバー先、または手動での切り替え先である *node_name*、および *timestamp* の形式で一覧表示されます。

```
# scha_cluster_get -O RG_FAILOVER_LOG 7
rg1,psnow4,Thu Oct 17 04:17:31 2013
rg2,psnow4,Thu Oct 17 04:17:31 2013
rg1,psnow4,Thu Oct 17 04:30:56 2013
rg2,psnow4,Thu Oct 17 04:30:56 2013
```

```
rg_fo_nfs,psnow4,Tue Oct 22 22:42:43 2013
rg_fo_nfs,psnow4,Tue Oct 22 22:46:08 2013
testrg-lh-1,psnow4,Tue Oct 22 22:47:43 2013
rg_test,psnow4,Tue Oct 22 22:51:50 2013
rg_test,psnow4,Tue Oct 22 22:51:55 2013
rg_test,psnow4,Tue Oct 22 22:52:01 2013
RG1,psnow4,Tue Oct 22 23:09:14 2013
RG1,psnow4,Tue Oct 22 23:15:39 2013
RG4,psnow4,Tue Oct 22 23:16:10 2013
RG3,psnow4,Tue Oct 22 23:16:16 2013
RG2,psnow4,Tue Oct 22 23:16:20 2013
RG1,psnow4,Tue Oct 22 23:16:26 2013
RG4,psnow4,Tue Oct 22 23:16:51 2013
RG2,psnow4,Tue Oct 22 23:16:51 2013
RG1,psnow4,Tue Oct 22 23:17:07 2013
RG3,psnow4,Tue Oct 22 23:17:10 2013
RG1,psnow4,Tue Oct 22 23:18:08 2013
RG2,psnow4,Tue Oct 22 23:18:08 2013
RG3,psnow4,Tue Oct 22 23:18:08 2013
```

次の終了ステータスコードが返されます。

`0`	正常終了。
0 以外	エラーが発生しました。 エラーコードは、1047 ページの`scha_calls`(3HA) で説明されています。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	安定

Unresolved link to " awk1", Unresolved link to " logger1", Unresolved link to " sh1", 675 ページの`scha_cmds`(1HA), Unresolved link to " syslog3C", 1047 ページの`scha_calls`(3HA), 1063 ページの`scha_cluster_get`(3HA), Unresolved link to " attributes5", 1319 ページの`rg_properties`(5)

名前

scha_cmds — scha_cluster_get、scha_control、scha_resource_get、scha_resourcegroup_get、scha_resourcetype_get、 scha_resource_setstatus のコマンド標準出力

scha-command -O *optag*...

Oracle Solaris Cluster の 669 ページのscha_cluster_get(1HA)、683 ページのscha_control(1HA)、691 ページのscha_resource_get(1l および 699 ページのscha_resource_setstatus(1HA) コマンドは、リソースタイプのコールバックメソッドのコマンド行実装です。645 ページのrt_callbacks(1HA) を参照してください。

リソースタイプは、クラスタの Resource Group Manager (RGM) 機能によって制御されるサービスを表します。これらのコマンドは、C 関数 1047 ページのscha_calls(3HA) の機能に対するコマンド行インタフェースを提供します。

get 関数はクラスタ構成情報にアクセスします。これらのコマンドすべては同じ標準的なインタフェースを持っています。これらのコマンドはすべて -O *optag* オペランドを取ります。このオペランドはアクセスされる情報を示します。これらのコマンドはすべて、結果を書式付き文字列として標準出力 (stdout) に送信します。コマンドとオペランド *optag* の値によっては、その他の引数を追加指定する必要があります。さまざまな *optag* の実行結果の書式については、「実行結果の書式」を参照してください。

注記 - すべての scha コマンドの *optag* オプションには、大文字と小文字の区別はありません。*optag* オプションを指定するときには、大文字と小文字の任意の組み合わせを使用できます。

---

683 ページのscha_control(1HA) コマンドにはまた、制御操作を示すが、標準出力への出力を生成しない -O *optag* オプションも指定できます。

699 ページのscha_resource_setstatus(1HA) コマンドは、RGM によって管理されるリソースの STATUS および STATUS_MSG プロパティーを設定します。

実行結果の書式

コマンドが標準出力に出力する文字列の書式は、-O オプションに指定された *optag* が指定する結果の型によって変わります。各型の書式は次の表のとおりです。表記法については、Unresolved link to " formats5" で説明されています。

結果の型	標準出力の書式
boolean	TRUE\n または FALSE\n
enum	%s\n、enum 値の文字列名。
extension	%s\n、拡張プロパティーの型属性。次の値のいずれか。STRING、INT、BOOLEAN、ENUM、STRINGARRAY。 型情報の後ろには、各型の書式に従ってプロパティー値が出力される。STRING は string として、INT は int として、BOOLEAN は boolean として、ENUM は enum として、STRINGARRAY は string_array として。
int	%d\n
status	%s\n%s\n、最初の文字列はステータスで、次の enum 値のいずれか。DEGRADED、FAULTED、OFFLINE、ONLINE、UNKNOWN。 2 番目の文字列はステータスメッセージ
string	%s\n
string_array	配列の各要素は、書式 %s\n で出力される。GLOBAL_RESOURCES_USED および INSTALLED_NODES プロパティーで、すべてのノードまたはリソースを示すアスタリスクを返すことができる。
unsigned_int	%u\n
unsigned_int_array	配列の各要素は、書式 %u\n で出力される。

optag の結果の型

次の表に、さまざまなコマンドに対する有効な *optag* 値と、以前の表で示した書式に従って出力される結果の型を示します。

669 ページのscha_cluster_get(1HA) の *optag* 値	結果の型
ALL_NODEIDS	unsigned_int_array
ALL_NODENAMES	string_array
ALL_PRIVATELINK_HOSTNAMES	string_array
ALL_RESOURCEGROUPS	string_array
ALL_RESOURCETYPES	string_array
CLUSTERNAME	string
NODEID_LOCAL	unsigned_int
NODEID_NODENAME	unsigned_int
NODENAME_LOCAL	string

669 ページのscha_cluster_get(1HA) の *optag* 値	結果の型
NODENAME_NODEID	string
NODESTATE_LOCAL	enum (UP, DOWN)
NODESTATE_NODE	enum (UP, DOWN)
PRIVATELINK_HOSTNAME_LOCAL	string
PRIVATELINK_HOSTNAME_NODE	string
SYSLOG_FACILITY	int

683 ページのscha_control(1HA) の *optag* 値
CHANGE_STATE_OFFLINE
CHANGE_STATE_ONLINE
CHECK_GIVEOVER
CHECK_RESTART
GIVEOVER
IGNORE_FAILED_START
RESOURCE_DISABLE
RESOURCE_IS_RESTARTED
RESOURCE_RESTART
RESTART

691 ページの scha_resource_get(1HA) の *optag* 値	結果の型
AFFINITY_TIMEOUT	int
ALL_EXTENSIONS	string_array
BOOT_TIMEOUT	int
CHEAP_PROBE_INTERVAL	int
CHEAP_PROBE_INTERVAL	int
EXTENSION	extension
EXTENSION_NODE	extension
FAILOVER_MODE	enum (NONE, HARD, SOFT, RESTART_ONLY, LOG_ONLY)
FINI_TIMEOUT	int
GROUP	string

691 ページの scha_resource_get(1HA) の *optag* 値	結果の型
INIT_TIMEOUT	int
LOAD_BALANCING_POLICY	string
LOAD_BALANCING_WEIGHTS	string_array
MONITORED_SWITCH	enum (DISABLED, ENABLED)
MONITORED_SWITCH_NODE	enum (DISABLED, ENABLED)
MONITOR_CHECK_TIMEOUT	int
MONITOR_START_TIMEOUT	int
MONITOR_STOP_TIMEOUT	int
NETWORK_RESOURCES_USED	string_array
NUM_RESOURCE_RESTARTS	int
NUM_RG_RESTARTS	int
ON_OFF_SWITCH	enum (DISABLED, ENABLED)
ON_OFF_SWITCH_NODE	enum (DISABLED, ENABLED)
PORT_LIST	string_array
POSTNET_STOP_TIMEOUT	int
PRENET_START_TIMEOUT	int
R_DESCRIPTION	string
RESOURCE_DEPENDENCIES	string_array
RESOURCE_DEPENDENCIES_OFFLINE_RESTART	string_array
RESOURCE_DEPENDENCIES_RESTART	string_array
RESOURCE_DEPENDENCIES_WEAK	string_array
RESOURCE_PROJECT_NAME	string
RESOURCE_STATE	enum (ONLINE, OFFLINE, START_FAILED, STOP_FAILED, MONITOR_FAILED, ONLINE_NOT_MONITORED, STARTING, STOPPING)
RESOURCE_STATE_NODE	enum (値については RESOURCE_STATE を参照)
RETRY_COUNT	int
RETRY_INTERVAL	int
SCALABLE	boolean
START_TIMEOUT	int
STATUS	status
STATUS_NODE	status
STOP_TIMEOUT	int

691 ページのscha_resource_get(1HA) の *optag* 値	結果の型
THOROUGH_PROBE_INTERVAL	int
TYPE	string
TYPE_VERSION	string
UDP_AFFINITY	boolean
UPDATE_TIMEOUT	int
VALIDATE_TIMEOUT	int
WEAK_AFFINITY	boolean

scha_resource_get (1HA) および scha_resourcetype_get(1HA) の *optag* 値	結果の型
API_VERSION	int
BOOT	string
FAILOVER	boolean
FINI	string
GLOBAL_ZONE	boolean
INIT	string
INIT_NODES	enum (RG_PRIMARIES, RT_INSTALLED_NODES)
INSTALLED_NODES	string_arrayすべてのノードを表すアスタリスク (*) が返される
IS_LOGICAL_HOSTNAME	boolean
IS_SHARED_ADDRESS	boolean
MONITOR_CHECK	string
MONITOR_START	string
MONITOR_STOP	string
PER_NODE	boolean
PKGLIST	string_array
POSTNET_STOP	string
PRENET_START	string
PROXY	boolean
RT_BASEDIR	string
RT_DESCRIPTION	string
RT_SYSTEM	boolean
RT_VERSION	string

scha_resource_get (1HA) および scha_resourcetype_get(1HA) の *optag* 値	結果の型
SINGLE_INSTANCE	boolean
START	string
STOP	string
UPDATE	string
VALIDATE	string

scha_resourcegroup_get (1HA) の *optag* 値	結果の型
AUTO_START_ON_NEW_CLUSTER	boolean
DESIRED_PRIMARIES	int
FAILBACK	boolean
GLOBAL_RESOURCES_USED	string_array (すべてのリソースを表すアスタリスク (*) が返される)
IMPLICIT_NETWORK_DEPENDENCIES	boolean
MAXIMUM_PRIMARIES	int
NODELIST	string_array
PATHPREFIX	string
PINGPONG_INTERVAL	int
RESOURCE_LIST	string_array
RG_AFFINITIES	string_array
RG_DEPENDENCIES	string_array
RG_DESCRIPTION	string
RG_IS_FROZEN	boolean
RG_MODE	enum (FAILOVER, SCALABLE)
RG_PROJECT_NAME	string
RG_SLM_CPU	decimal
RG_SLM_CPU_MIN	decimal
RG_SLM_PSET_TYPE	enum (DEFAULT, DEDICATED_STRONG, DEDICATED_WEAK)
RG_SLM_TYPE	enum (AUTOMATED, MANUAL)
RG_STATE	enum (UNMANAGED、ONLINE、 OFFLINE、PENDING_ONLINE、PENDING_OFFLINE、 ERROR_STOP_FAILED、ONLINE_FAULTED、 PENDING_ONLINE_BLOCKED)
RG_STATE_NODE	enum (値については RG_STATE を参照)

scha_resourcegroup_get (1HA) の optag 値	結果の型
RG_SYSTEM	boolean
SUSPEND_AUTOMATIC_RECOVERY	boolean

すべての scha コマンドには、一連の終了ステータスコードが使用されます。

終了ステータスコードは、1047 ページのscha_calls(3HA) で説明されている、対応する C 関数の scha_err_t リターンコードの数値です。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	安定

Unresolved link to " awk1", 645 ページのrt_callbacks(1HA),
669 ページのscha_cluster_get(1HA), 683 ページのscha_control(1HA),
691 ページのscha_resource_get(1HA),
703 ページのscha_resourcegroup_get(1HA),
709 ページのscha_resourcetype_get(1HA),
699 ページのscha_resource_setstatus(1HA), 1047 ページのscha_calls(3HA),
Unresolved link to " attributes5", Unresolved link to " formats5",
1287 ページのr_properties(5), 1319 ページのrg_properties(5),
1335 ページのrt_properties(5)

名前

scha_control — リソースおよびリソースグループ制御を要求する

```
scha_control -O optag -G group -R resource [-Z zonename]
```

`scha_control` コマンドは、クラスタの Resource GroupManager (RGM) の制御下にあるリソースまたはリソースグループの再起動または再配置を要求します。このコマンドは、リソースモニターのシェルスクリプト実装で使用します。このコマンドは、1101 ページの`scha_control`(3HA) 関数と同じ機能を提供します。

このコマンドの終了コードは、要求されたアクションが拒絶されたかどうかを示します。要求が受理された場合、このコマンドは、リソースグループまたはリソースがオフラインになったあと完全にオンラインに戻った時点で終了します。683 ページの`scha_control`(1HA) を呼び出した障害モニターは、リソースまたはリソースグループがオフラインになった結果として停止される場合があります。結果として、障害モニターは、成功した要求の戻りステータスを一度も受け取らない場合があります。

このコマンドを使用するには `solaris.cluster.resource.admin` の役割に基づくアクセス制御 (RBAC) が必要です。Unresolved link to " rbac5" を参照してください。

さらにこのコマンドを使用するにあたっては、Oracle Solaris Cluster コマンド権プロファイルを割り当てておく必要があります。認証されたユーザーは、Unresolved link to " pfsh1"、Unresolved link to " pfcsh1"、または Unresolved link to " pfksh1" プロファイルシェルから、Oracle Solaris Cluster の特権コマンドをコマンド行で実行できます。プロファイルシェルは、Oracle Solaris Cluster コマンド権プロファイルに割り当てられた Oracle Solaris Cluster の特権コマンドへのアクセスを可能にする特別なシェルです。プロファイルシェルは、Unresolved link to " su1M" を実行して役割を引き受けると起動されます。Unresolved link to " pfexec1" を使用しても、Oracle Solaris Cluster の特権コマンドを実行できます。

次のオプションがサポートされています。

`-G` *group*

再起動または再配備するリソースグループ、あるいは再起動または再配置するリソースを含むリソースグループの名前。リクエスト元ノードでリソースグループがオンラインになっていない場合、リクエストは拒否されます。

-O *optag*

optag オプションをリクエストします。

注記 - CHECK_GIVEOVER や CHECK_RESTART などの *optag* オプションには、大文字と小文字の区別は*ありません*。*optag* オプションを指定するときには、大文字と小文字の任意の組み合わせを使用できます。

次の *optag* 値がサポートされています。

CHANGE_STATE_OFFLINE

-R オプションで指定されたプロキシリソースがローカルノードでオフラインになるようにリクエストします。*プロキシリソース*は、リソースの状態を Oracle Clusterware などの別のクラスタからインポートする Oracle Solaris Cluster リソースです。Oracle Solaris Cluster ソフトウェアのコンテキスト内での、状態のこの変化には、外部リソースの状態の変化が反映されます。

プロキシリソースの状態をこの *optag* 値で変更すると、プロキシリソースのメソッドは実行されません。

ノード上にある「依存先の」リソースで障害が発生して、リソースを回復できない場合、モニターはそのノード上のそのリソースをオフラインにします。モニターは CHANGE_STATE_OFFLINE *optag* 値を指定して scha_control コマンドを呼び出すことでリソースをオフラインにします。また、モニターは、依存先リソースのオフライン、再起動に従属する対象をすべて、それらに対して再起動をトリガーすることにより、オフラインにします。クラスタ管理者が障害を解決し、依存先のリソースを再度有効にすると、モニターは、リソースのオフライン再起動依存リソースも再度オンラインにします。

CHANGE_STATE_ONLINE

-R オプションで指定されたプロキシリソースがローカルノードでオンラインになるようにリクエストします。*プロキシリソース*は、リソースの状態を Oracle Clusterware などの別のクラスタからインポートする Oracle Solaris Cluster リソースです。Oracle Solaris Cluster ソフトウェアのコンテキスト内での、状態のこの変化には、外部リソースの状態の変化が反映されます。

プロキシリソースの状態をこの *optag* 値で変更すると、プロキシリソースのメソッドは実行されません。

CHECK_GIVEOVER

-G オプションで指定されたリソースグループに対して GIVEOVER を実行するために必要な有効性チェックをすべて実行します。ただし、実際にリソースグループを再配置することはありません。

CHECK_RESTART

-G オプションで指定されたリソースグループに対して RESTART を実行するために必要な有効性チェックをすべて実行します。ただし、実際にリソースグループを再起動することはありません。

GIVEOVER

-G オプションで指定されたリソースグループをローカルノード上でオフラインにし、RGM によって選択された別のノード上で再度オンラインにするようにリクエストします。リソースグループが 2 つ以上のノード上で現在オンライン状態にあり、これ以上リソースグループをオンライン化できるノードが存在しない場合、このリソースグループは、ローカルノード上でオフラインになったままオンラインにならない可能性があります。またこうした処理要求は、各種のチェックの結果を受けた結果として、拒絶される場合もあります。たとえば、あるノード上で GIVEOVER リクエストにより、グループが PINGPONG_INTERVAL プロパティーで指定された間隔でオフラインになったために、このノードがホストとして拒絶されることがあります。

クラスタ管理者が 1 つまたは複数のリソースグループの RG_Affinities プロパティーを構成している場合、あるリソースグループで **scha_control GIVEOVER** 要求を発行すると、複数のリソースグループが再配置されることがあります。RG_Affinities プロパティーは、1319 ページのrg_properties(5) で説明されています。

MONITOR_CHECK メソッドは、リソースを含むリソースグループが障害モニターからの scha_control コマンドまたは scha_control() 関数の呼び出しの結果として新しいノードに再配置される前に呼び出されます。

MONITOR_CHECK メソッドは、リソースグループの潜在的な新しいマスターであるノード上で呼び出すことができます。MONITOR_CHECK メソッドは、リソース実行するのに十分なほど健全であるかどうかを評価します。MONITOR_CHECK メソッドは、並行して実行されるほかのメソッドと衝突しない方法で実装する必要があります。

MONITOR_CHECK が失敗した場合、コールバックが呼び出されたノードへのリソースグループの再配置は拒否されます。

IGNORE_FAILED_START

現在実行中の Prenet_start または Start メソッドが失敗した場合、Failover_mode プロパティーの設定とは関係なく、リソースグループをフェイルオーバーしないようにリクエストします。

言い換えると、この *optag* 値は、Failover_Mode プロパティーが SOFT または HARD に設定されているリソースに対して、通常、そのリソースが起動に失敗したときに行われる回復アクションを無効にします。通常、リソースグループは異なるノードにフェイルオーバーします。ところが、(この値が設定されている場合)、リソースは Failover_Mode が NONE に設定されているかのように動作します。このリソースは START_FAILED 状態になり、その他のエラーが発生していない場合、リソースグループは ONLINE_FAULTED 状態で終了します。

この optag 値は、0 以外の終了コードまたはタイムアウトで終了する Start または Prenet_start メソッドから呼び出された場合にのみ有効です。また、この optag 値は、現在の Start または Prenet_start メソッドの呼び出しに対してのみ有効です。Start メソッドにより、リソースを別のノード上で正常に起動できないと判断された場合は、この optag 値を指定して scha_control コマンドを呼び出すようにしてください。この optag 値がその他のメソッドによって呼び出された場合、SCHA_ERR_INVAL

エラーが返されます。この optag 値により、発生するはずのリソースグループの「ping pong」フェイルオーバーが行われません。

RESOURCE_DISABLE

-R オプションで指定されたリソースを scha_control コマンドが呼び出されるノードで無効にします。

ノード上にある「依存先の」リソースで障害が発生して、リソースを回復できない場合、モニターはそのノード上のそのリソースをオフラインにします。モニターは、scha_control コマンドを RESOURCE_DISABLE *optag* 値とともに呼び出して、リソースをオフラインにします。また、モニターは、依存先リソースのオフライン、再起動に従属する対象をすべて、それらに対して再起動をトリガーすることにより、オフラインにします。クラスタ管理者が障害を解決し、依存先のリソースを再度有効にすると、モニターは、リソースのオフライン再起動依存リソースも再度オンラインにします。

RESOURCE_IS_RESTARTED

-R オプションで指定したリソースのリソース再起動カウンタを、実際にリソースを再起動することなく、ローカルノード上で増分するようにリクエストします。

scha_control の RESOURCE_RESTART オプションを呼び出すことなく (たとえば、735 ページのpmfadm(1M) を使用して) リソースを直接再起動するリソースモニターは、このオプションを使用して、リソースが再起動されたことを RGM に通知できます。この増分は、691 ページのscha_resource_get(1HA) の後続の NUM_RESOURCE_RESTARTS クエリーで反映されます。

リソースタイプが RETRY_INTERVAL 標準プロパティーの宣言に失敗した場合、scha_control コマンドの RESOURCE_IS_RESTARTED オプションは使用できなくなります。その結果、scha_control コマンドは失敗し、終了ステータスコード 13 (SCHA_ERR_RT) を生成します。

RESOURCE_RESTART

-R オプションで指定したリソースをオフラインにし、リソースグループ内のその他のリソースを停止することなく、ローカルノード上で再度オンラインにするようリクエストします。リソースの停止と起動は、ローカルノード上で次の順序でメソッドを適用することで行われます:

```
MONITOR_STOP
STOP
START
MONITOR_START
```

リソースタイプが STOP メソッドおよび START メソッドを宣言していない場合は、代わりに POSTNET_STOP および PRENET_START を使用してリソースが再起動されます。

```
MONITOR_STOP
POSTNET_STOP
PRENET_START
MONITOR_START
```

リソースタイプが MONITOR_STOP メソッドおよび MONITOR_START メソッドを宣言していない場合、STOP メソッドと START メソッド、または POSTNET_STOP メソッドと PRENET_START メソッドだけが呼び出され、再起動が実行されます。

リソースの再起動時にメソッドの呼び出しに失敗した場合、RGM はリソースの FAILOVER_MODE プロパティーの設定に応じて、エラー状態の設定、リソースグループの再配置、またはノードのリブートを実行します。詳細は、1287 ページのr_properties(5) の FAILOVER_MODE プロパティーを参照してください。

このオプションを使用してリソースを再起動するリソースモニターは、691 ページのscha_resource_get(1HA) の NUM_RESOURCE_RESTARTS クエリーを使用して、再起動を試みた最近の回数を追跡できます。

RESOURCE_RESTART 関数は、PRENET_START、POSTNET_STOP、またはこの両方のメソッドを持つリソースタイプによって使用されなければなりません。こうしたリソースに対して使用できるのは MONITOR_STOP、STOP、START、MONITOR_START のメソッドのみです。このリソースが依存するネットワークアドレスリソースは再起動されず、オンライン状態のままになります。

ノード上にある「依存先の」リソースで障害が発生して、リソースを回復できない場合、モニターはそのノード上のそのリソースをオフラインにします。モニターは RESOURCE_RESTART *optag* 値を指定して scha_control コマンドを呼び出すことでリソースをオフラインにします。また、モニターは、依存先リソースのオフライン、再起動に従属する対象をすべて、それらに対して再起動をトリガーすることにより、オフラインにします。クラスタ管理者が障害を解決し、依存先のリソースを再度有効にすると、モニターは、リソースのオフライン再起動依存リソースも再度オンラインにします。

RESTART

-G オプションで指定したリソースグループをオフラインにし、別のノードに再配置することなく再度オンラインにするようにリクエストします。ただし、この要求を実行する際にグループ内のリソースの再起動が失敗した場合は、最終的にリソースグループが再配置されることもあります。このオプションを使用してリソースグループを再起動するリソースモニターは、691 ページのscha_resource_get(1HA) の NUM_RG_RESTARTS クエリーを使用して、再起動を試みた最近の回数を追跡できます。

CHECK_GIVEOVER と CHECK_RESTART の *optag* 値は、ギブオーバーや再起動を実行する際、scha_control コマンドを呼び出すのではなく、リソースに対して直接的なアクション (プロセスの強制終了と再起動、ノードのリブートなど) を取るリソースモニターによって使用されます。チェックに失敗した場合、モニターは、再起動やフェイルオーバーのアクションを呼び出す代わりに、しばらく休止したあと検証を再開します。詳細は、1101 ページのscha_control(3HA) を参照してください。

-R *resource*

リソースグループ内のリソースの名前で、このリソースはおそらく、そのモニターが683 ページのscha_control(1HA) 要求を行っていたものです。指定されたリソースがリソースグループ内に存在しない場合、要求は拒否されます。

指示されたリソースの Failover_mode プロパティーの設定は、要求された scha_control アクションを抑制する場合があります。Failover_mode が RESTART_ONLY である場合、scha_control GIVEOVER と scha_control CHECK_GIVEOVER を除くすべての要求が許可されます。GIVEOVER 要求および CHECK_GIVEOVER 要求は SCHA_ERR_CHECKS 終了コードを返し、要求されたギブオーバーアクションは実行されず、syslog メッセージのみを作成します。

Retry_count プロパティーと Retry_interval プロパティーがリソースで設定されている場合、リソース再起動の回数は Retry_interval 内での Retry_count の試行回数に制限されます。Failover_mode が LOG_ONLY である場合、任意の scha_control ギブオーバー、再起動、または無効の要求は SCHA_ERR_CHECKS 終了コードを返し、また要求されたギブオーバーまたは再起動アクションは実行されず、syslog メッセージのみを作成します。

-Z *zonename*

リソースグループが実行を構成されているゾーンの名前です。

Global_zone プロパティーに TRUE が設定されている場合、リソースを含むリソースグループが非大域ゾーンで動作しているときでも、メソッドは大域ゾーンで実行されます。このオプションは、リソースグループが実行するよう構成されている非大域ゾーンの名前を指定します。

Global_zone プロパティーが TRUE に設定されているリソースタイプにのみ、-Z オプションを使用します。Global_zone プロパティーが FALSE に設定されている場合、このオプションは必要ありません。Global_zone プロパティーの詳細は、1335 ページのrt_properties(5) のマニュアルページを参照してください。

次の終了ステータスコードが返されます。

0	コマンドは正常に完了しました。
0 以外	エラーが発生しました。 障害エラーコードは、1047 ページのscha_calls(3HA) で説明されています。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	安定

735 ページのpmfadm(1M), 645 ページのrt_callbacks(1HA),
675 ページのscha_cmds(1HA), 691 ページのscha_resource_get(1HA),

1047 ページの`scha_calls`(3HA), 1101 ページの`scha_control`(3HA), 1107 ページの`scha_control_zone`(3HA), Unresolved link to " attributes5", 1287 ページの`r_properties`(5), Unresolved link to " rbac5", 1319 ページの`rg_properties`(5), 1335 ページの`rt_properties`(5)

名前

scha_resource_get — リソース情報へのアクセス

```
scha_resource_get [-Q] -O optag -R resource [-G group] [-Z
    zoneclustername] [args]
```

scha_resource_get コマンドは、Resource Group Manager (RGM) の制御下にあるリソースの情報にアクセスします。このコマンドを使用して、1287 ページのr_properties(5) で説明されているリソースのプロパティーと同様に、1335 ページのrt_properties(5) で説明されているリソースタイプのプロパティーのクエリーを実行できます。

scha_resource_get コマンドは、クラスタの RGM で制御されるサービスを表すリソースタイプのコールバックメソッドのシェルスクリプト実装で使用します。このコマンドは、C 関数 1127 ページのscha_resource_get(3HA) と同じ情報を提供します。

これらの情報は、このコマンドによって、675 ページのscha_cmds(1HA) で説明されている書式付き文字列で stdout に別々の行で生成されます。あとでスクリプトで使用できるように、この出力をシェル変数に格納し、シェル機能または Unresolved link to " awk1" で解析できます。

このコマンドを使用するには solaris.cluster.resource.read の役割に基づくアクセス制御 (RBAC) が必要です。Unresolved link to " rbac5" を参照してください。

さらにこのコマンドを使用するにあたっては、Oracle Solaris Cluster コマンド権プロファイルを割り当てておく必要があります。認証されたユーザーは、Unresolved link to " pfsh1"、Unresolved link to " pfcsh1"、または Unresolved link to " pfksh1" プロファイルシェルから、Oracle Solaris Cluster の特権コマンドをコマンド行で実行できます。プロファイルシェルは、Oracle Solaris Cluster コマンド権プロファイルに割り当てられた Oracle Solaris Cluster の特権コマンドへのアクセスを可能にする特別なシェルです。プロファイルシェルは、Unresolved link to " su1M" を実行して役割を引き受けると起動されます。Unresolved link to " pfexec1" を使用しても、Oracle Solaris Cluster の特権コマンドを実行できます。

次のオプションがサポートされています。

-G *group*

内部でリソースが構成されているリソースグループの名前です。この引数はオプションですが、これを含めるとコマンドの実行効率が上がります。

-O *optag*

アクセスする情報を指定します。指定する *optag* 値によっては、情報を取得するクラスタノードを示すための追加の値を含める必要が生じる場合があります。

注記 - AFFINITY_TIMEOUT や BOOT_TIMEOUT などの *optag* 値には、大文字と小文字の区別はありません。*optag* 値を指定するときには、大文字と小文字の任意の組み合わせを使用できます。

次の *optag* 値は、対応するリソースのプロパティーを取得します。リソースの名前付きプロパティーの値が生成されます。NUM_RG_RESTARTS、NUM_RESOURCE_RESTARTS、MONITORED_SWITCH、ON_OFF_SWITCH、RESOURCE_STATE、および STATUS プロパティーは、コマンドが実行されたノード上の値を参照します。次の *optag* 値に対応するリソースプロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。次のリストにあるいくつかの *optag* 値は、1287 ページのr_properties(5) のマニュアルページ内にではなく、このリストのあとに説明があることに注意してください。

```
AFFINITY_TIMEOUT
ALL_EXTENSIONS
APPLICATION_USER
BOOT_TIMEOUT
CHEAP_PROBE_INTERVAL
EXTENSION
EXTENSION_NODE
FAILOVER_MODE
FINI_TIMEOUT
GLOBAL_ZONE_OVERRIDE
GROUP
INIT_TIMEOUT
LOAD_BALANCING_POLICY
LOAD_BALANCING_WEIGHT
SMONITORED_SWITCH
MONITORED_SWITCH_NODE
MONITOR_CHECK_TIMEOUT
MONITOR_START_TIMEOUT
MONITOR_STOP_TIMEOUT
NETWORK_RESOURCES_USED
NUM_RESOURCE_RESTARTS
NUM_RESOURCE_RESTARTS_ZONE
NUM_RG_RESTARTS
NUM_RG_RESTARTS_ZONE
ON_OFF_SWITCH
ON_OFF_SWITCH_NODE
PORT_LIST
POSTNET_STOP_TIMEOUT
PRE_EVICT
PRENET_START_TIMEOUT
RESOURCE_DEPENDENCIES
RESOURCE_DEPENDENCIES_OFFLINE_RESTART
RESOURCE_DEPENDENCIES_RESTART
RESOURCE_DEPENDENCIES_WEAK
```

RESOURCE_PROJECT_NAME
RESOURCE_STATE
RESOURCE_STATE_NODE
RETRY_COUNT
RETRY_INTERVAL
R_DESCRIPTION
SCALABLE
START_TIMEOUT
STATUS
STATUS_NODE
STOP_TIMEOUT
THOROUGH_PROBE_INTERVAL
TYPE
TYPE_VERSION
UDP_AFFINITY
UPDATE_TIMEOUT
VALIDATE_TIMEOUT
WEAK_AFFINITY

次の *optag* 値は、1287 ページのr_properties(5) のマニュアルページで説明されていません。

ALL_EXTENSIONS

リソースの拡張プロパティー名を複数の行に出力します。

EXTENSION

プロパティーの型とその値を連続した複数の行に生成します。プロパティーがノード単位の拡張プロパティーである場合、返される値は scha_resource_get が実行されるノード上のプロパティー値です。フラグが設定されていない引数を使って、リソースの拡張プロパティーを指定する必要があります。例 に示すように、シェルスクリプトは、値を取得するためにタイプを破棄する必要がある場合があります。

明示的な値が割り当てられていないノード上でユーザーがこのプロパティーの値をリクエストすると、リソースタイプ登録 (RTR) ファイルで宣言されているデフォルト値が返されます。1251 ページのrt_reg(4) のマニュアルページを参照してください。

EXTENSION_NODE

指定のノード用に、プロパティーのタイプに続いて、その値を後続の行に生成します。この値には、特定のノード上のリソースの拡張子を指定する、2 つのフラグなしの引数が次の順序で必要です:

- 拡張プロパティー名
- ノード名

シェルスクリプトは、値を取得するためにタイプを破棄する必要がある場合があります。

明示的な値が割り当てられていないノード上でこのプロパティーの値をユーザーがリクエストすると、RTR ファイルで宣言されているデフォルト値が返されます。1251 ページのrt_reg(4) のマニュアルページを参照してください。

GROUP

リソースを構成するリソースグループの名前を指定します。

RESOURCE_DEPENDENCIES_NODE

指定のノード用に、リソースの RESOURCE_DEPENDENCIES プロパティーの値を生成します。ノードを指定する、フラグが設定されていない引数が必要です。

RESOURCE_DEPENDENCIES_OFFLINE_RESTART_NODE

指定のノード用に、リソースの RESOURCE_DEPENDENCIES_OFFLINE_RESTART プロパティーの値を生成します。ノードを指定する、フラグが設定されていない引数が必要です。

RESOURCE_DEPENDENCIES_RESTART_NODE

指定のノード用に、リソースの RESOURCE_DEPENDENCIES_RESTART プロパティーの値を生成します。ノードを指定する、フラグが設定されていない引数が必要です。

RESOURCE_DEPENDENCIES_WEAK_NODE

指定のノード用に、リソースの RESOURCE_DEPENDENCIES_WEAK プロパティーの値を生成します。ノードを指定する、フラグが設定されていない引数が必要です。

RESOURCE_STATE_NODE

指定のノード用に、リソースの RESOURCE_STATE プロパティーの値を生成します。ノードを指定する、フラグが設定されていない引数が必要です。

STATUS_NODE

指定のノード用に、リソースの STATUS プロパティーの値を生成します。ノードを指定する、フラグが設定されていない引数が必要です。

次の *optag* 値は、対応するリソースタイプのプロパティーを取得します。リソースのタイプの名前付きプロパティーの値が生成されます。

注記 - API_VERSION や BOOT などの *optag* 値には、大文字と小文字の区別はありません。*optag* 値を指定するときには、大文字と小文字の任意の組み合わせを使用できます。

リソースタイププロパティーについては、1335 ページの rt_properties(5) を参照してください。

API_VERSION
BOOT
FAILOVER
FINI
GLOBAL_ZONE
INIT
INIT_NODES
INSTALLED_NODES

```
IS_LOGICAL_HOSTNAME
IS_SHARED_ADDRESS
MONITOR_CHECK
MONITOR_START
MONITOR_STOP
PKGLIST
POSTNET_STOP
PRENET_START
PROXY
RT_BASEDIR
RT_DESCRIPTION
RT_SYSTEM
RT_VERSION
SINGLE_INSTANCE
START
STOP
UPDATE
VALIDATE
```

このリソースタイプが GLOBAL_ZONE_OVERRIDE リソースプロパティーを宣言した場合、GLOBAL_ZONE *optag* によって取得された値は、GLOBAL_ZONE プロパティーの値ではなく、GLOBAL_ZONE_OVERRIDE プロパティーの現在の値となります。詳細は、1335 ページのrt_properties(5) のマニュアルページの Global_zone プロパティー、および 1287 ページのr_properties(5) のマニュアルページの Global_zone_override プロパティーの説明を参照してください。

-Q

指定された修飾子は、いずれもリソース依存性リストに含めます。{LOCAL_NODE}、{ANY_NODE}、@nodename、および {FROM_RG_AFFINITIES} 修飾子は、1287 ページのr_properties(5) のマニュアルページで説明されています。-Q オプションを省略すると、リソースの依存関係リストの戻り値には、ローカルノードで適用可能な依存関係のリソース名 (宣言された修飾子を含まない) のみが含まれます。

-R *resource*

RGM クラスタ機能によって管理されているリソースの名前

-Z *zoneclustername*

リソースグループが存在し、操作するクラスタを指定します。このオプションは、コマンドが大域ゾーンで実行されるものの、指定されたゾーンクラスタで動作する必要がある場合に使用します。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

zoneclustername　　クエリーが *zoneclustername* という名前のゾーンクラスタで実行されるように指定します。

-Z オプションを省略すると、コマンドが実行されるクラスタでクエリーが実行されます。

例 273 scha_resource_get コマンドを使用したサンプルスクリプト

次のスクリプトには、必要なリソース名とリソースグループ名を指定する -R 引数と -G 引数が渡されます。次に、scha_resource_get コマンドは、リソースの Retry_count プロパティーと enum 型の LogLevel 拡張プロパティーにアクセスします。

```
#!/bin/sh

while getopts R:G: opt
do
     case $opt in
           R)      resource="$OPTARG";;
           G)      group="$OPTARG";;
     esac
done

retry_count=`scha_resource_get -O Retry_count -R $resource \\
-G $group`
printf "retry count for resource %s is %d\n" $resource \\
$retry_count

LogLevel_info=`scha_resource_get -O Extension -R $resource \\
-G $group LogLevel`

# Get the enum value that follows the type information
# of the extension property.  Note that the preceding
# assignment has already changed the newlines separating
# the type and the value to spaces for parsing by awk.

loglevel=`echo $LogLevel_info | awk '{print $2}'`
```

例 274 リソース依存性のクエリーを行うために -Q オプション付きまたはオプションなしで scha_resource_get コマンドを使用

この例では、clresource コマンドを使用して、myres という名前のリソースを作成する方法を示します。このリソースは、{LOCAL_NODE} スコープ修飾子または {ANY_NODE} スコープ修飾子があるか、あるいはスコープ修飾子がない複数のリソース依存性を持つという前提です。この例では、次に、scha_resource_get コマンドを使用して、Resource_dependencies プロパティーのクエリーを実行する方法を示します。-Q オプションがない場合は、リソース名のみが返されます。-Q オプションがある場合は、宣言されたスコープ修飾子も返されます。

```
# clresource create -g mygrp -t myrestype \
-p Resource_dependencies=myres2{LOCAL_NODE},myres3{ANY_NODE},myres4 \
myres
# scha_resource_get -O Resource_dependencies -R myres -G mygrp

myres2
```

```
myres3
myres4
# scha_resource_get -Q -O Resource_dependencies -R myres -G mygrp

myres2{LOCAL_NODE}
myres3{ANY_NODE}
myres4
#
```

例 275 リソースの依存関係のプロパティーの表示

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

ptrancos1 ノードから:

```
ptrancos1(/root)$ scha_resource_get -O RESOURCE_DEPENDENCIES -R gds-rs

trancos-3-rs
ptrancos1(/root)$ scha_resource_get -O RESOURCE_DEPENDENCIES_NODE -R gds-rs ptrancos1

trancos-3-rs
ptrancos1(/root)$ scha_resource_get -O RESOURCE_DEPENDENCIES_NODE -R gds-rs ptrancos2

trancos-4-rs
ptrancos1(/root)$ scha_resource_get -Q -O RESOURCE_DEPENDENCIES -R gds-rs

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

trancos-3-rs
```

ptrancos2 ノードから:

```
ptrancos2(/root)$ scha_resource_get -O RESOURCE_DEPENDENCIES -R gds-rs

trancos-4-rs
ptrancos2(/root)$ scha_resource_get -O RESOURCE_DEPENDENCIES_NODE -R gds-rs ptrancos1

trancos-3-rs
ptrancos2(/root)$ scha_resource_get -O RESOURCE_DEPENDENCIES_NODE -R gds-rs ptrancos2

trancos-4-rs
ptrancos2(/root)$ scha_resource_get -Q -O RESOURCE_DEPENDENCIES -R gds-rs

trancos-3-rs@ptrancos1
```

trancos-4-rs@ptrancos2

```
ptrancos2(/root)$ scha_resource_get -O NETWORK_RESOURCES_USED -R gds-rs
```

trancos-4-rs

次の終了ステータスコードが返されます。

`0`	コマンドは正常に完了しました。
0 以外	エラーが発生しました。 障害エラーコードは、1047 ページの`scha_calls`(3HA) で説明されています。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	発展中

Unresolved link to " awk1", 675 ページの`scha_cmds`(1HA),
1047 ページの`scha_calls`(3HA), 1127 ページの`scha_resource_get`(3HA),
1251 ページの`rt_reg`(4), Unresolved link to " attributes5",
1271 ページの`property_attributes`(5), 1287 ページの`r_properties`(5),
1335 ページの`rt_properties`(5)

名前

scha_resource_setstatus — リソースステータスの設定

```
scha_resource_setstatus -R resource -G group -s status [-m msg]
     [-Z zonename]
```

scha_resource_setstatus コマンドは、Resource Group Manager (RGM) が管理するリソースの Status プロパティーと Status_msg プロパティーを設定します。リソースのモニターは、このコマンドを使用して、認識したリソースのステータスを示します。このコマンドは、C 関数 1175 ページのscha_resource_setstatus(3HA) と同じ機能を提供します。

699 ページのscha_resource_setstatus(1HA) コマンドを実行した場合、リソースの Status および Status_msg プロパティーは、指定した値で更新されます。Oracle Solaris Cluster は、この変更をクラスタシステムログのリソースステータスにログとして記録します。このログはクラスタ管理ツールで表示することができます。

このコマンドを使用するためには、solaris.cluster.resource.admin RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

さらにこのコマンドを使用するにあたっては、Oracle Solaris Cluster コマンド権プロファイルを割り当てておく必要があります。認証されたユーザーは、Unresolved link to " pfsh1"、Unresolved link to " pfcsh1"、または Unresolved link to " pfksh1" プロファイルシェルから、Oracle Solaris Cluster の特権コマンドをコマンド行で実行できます。プロファイルシェルは、Oracle Solaris Cluster コマンド権プロファイルに割り当てられた Oracle Solaris Cluster の特権コマンドへのアクセスを可能にする特別なシェルです。プロファイルシェルは、Unresolved link to " su1M" を実行して役割を引き受けると起動されます。Unresolved link to " pfexec1" を使用しても、Oracle Solaris Cluster の特権コマンドを実行できます。

次のオプションがサポートされています。

-G *group*

このリソースを含むリソースグループを指定します。

-m *msg*

このリソースの Status_msg プロパティーに割り当てるテキスト文字列を指定します。このオプションを指定しないと、リソースの Status_msg の値は NULL に設定されます。

-R *resource*

ステータスを設定するリソースを指定します。

-s *status*

status の値 (OK、DEGRADED、FAULTED、UNKNOWN、または OFFLINE) を指定します。

-Z *zonename*

リソースグループを実行するように構成されるゾーンクラスタの名前を指定します。

Global_zone プロパティーに TRUE が設定されている場合、リソースを含むリソースグループがゾーンクラスタに構成されている場合でも、メソッドは大域ゾーンで実行されます。-Z オプションは、メソッドが動作する大域ゾーンではなく、リソースグループが動作する非大域ゾーンのステータスを設定します。

Global_zone プロパティーが TRUE に設定されているリソースタイプにのみ、-Z オプションを使用します。Global_zone プロパティーが FALSE に設定されている場合、このオプションは必要ありません。Global_zone プロパティーの詳細は、1335 ページのrt_properties(5) のマニュアルページを参照してください。

例 276 Status_msg を使用したリソース R1 のステータスの設定

次のコマンドでは、リソースグループ RG2 内のリソース R1 のステータスを OK に設定し、Status_msg を Resource R1 is OK に設定します:

```
scha_resource_setstatus -R R1 -G RG2 -s OK -m "Resource R1 is OK"
```

例 277 Status_msg を使用しないリソース R1 のステータスの設定

次のコマンドでは、リソースグループ RG2 内の R1 のステータスを DEGRADED に設定し、Status_msg を NULL に設定します:

```
scha_resource_setstatus -R R1 -G RG2 -s DEGRADED
```

例 278 Status_msg を使用したゾーン Zone1 内でのリソース R1 のステータスの設定

次の例に、シェルスクリプトとして実装されたリソースタイプメソッドまたはモニターを示します。次のシェルスクリプトでは、ゾーン $localzone 内のリソースグループ $rg のリソース $resource のステータスを OK に設定する方法を示します。このシェルスクリプトはまた、Status_msg を Resource R1 is OK に設定します。この場合、リソースタイププロパティー Global_zone は TRUE に設定されていることが前提であるため、-Z オプションを指定する必要があります。

```
resource=
rg=""
localzone=""
```

```
zflag=""
while getopts R:G:Z:
do
      case $c in
      R)   resource=$OPTARG;;
      G)   rg=$OPTARG;;
      Z)   zflag="-Z"
           localzone=$OPTARG;;
      esac
done
...
scha_resource_setstatus -R $resource -G $rg $zflag $localzone -s OK -m
    "Resource R1 is OK"
```

次の終了ステータスコードが返されます。

`0`	コマンドは正常に完了しました。
0 以外	エラーが発生しました。 障害エラーコードは、1047 ページの`scha_calls`(3HA) で説明されています。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	安定

675 ページの`scha_cmds`(1HA), 1047 ページの`scha_calls`(3HA), 1175 ページの`scha_resource_setstatus`(3HA), Unresolved link to " attributes5", Unresolved link to " rbac5", 1335 ページの`rt_properties`(5)

名前

scha_resourcegroup_get — リソースグループ情報へのアクセス

```
scha_resourcegroup_get -O optag -G group [-Z zonecluster] [args]
```

`scha_resourcegroup_get` コマンドは、クラスタの Resource Group Manager (RGM) の制御下にあるリソースグループの情報にアクセスします。

このコマンドは、リソースタイプに対するコールバックメソッドのシェルスクリプト実装で使用するためのものです。このようなリソースタイプは、クラスタの RGM によって制御されるサービスを表しています。このコマンドは、C 関数 1185 ページの`scha_resourcegroup_get`(3HA) と同じ情報を提供します。

これらの情報は、このコマンドによって、675 ページの`scha_cmds`(1HA) で説明されている書式付き文字列で標準出力 (`stdout`) に生成されます。各行に単一または複数の文字列が出力されます。スクリプトでさらに使用するために、この出力をシェル変数に格納し、シェル機能または Unresolved link to " awk1" を使用して解析できます。

このコマンドを使用するには `solaris.cluster.resource.read` の役割に基づくアクセス制御 (RBAC) が必要です。Unresolved link to " rbac5" を参照してください。

さらにこのコマンドを使用するにあたっては、Oracle Solaris Cluster コマンド権プロファイルを割り当てておく必要があります。認証されたユーザーは、Unresolved link to " pfsh1"、Unresolved link to " pfcsh1"、または Unresolved link to " pfksh1" プロファイルシェルから、Oracle Solaris Cluster の特権コマンドをコマンド行で実行できます。プロファイルシェルは、Oracle Solaris Cluster コマンド権プロファイルに割り当てられた Oracle Solaris Cluster の特権コマンドへのアクセスを可能にする特別なシェルです。プロファイルシェルは、Unresolved link to " su1M" を実行して役割を引き受けると起動されます。Unresolved link to " pfexec1" を使用しても、Oracle Solaris Cluster の特権コマンドを実行できます。

次のオプションがサポートされています。

`-G` *group*
: リソースグループの名前。

-O *optag*

アクセスする情報を指定します。指定した *optag* によっては、情報を取得するクラスタノードまたはゾーンを示すオペランドを追加する必要があります。

注記 - DESIRED_PRIMARIES や FAILBACK などの *optag* 値には、大文字と小文字の区別はありません。*optag* オプションを指定するときには、大文字と小文字の任意の組み合わせを使用できます。

次の *optag* 値は、対応するリソースグループのプロパティーを取得します。リソースグループの名前付きプロパティー値が生成されます。RG_STATE プロパティーは、コマンドの実行元である特定のノード上の値を示します。

```
ALL_LOAD_FACTORS
ALL_LOAD_FACTOR_NAMES
AUTO_START_ON_NEW_CLUSTER
DESIRED_PRIMARIES
FAILBACK
GLOBAL_RESOURCES_USED
IMPLICIT_NETWORK_DEPENDENCIES
LOAD_FACTOR
MAXIMUM_PRIMARIES
NODELIST
PATHPREFIX
PINGPONG_INTERVAL
PREEMPTION_MODE
PRIORITY
RESOURCE_LIST
RG_AFFINITIES
RG_DEPENDENCIES
RG_DESCRIPTION
RG_IS_FROZEN
RG_MODE
RG_PROJECT_NAME
RG_SLM_TYPE
RG_SLM_PSET_TYPE
RG_SLM_CPU
RG_SLM_CPU_MIN
RG_STATE
RG_STATE_NODE
RG_SYSTEM
SUSPEND_AUTOMATIC_RECOVERY
TARGET_NODES
```

-Z *zoneclustername*

リソースグループが存在し、操作するクラスタを指定します。このオプションは、コマンドが大域ゾーンで実行されるものの、指定されたゾーンクラスタで動作する必要がある場合に使用します。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

zoneclustername クエリーが *zoneclustername* という名前のゾーンクラスタで実行されるように指定します。

-Z オプションを省略すると、コマンドが実行されるクラスタでクエリーが実行されます。

グローバルクラスタのリソースグループ状態など、ゾーン別のプロパティー値をクエリーするには、-Z オプションは使用しないでください。代わりに、クエリータグのゾーン別の書式を使用します。たとえば、RG_STATE の代わりに RG_STATE_NODE を使用し、*nodename:zonename* という形式の追加のコマンド行引数を指定します。

注記 - RG_STATE_NODE には、ノードを指定するフラグなし引数が必要です。この *optag* 値は、指定したノードに対するリソースグループの RG_STATE プロパティーの値を生成します。フラグなしの引数が非大域ゾーンを指定する場合、その形式は *nodename*:*zonename* となります。

例 279 scha_resourcegroup_get を使用するサンプルスクリプト

次のスクリプトには、必要なリソースグループ名を示す引数 -G が渡されます。次に、scha_resourcegroup_get コマンドにより、リソースグループ内のリソースのリストが取得されます。

```
#!/bin/sh

while getopts G: opt
do
     case $opt in
          G)      group="$OPTARG";;
     esac
done

resource_list=`scha_resourcegroup_get -O Resource_list -G $group`

for resource in $resource_list
do
     printf "Group: %s contains resource: %s\n" "$group" "$resource"
done
```

例 280 scha_resourcegroup_get コマンドを使用して、リソースグループのすべての負荷係数をクエリーする

rg1 というリソースグループのすべての負荷係数を表示するには、次のコマンドを使用します。

```
# scha_resourcegroup_get -O ALL_LOAD_FACTORS -G rg1
factor1=50
factor2=1
factor3=0
```

例 281 scha_resourcegroup_get コマンドを使用して、リソースグループに定義されているすべての負荷係数名を一覧表示する

rg1 というリソースグループに定義されているすべての負荷係数のリストを取得するには、次のコマンドを使用します。

```
# scha_resourcegroup_get -O ALL_LOAD_FACTOR_NAMES -G rg1

factor1
factor2
factor3
```

例 282 scha_resourcegroup_get コマンドを使用して、リソースグループの特定の負荷係数をクエリーする

rg1 というリソースグループに定義されている特定の負荷係数を表示するには、次のコマンドを使用します。

```
# scha_resourcegroup_get -O LOAD_FACTOR -G rg1 factor1

50
```

例 283 scha_resourcegroup_get コマンドを使用して、リソースグループの優先順位をクエリーする

rg1 というリソースグループに設定されている優先順位を表示するには、次のコマンドを使用します。

```
# scha_resourcegroup_get -O PRIORITY -G rg1
501
```

例 284 scha_resourcegroup_get コマンドを使用して、リソースグループのプリエンプションモードをクエリーする

rg1 というリソースグループに設定されているプリエンプションモードを表示するには、次のコマンドを使用します。

```
# scha_resourcegroup_get -O PREEMPTION_MODE -G rg1
Has_Cost
```

次の終了ステータスコードが返されます。

0 コマンドは正常に完了しました。

0 以外 エラーが発生しました。

障害エラーコードは、1047 ページのscha_calls(3HA) で説明されています。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	安定

Unresolved link to " awk1", 185 ページのclnode(1CL), 675 ページのscha_cmds(1HA), 1047 ページのscha_calls(3HA), 1185 ページのscha_resourcegroup_get(3HA), Unresolved link to " attributes5", 1319 ページのrg_properties(5), Unresolved link to " rbac5"

名前

scha_resourcetype_get — リソースタイプ情報にアクセス

```
scha_resourcetype_get -O optag -T type [-Z zoneclustername]
```

`scha_resourcetype_get` コマンドは、Resource Group Manager (RGM) で登録されているリソースタイプの情報にアクセスします。

このコマンドは、クラスタの RGM で制御されるサービスを表すリソースタイプのコールバックメソッドのシェルスクリプト実装で使用します。このコマンドは、C 関数 1215 ページの`scha_resourcetype_get`(3HA) と同じ情報を提供します。

これらの情報は、このコマンドによって、675 ページの`scha_cmds`(1HA) のマニュアルページで説明されている書式付き文字列で標準出力 (`stdout`) に出力されます。各行に単一または複数の文字列が出力されます。出力はシェル変数に格納できます。また、出力を Unresolved link to " awk1" コマンドまたはほかのシェルコマンドを使用して、スクリプトがさらに使用できるように解析することもできます。

このコマンドを使用するためには、`solaris.cluster.resource.read` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

認証されたユーザーは、Unresolved link to " pfsh1"、Unresolved link to " pfcsh1"、または Unresolved link to " pfksh1" プロファイルシェルから、Oracle Solaris Cluster の特権コマンドをコマンド行で実行できます。プロファイルシェルは、Oracle Solaris Cluster コマンド権プロファイルに割り当てられた Oracle Solaris Cluster の特権コマンドへのアクセスを可能にする特別なシェルです。プロファイルシェルは、Unresolved link to " su1M" を実行すると起動されます。Unresolved link to " pfexec1" を使用しても、Oracle Solaris Cluster の特権コマンドを実行できます。

次のオプションがサポートされています。

-O *optag*
: アクセスする情報を指定します。

注記 - `API_VERSION` や `BOOT` などの *optag* オプションには、大文字と小文字の区別はありません。*optag* オプションを指定するときには、大文字と小文字の任意の組み合わせを使用できます。

次の *optag* 値は、対応するリソースタイプのプロパティーを取得します。出力は、リソースタイプの名前付きプロパティーの値です。

```
API_VERSION
BOOT
FAILOVER
FINI
GLOBAL_ZONE
INIT
INIT_NODES
INSTALLED_NODES
IS_LOGICAL_HOSTNAME
IS_SHARED_ADDRESS
MONITOR_CHECK
MONITOR_START
MONITOR_STOP
PKGLIST
POSTNET_STOP
PRENET_START
PROXY
RESOURCE_LIST
RT_BASEDIR
RT_DESCRIPTION
RT_SYSTEM
RT_VERSION
SINGLE_INSTANCE
START
STOP
UPDATE
VALIDATE
```

-T *type*

クラスタの RGM で使用するように登録されたリソースタイプの名前です。

-Z *zoneclustername*

操作するクラスタを指定します。このオプションは、コマンドが大域ゾーンで実行されるものの、指定されたゾーンクラスタで動作する必要がある場合に使用します。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

zoneclustername クエリーが *zoneclustername* という名前のゾーンクラスタで実行されるように指定します。

-Z オプションを省略すると、コマンドが実行されるクラスタでクエリーが実行されます。

次の終了値が返されます。

0	コマンドは正常に完了しました。
0 以外	エラーが発生しました。 障害エラーコードは、1047 ページのscha_calls(3HA) で説明されています。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	安定

Unresolved link to " awk1", 675 ページのscha_cmds(1HA),
1047 ページのscha_calls(3HA), 1215 ページのscha_resourcetype_get(3HA),
Unresolved link to " attributes5", 1335 ページのrt_properties(5)

OSC4 1m

名前

ccradm — CCR テーブルファイル管理コマンド

```
/usr/cluster/lib/sc/ccradm subcommand [-?]

/usr/cluster/lib/sc/ccradm addkey [-Z zoneclustername] -v value -k key ccrtablefile

/usr/cluster/lib/sc/ccradm addtab [-Z zoneclustername] ccrtablefile

/usr/cluster/lib/sc/ccradm changekey [-Z zoneclustername] -v value -k key ccrtablefile

/usr/cluster/lib/sc/ccradm delkey [-Z zoneclustername] -k key ccrtablefile

/usr/cluster/lib/sc/ccradm recover [-Z zoneclustername] -f -o ccrtablefile

/usr/cluster/lib/sc/ccradm remtab [-Z zoneclustername] ccrtablefile

/usr/cluster/lib/sc/ccradm replace [-Z zoneclustername] -i newdatafile ccrtablefile

/usr/cluster/lib/sc/ccradm showkey [-Z zoneclustername] -k key ccrtablefile
```

`ccradm` コマンドは、クラスタ構成リポジトリ (CCR) 情報をサポートします。

CCR 情報は、`/etc/cluster/ccr` ディレクトリに存在します。グローバルクラスタに関する CCR 情報は、`/etc/cluster/ccr/global` ディレクトリに存在します。ゾーンクラスタ *zoneclustername* に関する CCR 情報は、`/etc/cluster/ccr/zoneclustername` ディレクトリに存在します。CCR 情報は、サポートされているプログラミングインタフェースからのみアクセスできるようにしてください。ファイルのアクセス権は、CCR 情報に直接アクセスできないように意図的に設定されています。

CCR 情報は、テーブルの形式で格納され、1 つのテーブルがその専用のファイルに格納されます。CCR テーブルファイルの各行は、キーと値の 2 つの ASCII 文字列で構成されます。各 CCR ファイルは、世代番号 (`ccr_gennum`) およびチェックサム (`ccr_checksum`) で始まります。

`ccr_gennum` は、CCR テーブルファイルの現在の世代番号を示します。システムは `ccr_gennum` を管理します。もっとも大きい番号がファイルの最新バージョンです。

`ccr_checksum` は、CCR テーブルコンテンツのチェックサムを示し、テーブルのデータの整合性検査を提供します。システムは、無効なチェックサムでは CCR テーブルファイルを使用しません。

ccrtablefile ファイルは、ローカルノード上の CCR テーブルを表すファイルの名前です。`-Z` オプションを指定すると、*ccrtablefile* は指定したゾーンクラスタに属します。`-Z` オプションを指定し

ないと、*ccrtablefile* はグローバルクラスタに属します。グローバルクラスタおよびゾーンクラスタは、それぞれ同じ名前の *ccrtablefile*を持つことができますが、含まれる情報は異なります。

このコマンドは、大域ゾーンだけで使用できます。

このコマンドには、次のサブコマンドがあります:

`addkey`

指定したクラスタの CCR テーブルファイルにキーと値のペアを追加します。`-s` オプションとともに使用すると、データは文字列値になります。`-f` オプションとともに使用すると、値はファイルの最初の文字列になり、ファイルに 1 つの文字列のみが含まれます。ファイルがこの形式でない場合、コマンドによってエラーが返されます。

このサブコマンドはクラスタモードでのみ使用できます。

`addtab`

指定したクラスタの CCR にテーブルを作成します。テーブルには、最初に `ccr_gennum` および `ccr_checksum` のみが含まれます。

このサブコマンドはクラスタモードでのみ使用できます。

`changekey`

指定したキーと新しい値に基づいて、CCR テーブルファイルのキーの値を変更します。キーが CCR テーブルファイルに見つからない場合、コマンドは `ESPIPE` を返します。`-s` オプションとともに使用すると、データは文字列値になります。`f` オプションとともに使用すると、値はファイルの最初の文字列になり、ファイルに 1 つの文字列のみが含まれます。ファイルがこの形式でない場合、コマンドによってエラーが返されます。

このサブコマンドはクラスタモードでのみ使用できます。

`delkey`

指定したキーに基づいて、キーと値のペアを CCR テーブルファイルから削除します。キーが CCR テーブルファイルに見つからない場合、コマンドは `ESPIPE` を返します。

このサブコマンドはクラスタモードでのみ使用できます。

`recover`

注意 - このサブコマンドは、CCR の内部処理に精通しているエンジニアのみが使用するようにしてください。このサブコマンドは手動による復旧操作をサポートしています。一般のユーザーはこのサブコマンドを使用しないようにしてください。

`recover` サブコマンドは、常に `ccr_gennum` の値を設定し、チェックサムを再計算して、CCR テーブルファイルの `ccr_checksum` の値を設定します。

`-o` オプションを指定せずに `recover` サブコマンドを使用すると、`recover` サブコマンドは世代番号を `INIT_VERSION` (`-1`) に設定します。`INIT_VERSION` の世代番号は、ローカルノー

ドがクラスタを再結合するまでの間のみ CCR テーブルファイルが有効であることを意味し、再結合時点で、クラスタは CCR テーブルファイルの内容を、クラスタ内の別のノードの CCR テーブルファイルの内容で置き換えます。前提条件は、クラスタ内のその他のノードのいずれかで、CCR テーブルファイルにオーバーライドバージョンを設定するか、その他のノードの少なくとも 1 つに、CCR テーブルファイルの有効なコピーがある必要があるということです。CCR テーブルファイルが有効になるのは、有効なチェックサムを持ち、その世代番号が 0 以上の場合です。

すべてのノード上で CCR テーブルファイルの世代番号が `INIT_VERSION` である場合は、復旧が完了したあとも CCR テーブルは無効のままになります。そのため、クラスタ内の少なくとも 1 つのノードの CCR テーブルファイルでは `recover` サブコマンドとともに `-o` オプションを使用する必要があります。

`-o` オプションとともに recover サブコマンドを使用すると、`recover` サブコマンドは世代番号を `OVRD_VERSION` (`-2`) に設定します。`OVRD_VERSION` の世代番号は、システムがローカルノードの CCR テーブルファイルの内容をほかのすべてのクラスタノードに伝えることを意味します。内容をほかのノードに伝えると、システムは世代番号を 0 に変更します。`OVRD_VERSION` の値を持つ CCR テーブルファイルは 1 つのノードのみに含めるようにしてください。`OVRD_VERSION` が複数のノードで同じ CCR テーブルファイルに設定されている場合、システムは任意に 1 つの CCR テーブルファイルの内容を使用します。

このサブコマンドは、非クラスタモードでのみ使用できます。

`remtab`

CCR からテーブルを削除します。

このサブコマンドはクラスタモードでのみ使用できます。

`replace`

注意 - このサブコマンドは、CCR の内部処理に精通しているエンジニアのみが使用するようにしてください。このサブコマンドは手動による復旧操作をサポートしています。一般のユーザーはこのサブコマンドを使用しないようにしてください。

ccrdatafile の内容を *newdatafile* の内容で置き換えます。チェックサムが再計算され、世代番号は 0 にリセットされます。

このサブコマンドはクラスタモードでのみ使用できます。

`showkey`

CCR テーブルファイルの指定されたキーの値を表示します。キーが CCR テーブルファイルに見つからない場合、コマンドは `ESPIPE` を返します。`showkey` コマンドは、指定したキーの値文字列のみ (そのあとに行末が続く) を標準出力に書き込みます。エラーが発生した場合、コマンドは何も書き込みません。

このサブコマンドはクラスタモードでのみ使用できます。

このコマンドには、次のオプションがあります:

-?
--help

ヘルプ情報を表示します。

このオプションはサブコマンド付きでもサブコマンドなしでも指定できます。

サブコマンドを指定しない場合、使用可能なすべてのサブコマンドのリストが表示されます。

サブコマンドを指定した場合、そのサブコマンドの使用法が表示されます。

このオプションとその他のオプションを指定すると、その他のオプションは無視されます。

-f
--force

ノードがクラスタメンバーとしてブートされる場合は recover サブコマンドを強制することを指定します。

-i *newdatafile*
--input=*newdatafile*
--input *newdatafile*

復旧操作に使用する CCR テーブルファイルを指定します。

-k
--key

追加、削除、または変更するキーの名前を指定します。

-o
--override

オーバーライドオプションは recover サブコマンドとともに使用します。世代番号を OVRD_VERSION に設定します。

このオプションは、1 つの CCR テーブルファイルをマスターコピーにするよう指定するために使用します。このマスターバージョンの CCR テーブルファイルは、復旧時に残りのノード上のほかのバージョンのファイルをオーバーライドします。複数のノードで CCR テーブルファイルの世代番号が OVRD_VERSION である場合は、1 つのファイルのみが選択され、警告メッセージが 1 つのノードのコンソールに出力されます。復旧後、テーブルの世代番号は 0 にリセットされます。

このオプションは、非クラスタモードモードでのみ使用できます。

-v *value*
--value=*value*
--value *value*

CCR テーブルのキーの値を指定します。値文字列に空白文字は使用できません。つまり、スペース、タブ、キャリッジリターン、または改行は使用できません。

-Z {*zoneclustername* | global}
--zoneclustername= {*zoneclustername* | global}
--zoneclustername {*zoneclustername* | global}

CCR トランザクションが実行されるクラスタを指定します。このオプションは、すべてのサブコマンドでサポートされています。このオプションを指定する場合は、次のいずれかの引数も指定する必要があります。

zoneclustername

このオプションを使用するコマンドが、*zoneclustername* という名前のゾーンクラスタでのみ指定されたすべてのリソースグループで作動するように指定します。

global

このオプションを使用するコマンドが、広域クラスタでのみ指定されたすべてのリソースグループで作動するように指定します。

-Z オプションは、クラスタモードでのみ使用できます。

次のオペランドだけがサポートされています。

ccrtablefile

管理対象となる CCR テーブルファイルを指定します。指定できるのは 1 つの *ccrtablefile* のみです。

ccradm コマンドは、CCR テーブルファイルの管理アクションに使用できます。

例 285 破損した CCR テーブルの修復および checksum の再計算

破損した CCR テーブルを修復するためのこれらの手順は、緊急の修復手順の一部として、承認された Oracle 担当者によって指示された場合にのみ実行してください。

次の例では、CCR テーブル ccr-file を修復します。

1. すべてのノードを非クラスタモードでリブートします。
2. すべてのノードのファイルを編集し、正しいデータが含まれるようにします。ファイルはすべてのノードで同じである必要があります。ファイルはすべてのノード同じであるため、すべてのノードでオーバーライドバージョンとして指定することもできます。
3. チェックサムを再計算し、すべてのノードで次のコマンドを実行して、この CCR テーブルファイルをオーバーライドバージョンに指定します (ccr-file は CCR テーブルの名前です)。

```
# ccradm recover -o ccr-file
```

4. すべてのノードをクラスタモードでリブートします。

例 286 バックアップバージョンからの破損した CCR テーブルの復元

次の例では、CCR テーブル yyy をバックアップバージョン (yyy.bak ファイル) の内容で置き換えます。コマンドは、クラスタモードの 1 つのノードから実行します。

```
# ccradm replace -Z global -i /etc/cluster/ccr/global/yyy.bak /etc/cluster/ccr/global/yyy
```

例 287 CCR テーブルの作成

次の例では、一時 CCR テーブル foo をゾーンクラスタ zc1 に作成します。コマンドは、クラスタモードの 1 つのノードから実行します。

```
# ccradm addtab -Z zc1 foo
```

例 288 CCR テーブルの削除

次の例は、グローバルクラスタからの CCR テーブル foo の削除を示しています。コマンドは、クラスタモードの 1 つのノードから実行します。

```
# ccradm remtab foo
```

例 289 CCR テーブルの変更

次の例では、グローバルクラスタ CCR テーブル rgm_rg_nfs-rg の Pingpong_interval プロパティーの値を 5400 に変更します。コマンドは、クラスタモードの 1 つのノードから実行します。

```
# ccradm changekey -s 5400 -k Pingpong_interval rgm_rg_nfs-rg
```

例 290 CCR テーブルのキー値の表示

次の例では、CCR テーブル rgm_rg_nfs-rg の Pingpong_interval プロパティーの値を表示します。

```
# ccradm showkey -k Pingpong_interval rgm_rg_nfs-rg
5400
```

次の終了値が返されます。

0
エラーは発生していません。

>0
エラーが発生しました。

次の属性の詳細は、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

名前

cl_eventd — クラスタイベントデーモン

```
/usr/cluster/lib/sc/cl_eventd [-v]
```

`cl_eventd` デーモンはブート時に開始され、ほかのクラスタコンポーネントで生成されたシステムイベントをモニターして、ほかのクラスタノードへ転送します。さらに、このデーモンは、これらのイベントをほかのクラスタノードに転送します。ただし転送されるイベントは、クラス `EC_Cluster` のイベントのみに限られます。

次のオプションがサポートされています。

-v　　Unresolved link to " syslogd1M"に追加のトラブルシューティング情報やデバッグ情報を送信します。

`/usr/cluster/lib/sc/cl_eventd`　　クラスタイベントデーモン

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core

Unresolved link to " syseventd1M", Unresolved link to " syslog3C"

`cl_eventd` デーモンは、アクセス可能なインタフェースを公開しません。

名前

cl_pnmd — Public Network Management (PNM) サービスデーモン

```
/usr/cluster/bin/cl_pnmd [-d [-t [tracefile]]]
```

`cl_pnmd` は、Public Network Management (PNM) モジュールのサーバーデーモンです。このデーモンが開始するのは通常、システムブート時です。このデーモンが起動すると、PNM サービスが開始されます。

Unresolved link to " in.mpathd1M" デーモンは、アダプタのテストおよびノード内フェイルオーバーをローカルホスト内のすべての IP ネットワークマルチパス (IPMP) グループに対して実行します。

`cl_pnmd` は、ローカルホストの IPMP 状態を追跡し、すべての IPMP グループのノード内のフェイルオーバーをサポートします。

次のオプションがサポートされています。

`-d`	`stderr` にデバッグメッセージを表示します。
`-t` *tracefile*	`-d` オプションと併用することで、すべてのデバッグメッセージを *tracefile* にリダイレクトさせます。*tracefile* が省略されている場合は、`/var/cluster/run/cl_pnmd.log` が使用されます。

`cl_pnmd` は、デーモンであり、`stdin`、`stdout`、または `stderr` を使用して外部へ直接接続できません。すべての診断メッセージは、Unresolved link to " syslog3C" を使用して記録されます。

`cl_pnmd` はスーパーユーザーモードで実行する必要があります。

生成されるデバッグメッセージは膨大な量になる可能性があるため、`-t` オプションを長期間使用することは避けてください。

`cl_pnmd` は、`pnm` 起動スクリプトにより開始されます。サービス管理機能がデーモンを起動および終了します。

`SIGTERM` シグナルを使用することで、`cl_pnmd` を正常に終了させることができます。同デーモンを終了させる場合、その他のシグナルは使用しないでください。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

Unresolved link to " ifconfig1M", Unresolved link to " in.mpathd1M", Unresolved link to " syslog3C", Unresolved link to " attributes5"

名前

dcs_config — DCS のクエリー

```
/usr/cluster/lib/sc/dcs_config -c info [-s service-name | -C
     service-class | -d device-path]

/usr/cluster/dtk/bin/dcs_config -c status [-s service-name]

/usr/cluster/lib/sc/dcs_config -c remove -s service-name
```

`dcs_config` コマンドは、Device Configuration System (DCS) を直接更新するように設計された緊急のコマンド行インタフェースです。`dcs_config` の更新オプションは、承認された Oracle サポート担当者に指示された場合にのみ使用するようにしてください。`cldevicegroup` コマンドを使用して、DCS に対する通常の変更をすべて実行します。

デバイスのサービスをクエリーするには、コマンドの `info` または `status` 形式を使用します。`info` 形式では、サービスに関する一般的な構成情報が提供されます。`status` 形式では、サービスの現在の状態に関する情報が提供されます。追加の修飾オプションなしの `info` および `status` コマンドは、使用中のすべてのサービスクラスとサービスを示します。

このコマンドは、大域ゾーンだけで使用できます。

次のオプションがサポートされています。

`-c` *command*

実行する *command* を指定します:

`info`

指定されたサービス名に関する情報を表示し、指定されていない場合はすべてのサービスを表示します。出力はサービスのタイプに応じて異なり、サービスクラス、セカンダリ、スイッチバック、レプリカ、生成、デバイスなどが含まれます。

`status`

指定されたサービスのサービス状態またはすべての状態 (サービス名が指定されていない場合) を表示します。

`remove`

指定されたサービス名を DCS から削除します。これは、サービス名をクラスタから削除するだけです。Oracle Solaris からは削除しません。たとえば、`dcs_config` を使用

してメタセットを削除すると、Solaris Volume Manager ではディスクセットは削除されません。

-C *service-class*

サービスクラスを指定します。有効なサービスクラスは、SUNWmd、DISK、TAPE、および SUNWlocal です。

-d *device-path*

デバイスパスを指定します。

-s *service-name*

サービス名を指定します。有効なサービス名には、メタセットとディスクが含まれます。

例 291 ディスクに関する情報の表示

この例では、ディスク dsk/d5 に関する情報を表示します:

```
# dcs_config -c info -s dsk/d5
Service name: dsk/d5
Service class: DISK
Switchback Enabled: False
Number of secondaries: All
Replicas: (Node id --> 1, Preference --> 0)(Node id --> 2, Preference --> 0)
Devices: (239, 160-167)
Properties:
   gdev --> d5
   autogenerated --> 1
```

例 292 認識されない Solaris Volume Manager metaset の削除

この例では、クラスタソフトウェアがメタセット nfs-set を認識しますが、Solaris Volume Manager では認識しません。cldevicegroup status コマンドはメタセットを表示します:

```
=== Cluster Device Groups ===

--- Device Group Status ---
Device Group Name     Primary     Secondary     Status
-----------------     -------     ---------     ------
nfs-set                -           -             Offline
```

metaset コマンドでは、セットを認識しません:

```
# metaset -s nfs-set
metaset: setname "nfs-set": no such set
```

1 つのノードから実行し、次の dcs_config コマンドがクラスタから nfs-set を削除します:

```
# dcs_config -c remove -s nfs-set
```

例 293 `metaset` のステータスの表示

この例では、`nfs-set` メタセットのステータスを表示します。

```
# dcs_config -c status -s nfs-set
Service Name: nfs-set
Active replicas: (1. State - Primary)(2. State - Secondary)
Service state: SC_STATE_ONLINE
```

次の属性の詳細は、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

79 ページの`cldevicegroup`(1CL), Unresolved link to " metaset1M"

名前

halockrun — ファイルロック保持中の子プログラムの実行

```
/usr/cluster/bin/halockrun [-nsv] [-e exitcode] lockfilename
     prog [args]
```

`halockrun` ユーティリティーは、ファイルロックを要求し、そのロックを保持したままプログラムを実行するための便利な方法を備えています。このユーティリティーではスクリプトロックがサポートされるので、Bourne シェルなどのスクリプト言語でプログラミングする際には役立ちます。Unresolved link to " sh1" を参照してください。

`halockrun` は、ファイル *lockfilename* を開き、ファイル全体に対して排他モードでファイルロックを要求します。Unresolved link to " fcntl2" を参照してください。次に、引数 *args* でプログラム *prog* を子プロセスとして実行し、子プロセスが終了するまで待機します。子が終了すると、`halockrun` はロックを解除し、子が終了したのと同じ終了コードで終了します。

子プロセスで、*prog* は危険領域で実行されます。この危険領域は形がよく整われているため、子プロセスがどのような方法で終了してもロックが解除されることです。これらの方法は全体的に効果的です。

ファイル *lockfilename* を開けない場合や作成できない場合、`halockrun` は `stderr` にエラーメッセージを出力し、終了コード `99` で終了します。

次のオプションがサポートされています。

`-e` *exitcode*

通常、`halockrun` によって検出されたエラーは終了コード `99` で終了します。`-e` オプションでは、この特別な終了コードの値を変更できます。

`-n`

ロックを非ブロックモードでリクエストします。ロックがすぐ許可されない場合、`halockrun` は *prog* を実行せずに終了コード `1` でただちに終了します。この動作は `-e` オプションの影響を受けません。

`-n` オプションを省略すると、ブロックモードでロックを要求します。つまり、`halockrun` ユーティリティーは、ロックが使用可能な状態になるまで待機します。

`-s`

排他モードではなく共有モードのファイルロックを要求します。

-v

標準エラー出力 に冗長出力します。

子プロセスが起動しないなどの `halockrun` 自体によるエラーが検出された場合、`halockrun` は終了コード `99` で終了します。`-e` オプションで終了コードの値をそれ以外の値に変更できます。「**オプション** 」を参照してください。

それ以外の場合、`halockrun` は子プロセスの終了コードで終了します。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

Unresolved link to " fcntl2", Unresolved link to " attributes5"

名前

hatimerun — タイムアウト制御下での子プログラムの実行

```
/usr/cluster/bin/hatimerun -t timeOutSecs [-adv] [-e exitcode] prog args

/usr/cluster/bin/hatimerun -t timeOutSecs [-dv] [-e exitcode] [-k signalname] prog args
```

`hatimerun` ユーティリティーでは、タイムアウト可能な別の子プログラムを実行するための便利な機能を備えています。Bourne シェルなどのスクリプト言語でのプログラミングに使用すると便利です。Unresolved link to " sh1" を参照してください。

`hatimerun` ユーティリティーでは、プログラム *prog* を引数 *args* で、タイムアウト制御化での子サブプロセスとして、独自のプロセスグループとして実行します。タイムアウトは、`-t` *timeOutSecs* オプションを使用して秒単位で指定します。タイムアウトになると、`hatimerun` は SIGKILL シグナルで子サブプロセスのプロセスグループを強制終了し、終了コード 99 で終了します。

次のオプションがサポートされています。

`-a`
: `hatimerun` の機能を大幅に変更します。`hatimerun` ユーティリティーは、タイムアウトになっても子プロセスを強制終了しません。子を非同期で実行したまま、終了コード 99 で終了します。

  `-a` オプションと `-k` オプションを同時に指定できません。

`-d`
: タイムアウトの遅延を有効にします。このオプションは、プログラム `_prog_` の実行が開始されるまでタイムアウトクロックの開始を遅延させます。負荷の高いシステムでは、子プロセスがフォークされる時点から指定されたプログラムの実行が開始される時点まで、数秒間の遅延が発生する可能性があります。`-d` オプションを使用すれば、割り当てられたタイムアウト期間に対して追加の実行前時間がカウントされなくなります。

`-e`
: タイムアウトの終了コードを 99 以外の値に変更します。

`-k`
: 子プロセスグループを強制終了するシグナルを指定します。Unresolved link to " kill1" コマンドで認識されるものと同じシグナル名を指定できます。特に、シグナル名は、`<signal.h>` の説明に定義されているシンボリック名のいずれかにするようにしてください。シグナル名ではアルファベットの大文字と小文字が区別されません。また、接頭辞 `SIG` は省略します。シグナル番号を使用するときは、`-k` オプションに数値引数を指定します。

  `-a` オプションと `-k` オプションを同時に指定できません。

-t　　タイムアウト期間を秒単位で指定します。

-v　　**標準エラー出力**　に冗長出力します。

タイムアウトになったとき、hatimerun は終了コード 99 で終了します。この値は、-e オプションで値を指定して変更できます。

タイムアウトにはなっていないがその他のエラー (子プログラムではなく hatimerun ユーティリティーによって) が検出された場合、hatimerun は終了コード 98 で終了します。

それ以外の場合、hatimerun は子プログラムと同じ終了ステータスで終了します。

hatimerun ユーティリティーは、シグナル SIGTERM をキャッチします。hatimerun は、このシグナルへの応答として、タイムアウトのときと同様に子プログラムを強制終了し、終了コード 98 で終了します。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

Unresolved link to " kill1", Unresolved link to " sh1", Unresolved link to " attributes5"

名前

pmfadm — プロセスモニター機能の管理

```
/usr/cluster/bin/pmfadm -c nametag [-a action] [[-e ENV_VAR=
     env.var…] | -E] [-n retries] [-t period] [-C level#]
     command [args-to-command…]

/usr/cluster/bin/pmfadm -k nametag [-w timeout] [signal]

/usr/cluster/bin/pmfadm -L [-h host]

/usr/cluster/bin/pmfadm -l nametag [-h host]

/usr/cluster/bin/pmfadm -m nametag [-n retries] [-t period]

/usr/cluster/bin/pmfadm -q nametag [-h host]

/usr/cluster/bin/pmfadm -s nametag [-w timeout] [signal]
```

`pmfadm` は、プロセスモニター機能に管理用のコマンド行インタフェースを提供するためのユーティリティーです。

プロセスモニター機能は、プロセスおよび派生プロセスのモニタリングを行い、途中で終了したプロセスに対しては再起動することができます。また発生する障害について、許容される総数および期間を指定することができます。指定期間内に発生した障害の総数が許容値を超過した場合、コンソールにメッセージが記録され、プロセスの再起動は行われなくなります。

action プログラムを指定しておくと、障害の発生数が許容値を超過した場合に、指定されたプログラムが呼び出されます。なお、*action* プログラムがゼロ以外のステータスで終了した場合は、プロセスの nametag がプロセスモニター機能から削除されます。それ以外の場合はプロセスが再起動され、オリジナルのパラメータが `pmfadm` に渡されます。

プロセスモニターの制御のもとで起動されるプロセスは、リクエストを開始したユーザーの有効なユーザー ID (`euid`) と有効なグループ ID (`egid`) のもとで実行されます。これらのプロセスに関連付けられた nametag を操作できるのは、実行ユーザーまたは root のユーザーのみです。ただしステータス情報は、ローカルかリモートかを問わず、すべての呼び出し元から使用可能です。

最初に実行されたプロセスと、そこから生成されたプロセスおよび、その派生プロセスがすべてモニターされます。最後のプロセス/サブプロセスが終了した場合のみ、プロセスモニターはそのプロセスの再起動を試みます。

次のオプションがサポートされています。

-a *action*

プロセスが中断した場合に呼び出されるアクションプログラムです。該当するプログラムについては -a オプションに単独の引数として指定する必要がありますが、複数のコンポーネントを引用符で囲んだ文字列として指定することも可能です。いずれの場合も、文字列は指定通りに実行され、2 つの追加引数、発生したイベント (現状では failed のみ)、プロセスに関連付けられた nametag が付加されます。カレントディレクトリと PATH 環境変数は、コマンドが実行される前に再インスタンス化されます。その他の環境変数は、基本的に保持されるものとして処理されます。

アクションプログラムが 0 ステータスで終了した場合、pmfadm に渡されたオリジナルの引数を使用して、プロセスが再起動されます。その他のステータスで終了した場合は、プロセスモニターのモニター範囲内で nametag が終了します。

-a オプションが指定されていない場合は、0 ステータス以外で常に終了となるアクションスクリプトが指定されている場合と同様の結果になります。

-C *level#*

プロセスを開始する際に、該当プロセスおよびその子孫を最大 *level#* レベルまでモニターします。なお *level#* に指定する値は、0 以上の整数にする必要があります。各プロセスの実行レベルについては、オリジナルのプロセスはレベル 0 で実行され、その子供はレベル 1 で実行され、孫に当たるプロセスはレベル 2 で実行されるというようになります。新しいフォーク操作により、新しい子のレベルが生成されます。

このオプションを使用することで、モニターするプロセスに対して、より細かなコントロールを施すことができます。これは、新規プロセスをフォークするサーバーをモニターする際に有用です。

このオプションを使用しない場合、すべての子プロセスがモニターの対象とされ、オリジナルおよびそのすべての子孫が実行を停止するまで、オリジナルのプロセスは再起動されないことになります。

サーバーがクライアントの要求を扱う際に新規プロセスをフォークさせる場合などは、このサーバーのみに限定してモニターできた方が都合がよい場合があります。これは、こうしたサーバーが活動を停止したような場合、そのクライアントプロセスが実行中であっても、サーバー本体を再起動する必要があるからです。適切なモニタリングレベルは -C 0 です。

子プロセスをフォークした場合、親プロセスは終了して、モニタリングすべき対象は子の側に移ります。子プロセスのモニターに用いるレベルは -C 1 です。双方のプロセスが終了した場合は、サーバーを再起動します。

-c *nametag*

nametag を識別子としてプロセスを開始させます。コマンド行フラグに続くすべての引数は、該当するプロセスとして処理されます。カレントディレクトリと PATH 環境変数は、コマンドが実行される前にプロセスモニター機能によって再インスタンス化されます。その他の環境変数は、基本的に保持されるものとして処理されます。

nametag がすでに存在する場合、pmfadm は終了ステータスを 1 として終了し、副次的な作用を及ぼすことはありません。

入出力先の変更は、コマンド行引数ではサポートされていません。こうした操作が必要な場合は、入出力先を変更するためのスクリプトを作成し、pmfadm の実行するコマンドに指定してください。

-E

pmfadm 環境全体を新規プロセスへ渡します。デフォルトでは、このオプションは使用されず、rpc.pmfd 環境と pmfadm 環境のパスが渡されます。

-e と -E オプションは排他的な関係にあるため、同じコマンド内で両者を併用できません。

-e ENV_VAR= *env.value*

ENV_VAR=*env.value* の形式で使用される環境変数の 1 つで、新規プロセスの実行環境に渡されます。このオプション指定は繰り返しが可能で、複数の環境変数を渡すことができます。デフォルトでは、このオプションは使用されず、rpc.pmfd 環境と pmfadm 環境のパスが渡されます。

-h *host*

アクセスするホスト名を指定します。デフォルトは localhost です。

-k *nametag*

指定されたシグナルを *nametag* に関連付けられたプロセスに送信するためのものです。アクションプログラムが現在実行中であれば、それに関連付けられたすべてのプロセスが該当します。特に指定されない場合は、デフォルトシグナルの SIGKILL が送信されます。プロセスおよびその子孫が終了した状況で再試行が可能であれば、プロセスモニターはプロセスを再起動させます。指定されるシグナルは、kill コマンドが認識する名前のセットと同じです。

-L

該当コマンドを実行させたユーザーに属するすべての実行中のタグを一覧表示させるためのもので、root ユーザの場合は、サーバー上で実行中のすべてのタグが表示されます。

-l *nametag*

nametag に関するステータス情報を出力させます。このコマンドによる出力はシステムの診断をする際に有用ですが、内容は変更される可能性もあります。

-m *nametag*

nametag での再試行回数または再試行の監視時間を変更させるためのものです。なお、これらのパラメータを変更した場合、過去の障害発生に関する履歴はクリアされます。

-n *retries*

指定期間内の再試行回数を指定します。このフィールドのデフォルト値は 0 であり、この場合いったん終了したプロセスは再起動されません。許容される最大値は 100 です。値 -1 は、再試行回数が無限大であることを示します。

-q *nametag*

nametag がプロセスモニターに登録されて実行されているかについて、その確認を行うためのものです。登録されていれば `0` が返され、そうでなければ `1` が返されます。その他の戻り値は、エラーを意味します。

-s *nametag*

nametag に関連付けられたコマンドの再起動を停止させます。シグナルが指定された場合、すべてのプロセスに送信されますが、これには現在実行中のアクションスクリプトおよびそのプロセスも該当します。シグナルが指定されない場合は、何も送信されません。ただし、プロセスのモニタリングを停止しても、そのままプロセス自体が消滅するわけではありません。プロセス本体およびそのすべての派生プロセスが終了するまで、該当するプロセスは実行され続けます。指定されるシグナルは、`kill` コマンドが認識する名前のセットと同じです。

-t *period*

障害発生をカウントし続ける時間を分単位で指定します。このフラグのデフォルト値は `-1` で、この場合はカウント時間は無限大になります。このパラメータを指定すると、指定期間外でのプロセスへの障害発生はカウントされなくなります。

-w *timeout*

-s *nametag* または -k *nametag* フラグと併用することで、*nametag* に関連付けられたプロセスが終了するまで、指定された時間 (秒) 待機させます。待機期限切れになった場合は、`pmfadm` が終了し、その終了ステータスは `2` となります。このフラグのデフォルト値は `0` であるため、いずれのプロセスの終了も待つことなく、コマンドはただちに戻り値を返します。

値として `-1` を指定すると、タグに関連付けられたプロセスが終了するまで、`pmfadm` は無限に待機し続けます。RPC タイムアウト期間に到達するまで、`pmfadm` プロセスは使用している RPC サーバースレッドを解放しません。したがって、特に必要でないかぎり、-w *timeout* の値には -1 を設定しないでください。

例 294 再起動することのない休眠プロセスの起動

次の例は、`sleep.once` という名前の休眠プロセスを起動させるもので、これはいったん終了したあとは再起動することがありません。

```
example% pmfadm -c sleep.once /bin/sleep 5
```

例 295 休眠プロセスの起動とその再起動

次の例は休眠プロセスを起動させたあと、最大 1 回再起動させます。

```
example% pmfadm -c sleep.twice –n 1 /bin/sleep 5
```

例 296 休眠プロセスの起動とその再起動

次の例は休眠プロセスを起動させたあと、最大で毎分 2 回再起動させます。この実行の中断数が許容回数を超過すると、/bin/true が呼び出されます。

```
example% pmfadm -c sleep.forever –n 2 –t 1 –a /bin/true /bin/sleep 60
```

例 297 sleep.forever nametag の現在のステータスの一覧表示

次のコマンドは、sleep.forever nametag の現在のステータスを一覧表示させます。

```
example% pmfadm -l sleep.forever
```

例 298 すべてのプロセスへの SIGHUP の送信

次のコマンドは、sleep.forever に関連付けられたすべてのプロセスに SIGHUP を送信させるもので、すべてのプロセスが終了するまで最大で 5 秒間待機させます。

```
example% pmfadm -w 5 -k sleep.forever HUP
```

例 299 プロセスのモニタリングの停止および SIGHUP の送信

次のコマンドは、sleep.forever プロセスのモニタリング (および再起動) を停止させるもので、これに関連したすべてのプロセスに SIGHUP を送信させます。このコマンドはシグナルの送信後ただちに戻り値を返しますが、その時点ではすべてのプロセスが終了しきっていない可能性もあります。

```
example% pmfadm -s sleep.forever HUP
```

例 300 該当ユーザーに属するすべての実行タグを一覧表示

ここでは、ユーザーが次の一連のコマンドを実行したものとします。

```
example% pmfadm -c sleep.once /bin/sleep 30
example% pmfadm -c sleep.twice /bin/sleep 60
example% pmfadm -c sleep.forever /bin/sleep 90
```

これらの次に、次のコマンドを実行するものとします。

```
example% pmfadm -L
```

is

```
sleep.once sleep.twice sleep.forever
```

次の終了値が返されます。

`0`	正常終了。
`1`	*nametag* が存在しなかったか、既存の nametag を作成しようとした場合。
`2`	コマンドがタイムアウトした場合。
ゼロ以外	エラーが発生しました。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

Unresolved link to " kill1", 743 ページの`rpc.pmfd`(1M), Unresolved link to " attributes5"

名前

rpc.pmfd, pmfd — RPC ベースのプロセスモニターサーバー

```
/usr/cluster/lib/sc/rpc.pmfd
```

`rpc.pmfd` デーモンは、Oracle Solaris Cluster ソフトウェアで使用されるプロセスモニター機能を提供するための Oracle の ONC RPC サーバーです。このデーモンが最初に実行開始されるのは、システムの起動時です。

モニターするコマンドを、各コマンドを起動したユーザーで実行するため、`rpc.pmfd` デーモンはスーパーユーザーで起動する必要があります。

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

Unresolved link to " truss1"Unresolved link to " attributes5"

診断メッセージは通常、コンソールにログが記録されます。

ほかの制御プロセスとの衝突を防ぐため、`truss` コマンドは、ほかのプロセスが `/proc` インタフェースを介して制御しているプロセスの追跡を許可しません。

名前

rpc.pmfd, pmfd — RPC ベースのプロセスモニターサーバー

```
/usr/cluster/lib/sc/rpc.pmfd
```

`rpc.pmfd` デーモンは、Oracle Solaris Cluster ソフトウェアで使用されるプロセスモニター機能を提供するための Oracle の ONC RPC サーバーです。このデーモンが最初に実行開始されるのは、システムの起動時です。

モニターするコマンドを、各コマンドを起動したユーザーで実行するため、`rpc.pmfd` デーモンはスーパーユーザーで起動する必要があります。

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

Unresolved link to " truss1"Unresolved link to " attributes5"

診断メッセージは通常、コンソールにログが記録されます。

ほかの制御プロセスとの衝突を防ぐため、`truss` コマンドは、ほかのプロセスが `/proc` インタフェースを介して制御しているプロセスの追跡を許可しません。

名前

sc_zonesd — Oracle Solaris Cluster ゾーン管理デーモン

```
/usr/cluster/lib/sc/sc_zonesd
```

`sc_zonesd` デーモンは、Oracle Solaris Cluster ソフトウェアによって使用されるシステムデーモンです。このデーモンが最初に実行開始されるのは、システムの起動時です。

このデーモンは大域ゾーンだけで動作します。

すべての診断メッセージは、`syslog` 関数を使用して記録されます。

`sc_zonesd` デーモンは、スーパーユーザーモードで起動する必要があります。

`sc_zonesd` デーモンは、SMF サービス `sc_zones` により制御されます。このデーモンが終了している場合、あるいは、SMF サービスが無効である場合、クラスタノードはパニック状態になります。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	プライベート

Unresolved link to " syslog3C", Unresolved link to " attributes5"

名前

scconf — Oracle Solaris Cluster ソフトウェア構成の更新

```
scconf -a [-Hv] [-h node_options] [-A adapter_options] [-B
     switch_options] [-m cable_options] [-P privatehostname_options]
     [-q quorum_options] [-D devicegroup_options] [-T
     authentication_options]

scconf -c [-Hv] [-C cluster_options] [-A adapter_options] [-B
     switch_options] [-m cable_options] [-P privatehostname_options]
     [-q quorum_options] [-D devicegroup_options] [-S slm_options]
     [-T authentication_options] [-w heartbeat_options]

scconf -r [-Hv] [-h node_options] [-A adapter_options] [-B
     switch_options] [-m cable_options] [-P privatehostname_options]
     [-q quorum_options] [-D devicegroup_options] [-T
     authentication_options]

scconf -p [-Hv [v]]

scconf [-H]
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

`scconf` コマンドは、Oracle Solaris Cluster ソフトウェアの構成を管理します。`scconf` を使用して、構成への項目の追加、構成済み項目のプロパティーの変更、構成からの項目の削除などを行うことができます。これらの 3 つの形式の各コマンド内で、コマンド行に入力された順にオプションが処理されます。各オプションの処理が開始されるのは、その直前のオプションによる更新処理がすべて完了してからになります。

また、`scconf` コマンドは Solaris Volume Manager メタセットおよび raw ディスクグループが、データ可用性のためにコントローラベースのレプリケーションを使用するディスクで構成されている場合に、これらのディスクグループおよびメタセットを登録するためにも使用できます。`scconf` コマンドを使用してディスクグループとメタセットを登録する前に、ディスクグループのすべてのディスクがレプリケートされているか、レプリケートされていないかのどちらかであり、その両方ではないことを確認してください。また、`scdidadm` コマンドに `-T` または `-t` オプションを付けて実行するか、`cldevice replicate` コマンドを実行する必要があります。これらのコマン

ドは、コントローラベースのレプリケーションを使用するように DID デバイスを構成します。詳細は、793 ページのscdidadm(1M) のマニュアルページまたは 61 ページのcldevice(1CL) のマニュアルページを参照してください。

なお scconf コマンドを実行できるのは、有効なクラスタノードからだけです。クラスタ内のアクティブなノードであれば、どのノードからコマンドを実行しても違いはありません。コマンドを実行して得られる結果は、実行するノードに関係なく、常に同じです。

scconf に -p オプションを指定すると、現在の構成が一覧表示されます。

-H オプションは、scconf コマンドのすべての形式で使用できます。-H を指定すると、ヘルプ情報が表示されます。その他のオプションは無視されるため、実行されません。ヘルプ情報は、オプションなしで scconf を呼び出した場合も出力されます。

このコマンドは、大域ゾーンだけで使用できます。

基本オプション

次のオプションは、すべての形式の scconf コマンドに共通です。

-H

このオプションをコマンド行のいずれかの場所に指定すると、ヘルプ情報が出力されます。他のすべてのオプションは無視され、実行されません。ヘルプ情報は、オプションなしで scconf を呼び出した場合も出力されます。

このオプションは大域ゾーンだけで使用できます。

次のオプションは、scconf コマンドの基本的な形式および機能を変更します。これらのオプションは、1 つのコマンド行で複数を同時使用できません。

-a

add 形式の scconf コマンドを指定します。

このオプションは大域ゾーンだけで使用できます。

-a オプションは、Oracle Solaris Cluster のソフトウェア構成の定義に使用されるほとんどの項目の追加や初期化に使用できます。追加する要素 (たとえば、アダプタ、スイッチ、デバイスグループなどのオプション) やその関連プロパティーを指定する場合は、-a と一緒に追加オプションを使用します。1 つのコマンド行で同時に使用できる追加オプションの数は、-a オプションと併用可能であれば、特に制限はありません。

-c

change 形式の scconf コマンドを指定します。

このオプションは大域ゾーンだけで使用できます。

-c オプションは、Oracle Solaris Cluster ソフトウェア構成の一部としてすでに構成されている項目のプロパティーを変更するときに使用します。新規または変更済みのプロパティーを指定する場合は、-c と一緒に追加オプションを使用します。1 つのコマンド行で同時に使用できる追加オプションの数は、-c オプションと併用可能であれば、特に制限はありません。

-p

print 形式の scconf コマンドを指定します。

このオプションは大域ゾーンだけで使用できます。

-p オプションを指定すると、scconf を使って構成できる Oracle Solaris Cluster の現在の構成要素とその関連プロパティーが一覧表示されます。このオプションに -v オプションを併用して、冗長形式で出力させることも可能です。

-r

remove 形式の scconf コマンドを指定します。

このオプションは大域ゾーンだけで使用できます。

-r オプションは、Oracle Solaris Cluster ソフトウェア構成から項目を削除するときに使用します。-r では、構成から削除する項目を指定する追加オプションを使用できます。1 つのコマンド行で同時に使用できる追加オプションの数は、-r オプションと併用可能であれば、特に制限はありません。

追加オプション

次の追加オプションは、上で述べた基本オプションの 1 つまたは複数と併用できます。scconf の各形式で使用できるオプションについては、SYNOPSIS セクションを参照してください。

使用可能なオプションを次に示します。

-A *adapter_options*

クラスタトランスポートアダプタに、プロパティーの追加、削除、変更を行います。処理にあたって、指定されたアダプタをホストするノードは、クラスタ内でアクティブとなっている必要はありません。-A が使用可能なコマンドの 3 つの形式で、-A *adapter_options* はそれぞれ次のようになります。

- add 形式のコマンドの場合、-A *adapter_options* を指定するには次の構文を使用します:

```
–A name=adaptername,node=
node[,vlanid=vlanid][,state=
state] \
[,other_options]
```

- `change` 形式のコマンドの場合、`-A` *adapter_options* を指定するには次の構文を使用します:

  ```
  –A name=adaptername,node=
  node[,state=state] \
  [,other_options]
  ```

- `remove` 形式のコマンドの場合、`-A` *adapter_options* を指定するには次の構文を使用します:

  ```
  –A name=name,node=node
  ```

`-A` オプションは、次のサブオプションをサポートしています。

`name=`*adaptername*

特定のノード上にあるアダプタ名を指定します。`-A` オプションが指定されるごとに、このサブオプションが含まれている必要があります。

adaptername は 1 つの *device name* から構成され、その直後に 1 つの *physical-unit* 番号が続きます (`hme0` など)。

`node=`*node*

特定のノード上にあるアダプタ名を指定します。`-A` オプションが指定されるごとに、`node` サブオプションが必要です。

node には、ノード名またはノード ID を指定できます。

`state=`*state*

アダプタの状態を変更します。このサブオプションは、`change` 形式のコマンドで使用できます。`state` には、`enabled` または `disabled` を指定できます。

構成にアダプタを追加した場合、その状態は常に `disabled` に設定されます。アダプタ上のいずれかのポートにケーブルを追加すると、デフォルトでは、ポートおよびアダプタの状態が `enabled` に変わります。`-m` *cable_options* を参照してください。

あるアダプタを使用不可にすると、そのアダプタに関連付けられたすべてのポートも使用不可になります。これに対して、アダプタを使用可能にしても、関連ポートは使用可能にはなりません。アダプタポートを有効にするためには、ポートが接続されているケーブルを有効にする必要があります。

`trtype=`*type*

トランスポートタイプを指定します。`add` 形式のコマンドで `-A` が使用されている場合は、このサブオプションが含まれている必要があります。

トランスポート *type* の例は `dlpi` です。1483 ページの`sctransp_dlpi(7p)` を参照してください。

[vlanid=*vlanid*]

タグ付けされた VLAN アダプタの VLAN ID を指定します。

[*other_options*]

特定のアダプタの型に使用可能なほかのオプションには、add および change 形式のコマンドで -A オプションと併用するものがあります。特殊なオプションについては、クラスタトランスポートアダプタのマニュアルページを参照してください。

このコマンドオプションを、-a、-c、または -r と一緒に使用するには、solaris.cluster.transport.modify の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-B *switch_options*

クラスタトランスポートスイッチ (トランスポート接続点とも呼ぶ) のプロパティーを追加、削除、または変更します。

このようなデバイスの例としては、Ethernet ハブ、ほかの各種タイプのスイッチ、リングなどが該当しますが、これらだけに限られるわけではありません。

-B が使用可能なコマンドの 3 つの形式で、-B *switch_options* はそれぞれ次のようになります。

- add 形式のコマンドの場合、-B *switch_options* を指定するには次の構文を使用します:

  –B type=*type*,name=*name*

  [,*other_options*]

- change 形式のコマンドの場合、-B *switch_options* を指定するには次の構文を使用します:

  –B name=*name*[,state=

  state][,*other_options*]

- remove 形式のコマンドの場合、-B *switch_options* を指定するには次の構文を使用します:

  –B name=*name*

-B オプションは、次のサブオプションをサポートしています。

name=*name*

クラスタトランスポートスイッチの名前を指定します。-B オプションが指定されるごとに、name サブオプションが含まれている必要があります。

name は最大長 256 文字の文字列が指定できます。名前には数字や文字が使えますが、最初のキャラクタは文字でなければなりません。個々のトランスポートスイッチの名前は、クラスタの名前空間全体で一意でなければなりません。

`state=`*state*

クラスタトランスポートスイッチの状態を変更します。このサブオプションは、`-B change` コマンドで使用できます。`state` には、`enabled` または `disabled` を指定できます。

構成にスイッチを追加した場合、その状態は `disabled` に設定されます。スイッチ上のいずれかのポートにケーブルを追加すると、デフォルトでは、ポートおよびスイッチの状態が `enabled` に変わります。`-m` *cable_options* を参照してください。

あるスイッチを使用不可にすると、そのスイッチに関連付けられたすべてのポートも使用不可になります。これに対して、スイッチを使用可能にしても、関連ポートは使用可能にはなりません。スイッチポートを有効にするためには、ポートが接続されているケーブルを有効にする必要があります。

`type=`*type*

クラスタトランスポートスイッチの型を指定します。`add` 形式のコマンドで `-B` が使用されている場合は、このサブオプションが含まれている必要があります。

Ethernet ハブとは、`switch` タイプのクラスタトランスポートスイッチの例です。詳細は、789 ページの`scconf_transp_jct_etherswitch`(1M) のマニュアルページを参照してください。

[*other_options*]

特定のスイッチの型にほかのオプションを使用できる場合、`add` および `change` 形式のコマンドで `-B` とともにそれらを使用できます。特殊なオプションについては、789 ページの`scconf_transp_jct_etherswitch`(1M) クラスタトランスポートスイッチのマニュアルページを参照してください。

このコマンドオプションを、`-a`、`-c`、または `-r` と一緒に使用するには、`solaris.cluster.transport.modify` の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

`-C` *cluster_options*

クラスタ名を変更します。このオプションが使用できるのは `change` 形式のコマンドだけです。

cluster_options は、`change` 形式のコマンドに対して、次のように指定します:

`-C cluster=`*clustername*

この形式のコマンドでは、クラスタの名前が *clustername* に変更されます。

`-D` *devicegroup_options*

ディスクデバイスグループを構成に追加するか、既存のデバイスグループのプロパティーを変更またはリセットするか、Oracle Solaris デバイスグループの構成からグループを削除します。その他のデバイスグループオプション (*other_options*) は、デバイスグループおよびそのオプションの追加と変更で、重要な役割を担っています。デバイスグループを構成する際は、マニュアルページのタイプ依存型デバイスグループオプションに特に注意してください (例: 773 ページの`scconf_dg_svm`(1M) および 769 ページの`scconf_dg_rawdisk`(1M))。デバイスグループの種類によっては、`-D` オプ

ションの 3 つの形式すべてをサポートしていないものもあります。svm デバイスグループで通常使用できるものは change 形式のコマンドのみで、ノード優先順位リストの順序など、特定の属性を変更することしかできません。

add 形式のコマンドを使用できるのは、新規にデバイスグループを作成する場合か、既存のデバイスグループにノードを追加する場合です。デバイスグループのタイプによっては、add 形式を使ってデバイスをグループに追加できる場合があります。コマンドの change 形式は、グループに関連するいくつかの属性を変更するために、更新を登録します。remove 形式のコマンドを使用できるのは、デバイスグループ全体を削除する場合か、グループのコンポーネントをいくつか削除する場合です。

-D が使用可能な scconf コマンドの 3 つの形式で、-D *devicegroup_options* はそれぞれ次のようになります:

追加:

```
-D type=type,name=
name[,nodelist=node[:
node]…]
        [,preferenced={true | false}]
        [,numsecondaries=integer]
        [,failback={enabled | disabled}][,other_options]
```

変更:

```
-D name=name[,nodelist=
node[:node]…]
        [,preferenced={true | false}]
        [,numsecondaries=integer]
        [,failback={enabled | disabled}][,other_options]
```

削除:

```
-D name=name,nodelist=
node[:node]…
```

-D オプションは、次のサブオプションをサポートしています。

name=*name*

デバイスグループの名前。この名前は、コマンドの 3 つの形式すべてで指定してください。

nodelist=*node*[: *node*]…

潜在的なプライマリノードのリストです。このリストは、グループをクラスタに追加するときに、デバイスグループタイプによっては指定が必要です。詳細は、タイプ依存型デバイスグループのマニュアルページを参照してください。

preferenced サブオプションを true に設定する場合は、nodelist サブオプションが必要です。

add 形式のコマンドの場合、デフォルトでは nodelist は、デバイスグループでプライマリノードとなるべきノードの優先順序を指定する順序リストです。ただし、preferenced

サブオプションが false (次のサブセクションを参照) に設定されている場合は、グループ内のデバイスにアクセスする最初のノードが自動的にそのグループのプライマリノードになります。preferenced サブオプションは、既存のデバイスグループにノードを追加する場合は使用できません。ただし、そのグループをはじめて作成する場合や、change 形式のコマンドで使用する場合は、preferenced サブオプションを使用できます。

プライマリノードの優先順位を変更する場合は、nodelist に、必要な順序でクラスタノードの完全なリストを指定する必要があります。また、preferenced サブオプションも true に設定する必要があります。

remove 形式のコマンドで使用する場合、nodelist サブオプションは、指定したノードをデバイスグループから削除するために使用します。デバイスグループ全体が削除されるのは、nodelist を指定しない場合だけです。デバイスグループからすべてのノードを削除しただけでは、必ずしもグループそのものを削除したことにはなりません。

type=*type*

デバイスグループのタイプ。このタイプは、作成するデバイスグループのタイプ (rawdisk など) を示すために、add 形式のコマンドで使用する必要があります。

[failback={enabled | disabled}]

add または change 形式のコマンドを使用して、デバイスグループの failback 動作を有効または無効にします。

システムの動作を指定する理由は、デバイスグループのプライマリノードがクラスタメンバーシップから切り離され、あとでふたたび戻ることがあるためです。

こうしたノードがクラスタメンバーシップから切り離された段階で、デバイスグループはセカンダリノードに処理を継続します。そして障害の発生したノードがクラスタメンバーシップに再結合すると、デバイスグループは、そのままセカンダリノードにマスターされ続けるか、あるいは、オリジナルのプライマリノードにフェイルバックするか、のいずれかの挙動を取ります。

failback が enabled に設定されている場合、デバイスグループは、オリジナルのプライマリノードにマスターされるようになります。failback が disabled に設定されている場合、デバイスグループは、セカンダリノードによって引き続きマスターされます。

デフォルトでは、failback が disabled に設定されています。

[numsecondaries=*integer*]

デバイスグループのセカンダリノードの数を必要に応じて動的に変更できます。デバイスグループは HA サービスの 1 つであるため、1 つのプライマリノードと 1 つ以上のセカンダリノードが必要です。デバイスグループ内のこうしたセカンダリノードは、現在のプライマリノードに問題が生じた場合に、新たなプライマリノードとして機能します。

ここには 1 以上の整数を指定しますが、指定したグループ内に存在するノードの総数以下にする必要があります。デフォルトは 1 です。

numsecondaries サブオプションを使用することにより、システム管理者は、特定のレベルの可用性を維持しながらデバイスグループのセカンダリノードの数を変更できま

す。デバイスグループのあるノードがセカンダリノードのリストから削除されると、そのノードはセカンダリノードに戻るまで、プライマリノードの役割を引き継ぐことはできません。セカンダリノードの数を変更するに当たっては、セカンダリグローバルファイルシステムに与える影響を事前に考察しておく必要があります。

numsecondaries サブオプションは、現在クラスタモードにあるデバイスグループ内のノードにのみ適用され、そのノードの preferenced サブオプションとともに使用できます。デバイスの preferenced サブオプションが有効になっている場合は、優先度のもっとも低いノードが最初にセカンダリノードリストから削除されます。優先フラグがついているノードがデバイスグループの中に 1 つもないと、クラスタはノードをランダムに選択し、削除します。

いくつかのノードに障害が発生した結果として、デバイスグループ内のセカンダリノードの実働数が指定値よりも少なくなった場合、過去にセカンダリノードリストから削除されたノードがあれば、これらはセカンダリノードリストに再登録されますが、その際には、これらのノードがクラスタ内に存在すること、該当するデバイスグループに所属していること、プライマリノードとしてもセカンダリノードとしても現在使用されていないことが再登録の条件となります。こうしたリストへの再登録は、デバイスグループ内でもっとも優先度の高いノードから始められ、セカンダリノードの指定数が満たされた段階で終了します。

デバイスグループ内のあるノードが既存のセカンダリノードよりも高い優先度をもち、クラスタに結合される場合、優先度がもっとも低いノードがセカンダリノードリストから削除され、新規に追加されたノードと交換されます。この交換は、実際のセカンダリノードが必要なレベルよりも多く存在するときに限って起こります。

必要なセカンダリノード数をシステムのデフォルトに設定する場合は (デフォルト値を知らずに)、次のどちらかのコマンドを使用します。

```
# scconf -aD type=svm,name=foo, \
nodelist=node1:node2,numsecondaries=
```

または

```
# scconf -cD name=foo,numsecondaries=
```

numsecondaries サブオプションを -a オプションとともに使用できるのは、デバイスグループが作成される場合だけです。既存のデバイスグループにホストを追加する場合は、numsecondaries サブオプションを -a オプションとともに使用できません。

[preferenced={true | false}]

デバイスグループに対する潜在的なプライマリノードの優先順位のステータスを指定します。preferenced サブオプションが false に設定されていないかぎり、新しく作成されたデバイスグループのノードリストは、各ノードがデバイスグループのプライマリノードとして引き継ぎを試みる優先順位を示します。

preferenced サブオプションを true に設定した場合は、ノードリスト全体を指定するために nodelist サブオプションも使用する必要があります。

デバイスグループを作成するために使用される add で preferenced サブオプションが指定されていない場合、このサブオプションはデフォルトで false になります。ただ

し、change で preferenced サブオプションが指定されていない場合、nodelist が指定されていると、このサブオプションはデフォルトで true に設定されます。

preferenced サブオプションは、確立されたデバイスグループにノードを追加するために使用する add では使用できません。このような場合は、過去に指定したノードプレファレンスリストが使用されます。

[*other_options*]

add または change 形式のコマンドでは、デバイスグループのほかのタイプ依存型オプションを使用できます。詳細は、該当するマニュアルページを参照してください (例: 773 ページのscconf_dg_svm(1M) および 769 ページのscconf_dg_rawdisk(1M))。

このコマンドオプションを、-a、-c、または -r と一緒に使用するには、solaris.cluster.device.modify の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-h *node_options*

クラスタ構成データベースに対して、ノードの追加または削除を行います。add 形式の scconf コマンドで使用した場合、クラスタ構成データベースに新規の名前および内部生成されたノード ID が追加されます。さらに新規ノードに対しては、ディスク予約キーが与えられ、定足数の投票数は 0 になります。クラスタインターコネクトでのノードのアクセス用に割り当てられる名前は、clusternode*nodeid*-priv に初期化されます。構成要素とそのプロパティーの出力については、-p オプションを参照してください。

scconf 単独では、クラスタに新規ノードを追加できません。scconf を使用できるのは、構成データベース自体を更新する場合だけです。scconf は、構成データベースを新しいノードにコピーしたり、必要なノード識別子を新しいノードに作成したりするわけではありません。ノードをクラスタに追加するには、819 ページのscinstall(1M) を使用します。

remove 形式の scconf で使用する場合、ノードへのすべての参照 (最後のトランスポートケーブル、すべてのリソースグループ参照、およびすべてのデバイスグループ参照を含む) を、scconf を使用してクラスタ構成からノードを完全に削除する前に削除する必要があります。

定足数デバイスに対して構成されているノードは、削除できません。また 3 ノードクラスタに関しては、少なくとも 1 つの共有定足数デバイスが構成されていないかぎり、ノードを削除できません。

クラスタノードの削除に関する詳しい説明は、Oracle Solaris Cluster ドキュメントのシステム管理手順を参照してください。

-h オプションが指定されるごとに、node=*node* サブオプションを指定する必要があります。add 形式のコマンドを使用する場合、*node* にノード名を指定する必要があります。

add 形式のコマンドの場合、-h *node_options* を指定するには次の構文を使用します。

–h node=*nodename*

remove 形式のコマンドを使用する場合、*node* には、ノード名またはノード ID を指定できます。remove 形式のコマンドの場合、-h *node_options* を指定するには次の構文を使用します:

–h node=*node*

このコマンドオプションを、-a、-c、または -r と一緒に使用するには、solaris.cluster.node.modify の RBAC 承認が必要です。Unresolved link to "rbac5" を参照してください。

-m *cable_options*

クラスタインターコネクトトポロジの設定を支援します。このオプションは、クラスタトランスポートのアダプタやスイッチに存在するさまざまなポートを接続するケーブルを構成するときに有益です。個々の新規ケーブルは、通常 2 つのクラスタトランスポートアダプタ間または 1 つのトランスポートスイッチ上のポートとアダプタ間で、接続の対応付けを行います。-m が使用可能な各形式のコマンドで、-m *cable_options* は、それぞれ次のようになります:

- add 形式のコマンドの場合、-m *cable_options* を指定するには次の構文を使用します:

  –m endpoint=[*node*:]
  name[@*port*],
      endpoint=[*node*:]*name*[@
  port][,noenable]

- change 形式のコマンドの場合、-m *cable_options* を指定するには次の構文を使用します:

  –m endpoint=[*node*:]
  name[@*port*],state=
  state

- remove 形式のコマンドの場合、-m *cable_options* を指定するには次の構文を使用します:

  –m endpoint=[*node*:]
  name[@*port*]

-m オプションは、次のサブオプションをサポートしています。

endpoint=[*node*:]*name*[@*port*]

-m オプションを指定するたびに、これを指定する必要があります。add 形式のコマンドでは、2 つの endpoint オプションを指定する必要があります。オプション引数の *name* コンポーネントは、ケーブルの一方の終端の、クラスタトランスポートスイッチまたはクラスタトランスポートアダプタの名前を指定します。*node* コンポーネントを指定する場合、*name* はクラスタトランスポートアダプタの名前です。そうでない場合は、*name* にクラスタトランスポートスイッチの名前を指定します。

port コンポーネントが指定されなかった場合、デフォルトのポート名を使って処理が試みられます。アダプタに対するデフォルトポートは、常に `0` になります。スイッチ終端に対するデフォルトポート名は、ケーブルの他端に接続されたノードのノード ID と等しくなります。*ポート*の割り当てやその他の要件の詳細は、クラスタトランスポートアダプタおよびクラスタトランスポートスイッチのマニュアルページを参照してください (例: 789 ページの`scconf_transp_jct_etherswitch`(1M))。ケーブルを追加する前に、ケーブルの 2 つの終端にある各アダプタおよびスイッチを構成しておく必要があります (`-A` および `-B` を参照)。

`noenable`

ケーブルを構成に追加するときにこれを使用できます。デフォルトでは、ケーブルの追加時に、ケーブル、それが接続される 2 つのポート、それらのポートがあるアダプタまたはスイッチの各状態が `enable` に設定されます。これに対して、ケーブルの追加時に `noenable` を指定すると、ケーブルおよびその 2 つの終端は disabled の状態で追加されます。ただし、これらのポートの位置するスイッチまたはアダプタの状態は、変更されません。

`state=`*state*

ケーブルとそれが接続されている 2 つの終端の状態を変更します。ケーブルが有効になると、ケーブル、その 2 つのポート、およびそれらの 2 つのポートに関連するアダプタやスイッチが、すべて有効になります。これに対して、ケーブルを使用不可にしても、使用不可となるのはケーブル自体とその 2 つのポートだけです。これら 2 つのポートに関連するスイッチまたはアダプタの状態は変更されません。デフォルトでは、ケーブルとその終端の状態は、ケーブルが構成に追加された時点で、常に `enabled` に設定されます。ただし、`disabled` 状態にあるケーブルを追加するには、追加操作の一部として `noenable` サブオプションを使用します。

このコマンドオプションを、`-a`、`-c`、または `-r` と一緒に使用するには、`solaris.cluster.transport.modify` の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

`-P` *privatehostname_options*

ノードの場合、プライベートホスト名を追加または変更します。

このコマンドの `add` (`-a`) 形式で使用するとき、`-P` オプションは次のアクションの 1 つを指定します。

- ノードが指定されている場合、このコマンドは、プライベートクラスタインターコネクトまたはトランスポートにおいて、指定されたノードの IP アクセスで使用するように、指定されたホスト名エイリアスを割り当てます。それ以外が割り当てられていない、あるいは、リセットされた場合、ノードのデフォルトのプライベートホスト名は `clusternode`*nodeid*`-priv` になります。
- このホスト名は、企業内のほかのノードで使用されていないものにしてください。

  クラスタ用に構成されているプライベート IP アドレス範囲は、クラスタ内で使用されるプライベート IP アドレスの増加をサポートできる必要があります。プライベート IP アドレ

スを割り当てる前に、プライベート IP アドレス範囲で、追加されるプライベート IP アドレスをサポートできることを確認します。詳細は、853 ページのscprivipadm(1M) のマニュアルページを参照してください。

change (-c) 形式のコマンドで使用する場合、-P オプションは、指定されたノードのホスト名エイリアスを変更します。

プライベートホスト名は、Unresolved link to " hosts4" データベースには格納しないようにしてください。プライベートホスト名のホスト名検索はすべて、nsswitch という特殊な機能により実行されます (Unresolved link to " nsswitch.conf4" を参照)。

-P が使用可能なコマンドの各形式で、*privatehostname_options* はそれぞれ次のようになります:

追加:

```
–P node=node[,privatehostname=hostalias]
```

変更:

```
–P node=node[,privatehostname=hostalias]
```

削除:

```
-P node=node
```

-P オプションは、次のサブオプションをサポートしています。

node=*node*

privatehostname サブオプションで指定されたプライベートホスト名 (またはホストのエイリアス) が割り当てられるノードの名前または ID を提供します。

privatehostname=*hostalias*

プライベートクラスタインターコネクトまたはトランスポートのノードアクセスに使用するホストのエイリアスを指定します。privatehostname サブオプションが指定されていない場合、指定された node のプライベートホスト名はデフォルトにリセットされます。

このコマンドオプションを、-a、-c、または -r と一緒に使用するには、solaris.cluster.transport.modify の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-q *quorum_options*

共有クラスタ定足数デバイスやさまざまなクラスタ定足数プロパティーを管理します。マニュアルページのタイプ依存型定足数デバイスオプションに特に注意してください (例: 781 ページのscconf_quorum_dev_scsi(1M))。

注意 - コントローラベースのレプリケーションを使用するデバイスは、Oracle Solaris Cluster 環境では定足数デバイスとしては使用できません。-q オプションを使用してコントローラベースのレプリケーションを使用するデバイスを指定する場合、scconf コマンドはエラーを返します。

構成への共有定足数デバイスの追加および削除には、add および remove 形式のコマンドを使用します。各種のクラスタ定足数の構成プロパティーや状態を変更するには、change 形式のコマンドを使用します。これら 3 つの各形式のコマンドで使用可能な -q *quorum_options* を使用すると、次のようにクラスタ定足数構成を変更できます:

追加:

```
–q name=devicename,
type={scsi}
```

For SCSI quorum devices only:

```
-q autoconfig[,noop]
```

変更:

```
-q node=node,{
maintstate | reset}
–q name=devicename,{
maintstate | reset}
–q reset
-q installmode
```

For SCSI quorum devices only:

```
-q autoconfig[,noop]
```

削除:

```
–q name=devicename
```

scconf による定足数に関係した処理中に割り込みや失敗が発生した場合、クラスタ構成データベースの定足数の構成情報の一貫性が維持できなくなることがあります。この状態が発生した場合は、同じ scconf コマンドを再度実行するか、または定足数情報をリセットするための reset サブオプションを指定して実行します。

-q オプションは、次のサブオプションをサポートしています。

autoconfig

add 形式のコマンドで使用された場合、2 ノードクラスタから 1 つの定足数デバイスを自動的に選択し、割り当てます。この定足数デバイスは、使用可能なデバイスから選択されます。定足数デバイスがすでに構成されている場合には、このコマンドは異常終了します。

change 形式のコマンドで使用された場合、1 つのデバイスを自動的に選択し、割り当てます。このデバイスが、2 ノードクラスタのすべての既存定足数デバイスを置き換えます。この定足数デバイスは、使用可能なデバイスから選択されます。

クラスタにあるすべてのデバイスは、定足数デバイスとしての資格を備えていなければなりません。autoconfig サブオプションは、使用可能なデバイスが定足数デバイスの資格を備えているかどうかを評価しません。

クラスタに 3 つ以上のノードが含まれている場合は、autoconfig サブオプションによって定足数構成が変更されることはありません。NAS デバイスを定足数として構成しようとしている場合は、autoconfig サブオプションを使用しないでください。

`installmode`

クラスタを強制的にインストールモードに戻します。`installmode` に設定されたノードは、ブート時に定足数構成のリセットを行いません。またこのモードにある間は、管理機能の多くが使用不可能になります。クラスタが最初にインストールされる際、`installmode` が設定された状態で設定されます。すべてのノードがはじめてクラスタに参加し、共有定足数デバイスが構成に追加されると、`scconf -c -q reset` を発行して投票数をデフォルト値にリセットし、`installmode` 設定をクリアします。

`name=`*`devicename`*

共有定足数デバイスをクラスタに追加したり、クラスタから削除する際に、接続されたストレージデバイスの名前を指定します。また、このサブオプションを `change` 形式のコマンドで使用して、定足数デバイスの状態を変更することもできます。

個々の定足数デバイスは、クラスタ内の最低 2 つのノードと接続またはポートする必要があります。非共有ディスクを、定足数デバイスにできません。

`change` 形式の `scconf` コマンドに `-q name` を指定することで、デバイスを保守状態に移行させたり、あるいはデバイスの定足数に関する構成をデフォルトにリセットしたりすることができます。保守状態に移行させた場合、デバイスは投票数が 0 となるため、定足数の形成には参加しません。デフォルトにリセットした場合、デバイスの投票数は N-1 に変更されます。この N は、デバイスへのポートを有する、投票数がゼロ以外のノードの数です。

`node=`*`node`*

`add` 形式のコマンドで使用する場合は、追加しようとする共有定足数デバイスのポートで構成すべきノードをこのサブオプションで選択します。また、このサブオプションを `change` 形式のコマンドで使用して、ノードの定足数状態を変更することもできます。

`node` サブオプションが `change` 形式の定足数更新コマンドで使用される場合、このサブオプションはノードを保守状態に移行するか、またはノードの定足数構成をデフォルトにリセットするために使用されます。

ノードを保守状態に移行する前にノードを停止する必要があります。またクラスタメンバーを保守状態に移行しようとすると、`scconf` はエラーを返します。

保守状態に移行させた場合、ノードの投票数は 0 になるため、定足数の形成には参加しません。また、ノードへのポートを構成した共有定足数デバイスについては、新しいノードの状態を反映するため、その投票数が 1 つ小さくなります。ノードがデフォルトにリセットされる場合、その投票数は 1 にリセットされ、共有定足数デバイスの投票数は元の値に再調整されます。クラスタが `installmode` にないかぎり、個々のノードに対する定足数の構成は、ブート時に自動的にリセットされます。

node には、ノード名またはノード ID を指定できます。

`type=`*`type`*

`add` 形式のコマンドで使用した場合、作成する定足数デバイスの種類を指定します。

scsi

共有ディスク定足数デバイスを指定します。SCSI タイプ固有のオプションについては、781 ページのscconf_quorum_dev_scsi(1M) を参照してください。

{maintstate}

change 形式のコマンドで、globaldev と node のいずれかのサブオプションのフラグとして使用する場合は、共有定足数デバイスまたはノードを定足数の保守状態に移行します。保守状態に移行させた共有デバイスやノードは、定足数の形成には参加しません。これは保守作業のためにノードやデバイスを長期に渡って停止させる場合に有用な機能です。通常の状況では、ノードをブートし、クラスタに戻すと、ノードは自動的に保守モードを脱します。

maintstate および reset オプションを、1 つの -q オプションに対して同時に指定できません。

[,noop]

autoconfig サブオプションとともに指定した場合に有効です。このコマンドは、autoconfig サブオプションによって追加または変更される定足数デバイスのリストを標準出力に出力します。autoconfig,noop サブオプションによって定足数構成が変更されることはありません。

{reset}

change 形式のコマンドでフラグとして使用する場合は、共有定足数デバイスまたはノードの、構成された定足数投票数をこれでリセットします。このオプションは、globaldev または node サブオプションと組み合わせることも、単独のサブオプションにすることもできます。

このオプションを単独で使用した場合、定足数のすべての構成は、デフォルトの投票数にリセットされます。また installmode が設定されていると、グローバルな定足数構成のリセットによりクリアされます。1 つ以上の共有定足数デバイスが正常に構成されている場合を除き、2 ノードクラスタでは installmode をリセットできません。

otheroptions

その他の定足数デバイスタイプ固有のオプションを使用できます。詳細は、781 ページのscconf_quorum_dev_scsi(1M) を参照してください。

このコマンドオプションを、-a、-c、または -r と一緒に使用するには、solaris.cluster.quorum.modify の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-S *slm_options*

change 形式の sconf コマンドで使用する場合、システムリソース制御を構成するためのプロパティーを設定します。これらのプロパティーに値を割り当てない場合、自動的にデフォルト値に設定されます。

-S オプションの構文は次のとおりです。

```
–S [node=node] \
```

```
[,globalzoneshares=integer] \
[,defaultpsetmin=integer]
```

-S オプションは、次のサブオプションをサポートしています。

`globalzoneshares=`*globalzoneshares*

大域ゾーンに割り当てられるシェアの数を設定します。`globalzoneshares` の下限は 1、上限は 65,535 です。この上限を理解するには、Unresolved link to " prctl1" のマニュアルページの `zone.cpu-shares` 属性を参照してください。`globalzoneshares` のデフォルト値は 1 です。動作しているクラスタ上で、大域ゾーンに CPU 制御が構成されたオンラインリソースグループが存在しなくなった場合、その大域ゾーンに割り当てられている CPU シェア数が `globalzoneshares` の値に設定されます。

`defaultpsetmin=`*defaultpsetmin*

デフォルトプロセッサセットで使用可能な CPU の最小数を設定します。デフォルト値は 1 です。`defaultpsetmin` の最小値は 1 です。Oracle Solaris Cluster は、使用可能な CPU の範囲内で、`defaultpsetmin` に設定する数値にできるだけ近い CPU の数を割り当てます。割り当てられた数が要求した数より少ない場合、Oracle Solaris Cluster は一定期間ごとに要求した数の CPU を割り当てるように試みます。このアクションは、いくつかの `dedicated_weak` プロセッサセットを破壊する可能性があります。`dedicated_weak` プロセッサセットについては、861 ページの`scrgadm`(1M) のマニュアルページを参照してください。

`node=`*node*

プロパティーを設定するノードを指定します。ノードの名前を指定することにより、これらのプロパティーを CPU 制御を行う各ノードで設定します。-S オプションの使用ごとに、1 つのノードを指定できます。

このコマンドオプションを -c と併用するには、`solaris.cluster.node.modify` の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-T *authentication_options*

このオプションは、クラスタ構成への追加を試みるノードに対して、その認証ポリシーを確立します。特に、マシンがクラスタノードとしてクラスタに追加されることをリクエストした場合 (819 ページの`scinstall`(1M) を参照)、そのノードに参加する権利があるかどうかを判定するためのチェックが行われます。ノードにその権利があれば、結合しようとするノードは許可されます。デフォルトでは、すべてのマシンがクラスタに自ら結合する権利をもっています。

-T オプションが使用可能なコマンドの 3 つの形式で、-T *authentication_options* はそれぞれ次のようになります:

追加:

```
–T node=nodename[,...][,authtype=
authtype]
```

変更:

–T authtype=*authtype*

削除:

–T {node=*nodename*[,...] | all}

-T オプションは、次のサブオプションをサポートしています。

all

scconf -r -T all を指定すると、すべてのノード名のリストをクリアできます。このように認証リストがクリアされている場合、任意のノードからクラスタへのインストールおよび構成の要求を出すことができます。

node=*nodename*

クラスタ内のノードとしてインストールおよび構成が可能なノードリストにホスト名を追加したり、そこからホスト名を削除したりします。add 形式のコマンドには少なくとも 1 つの node サブオプションが必要ですが、remove ではオプションになります。認証リストが空であれば、どのホストもクラスタ構成への追加要求を行うことができます。これに対して、リストに 1 つでも名前が登録されていれば、こうした要求のすべてに対して、認証リストによる確認が行われます。

nodename には、ドット文字 (.) など、本来は不正なノード名も許されます。ドット文字は特殊な機能をもっています。つまり、許可リストに *nodename* として . を追加すると、ほかのすべての名前が削除されます。この機能は、ホストによるクラスタへのインストールおよび構成の要求を阻止します。

authtype=*authtype*

add または change 形式のコマンドで使用します。

現在サポートされている認証タイプ (authtype) は des と sys (または unix) だけです。デフォルトの認証型は sys ですが、これは最低限のセキュリティー保護しか実行しません。

des (Diffie-Hellman) 認証を使用する場合、実際に scinstall コマンドを実行してノードを追加する前に、各クラスタノードの publickey データベースにエントリを追加するようにしてください。

このコマンドオプションを、-a、-c、または -r と一緒に使用するには、solaris.cluster.node.modify の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-v

-p オプションで使用する場合は、クラスタ構成のより冗長性のある (あるいは詳細な) リストを要求します。また、ほかのオプションと併用することで、エラー発生時の追加情報を出力させることなども可能です。

このコマンドオプションを -p と併用するには、solaris.cluster.device.read、solaris.cluster.transport.read、solaris.cluster.resource.read、および solaris.cluster.system.read の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-w *heartbeat_options*

クラスタの広域ハートビートパラメータを変更します。その結果、クラスタのすべてのアダプタのハートビートパラメータが変更されます。

Oracle Solaris Cluster は、プライベートインターコネクトを介してこれらのハートビートを使用することによって、クラスタノード間の通信障害を検出します。ハートビートタイムアウト値を小さくすると障害検出に必要な時間が短縮されるため、ハートビートタイムアウトを短縮することにより、Oracle Solaris Cluster がより迅速に障害を検出できます。そのため、Oracle Solaris Cluster はより短い時間で障害から回復し、クラスタの可用性が向上します。

-w オプションは、次のサブオプションをサポートしています。

`heartbeat_quantum=`*quantum_milliseconds*

ハートビートを送信する頻度を定義します。Oracle Solaris Cluster は、デフォルトでは 1 秒、つまり 1,000 ミリ秒のハートビート定足数を使用します。100 ミリ秒から 10,000 ミリ秒までの値を指定します。

`heartbeat_timeout=`*timeout_milliseconds*

この間隔の間にピアノードからハードビートが受信されないと、このパスはダウンしていると宣言されます。Oracle Solaris Cluster は、デフォルトでは 10 秒、つまり 10,000 ミリ秒のハートビートタイムアウトを使用します。2,500 ミリ秒から 60,000 ミリ秒までの値を指定します。

注記 - 理想的な条件下であっても、-w オプションでハードビートパラメータの値を減らすと、誤ったパスタイムアウトやノードパニックが起こるおそれが常にあります。より小さい値を実際にクラスタに導入する際には、適切な負荷条件のもとでハートビートパラメータの値を必ずテストし、入念に確認する必要があります。

-w オプションでは、ハートビートサブオプションを一度に 1 つしか変更できません。ハートビートパラメータの値を減らすときは、まず、`heartbeat_quantum` を変更し、次に、`heartbeat_timeout` を変更します。ハートビートパラメータの値を増やすときは、まず、`heartbeat_timeout` を変更し、次に、`heartbeat_quantum` を変更します。

注記 - `heartbeat_timeout` に指定する値は常に、`heartbeat_quantum` に指定する値の 5 倍以上である必要があります (`heartbeat_timeout` >= (5*`heartbeat_quantum`))。

-w を使用するには、`solaris.cluster.system.modify` RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

例 301 ハートビートの短縮

次の例では、ハートビート数量を、Oracle Solaris Cluster のデフォルト値である 1,000 ミリ秒から 100 ミリ秒に短縮します。さらに、この例では、ハートビートタイムアウトをデフォルト値の 10,000 ミリ秒から 2,500 ミリ秒に短縮します。

```
phys-schost-1# scconf -c -w heartbeat_quantum=100
phys-schost-1# scconf -c -w heartbeat_timeout=2500
```

heartbeat_timeout の値は常に heartbeat_quantum の 5 倍以上でなければならないため、heartbeat_quantum を先に設定する必要があります。そうしないと、5 倍以上という要件が満たされないおそれがあります。つまり、現在 heartbeat_quantum がデフォルトの 1,000 ミリ秒に設定されている場合、heartbeat_timeout が 2,500 ミリ秒に設定されていると、heartbeat_timeout は heartbeat_quantum の 5 倍*未満*です。その結果、scconf コマンドは失敗します。

しかし、heartbeat_quantum を正しい値に設定したあとであれば、要件が満たされ、heartbeat_timeout の値を減らすことができます。

例 302 Heartbeat の増加

次の例では、前の例で設定したハートビートタイムアウトとハートビート数量の値を Oracle Solaris Cluster のデフォルト値に戻します (増やします)。

```
phys-schost-1# scconf -c -w heartbeat_timeout=10000
phys-schost-1# scconf -c -w heartbeat_quantum=1000
```

heartbeat_timeout は常に heartbeat_quantum の 5 倍以上でなければならないため、heartbeat_timeoutを先に設定する必要があります。heartbeat_timeout を希望の値に設定したあと、heartbeat_quantum を増やして新しい値に設定できます。

例 303 インストール後に行う通常の設定処理

次は、2 ノードクラスタを新規に構成した場合、インストール後に通常行う設定処理のサンプルコマンドです。これらのコマンドが実行する処理は、クラスタへの共有定足数デバイスの追加、installmode の解除、クラスタトラスポート接続の二次セットの設定、クラスタへの追加が想定されるマシンに対するクラスタのセキュリティー保護などです。

```
phys-red# scconf -a -q globaldev=d0
phys-red# scconf -c -q reset
phys-red# scconf -a \
```

```
    -A trtype=dlpi,name=hme1,node=phys-red \
    -A trtype=dlpi,name=hme1,node=phys-green \
    -m endpoint=phys-red:hme1,endpoint=phys-green:hme1
phys-red# scconf -a -T node=.
```

次の終了値が返されます。

`0` コマンドは正常に完了しました。

0 以外 エラーが発生しました。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/system/core`
インタフェースの安定性	廃止

19 ページの`Intro`(1CL)、61 ページの`cldevice`(1CL)、769 ページの`scconf_dg_rawdisk`(1M)、773 ページの`scc`
link to " hosts4"、Unresolved link to " nsswitch.conf4"、Unresolved link to " publickey4"、Unresolved link to " attributes5"、1483 ページの`sctransp_dlpi`(7p)

クラスタのノードがすべて起動している場合に限り、`-w` オプションを使用します。クラスタのノードが 1 つでも停止している場合には、`-w` を使用しないでください。ほかのノードがハング状態やパニック状態になることがあります。

1 つまたは複数のシングル CPU ノードを含むクラスタや、8 つを超えるのノードを持つクラスタに、小さい値のハートビートパラメータを設定すると、タイムアウトやノードパニックが起こる可能性が高くなります。

注記 - 理想的な条件下であっても、`-w` オプションでハードビートパラメータの値を減らすと、誤ったパスタイムアウトやノードパニックが起こるおそれが常にあります。より小さい値を実際にクラスタに導入する際には、適切な負荷条件のもとでハートビートパラメータの値を必ずテストし、入念に確認する必要があります。

`scconf` で構成を変更したあとにすべてのノードのルートファイルシステムをバックアップするか、すべての変更のログをとるべきです。通常のシステムバックアップとバックアップの間に構成変更を回復する場合は、このログを使ってもっとも新しい構成に戻ることができます。

scconf コマンドに指定するオプションリストは、コマンド行に入力された順に実行されます。ただし特定のトランスポートオプション (-A、-B、-m) については、可能な限りクラスタ構成データベースに対する単独のトランザクションとなるように、scconf で処理されます。クラスタのオーバーヘッドを減らすために、関連するこのタイプのオプションはできるだけまとめて 1 つのコマンド行に指定すべきです。

名前

scconf_dg_rawdisk — raw ディスクデバイスグループ構成の追加、変更、または更新

```
scconf -a -D type=rawdisk, [generic_options] [,globaldev=gdev1,globaldev=
     gdev1,…] [,localonly=true]

scconf -a -D type=rawdisk, [generic_options] [,globaldev=gdev1,globaldev=
     gdev1,…] [,localonly=true | false]

scconf -c -D name=diskgroup,autogen=true

scconf -r -D device_service_name [,nodelist=node[:node]…]
     [,globaldev=gdev1,…]
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

次の情報は、`scconf` コマンドに固有です。同等のオブジェクト指向コマンドを使用するには、79 ページの`cldevicegroup`(1CL) のマニュアルページを参照してください。

`scconf_dg_rawdisk` ユーティリティーは、raw ディスクデバイスグループ構成を追加、変更、または更新します。

rawdisk とは、ボリュームマネージャーのボリュームまたはメタデバイスの一部として使用されていないディスクのことです。raw ディスクデバイスグループを使用すると、ディスクデバイスグループ内にディスクセットを定義できます。

デフォルトでは、システムブート時に、raw ディスクデバイスグループが、構成内の Disk ID (DID) 擬似ドライバデバイスごとに作成されます。慣例的に、raw ディスクデバイスグループ名は初期化時に割り当てられ、DID 名から派生します。raw ディスクデバイスグループに追加されるノードごとに、`scconf` ユーティリティーは、グループのすべてのデバイスがそのノードのポートに物理的に接続されていることを確認します。

`scconf -a` (追加) コマンドは、複数のディスクデバイスが構成されている raw ディスクデバイスグループを作成する際に使用できます。raw ディスクデバイスグループは、ブート時に、クラスタにあるディスクデバイスごとに作成されます。

新しい raw ディスクデバイスグループを追加するには、事前に、新しいグループで使用するデバイスを、ブート時に作成されたデバイスグループから削除してください。これにより、これらのデバイスを含む新しい raw ディスクデバイスグループを作成できます。実際の操作としては、該当するデバイスのリストを globaldev オプションに指定するとともに、プライマリノードとなりうるプレファレンスリストを nodelist オプションに指定して、scconf コマンドを実行します。すでに存在しているデバイスグループに対しては、新規のノードおよびグローバルデバイスのみを追加することができ、既存のデバイスグループに属しているノードやデバイスは無視されます。

新しいデバイスグループを作成するために -a オプションとともに preferenced サブオプションが指定されていない場合、このサブオプションはデフォルトで false になります。ただし、既存のデバイスグループに対して true または false の値の preferenced サブオプションが指定されている場合は、エラーが返されます。これは既存の nodelist プレファレンスを維持するための措置です。

特定のノードのみに支配させるデバイスグループの場合、構成時に otheroption を localonly=true に指定します。localonly を指定してデバイスグループを作成する際は、ノードリストには 1 つのノードしか指定できません。

scconf -c (変更) コマンドは、プライマリノードとなり得るプレファレンスリストの順序の変更、フェイルバックの有無の変更、二次ノードの指定の設定、デバイスグループへの広域デバイスの再追加などを行う際に使用します。

ノードプレファレンスリストの順序を変更する場合は、デバイスグループ内のすべてのノードを nodelist に指定する必要があります。さらに、ノード優先順位の順序を変更している場合は、preferenced サブオプションも true に設定する必要があります。

change で preferenced サブオプションが指定されていない場合は、すでに確立された true または false の設定が使用されます。

変更形式のコマンドでは、新規ノードを追加できません。change オプションは、デバイスグループに関する localonly デバイスグループ指定の有無を変更する際にも使用します。デバイスグループを localonly デバイスグループに変更するには、otheroption に localonly=true を指定します。localonly デバイスグループに戻らなくするには、localonly=false を指定します。nodelist には事前に、リストとして 1 つのノードを設定しておく必要があり、そうでない場合はエラーが返されます。localonly を true に設定する場合は、change 形式のコマンドに nodelist を指定できます。ただし、リストには 1 つのノードが構成されているはずであり、それ以外は受け付けられないので、こうした指定は不要です。なお、事前に構成したノード以外を指定しようとすると、エラーが返されます。

scconf -r (削除) コマンドは、クラスタデバイスグループの構成から、ノード、広域デバイス、デバイスグループ名を削除する際に使用します。デバイスグループ名とともにノードまたは広域デバイスを指定した場合、このデバイスグループからはまずこれらのノードや広域デバイスが削除されます。クラスタ構成からデバイスグループが削除されるのは、このデバイスグループから最後のデバイスまたはノードが削除されてからになります。デバイスグループ名のみを指定した場合 (ノードまたはデバイスを何も指定しない場合) は、デバイスグループ全体が削除されます。

raw ディスクデバイス名が raw ディスクデバイスグループに登録されている場合、Solaris Volume Manager デバイスグループには登録できません。

サポートされている汎用オプションのリストについては、747 ページのscconf(1M) のマニュアルページを参照してください。

下記のアクションオプション一覧は、コマンドの動作を制御するためのものです。このコマンドで使用できるアクションオプションは 1 つだけです。

サポートされるアクションオプションには、次のものがあります。

-a
: 新しい raw ディスクデバイスグループをクラスタ構成に追加します。このオプションは、デバイスグループの構成を変更する際にも使用できます。

-c
: ノードプレファレンスリストの順序の変更、プレファレンスおよびフェイルバックポリシーの変更、二次ノードの指定数の変更、globaldev オプション指定によるデバイスグループへのデバイスの追加を行います。そのほかに、デバイスグループのローカル指定をする際にも使用します。

-r
: raw ディスクデバイスグループ名をクラスタから削除します。

  autogen フラグは、scconf コマンドのインジケータとして使用します。このコマンドは、-v コマンド行オプションを使用しないかぎり、デバイスの autogen プロパティーを一覧表示しません。change 形式 (c) の scconf コマンドを使用した場合、autogen=true が指定されないかぎり、このデバイスの autogen プロパティーはリセットされ false に設定されます。

例 304 scconf コマンドの使用法

次の scconf コマンドは、raw ディスクデバイスグループの作成、潜在的なプライマリノードの順番の変更、プレファレンスおよびフェイルバックポリシーの変更、セカンダリノードの指定数の変更、および raw ディスクデバイスグループのクラスタ構成からの削除を行います。

```
host1# scconf -a -D type=rawdisk,name=rawdisk_groupname,
nodelist=host1:host2:host3,preferenced=false,failback=enabled,
numsecondaries=,globaldev=d1,globaldev=d2

host1# scconf -a -D type=rawdisk,name=rawdisk_groupname,
nodelist=host1,globaldev=d1,globaldev=d2,localonly=true,
globaldev=d1,globaldev=d2

host1# scconf -c -D name=rawdisk_groupname,
nodelist=host3:host2:host1,preferenced=true,failback=disabled,
numsecondaries=2,globaldev=d4,globaldev=d5

host1# scconf -c -D name=rawdisk_groupname,localonly=true

host1# scconf -r -D name=rawdisk_groupname

host1# scconf -r -D name=rawdisk_groupname,nodelist=host1,host2
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
アーキテクチャー	SPARC
使用条件	ha-cluster/system/core
インタフェースの安定性	廃止

19 ページの Intro(1CL), 79 ページの cldevicegroup(1CL), 747 ページの scconf(1M), Unresolved link to " attributes5"

名前

scconf_dg_svm — Solaris Volume Manager デバイスグループ構成の変更

```
scconf -c -D [generic_options]
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページのIntro(1CL) のマニュアルページを参照してください。

次の情報は、scconf コマンドに固有です。同等のオブジェクト指向コマンドを使用するには、79 ページのcldevicegroup(1CL) のマニュアルページを参照してください。

Solaris ボリュームマネージャーでデバイスグループを定義する際には、名前、該当グループへアクセス可能なノード、ディスクセット内のデバイスに対するグローバルリスト、プライマリプレファレンスおよびフェイルバック動作などのアクション制御用プロパティーセットを指定します。

Solaris ボリュームマネージャーデバイスグループの場合、各デバイスグループに割り当てられるディスクセットは 1 つだけであり、グループ名とディスクセット名は常に一致している必要があります。

Solaris Volume Manager では、多重ホストまたは共有デバイスは、すべてのホストからアクセス可能であり、すべてのホスト上で同じデバイス名を持つ 2 つ以上のホストおよびディスクドライブのグループ化です。こうしたデバイス名の統一は、raw ディスクデバイスを用いてディスクセットを構築することにより行います。デバイス ID 疑似ドライバ (DID) を使用すると、多重ホストデバイスがクラスタ全体にわたって一貫性のある名前を持つようにすることができます。Solaris Volume Manager デバイスグループの nodelist に構成できるのは、該当するディスクセットへの構成が済んだホストだけです。共用ディスクセットに追加されたドライブを、他の共用ディスクセットに所属させることはできません。

Solaris Volume Manager の metaset コマンドはディスクセットを作成します。はじめに Solaris Volume Manager デバイスグループとしてディスクセットを作成および登録することも行います。次に、scconf コマンドを使用して、ノード優先順位リスト、preferenced、failback、および numsecondaries サブオプションを設定する必要があります。

ノードプレファレンスリストの順序やフェイルバックモードを変更する場合は、nodelist 内のデバイスグループに存在するすべてのノードを指定する必要があります。さらに、ノード優先順位の順序を変更している場合は、preferenced サブオプションも true に設定する必要があります。

「change」形式のコマンドで preferenced サブオプションを指定しない場合は、すでに確立された true または false の設定が使用されます。

Solaris Volume Manager のデバイスグループをクラスタ構成から削除する場合、scconf コマンドを使用できません。その代わりに、Solaris ボリュームマネージャーの metaset コマンドを使用します。デバイスグループを削除すると、Solaris ボリュームマネージャーディスクセットが削除されます。

サポートされている汎用オプションのリストについては、747 ページのscconf(1M) を参照してください。ディスクセットおよびデバイスグループの作成や削除を行うための metaset の関連コマンドの一覧については、Unresolved link to " metaset1M" を参照してください。

このコマンドで使用できるアクションオプションは 1 つだけです。サポートされるアクションオプションには、次のものがあります。

-c　　ノードプレファレンスリストの順序の変更、プレファレンスおよびフェイルバックポリシーの変更、二次ノード数の変更を行います。

例 305　　ディスクセットの作成と登録

次の例では、metaset コマンドにより、ディスクセット diskset を作成して、そのディスクセットを Solaris ボリュームマネージャーデバイスグループに登録します。

次に scconf コマンドを使用して、デバイスグループに対し潜在的に使用するプライマリノードの順序指定、プレファレンスおよびフェイルバック用オプションの変更、セカンダリノード数の変更を行います。

```
host1# metaset -s diskset1 -a -h host1 host2
host1# scconf -c -D name=diskset1,nodelist=host2:host1,
preferenced=true,failback=disabled,numsecondaries=1
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core

属性タイプ	属性値
インタフェースの安定性	廃止

19 ページの`Intro`(1CL)、79 ページの`cldevicegroup`(1CL)、747 ページの`scconf`(1M)、Unresolved link to " metaset1M"

名前

scconf_quorum_dev_quorum_server — 定足数サーバータイプの定足数デバイスの追加、削除、および構成

```
scconf [-q quorum-options]
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

Oracle Solaris Cluster には、定足数サーバーを定足数デバイスとして構成するオプションがあります。構成情報は、定足数デバイス全体で一意でなければならないデバイス名、定足数サーバーが動作するホストマシンのアドレス、および定足数サーバーが要求を待機するポート番号から構成されます。クラスタに複数の定足数デバイスが必要な場合、複数の定足数サーバーを構成するか、追加の定足数デバイス用のストレージデバイスを使用します。定足数サーバーは、クラスタごとに 1 つの定足数デバイスとしてのみ動作できます。

定足数サーバーを使用するようにクラスタを構成するには、すべてのクラスタノードからアクセス可能なマシンに、定足数サーバーソフトウェアをインストール、構成、および実行してください。このコマンドをクラスタノード上で実行するときには、定足数サーバー自身を構成および実行してください。定足数サーバーの構成については、Unresolved link to " clquorumserver1CL" を参照してください。

定足数サーバータイプの定足数デバイスを構成するには、次のパラメータが必要です。サポートされている汎用オプションのリストについては、Unresolved link to " scconf(1M)" を参照してください。

共有定足数デバイスを追加するには、このコマンドの `add` 形式を使用し、また、共有定足数デバイスを構成ファイルから削除するには、このコマンドの `remove` 形式を使用します。各種のクラスタ定足数の構成プロパティーや状態を変更するには、このコマンドの `change` 形式を使用します。クラスタ定足数構成を変更するには、次の定足数サーバー固有オプションを使用できます。

定足数サーバータイプの定足数デバイスの追加

定足数デバイスを追加する前に、次の準備を行います。

- 定足数サーバーが定足数サーバーホストマシン上で動作していることを確認してください。
- 定足数サーバーホスト名を `/etc/inet/hosts` ファイルに入力してください。
- 定足数サーバーホストのネットマスクを設定してください。

ホストファイルとネットマスクの要件については、Unresolved link to " Oracle Solaris Cluster システム管理 "にある定足数サーバー上の定足数デバイスの追加に関する手順を参照してください。定足数デバイスを追加したあと、パラメータは変更できなくなります。

```
# scconf –q -a name=devicename,type=quorum_server,qshost=qhost,port=portnumber
```

`name=`*devicename*

定足数サーバーの名前。この名前は、システムのすべての定足数デバイス間で一意でなければなりません。

`type=quorum_server`

作成するディスクデバイスグループのタイプを示します。定足数サーバータイプの定足数デバイスに対しては、このパラメータの値は `quorum_server` でなければなりません。

`qhost=`*qhost*

すべてのクラスタノードから到達でき、定足数サーバーが動作しているネットワーク上のマシンのホスト名。ホストの IPv4 または IPv6 構成に基づいて、このホスト名は、`/etc/hosts` ファイル、`/etc/inet/ipnodes` ファイル、あるいは、その両方にエントリを持つ必要があります。

`port=`*portnumber*

定足数サーバーが要求を待機するポート。

注記 - ホスト名を変更せずに、定足数サーバーのポート番号を変更する必要がある場合は、まず、定足数デバイスを削除し、ポート番号を変更し、ふたたび、定足数デバイスを追加します。

定足数サーバータイプの定足数デバイスの構成パラメータの変更

```
# scconf -c -q name=devicename,maintstate | reset
```

`qshost` や `port` などのその他のパラメータを変更する必要がある場合は、新しいパラメータを指定して新しい定足数デバイスを追加し、既存の定足数デバイスを取り外します。

定足数サーバータイプの定足数デバイスの削除

```
# scconf -q name=devicename
```

`scconf` コマンドによる定足数に関係した処理中に割り込みや失敗が発生した場合、クラスタ構成データベースの定足数の構成情報の一貫性が維持できなくなることがあります。このような

場合は、前回と同じ scconf コマンドを再実行するか、scconf コマンドに reset オプションを指定して定足数情報をリセットしてください。

例 306 定足数サーバータイプの定足数デバイスの追加

次の scconf コマンドは、ポート番号を 9000 に構成して、定足数サーバータイプの定足数デバイスを追加します。

```
# scconf –q -a name=qd1,type=quorum_server,qshost=scclient1,port=9000
```

例 307 定足数サーバータイプの定足数デバイスの削除

次の scconf コマンドは、qd1 という名前の定足数サーバータイプの定足数デバイスを削除します。

```
# scconf -r –q name=qd1
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
安定性	廃止

19 ページの Intro(1CL), 239 ページの clquorum(1CL), Unresolved link to " clquorumserver1CL", 551 ページの cluster(1CL), Unresolved link to " scconf(1M)", Unresolved link to " gateways4", Unresolved link to " hosts4"

名前

scconf_quorum_dev_scsi — 共有 SCSI 定足数デバイスの追加と削除、およびさまざまな SCSI クラスタ定足数構成のプロパティーや状態の変更

```
scconf {-a|-c|-r} -q globaldev=devicename otheroptions

scconf {-a|-c|-r} -q name=devicename otheroptions
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

SCSI 定足数デバイスは、Oracle Solaris Cluster がサポートし、そのクラスタの複数のノードに接続されているストレージであると考えられます。SCSI 定足数デバイスは DID で管理しなければならず、指定するデバイス名は DID デバイス名でなければなりません。

SCSI 定足数デバイスには、この他に指定可能なプロパティーはありません。

次のオプションは、共有ディスク定足数デバイスに特有のものです。サポートされている汎用オプションのリストについては、747 ページの`scconf`(1M) を参照してください。

構成への共有定足数デバイスの追加および削除には、`add` または `remove` 形式のコマンドを使用します。クラスタ定足数構成のさまざまなプロパティーを変更するには、`change` 形式のコマンドを使用します。これら 3 つの各形式のコマンドで使用可能な `-q` *quorum-options* を使用すると、次のようにクラスタ定足数構成を変更できます:

共有定足数デバイスの追加:

```
–q -a globaldev=devicename[,
node=node,node=
node[, ...]]
```

または

```
–q -a name= devicename,type=scsi
```

または

```
 -q -a autoconfig[,noop]
```

定足数構成のプロパティーまたは状態の変更:

```
–q -c globaldev=devicename,{
maintstate | reset}
```

または

```
-q -c autoconfig[,noop]
```

共有定足数デバイスの削除:

```
–q -r globaldev=devicename
```

または

```
–q -r name=devicename
```

`autoconfig`

`add` 形式のコマンドで使用された場合、2 ノードクラスタから 1 つの定足数デバイスを自動的に選択し、割り当てます。この定足数デバイスは、使用可能なデバイスから選択されます。定足数デバイスがすでに構成されている場合には、このコマンドは異常終了します。

`change` 形式のコマンドで使用された場合、1 つのデバイスを自動的に選択し、割り当てます。このデバイスが、2 ノードクラスタのすべての既存定足数デバイスを置き換えます。この定足数デバイスは、使用可能なデバイスから選択されます。

クラスタにあるすべてのデバイスは、定足数デバイスとしての資格を備えていなければなりません。`autoconfig` サブオプションは、使用可能なデバイスが定足数デバイスの資格を備えているかどうかを評価しません。

クラスタに 3 つ以上のノードが含まれている場合は、`autoconfig` サブオプションによって定足数構成が変更されることはありません。NAS デバイスを定足数として構成しようとしている場合は、`autoconfig` サブオプションを使用しないでください。

[,`noop`]

`autoconfig` サブオプションとともに指定した場合に有効です。このコマンドは、`autoconfig` サブオプションによって追加または変更される定足数デバイスのリストを標準出力に出力します。`autoconfig,noop` サブオプションによって定足数構成が変更されることはありません。

`scconf` による定足数に関係した処理中に割り込みや失敗が発生した場合、クラスタ構成データベースの定足数の構成情報の一貫性が維持できなくなることがあります。不一致が発生すると、同じ `scconf` コマンドを再実行するか、またはコマンドを `reset` オプションを指定して実行し、定足数情報をリセットします。

`add` 形式のコマンドで、`node` リストなしで `name` を指定した場合は、デバイスを接続するすべてのノードに対して定義されたポートとともに定足数デバイスが追加されます。しかし、`node` リスト

が指定済みの場合は、少なくとも 2 つのノードを指定し、リスト内の各ノードとデバイス間のポートを定義する必要があります。

例 308 SCSI 定足数デバイスの追加

次の `scconf` コマンドは、SCSI 定足数デバイスを追加します。

```
-a –q globaldev=/dev/did/rdsk/d4s2
    or
-a –q name=/dev/did/rdsk/d4s2,type=scsi
```

例 309 SCSI 定足数デバイスの変更

次の `scconf` コマンドは、SCSI 定足数デバイス構成を変更します。

```
-c -q globaldev=/dev/did/rdsk/d4s2,reset
    or
-c -q name=/dev/did/rdsk/d4s2,reset
```

例 310 SCSI 定足数デバイスの削除

次の `scconf` コマンドは、SCSI 定足数デバイスを削除します。`qd1`.

```
-r –q globaldev=qd1
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	廃止

19 ページの`Intro`(1CL), 239 ページの`clquorum`(1CL), 551 ページの`cluster`(1CL), 747 ページの`scconf`(1M)

名前

scconf_transp_adap_bge — bge トランスポートアダプタの構成

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

`bge` アダプタはクラスタトランスポートアダプタとして構成できます。これらのアダプタは、トランスポートタイプが `dlpi` の場合に限り使用可能です。`bge` アダプタは VLAN 対応です。

`bge` アダプタは、トランスポートスイッチか、別のノード上の `bge` アダプタに接続します。どちらの場合でも、接続にはトランスポートケーブルを使用します。

1 つのトランスポートスイッチが使用され、トランスポートケーブルの終端が `scconf` コマンド、`scinstall` コマンドなどのツールで構成されている場合は、そのトランスポートスイッチ上のポート名を指定するように求められます。希望するポート名を指定するか、デフォルトの設定を使用します。そのスイッチで一意のポート名を指定する必要があります。

デフォルトのポート名は、ケーブルのもう一方の端でアダプタをホストしているノードの ID と等しくなります。

このタイプのクラスタトランスポートアダプタのプロパティーの構成は、ユーザーには許可されていません。

19 ページの`Intro`(1CL), 119 ページの`clinterconnect`(1CL), 185 ページの`clnode`(1CL), 747 ページの`scconf`(1M), 819 ページの`scinstall`(1M), Unresolved link to " bge7D"

名前

scconf_transp_adap_e1000g — Intel PRO/1000 ネットワークアダプタの構成

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

`e1000g` Intel PRO/1000 ネットワークアダプタは、クラスタトランスポートアダプタとして構成できます。これらのアダプタは、トランスポートタイプが `dlpi` の場合に限り使用可能です。

`e1000g` ベースのネットワークアダプタは、トランスポートスイッチか、別のノード上の Ethernet アダプタに接続します。どちらの場合でも、接続にはトランスポートケーブルを使用します。

1 つのトランスポートスイッチが使用され、トランスポートケーブルの終端が `scconf`、`scinstall` などのツールで構成されている場合は、そのトランスポートスイッチ上のポート名を指定するように求められます。希望するポート名を指定するか、デフォルトの設定を使用します。そのスイッチで一意のポート名を指定する必要があります。

デフォルトのポート名は、ケーブルの他方の端でアダプタをホストしているノード ID と等しくなります。

このタイプのクラスタトランスポートアダプタのプロパティーの構成は、ユーザーには許可されていません。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
アーキテクチャー	x86
使用条件	廃止

19 ページの`Intro`(1CL)、119 ページの`clinterconnect`(1CL)、185 ページの`clnode`(1CL)、747 ページの`scconf` link to " e1000g7D"

名前

scconf_transp_jct_etherswitch — Ethernet クラスタトランスポートスイッチの構成

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

Ethernet スイッチは、クラスタトランスポートスイッチ (トランスポート接続点とも呼ぶ) として使用できます。スイッチタイプは `switch` です。ユーザーが構成できるプロパティーはありません。

19 ページの`Intro`(1CL), 119 ページの`clinterconnect`(1CL), 185 ページの`clnode`(1CL)

名前

scconf_transp_jct_ibswitch — InfiniBand クラスタトランスポートスイッチの構成

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

InfiniBand スイッチは、クラスタトランスポートスイッチ (トランスポート接続点とも呼ぶ) として使用できます。スイッチタイプは `switch` です。ユーザーが構成できるプロパティーはありません。

19 ページの`Intro`(1CL), 119 ページの`clinterconnect`(1CL), 185 ページの`clnode`(1CL)

名前

scdidadm — デバイス識別子構成および管理用ユーティリティーラッパー

```
/usr/cluster/bin/scdidadm -b combined-did-instance

/usr/cluster/bin/scdidadm -C

/usr/cluster/bin/scdidadm -c

/usr/cluster/bin/scdidadm -F
     {pathcount | scsi3 | useglobal} instance

/usr/cluster/bin/scdidadm -G

/usr/cluster/bin/scdidadm -G {pathcount | prefer3}

/usr/cluster/bin/scdidadm {-l | -L} [-h] [-o fmt]… [path |
     instance]

/usr/cluster/bin/scdidadm -R {path | instance | all}

/usr/cluster/bin/scdidadm -r

/usr/cluster/bin/scdidadm -T remote-nodename -e replication-type

/usr/cluster/bin/scdidadm -t
     source-instance:destination-instance -e replication-type [-g
     replication-device-group]

/usr/cluster/bin/scdidadm [-u] [-i]

/usr/cluster/bin/scdidadm -v
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

`scdidadm` ユーティリティーは、デバイス識別子 (DID) の擬似デバイスドライバ `did` を管理します。

`scdidadm` ユーティリティーは、主に次の処理を行います。

- ドライバ構成ファイルの作成
- ファイル内のエントリの変更
- 現在の構成のカーネルへの読み込み

■ デバイスエントリと did ドライバのインスタンス番号間の対応付けの一覧表示

起動スクリプトの /etc/init.d/bootcluster は、did ドライバの初期化をする際に scdidadm ユーティリティーを使用します。さらに、scdidadm は、存在するデバイスと、対応するデバイス識別子および did ドライバインスタンス番号との間のデバイス対応付けを更新または照会するときにも使用されます。

Unresolved link to " devfsadm1M" コマンドは、ファイルシステムのデバイスエントリポイントを作成します。

このコマンドは、大域ゾーンだけで使用できます。

次のオプションがサポートされています。

-b

レプリケートされた DID インスタンスを、2 つの別々な DID インスタンスになる前の状態に戻します。構成の間違いを修正したり、元の DID インスタンスに影響を与える構成変更を準備したりするには、このオプションを使用します。

なおこのオプションは、クラスタモードでブートされたノードからしか実行できません。このオプションは大域ゾーンだけで使用できます。

-b オプションを使用する前に、このオプションを使用するすべてのデバイスグループから、レプリケートされたデバイスを削除します。次に、結合された DID インスタンスに吸収された DID インスタンスを持つノードの 1 つから、このオプションを指定します。

このコマンドオプションを使用するためには、solaris.cluster.device.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

-C

現在のノードから排除されたデバイスに関して、すべての DID 参照を削除するよう指定します。

なおこのオプションは、クラスタモードでブートされたノードからしか実行できません。このオプションは大域ゾーンだけで使用できます。

このオプションは、Solaris のデバイス関連のコマンドでクラスタノード上に存在しないデバイスの参照を削除したあとに指定します。

-F オプションは、構成済みの定足数デバイスのフェンシングプロトコルには影響しません。

このコマンドオプションを使用するためには、solaris.cluster.device.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

-c

デバイスおよび物理デバイスのカーネル部に対する整合性検査を指定します。

このオプションは大域ゾーンだけで使用できます。

この整合性検査で問題が発見されると、エラーメッセージが表示されます。またこの処理は、すべてのデバイスに対する検査が終了するまで継続されます。

このコマンドオプションを使用するためには、`solaris.cluster.device.read` の RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

`-e` *type*

複製タイプを指定します。SRDF レプリケーションタイプを指定する場合、このオプションと一緒に使用できるのは `-t` オプションのみです。

`-F`

指定した個々のデバイスの、グローバルなデフォルトフェンシングアルゴリズムに優先します。

デバイスのデフォルトのフェンシングアルゴリズムは、次の値の 1 つに設定できます。

`pathcount` 共有デバイスに接続されている DID パスの数でフェンシングプロトコルを決定します。

- 使用する DID パスが 3 未満のデバイスには、このコマンドは SCSI-2 プロトコルを設定します。
- 使用する DID パスが 3 以上のデバイスには、このコマンドは SCSI-3 プロトコルを設定します。

`scsi3` SCSI-3 プロトコルを設定します。そのデバイスが SCSI-3 プロトコルをサポートしない場合、フェンシングプロトコルの設定は変更されません。

`useglobal` 指定されたデバイスのグローバルなデフォルトのフェンシング設定を行います。

デフォルトでは、グローバルなデフォルトのフェンシングアルゴリズムは `pathcount` に設定されます。グローバルなデフォルトのフェンシング設定についての詳細は、`-G` オプションの説明を参照してください。

変更するデバイスはそのインスタンス番号で指定できます。このコマンドは、複数のデバイスのスペース区切りのリストを受け入れます。デバイス名の `instance` 形式の詳細は、`-o` オプションの説明を参照してください。

`-F` オプションは、構成済みの定足数デバイスのフェンシングプロトコルには影響しません。

このコマンドオプションを使用するためには、`solaris.cluster.device.modify` RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

`-G`

すべての共有デバイスの現在のグローバルなデフォルトのフェンシングアルゴリズムを設定または表示します。

単独で指定すると、`-G` オプションは現在のグローバルなデフォルトのフェンシングアルゴリズムの設定を表示します。

設定値を指定すると、-G オプションは、すべてのデバイスのグローバルなデフォルトのフェンシングアルゴリズムをその値に設定します。グローバルなデフォルトフェンシングは、次の値のいずれかに設定できます。

prefer3
: すべてのデバイスのデバイスフェンシングアルゴリズムを SCSI-3 プロトコルに設定します。SCSI-3 プロトコルをサポートしないデバイスには、pathcount 設定が割り当てられます。

pathcount
: 共有デバイスに接続されている DID パスの数でフェンシングプロトコルを決定します。

- 使用する DID パスが 3 つ未満のデバイスには、このコマンドは SCSI-2 プロトコルを設定します。
- 使用する DID パスが 3 つ以上のデバイスには、このコマンドは SCSI-3 プロトコルを設定します。

デフォルトでは、グローバルなデフォルトのフェンシングアルゴリズムは pathcount に設定されます。

-G オプションは、構成済みの定足数デバイスのフェンシングプロトコルには影響しません。

このコマンドオプションを使用するためには、solaris.cluster.device.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

-g

レプリケーションデバイスグループを指定します。

-h

デバイスの対応付けを一覧表示させる際に、ヘッダーを出力するよう指定します。

このオプションは、-l および -L のオプションを指定した場合にのみ有用です。

-i

did ドライバを初期化するよう指定します。

このオプションは大域ゾーンだけで使用できます。

did ドライバへの入出力要求を有効にする場合は、このオプションを使用します。

このコマンドオプションを使用するためには、solaris.cluster.device.modify RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

-L

リモートホストや DID 構成ファイル内のデバイスなどに関する、すべてのパスを一覧表示させます。

このオプションは大域ゾーンだけで使用できます。

なおこのコマンドによる出力は、-o オプションを使用することでカスタマイズできます。-o オプションを指定しない場合、デフォルトの一覧表示である*インスタンス*番号、ローカルおよびリモートのすべての*フルパス*、*フルネーム*が出力されます。

このコマンドオプションを使用するためには、solaris.cluster.device.read の RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

-l

DID 構成内のローカルデバイスを一覧表示するよう指示します。

このオプションは大域ゾーンだけで使用できます。

なおこのコマンドによる出力は、-o オプションを使用することでカスタマイズできます。-o オプションを指定しない場合、デフォルトの一覧表示である*インスタンス*番号、ローカルの*フルパス*、*フルネーム*が出力されます。

このコマンドオプションを使用するためには、solaris.cluster.device.read の RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

-o *fmt*

現在 did ドライバが認識しているデバイスを、書式指定 *fmt* に従って一覧表示させます。

-o オプションは複数指定することが可能です。*fmt* の指定は、コンマ切りの一覧による書式指定用のオプション引数と解釈されます。このオプションは、-l および -L のオプションを指定した場合にのみ有用です。使用可能な書式指定用のオプション引数は次のとおりです。

instance	did ドライバに対応するデバイスのインスタンス番号を出力させる引数で、たとえば、1 などが返されます。
path	デバイス識別子に対応したデバイスの物理パス名を出力させる引数で、たとえば、/dev/rdsk/c0t3d0 などが返されます。
fullpath	このデバイス識別子に対応したデバイスの物理パス名をフルパスで出力させます。このパス名には、ホスト (たとえば、phys-hostA:/dev/rdsk/c0t3d0) が含まれています。
host	-L オプションと併用した場合、指定されたデバイスに接続するすべてのホスト名を一覧表示させます。-l オプションと併用した場合、指定されたデバイスに接続するローカルホストの名前を一覧表示させます。
name	このデバイス識別子に対応したデバイスの DID 名を出力させる引数で、たとえば、d1 などが返されます。
fullname	デバイス識別子に対応したデバイスのフル DID パス名を出力させる引数で、たとえば、/dev/did/rdsk/d1 などが返されます。
diskid	一覧されたデバイスのインスタンスに対応したデバイス識別子を、16 進数表示で出力させます。
asciidiskid	一覧されたデバイスのインスタンスに対応したデバイス識別子を、ASCII 表示で出力させます。

`defaultfencing` デバイスに設定されているデフォルトのフェンシングアルゴリズムを出力します。

-R {*path* | *instance* | all}

指定されたデバイスインスタンスに対して、修復処理の実行を指定します。

このオプションは大域ゾーンだけで使用できます。

この場合のコマンドの引数には、新しいデバイスと交換した物理デバイスの*パス*か、あるいは交換したデバイスの*インスタンス*を指定します。`all` のキーワードを併用することにより、`scdidadm` ユーティリティーは、このノードに接続されたすべてのデバイスの構成データを更新します。

なおこのオプションは、クラスタモードでブートされたノードからしか実行できません。

このコマンドオプションを使用するためには、`solaris.cluster.device.modify` RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

-r

データベースの再構成を指定します。

このオプションは大域ゾーンだけで使用できます。

このオプションを指定すると、`rdsk` と `rmt` デバイスツリーが完全に検索されます。それまでに認識されていないすべてのデバイス識別子に対して新しいインスタンス番号が割り当てられます。新規に認識された個々のデバイスに対しては、それぞれ新しくパスが追加されます。

なおこのオプションは、クラスタモードでブートされたノードからしか実行できません。

このコマンドオプションを使用するためには、`solaris.cluster.device.modify` RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

-T *remote-nodename*

-e *replication—type* 引数で指定するストレージベースのレプリケーションタイプとともに使用する、DID デバイスを構成します。

このオプションは大域ゾーンだけで使用できます。

このオプションは、レプリケートされたデバイスが構成されているいずれか 1 つ のノードだけから実行します。*remote-nodename* オプション引数を使用して、リモートノードの名前を指定します。

ローカルノード上の DID インスタンスはリモートノード上の対応する DID インスタンスと結合され、レプリケートされたデバイスの各ペアは単一の論理 DID デバイスにマージされます。

このコマンドオプションを使用するためには、`solaris.cluster.device.modify` RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

-t *source-instance:destination-instance*

DID インスタンスを本来の場所から新しい宛先インスタンス (*destination-instance* オプション引数で指定) に移動します。

このオプションは大域ゾーンだけで使用できます。

このオプションは、ローカルのインスタンスを間違って変更した場合に、DID インスタンスを本来の場所に戻すときに使用します。共有ストレージに接続されたすべてのクラスタノード上でコマンドを実行したあとで、`devfsadm` および `scgdevs` コマンドを 1 つのノードから実行して、構成変更により広域デバイスのネームスペースを更新します。

destination-instance がクラスタ内に存在しない場合、*source-instance* 引数の値に対応する DID デバイスパスは削除され、指定した *destination-instance* で作成し直されます。

destination-instance がクラスタ内にすでに存在する場合、*source-instance* のパス (単数または複数) は *destination-instance* のパス (単数または複数) と結合され、両方のインスタンスのすべてのパスを含む単一の DID 宛先インスタンスになります。

レプリケーションタイプを指定する場合は、`-e` オプションを含めます。SRDF レプリケーションデバイスを使用している場合は、`-g` オプションを使用して、レプリケーションデバイスグループを指定します。

このコマンドオプションを使用するためには、`solaris.cluster.device.modify` RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

`-u`

デバイス識別子の構成テーブルをカーネルに読み込むよう指定します。

このオプションは大域ゾーンだけで使用できます。

このオプションを指定した場合、デバイスのパスおよび対応するインスタンス番号に関して、現在認識されているすべての構成情報がカーネルに読み込まれます。

このコマンドオプションを使用するためには、`solaris.cluster.device.modify` RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

`-v`

プログラムのバージョン番号の出力を指定します。

このオプションは大域ゾーンだけで使用できます。

例 311 ローカルホストに接続したデバイスの CCR への追加

```
# scdidadm -r
```

例 312 デバイスの物理パスの一覧表示

次のサンプルコードは、`did` ドライバのインスタンス `2` に対応するデバイス物理パスを一覧表示させます。

```
% scdidadm -l -o path 2
/dev/dsk/c1t4d0
```

例 313 書式指定用のオプション引数の複数指定

書式指定用のオプション引数を複数指定するには、次のいずれかの形式で記述します。

```
% scdidadm –l –o path –o name 2
```

```
% scdidadm -l -o path,name 2
```

いずれの形式の場合も、得られる結果は次のようになります。

```
/dev/dsk/c1t4d0 d1
```

例 314 EMC SRDF ストレージベースのレプリケーションで使用するための DID デバイスの構成

次の例では、EMC SRDF ストレージベースのレプリケーションで使用するために、ローカルの DID デバイスとリモートの DID デバイスを構成します。このコマンドは、レプリケートされたデバイスが構成されているローカルのソースノードから実行します。ソースノード上の DID インスタンスは、リモートの宛先ノード `phys-schost-1` 上の対応する DID インスタンスと結合されます。

```
# scdidadm -t 15:10 -e srdf -g devgroup1
```

例 315 レプリケートされた DID デバイスの構成解除

次の例では、レプリケートされた DID デバイス `d25` を、元の DID デバイスコンポーネントに戻します。これは、`phys-schost-1` 上のパス `d15` と、`phys-schost-2` 上のパス `d25` の結合により作成された DID デバイスです。2 つのパスが結合された際、パス `d15` はパス `d25` に統合されたため、`phys-schost-1` からコマンドを実行して、パス `d15` を確実に復元します。

```
phys-schost-1# scdidadm -b 25
```

例 316 DID インスタンスの移動

次の例は、ソースインスタンス `15` 上の DID インスタンスを、新しい DID インスタンス `10` に移動し、続いてグローバルデバイスのネームスペース内の構成変更を更新します。

```
# scdidadm -t 15:10
# devfsadm
# scgdevs
```

例 317 修復処理の実行

次のサンプルコードは、指定したデバイスのパスに対して、修復処理を実施させます。デバイス `/dev/dsk/c1t4d0` は、新しいデバイス識別子と対応付けられている新しいデバイスと交換され

ています。この新しいデバイス識別子が、古いデバイス識別子と対応付けられていたインスタンス番号に対応していることを示すため、データベースが更新されます。

```
# scdidadm -R c1t4d0
```

例 318 修復処理の実行

修復処理は、デバイスのパスと関連付けられたインスタンス番号を指定しても実行できます。たとえば、先の例で `c1t4d0` のデバイスのインスタンス番号 `2` であるとすれば、次のサンプルコードのように指定しても、前回と同様の結果が得られます。

```
# scdidadm -R 2
```

例 319 SCSI プロトコルのグローバルな設定

次の例では、クラスタ内のすべての SCSI デバイス (ただし、構成済みの定足数デバイスと SCSI-3 プロトコルをサポートしないデバイスを除く) を SCSI-3 プロトコルに設定します。SCSI-3 プロトコルをサポートしないデバイスは、`pathcount` に設定されます。

```
# scdidadm -G prefer3
```

例 320 単一デバイスの SCSI プロトコルの表示

次の例では、デバイス `/dev/rdsk/c0t3d0` の SCSI プロトコル設定を表示します。

```
# scdidadm -L -o defaultfencing /dev/rdsk/c0t3d0
```

例 321 単一デバイスの SCSI プロトコルの設定

次の例では、デバイス `11` (インスタンス番号で指定) を SCSI-3 プロトコルに設定します。このデバイスは、構成済みの定足数デバイスではなく、SCSI-3 プロトコルをサポートします。

```
# scdidadm -F scsi3 11
```

次の終了値が返されます。

`0` コマンドは正常に完了しました。

`1` エラーが発生しました。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	安定

19 ページの`Intro`(1CL)、61 ページの`cldevice`(1CL)、Unresolved link to " devfsadm1M"、815 ページの`scgdevs`(1M)、1479 ページの`did`(7)

Unresolved link to " Oracle Solaris Cluster システム管理 "

マルチポート式のテープドライブまたは CD-ROM ドライブはそれぞれ、個々の物理接続をするごとにネームスペースに表示されます。

名前

scdpm — ディスクパスモニタリングデーモンの管理

```
scdpm [-a] {node | all}

scdpm -f filename

scdpm -m {[node | all][:/dev/did/rdsk/]dN | [:/dev/rdsk/]cNtXdY | all}

scdpm -n {node | all}

scdpm -p [-F] {[node | all][:/dev/did/rdsk/]dN | [/dev/rdsk/]cNt
     XdY | all}

scdpm -u {[node | all][:/dev/did/rdsk/]dN | [/dev/rdsk/]cNtXdY | all}
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページのIntro(1CL) のマニュアルページを参照してください。

---

scdpm コマンドは、クラスタ内で動作するディスクパスモニタリングデーモンを管理します。このコマンドは、ディスクパスをモニターおよびモニター解除するのに使用します。このコマンドは、ディスクパスまたはノードのステータスを表示するのにも使用できます。その場合には、クラスタまたは特定のノードにあるアクセス可能なすべてのディスクパスが標準出力に出力されます。このコマンドは、オンラインであり、クラスタモードであるクラスタノードで実行してください。

新しいディスクパスをモニターする際には、広域名を指定することも UNIX 名を指定することもできます。さらに、ディスク構成全体の再読み込みをデーモンに指示することもできます。

このコマンドは、大域ゾーンだけで使用できます。

次のオプションがサポートされています。

-a

次の条件に適合する場合に、モニターしているすべてのディスクパスが失敗したときのノードの自動リブートを有効にします。

- 該当ノード上でモニター対象のすべてのディスクパスが失敗した。
- 少なくとも 1 つのディスクがクラスタ内の異なるノードからアクセス可能である。

このオプションは大域ゾーンだけで使用できます。

ノードがリブートすると、そのノード上でマスターされているすべてのリソースとデバイスグループが別のノード上で再起動します。

ノードが自動リブートしたあと、ノード上のすべてのモニター対象ディスクパスがアクセス不能のままである場合、そのノードはふたたび自動リブートしません。ただし、ノードがリブートしたあとにモニター対象ディスクパスのどれかが使用可能になり、その後、すべてのモニター対象ディスクパスでふたたび問題が発見されると、ノードはふたたび自動的にリブートします。

このコマンドを使用するには `solaris.cluster.device.admin` の役割に基づくアクセス制御 (RBAC) が必要です。Unresolved link to " rbac5" を参照してください。

`-F`

`-F` オプションを `-p` オプションとともに指定すると、`scdpm` はクラスタに存在する障害のあるディスクパスも出力します。`-p` オプションは、ノードまたは指定されたディスクパスの現在のステータスを、このストレージに接続されているすべてのノードから出力します。

`-f` *filename*

filename にあるモニターまたはモニター解除するディスクパスのリストを読み込みます。

このオプションは大域ゾーンだけで使用できます。

次の例では、*filename* の内容を示します。

```
u schost-1:/dev/did/rdsk/d5
m schost-2:all
```

このファイルの各行には、ディスクパスをモニターまたはモニター解除するかどうか、ノード名、およびディスクパス名を指定してください。モニターする場合は、`m` オプション、モニターを解除する場合は、`u` オプションを指定します。コマンドとノード名の間には、空白を挿入してください。また、ノード名とディスクパス名の間にコロン (:) を挿入することもできます。

このオプションを使用するには、`solaris.cluster.device.admin` RBAC 認証が必要です。Unresolved link to " rbac5" を参照してください。

`-m`

node:*diskpath* で指定される新しいディスクパスをモニターします。

このオプションは大域ゾーンだけで使用できます。

このオプションを使用するには、`solaris.cluster.device.admin` RBAC 認証が必要です。Unresolved link to " rbac5" を参照してください。

`-n`

モニターされているすべてのディスクパスが失敗したときのノードの自動リブートを無効にします。

このオプションは大域ゾーンだけで使用できます。

ノード上にあるモニターされているすべてのディスクパスが失敗した場合、そのノードはリブートしません。

このオプションを使用するには、solaris.cluster.device.admin RBAC 認証が必要です。Unresolved link to " rbac5" を参照してください。

-p

指定されたディスクパスの現在のステータスを、このストレージに接続されているすべてのノードから出力します。

このオプションは大域ゾーンだけで使用できます。

-F オプションを指定すると、scdpm はクラスタに存在する障害のあるディスクパスを出力します。

ディスクパスの有効なステータス値は、Ok、Fail、Unmonitored、または Unknown です。

ノードの有効なステータス値は、Reboot_on_disk_failure です。Reboot_on_disk_failure ステータスの詳細は、-a オプションと -n オプションを参照してください。

このオプションを使用するには、solaris.cluster.device.read RBAC 認証が必要です。Unresolved link to " rbac5" を参照してください。

-u

ディスクパスのモニターを解除します。各ノードのデーモンは、指定されたパスのモニタリングを停止します。

このオプションは大域ゾーンだけで使用できます。

このオプションを使用するには、solaris.cluster.device.admin RBAC 認証が必要です。Unresolved link to " rbac5" を参照してください。

例 322 クラスタインフラのすべてのディスクパスをモニタリング

次のコマンドでは、クラスタインフラのすべてのディスクパスをモニターします。

```
# scdpm -m all
```

例 323 新しいディスクパスのモニタリング

次のコマンドは、新しいディスクパスをモニターします。すべてのノードは、このパスが有効な /dev/did/dsk/d3 をモニターします。

```
# scdpm -m /dev/did/dsk/d3
```

例 324 単一ノードの新しいディスクパスをモニタリング

次のコマンドでは、単一ノードで新しいパスをモニターします。schost-2 ノードのデーモンは、/dev/did/dsk/d4 ディスクと /dev/did/dsk/d5 ディスクへのパスをモニターします。

```
# scdpm -m schost-2:d4 -m schost-2:d5
```

例 325 すべてのディスクパスとそのステータスの出力

次のコマンドでは、クラスタのすべてのディスクパスとそれらのステータスを出力します。

```
# scdpm -p
     schost-1:reboot_on_disk_failure  enabled
     schost-2:reboot_on_disk_failure  disabled
     schost-1:/dev/did/dsk/d4         Ok
     schost-1:/dev/did/dsk/d3         Ok
     schost-2:/dev/did/dsk/d4         Fail
     schost-2:/dev/did/dsk/d3         Ok
     schost-2:/dev/did/dsk/d5         Unmonitored
     schost-2:/dev/did/dsk/d6         Ok
```

例 326 障害のあるすべてのディスクパスを出力

次のコマンドでは、schost-2 ノードにある障害のあるすべてのディスクパスを出力します。

```
# scdpm -p -F all
     schost-2:/dev/did/dsk/d4         Fail
```

例 327 単一ノードからのすべてのディスクパスのステータスの出力

次のコマンドでは、schost-2 ノードでモニターされているすべてのディスクのディスクパスとそのステータスを出力します。

```
# scdpm -p schost-2:all
     schost-2:reboot_on_disk_failure   disabled
     schost-2:/dev/did/dsk/d4          Fail
     schost-2:/dev/did/dsk/d3          Ok
```

次の終了値が返されます。

`0` コマンドは正常に完了しました。

`1` コマンドは完全に失敗しました。

`2` コマンドは部分的に失敗しました。

注記 - ディスクパスはノード名とディスク名で表されます。ノード名はホスト名または `all` でなければなりません。ディスク名は、グローバルなディスク名、UNIX パス名、または `all` でなければなりません。ディスク名は、グローバルなフルパス名またはディスク名のどちらでもかまいません。たとえば、`/dev/did/dsk/d3` または `d3` などです。ディスク名は、UNIX のフルパス名でもかまいません。たとえば、`/dev/rdsk/c0t0d0s0` などです。

ディスクパスのステータスの変化を記録する場合は、syslogd LOG_INFO 機能レベルを使用します。すべての障害を記録する場合は、LOG_ERR 機能レベルを使用します。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
安定性	発展中

19 ページのIntro(1CL), 61 ページのcldevice(1CL), 185 ページのclnode(1CL), Unresolved link to " attributes5"

Unresolved link to " Oracle Solaris Cluster システム管理 "

名前

sceventmib — Oracle Solaris Cluster イベント MIB モジュールの管理

```
sceventmib -a -c community -h host ...

sceventmib -a -t auth-type -u username [-f password-file]

sceventmib -d -s security-level -u username

sceventmib {-e | -n}

sceventmib -l protocol

sceventmib -m -t auth-type -u username

sceventmib -p {all | hosts | users}

sceventmib -r -c community -h host...

sceventmib -r -u username
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページのIntro(1CL) のマニュアルページを参照してください。

`sceventmib` コマンドは、Oracle Solaris Cluster イベント管理情報ベース (MIB) モジュールを有効化、無効化、および構成します。このコマンドをあるクラスタノード上で実行した場合、そのノード上の MIB モジュールの構成だけに影響します。各クラスタノードの MIB モジュールは、クラスタ内のほかの MIB とは無関係に動作します。

このコマンドは、あるクラスタノード上の MIB モジュールを有効または無効にするのに使用できます。このコマンドは、SNMP トラップ通知のバージョンやトラップ通知を送信する IP アドレスのホスト名など、構成プロパティーを設定するのにも使用できます。Oracle Solaris Cluster イベント MIB は、ポート 11162 にトラップ通知を送信します。SNMP ツリーはポート 11161 に表示されます。

このコマンドは、大域ゾーンだけで使用できます。

基本オプション

次のオプションで、基本的なコマンドの形式および機能を指定します。

-a

指定した SNMP ホストとコミュニティー用のエントリ、または、指定したユーザー用のエントリをノード構成ファイルに追加します。

このオプションは大域ゾーンだけで使用できます。

-d

SNMPv3 プロトコルを指定するときに使用するデフォルトのセキュリティーレベルとユーザーを設定します。

このオプションは大域ゾーンだけで使用できます。

SNMPv3 を指定するときのデフォルトのユーザーを指定してください。SNMPv3 を使用すると、複数のユーザーを MIB モジュール用に構成できます。デフォルトのユーザーは常に 1 人だけです。このオプションの設定にかかわらず、最初に追加したユーザーが自動的にデフォルトのユーザーとして定義されます。この設定では、デフォルトのユーザーは Oracle Solaris OS ユーザーと同じであるというわけではありません。

-e

ノード上の Oracle Solaris Cluster の イベント MIB モジュールを有効にします。ノードをリブートしたあとでも、この設定は変更するまで有効のままです。

このオプションは大域ゾーンだけで使用できます。

-l *protocol*

MIB で使用するために SNMP プロトコルのバージョンを 設定します。

このオプションは大域ゾーンだけで使用できます。

protocol には、SNMPv2 または SNMPv3 のどちらかを指定できます。少なくとも 1 人の SNMPv3 ユーザーを構成するまで、SNMPv3 プロトコルは指定できません。

-m

SNMP ユーザーの認証タイプを変更します。

このオプションは大域ゾーンだけで使用できます。

-n

ノード上の Oracle Solaris Cluster の イベント MIB モジュールを無効にします。ノードをリブートしたあとでも、この設定は変更するまで有効のままです。

このオプションは大域ゾーンだけで使用できます。

-p {all | hosts | users}

MIB 構成情報を次のいずれかのタイプで表示します。

all	すべての MIB モジュール構成情報
hosts	MIB モジュールで使用するように構成されている SNMP ホストの構成情報のみ。

`users`
: MIB モジュールを使用するように構成されている SNMP ユーザーの構成情報のみ。

このオプションは大域ゾーンだけで使用できます。

`-r`
: 指定されている SNMP ホストとコミュニティー用のエントリ、または、指定した SNMP ユーザー用のエントリをノード構成ファイルから削除します。

このオプションは大域ゾーンだけで使用できます。

追加オプション

基本オプションに追加オプションを併用することで、各形式のコマンドのデフォルト動作を変更できます。各形式の `sceventmib` で使用可能なオプションについては、「形式」セクションを参照してください。

サポートされる追加オプションには、次のものがあります。

`-c` *community*
: ノード構成ファイルに追加する、または、ノード構成ファイルから削除する SNMP コミュニティーの名前を指定します。

`-f` *password-file*
: 1 つまたは複数の SNMP ユーザー名とそれに対応するパスワードが含まれるパスワードファイルの名前を指定します。

password-file ファイルに含まれる各行には、次の構文を使用します。

user:*password*

たとえば、Joseph Bloggs と Andrew Smith というユーザーには、次のような行を指定します。

```
jbloggs:fgrxty_0
asmith:artfli!9
```

`-h` *host* …
: SNMP ホストの名前を指定します。*host* 用の IP アドレスまたはホスト名のどちらかを指定できます。

1 つのホストを複数のコミュニティーに含めることができます。しかし、同じコミュニティーに同じ名前のホストがすでに存在する場合、エラーが返されます。

`-s` *security-level*
: 指定した SNMPv3 ユーザーのセキュリティーレベルを指定します。この設定は、ユーザーが SNMP MIB モジュールにアクセスできる程度を決定します。

1 人のユーザーに複数のセキュリティーレベルを割り当てることができます。

security-level の **case-sensitive** 設定には、次の 1 つを指定します。

`authNoPriv` 認証セキュリティー対策は必須ですが、プライバシーセキュリティー対策は必要ありません。

`authPriv` 認証とプライバシーの両方のセキュリティー対策が必要です。

`noAuthNoPriv` 認証とプライバシーのセキュリティー対策は必要ありません。

`-t` *auth-type*

使用する認証暗号化メカニズムを指定します。*auth-type* には、`MD5` または `SHA` のどちらかを指定できます。

`-u` *username*

SNMPv3 ユーザーの名前を指定します。

ユーザーのエントリを追加するときに、同じユーザー名とセキュリティーレベルがすでに存在する場合、その情報は上書きされます。

デフォルトの SNMPv3 ユーザーを削除する場合、このコマンドは自動的に別のデフォルトのユーザーを選択します。

例 328 イベント MIB の有効化

次のコマンドは、イベント MIB を有効にします。

```
# sceventmib -e
```

例 329 SNMP ホストのコミュニティーへの追加

次のコマンドは、ホストを SNMP コミュニティー `public` に追加します。

- 1 番目の例では、ホストをホスト名 `sc-host` で指定します。

```
# sceventmib -a -h sc-host -c public
```

- 2 番目の例では、ホストを IP アドレス `10.0.0.25` で指定します。

```
# sceventmib -a -h 10.0.0.25 -c public
```

例 330 パスワードファイルなしの SNMP ユーザーの追加

次のコマンドは、ユーザー `jbloggs` を追加して、MD5 認証暗号化メカニズムを指定します。パスワードファイルが指定されていないため、このコマンドはユーザーにパスワードを入力するように求めます。

```
# sceventmib -a -t MD5 -u jbloggs
Enter password for user jbloggs: ********
```

例 331 SNMP ユーザーの追加とパスワードファイルの指定

次のコマンドは、ユーザー jbloggs を追加して、MD5 認証暗号化メカニズムとパスワードファイル pfile を指定します。パスワードファイルが指定されているため、このコマンドはユーザーにパスワードを入力するように求めません。

```
# cat pfile
jbloggs:fgrxty_0
# sceventmib -a -f pfile -t MD5 -u jbloggs
```

例 332 すべての SNMP 構成情報の表示

次のコマンドは、すべての SNMP 構成情報を表示します。

```
# sceventmib -p all
```

例 333 SNMP ホストに関する構成情報だけの表示

次のコマンドは、SNMP ホストに関する構成情報だけを表示します。

```
# sceventmib -p hosts
```

例 334 SNMP プロトコルのバージョンの設定

次のコマンドは、SNMP プロトコルのバージョンを SNMPv3 に設定します。

```
# sceventmib -l SNMPv3
```

例 335 SNMP のデフォルトのユーザーの設定

次のコマンドは、SNMP のデフォルトのユーザーを jbloggs に設定して、必要な認証とプライバシの両方のセキュリティ対策を指定します。

```
# sceventmib -d -s authPriv -u jbloggs
```

例 336 ユーザーの認証タイプの変更

次のコマンドは、ユーザー jbloggs の認証タイプを SHA に変更します。

```
# sceventmib -m -t SHA -u jbloggs
```

例 337 SNMP ホストの削除

次のコマンドは、コミュニティ `public` に含まれる IP アドレス `10.0.0.25` のSNMP ホストを削除します。

```
# sceventmib -r -c public -h 10.0.0.25
```

例 338 SNMP ユーザーの削除

次のコマンドは、SNMP ユーザー `jbloggs` を削除します。

```
# sceventmib -r -u jbloggs
```

このコマンドは、次の終了ステータスコードを返します。

`0` コマンドは正常に完了しました。

0 以外 エラーが発生しました。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/system/core`
インタフェースの安定性	発展中

`/usr/cluster/lib/mib/sun-cluster-event-mib.mib`

Oracle Solaris Cluster SNMP イベント MIB 定義ファイル

19 ページの`Intro`(1CL), 491 ページの`clsnmphost`(1CL), 499 ページの`clsnmpmib`(1CL), 509 ページの`clsnmpuser`(1CL), Unresolved link to " attributes5"

Unresolved link to " Oracle Solaris Cluster システム管理 "

名前

scgdevs — 広域デバイス名前空間の管理スクリプト

```
/usr/cluster/bin/scgdevs
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

`scgdevs` コマンドは、広域デバイスの名前空間を管理します。グローバルデバイスの名前空間は物理デバイスへの論理リンクで構成され、`/global` ディレクトリの下にマウントされます。`/dev/global` ディレクトリはクラスタ内の各ノードから見えるので、どのノードからでも個々の物理デバイスが確認できます。つまり、クラスタ内のどのノードからでも、グローバルデバイスの名前空間に追加された任意のディスク、テープ、または CD-ROM にアクセスできます。

`scgdevs` コマンドを使用すれば、システムのリブートなしで、グローバルデバイスの名前空間に新しいグローバルデバイス (テープドライブ、CD-ROM ドライブ、ディスクドライブなど) を付加できます。`scgdevs` コマンドを実行する前に、`devfsadm` コマンドを実行してください。

これらのコマンドを実行しない場合は、再構成リブートにより、広域名前空間を再構成し、新しいグローバルデバイスを付加することができます。再構成リブートの詳細は、Unresolved link to " boot1M" のマニュアルページを参照してください。

このコマンドは、現在のクラスタメンバーであるノードから実行してください。クラスタメンバー以外のノードからこのコマンドを実行した場合、システムの状態は変更されず、このコマンドはエラーコードとともに終了します。

このコマンドは、大域ゾーンだけで使用できます。

このコマンドを使用するためには、`solaris.cluster.system.modify` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

さらにこのコマンドを使用するにあたっては、Oracle Solaris Cluster コマンド権プロファイルを割り当てておく必要があります。認証されたユーザーは、`pfsh`、`pfcsh`、または `pfksh` プロファイルから、Oracle Solaris Cluster の特権コマンドをコマンド行で実行できます。プロファイルシェルは、Oracle Solaris Cluster コマンド権プロファイルに割り当てられた Oracle Solaris

Cluster の特権コマンドへのアクセスを可能にする特別なシェルです。プロファイルシェルは、su コマンドを実行して役割を引き受けたときに起動されます。pfexec コマンドを使用しても、Oracle Solaris Cluster の特権コマンドを実行できます。

次の終了値が返されます。

0	コマンドは正常に完了しました。
0 以外	エラーが発生しました。標準出力にエラーメッセージを出力。

/devices	デバイスノードディレクトリ
/global/.devices	広域デバイスノードディレクトリ
/dev/md/shared	Solaris ボリュームマネージャーのメタセットディレクトリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	廃止

Unresolved link to " pfcsh1"、Unresolved link to " pfexec1"、Unresolved link to " pfksh1"、Unresolved link to " pfsh1"、19 ページの Intro(1CL)、61 ページの cldevice(1CL)、Unresolved link to " boot1M"、Unresolved link to " devfsadm1M"、Unresolved link to " su1M"、1479 ページの did(7)

Unresolved link to " Oracle Solaris Cluster システム管理 "

ローカルノードから呼び出された scgdevs コマンドは、その作業をリモートノード上で非同期的に実行します。したがって、ローカルノード上でコマンドが完了しても、必ずしも、その作業がクラスタ規模で完了しているとは限りません。

このドキュメントは API を構成しません。/global/.devices ディレクトリおよび /devices ディレクトリは、存在しないか、将来のリリースで異なる内容または解釈を持つ可能性があります。

ただし、この記述がないドキュメントすべてが API を構成するわけではありません。このインタフェースは、不安定なインタフェースと見なす必要があります。

名前

scinstall — Oracle Solaris Cluster ソフトウェアの初期化と新しいクラスタノードの確立

```
/usr/cluster/bin/scinstall -i -F [-C clustername]
     [-T authentication-options] [-o]] [-A adapter-options]
     [-B switch-options] [-m cable-options] [-w netaddr-options]

/usr/cluster/bin/scinstall -i -N cluster-member [-C clustername]
     [-A adapter-options] [-B switch-options] [-m cable-options]

/usr/cluster/bin/scinstall -c net-image-source -U password-file
     -h nodename -n nodeip-mac-options -W software-specs -F
     [-C clustername] [-T authentication-options [-A adapter-options]
     [-B switch-options] [-m cable-options] [-w netaddr-options]

/usr/cluster/bin/scinstall -c net-image-source -U password-file
     -h nodename -n nodeip-mac-options -W software-specs
     -N cluster-member [-C clustername] [-A adapter-options]
     [-B switch-options] [-m cable-options]

/usr/cluster/bin/scinstall -c archive=archive-location[::cert=cert-file::
     key=key-file],action=initial -U password-file -h nodename
     -n nodeip-mac-options -F [-C clustername] [-f hostnames-map-file]
     [-T authentication-options] [-A adapter-options]
     [-B switch-options] [-m cable-options] [-o] [-w netaddr-options]

/usr/cluster/bin/scinstall -c archive=archive-location[::cert=cert-file::
     key=key-file],action=initial -U password-file -h nodename
     -n nodeip-mac-options -N cluster-member [-C clustername] [-f hostnames-map-file]
     [-T authentication-options] [-A adapter-options]
     [-B switch-options] [-m cable-options] [-o] [-w netaddr-options]

/usr/cluster/bin/scinstall -c archive=archive-location[::cert=cert-file::
     key=key-file],action=restore -h nodename [-F[-o]]
     -C clustername -n nodeip-mac-options [-T secureAI=yes]

/usr/cluster/bin/scinstall -c archive=archive-location[::cert=cert-file::
     key=key-file],action=replicate -h nodename [-F[-o]]
     -C clustername -n nodeip-mac-options
     [-T node=archive-source-node::node-to-install[,...] [,secureAI=yes]
     [-f hostnames-map-file] [-w netaddr-options] -U password-file

/usr/cluster/bin/scinstall -u upgrade-modes [upgrade-options]

/usr/cluster/bin/scinstall -u update upgrade-options [pkg_fmri_pattern ...]

/usr/cluster/bin/scinstall -r [-N cluster-member]

scinstall -p [-v]
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

`scinstall` コマンドは、Oracle Solaris Cluster ノードの作成およびアップグレードについて、次の各種のタスクを実行します。

- `-scinstall` の「初期化」形式 (`i`) は、新しい Oracle Solaris Cluster の構成メンバーとしてノードを確立します。その際、新規クラスタで最初のノードとして確立 (`-F`) するか、既存のクラスタへノードを追加 (`-N`) することができます。`scinstall` コマンドのこの形式は、常にクラスタを作成するノードまたはクラスタに追加されるノードから実行します。
- `scinstall` の「インストールクライアント追加」形式 (`-c`) は、指定された *nodename* を、コマンドが実行された AI インストールサーバー上のカスタム自動インストーラ (AI) クライアントとして確立します。この形式の `scinstall` コマンドは、常に AI インストールサーバーから実行します。
- `scinstall` の「削除」形式 (`-r`) は、クラスタ構成情報を削除し、クラスタノードから Oracle Solaris Cluster ソフトウェアをアンインストールします。
- `scinstall` の「アップグレード」形式 (`-u`) には複数のモードとオプションがあり、Oracle Solaris Cluster ノードをアップグレードします。`scinstall` コマンドのこの形式は、常にアップグレードするノードから実行します。
- `scinstall` の「リリース出力」形式 (`-p`) は、コマンドを実行したノードにインストールされた Oracle Solaris Cluster ソフトウェアについて、そのリリース情報およびパッケージバージョン情報を出力します。

オプションを指定しない場合、`scinstall` コマンドは対話型モードで実行を試みます。

「リリース出力」形式 (`-p`) を除くすべての `scinstall` コマンド形式は、スーパーユーザーとして実行します。

`ha-cluster/system/install` ソフトウェアパッケージには、`scinstall` コマンドのコピーが含まれています。

このコマンドは、大域ゾーンからのみ実行できます。

基本オプション

次のオプションで、基本的なコマンドの形式および機能を指定します。

ただし、複数のオプションを同一のコマンド行で指定できません。

-c

「インストールクライアント追加」形式の `scinstall` コマンドを指定します。このオプションは、指定された *nodename* を、コマンドが発行された AI サーバー上のカスタム自動インストーラ (AI) クライアントとして確立します。この `-c` オプションは、2 つの仕様 `-c` *net-image-source* および `-c archive=`*archive-location*`[::cert=`*cert-file*`::key=`*key-file*`],action={`*initial/restore|replicate*`}` を受け入れます。

このオプションは大域ゾーンだけで使用できます。

AI を使用して、IPS リポジトリから Oracle Solaris および Oracle Solaris Cluster ソフトウェアパッケージをインストールし、新しいクラスタを構成する場合は、*net-image-source* を指定する必要があります。これは、クラスタノードのアーキテクチャ (SPARC または i386) に基づき、`install-image` または `solaris-auto-install` IPS パッケージを取得するリポジトリとすることができます。

`-c publisher=`*repo*`[::cert=`*cert-file*`=`*key-file*`],arch={`*sparc|i386*`}`

net-image-source は、Oracle Solaris リリースの AI ISO イメージファイルとすることもできます。ファイルは、クラスタノードをインストールするように構成された、すでに確立された AI サーバーからアクセスできる必要があります (`-c` *iso-file*)。

統合アーカイブを使用して自動的にクラスタをインストールしたり、クラスタノードを復元したりする場合は、`archive=`*archive-location*`,action={initial|restore|replicate}` コマンドを使用してください。このコマンドは、統合アーカイブの場所を指定し、AI サーバーからアクセス可能なファイルシステム上のアーカイブへのフルパスか、HTTP または HTTPS の場所とすることができます。HTTPS の場所にアクセスする場合は、SSL 鍵および証明書ファイルを指定する必要があります。また、新しいクラスタを構成する (`action=initial`)、ノードを復元する (`action=restore`)、または同じハードウェア構成を持つ既存のクラスタから新しいクラスタをレプリケートする (`action=replicate`) 場合は、アーカイブの使用目的も指定する必要があります。`restore` アクションを使用する場合、アーカイブは、復元するノードと同じノード上で以前作成された復旧タイプのアーカイブでなければなりません。

この形式のコマンドを使用すると、各クラスタノード (つまり *nodename*) を、すでに確立された自動インストーラインストールサーバー上のカスタム AI クライアントとして容易に確立できるため、AI サーバーからの完全に自動化されたクラスタインストールが可能になります。

Oracle Solaris Cluster の場合、AI マニフェストファイルをカスタマイズできます。Unresolved link to " Oracle Solaris Cluster ソフトウェアのインストール の Oracle Solaris および Oracle Solaris Cluster ソフトウェアをインストールおよび構成する方法 (IPS リポジトリ)"および Unresolved link to " Oracle Solaris 11.2 システムのインストール "を参照してください。

scinstall コマンドを使用してカスタム Oracle Solaris Cluster AI クライアントとしてノードを設定する前に、まず AI インストールサーバーを確立する必要があります。AI インストールサーバーの設定の詳細は、Unresolved link to " Oracle Solaris 11.2 システムのインストール の第 8 章AI サーバーの設定"を参照してください。

-i

「初期化」形式の scinstall コマンドを指定します。この形式のコマンドは、新しいクラスタメンバーとしてノードを確立します。新規のノードは、scinstall コマンドを発行したノードとなります。

このオプションは大域ゾーンだけで使用できます。

-i オプションに -F オプションを併用すると、scinstall は新規クラスタの最初のノードを確立します。

-F オプションに -o オプションを併用すると、scinstall は単一ノードクラスタを確立します。

-i オプションに -N オプションを併用すると、scinstall は既存のクラスタにノードを追加します。

-p

コマンドを実行したノードにインストールされた Oracle Solaris Cluster ソフトウェアのリリース情報およびパッケージバージョン情報を出力します。これは、スーパーユーザー以外で実行できる唯一の scinstall の形式です。

このオプションは大域ゾーンだけで使用できます。

-r

クラスタノードから、クラスタの構成情報を削除し、Oracle Solaris Cluster のフレームワークおよびデータサービスソフトウェアをアンインストールします。この処理の実行後は、ノードを再インストールしたり、クラスタからノードを削除することができます。このコマンドは、アンインストールするノード上のクラスタソフトウェアによって使用されていないディレクトリから実行してください。ノードは、非クラスタモードとします。

このオプションは大域ゾーンだけで使用できます。

-u

scinstall コマンドを起動したノード上の Oracle Solaris Cluster ソフトウェアをアップグレードします。scinstall の「アップグレード」形式には複数の動作モードがあり、*upgrade-mode* で指定されます。アップグレードタイプの詳細は、次の **アップグレードオプション** を参照してください。

このオプションは大域ゾーンだけで使用できます。

追加オプション

基本オプションに追加オプションを併用することで、各形式のコマンドのデフォルト動作を変更できます。各形式の scinstall コマンドで使用可能なオプションについては、「SYNOPSIS」セクションを参照してください。

サポートされる追加オプションには、次のものがあります。

-h *nodename*

ノード名を指定します。-h オプションが利用できるのは、「インストールクライアント追加」(-c) 形式のコマンドのみです。

nodename は、カスタム AI インストールのために設定するクラスタノード (つまり、AI インストールクライアント) の名前です。

-v

リリース情報を冗長モードで出力します。-v オプションは、「リリース出力」(-p) 形式のコマンドで冗長モードを指定する場合にのみ使用します。

「リリース出力」の冗長モードでは、インストールされている各 Oracle Solaris Cluster ソフトウェアパッケージのバージョン文字列も出力されます。

-F [*config-options*]

クラスタ内の最初のノードを確立します。-F オプションは、「初期化」(-i) または「インストールクライアント追加」(-c) 形式のコマンドでのみ有効です。

最初のノードのインストールが完全にクラスタメンバーとしてインスタンス化され、新規クラスタノードの追加に必要なすべての処理を実行できる状態になるまでは、2 番目以降のノードは確立できません。-F オプションとともに -o オプションを使用すると、単一ノードクラスタが作成され、このクラスタ作成中は、新規にノードを追加できません。

-f *hostnames-map-file*

別のクラスタからクラスタをレプリケートしたり、initial アクションで復旧アーカイブを使用して新しいクラスタを作成したりするために使用する古いホスト名と新しいホスト名の組み合わせリストを含むテキストファイルを指定します。このファイルには、複数の行を含めることができます。各行は 2 列で構成されています。最初の列は、アーカイブが作成されるソースクラスタで使用されるホスト名または IP アドレスです。2 列目は、新しいクラスタに対応するホスト名または IP アドレスです。これらのホスト名は、論理ホスト名、共有アドレスリソース、およびゾーンクラスタ用に使用できます。

source-cluster-zc-hostname1	*target-cluster-zc-hostname1*
source-cluster-zc-hostname2	*target-cluster-zc-hostname2*
source-cluster-lh1	*target-cluster-lh1*
source-cluster-lh2	*target-cluster-lh2*

このオプションは大域ゾーンだけで使用できます。

-N *cluster-member* [*config-options*]

クラスタ番号を指定します。-N オプションは、「初期化」(-i)、「インストールクライアント追加」(-c)、または「削除」(-r) 形式のコマンドでのみ有効です。

-i または -c オプションとともに使用される場合、-N オプションは、既存のクラスタにノードを追加するために使用されます。通常、*cluster-member* にはクラスタに対して確立される最初のクラスタノードの名前を指定します。ただし、*cluster-member* は、すでにクラスタメ

ンバーとして参加している任意のクラスタノードの名前を指定します。初期化されるノードは、*cluster-member* がすでに有効なメンバーであるクラスタに追加されます。既存のクラスタへの新規ノードの追加処理では、新規ノードのローカルファイルシステムへの構成データベースのコピーに加え、指定された *cluster-member* 内の構成データベースが更新されます。

-r オプションにこの -N オプションを併用する場合は、クラスタのアクティブなメンバー以外の任意のノードを *cluster-member* に指定できます。scinstall コマンドは、指定された *cluster-member* にアクセスしてクラスタ構成の更新を行います。-N オプションが指定されないと、scinstall は既存のノードを探してアクセスを試みます。

構成オプション

config-options は -F オプションとともに使用します。

```
/usr/cluster/bin/scinstall{-i | -c net-image-source -U password-file -h nodename -n nodeip-
mac-options -W software-spec} -F [-C clustername] [-T authentication-options] [-A adapter-options]
 [-B switch-options] [-m endpoint=[this-node]:name[@port],endpoint= [node:]name[@port]] [-o]
 [-w netaddr-options]
```

```
/usr/cluster/bin/scinstall {-i | -c net-image-source -U password-file -h nodename -n nodeip-
mac-options -W software-spec} -N cluster-member [-C clustername] [-A adapter-options] [-B switch-
options] [-m endpoint=cable-options]
```

-m *cable-options*

クラスタインターコネクトを指定します。このオプションが利用できるのは、-F または -N オプションを指定する場合だけです。

-m オプションは、クラスタトランスポートのアダプタおよびスイッチに存在する各種ポートのケーブル接続を構成して、クラスタインターコネクトのトポロジ確立を支援します。この形式のコマンドで新規にケーブルを構成すると、現在のノード上のクラスタトランスポートアダプタからクラスタトランスポートスイッチ上のポートへの接続、またはクラスタ内に存在するほかのノード上のアダプタへの接続が確立されます。

-m オプションを指定しない場合、scinstall コマンドはデフォルトケーブルの構成を試みます。ただし、scinstall の指定インスタンスで 2 つ以上のトランスポートアダプタまたはスイッチを構成した場合、scinstall でデフォルトは構成されません。デフォルトでは、シングル構成のトランスポートアダプタからシングル構成 (またはデフォルト) のトランスポートスイッチに対して、1 つのケーブルが構成されます。

-m *cable-options* は次のように指定します。

```
-m endpoint=[this-node]:
name[@port],endpoint=[
node:]name[@port]
```

-m オプションの構文では、2 つの終端の少なくとも 1 つを、構成しているノード上のアダプタとしてください。この終端については、明示的に *this-node* を指定する必要はありません。次にケーブル追加のサンプルコードを示します。

```
-m endpoint=:net1,endpoint=switch1
```

この例では、このノード (scinstall が構成しているノード) 上の net1 トランスポートアダプタのポート 0 が、トランスポートスイッチ switch1 上のポートにケーブル接続されます。switch1 上で使用されるポート番号は、デフォルトでこのノードのノード ID 番号になります。

1 つの -m オプションに対して、常に 2 つの endpoint オプションを指定する必要があります。オプション引数の *name* コンポーネントは、ケーブルの一方の終端の、クラスタトランスポートアダプタまたはクラスタトランスポートスイッチの名前を指定します。

- *node* コンポーネントを指定すると、*name* はトランスポートアダプタの名前になります。
- *node* コンポーネントを指定しないと、*name* はトランスポートスイッチの名前になります。

port コンポーネントを指定しない場合、scinstall コマンドはデフォルトポート名の使用を試みます。アダプタのデフォルト *port* は、常に 0 になります。スイッチの終端の場合、ポートの *name* のデフォルト値は、クラスタに追加するノード ID と等しくなります。

*ポート*の割り当てやその他の要件の詳細は、119 ページのclinterconnect(1CL) のマニュアルページを参照してください。

ケーブルを構成する前に、ケーブルの 2 つの終端でアダプタやスイッチを構成しておく必要があります (-A および -B を参照)。

-n *nodeip-mac-options*

ノードの IP アドレスと MAC アドレスを指定します。このオプションは、-c オプションも指定されている場合にのみ有効です。

-n *nodeip-mac-options* の構文は次のとおりです。

```
-n ip=node-ipaddr/N,mac=
mac-address
```

-o

単一ノードクラスタの構成を指定します。このオプションが利用できるのは、-i および -F オプションを指定する場合だけです。

その他の -F オプションはサポートされていますが、必須ではありません。クラスタ名が指定されないと、ノードの名前がクラスタ名として使用されます。CCR に格納されるトランスポート構成オプションを指定できます。単一ノードクラスタが確立されたあとは、定足数デバイスを構成したり、installmode を無効にしたりする必要はありません。

-w *netaddr-options*

プライベートインターコネクト、つまり、クラスタトランスポートのネットワークアドレスを指定します。このオプションが利用できるのは、-F オプションを指定する場合だけです。

このオプションは、プライベートインターコネクトで使用するプライベートネットワークアドレスを指定するために使用します。このオプションは、デフォルトのプライベートネットワークアドレスが、すでにネットワークで使用中のアドレスと競合する場合に使用できます。このオプションは、プライベートインターコネクト用に予約されている IP アドレス範囲のサイズをカスタマイズするのにも使用できます。詳細は、Unresolved link to " networks4" および Unresolved link to " netmasks4" のマニュアルページを参照してください。

指定しない場合、プライベートインターコネクトのデフォルトのネットワークアドレスは 172.16.0.0 です。デフォルトのネットマスクは 255.255.240.0 です。この IP アドレス範囲は、最大 62 のノード、10 個のプライベートネットワーク、12 のゾーンクラスタ、および 3 つの排他的 IP ゾーンクラスタをサポートします。

-w *netaddr-options* は次のように指定します。

```
–w netaddr=netaddr[,netmask=
netmask]
–w netaddr=netaddr[,maxnodes=
nodes,maxprivatenets=maxprivnets,\
numvirtualclusters=zoneclusters, numxipvirtualclusters=
xipzoneclusters]
–w netaddr=netaddr[,netmask=
netmask,maxnodes=nodes,\maxprivatenets=
maxprivnets\
,numvirtualclusters=zoneclusters]
```

netaddr=*netaddr*

プライベートネットワークアドレスを指定します。このアドレスの末尾 2 つのオクテットは、常に 0 にする必要があります。

[netmask=*netmask*]

ネットマスクを指定します。指定した値は、デフォルト以上の IP アドレス範囲に一致している必要があります。

デフォルト未満の IP アドレス範囲を割り当てるには、maxnodes、maxprivatenets、および numvirtualclusters オペランドを指定します。

[,maxnodes=*nodes*,maxprivatenets= *maxprivnets*,numvirtualclusters=*zoneclusters*]

クラスタに含まれると予想されるノード、プライベートネットワーク、およびゾーンクラスタの最大数を指定します。このコマンドは指定された値を使用して、プライベートインターコネクトが指定された数のノード、プライベートネットワーク、およびゾーンクラスタをサポートするのに必要な最小のネットマスクを計算します。*nodes* の最大値は 62 で、最小値は 2 です。*maxprivnets* の最大値は 128 です。最小値は 2 です。*zoneclusters* には、値として 0 を設定できます。

[,netmask=*netmask*,maxnodes= *nodes*,maxprivatenets=*maxprivnets*¥
,numvirtualclusters=*zoneclusters*]

クラスタに含まれると予想されるノード、プライベートネットワーク、およびゾーンクラスタのネットマスクと最大数を指定します。ネットマスクには、指定した *nodes* 数、*privnets* 数、*zoneclusters* 数を十分にカバーできるものを指定する必要があります。*nodes* の最大値は 62 で、最小値は 2 です。*privnets* の最大値は 128 です。最小値は 2 です。*zoneclusters* には、値に 0 を設定できます。

netaddr サブオプションだけを指定する場合、このコマンドは 255.255.240.0 というデフォルトのネットマスクを割り当てます。この IP アドレス範囲には、最大で 62 個のノード、10 個のプライベートネットワーク、および 12 個のゾーンクラスタが含まれます。

クラスタの確立後にプライベートネットワークアドレスまたはネットマスクを変更するには、cluster コマンドまたは clsetup ユーティリティーを使用します。

-A *adapter-options*

トランスポートアダプタと、オプションでトランスポートタイプを指定します。このオプションが利用できるのは、-F または -N オプションも指定する場合のみです。

1 つの -A オプションで、scinstall コマンドを実行するノードに接続したクラスタトランスポートアダプタを 1 つ構成できます。

-A オプションが指定されない場合は、デフォルトのアダプタおよびトランスポートタイプが使用されます。デフォルトのトランスポートタイプは dlpi です。SPARC プラットフォームでは、デフォルトのアダプタは hme1 です。

アダプタのトランスポートタイプが dlpi であれば、trtype サブオプションを指定する必要はありません。この場合、次の 2 つのいずれかの形式で -A *adapter-options* を使用できます:

```
-A [trtype=type,]name=adaptername[,vlanid=
vlanid][,other-options]
-A adaptername
```

[trtype=*type*]

アダプタのトランスポートタイプを指定します。trtype オプションは、指定するアダプタのトランスポートタイプの -A オプションごとに使用します。トランスポートタイプ type の例は dlpi です。

デフォルトのトランスポートタイプは dlpi です。

name=*adaptername*

アダプタ名を指定します。-A オプションごとに name サブオプションを使用して、*adaptername* を指定する必要があります。*adaptername* は 1 つの *device name* から構成され、その直後に 1 つの *physical-unit* 番号が続きます (hme0 など)。

-A オプションにサブオプションを指定しない場合は、*adaptername* を -A オプションのスタンドアロン引数として指定できます (つまり、-A *adaptername* となります)。

vlanid=*vlanid*

タグ付けされた VLAN アダプタの VLAN ID を指定します。

[*other-options*]

追加のアダプタオプションを指定します。ある特定のアダプタがほかのオプションを提供する場合、これらのオプションは -A オプションから指定することができます。

-B *switch-options*

トランスポートスイッチ (トランスポート接続点とも呼ぶ) を指定します。このオプションが利用できるのは、-F または -N オプションを指定する場合だけです。

1 つの -B オプションで、クラスタトランスポートスイッチを 1 つ構成できます該当するデバイスとしては、Ethernet ハブなどの各種タイプのスイッチやリンクなどがありますが、これだけに限られるわけではありません。

-B オプションを指定しない場合、scinstall は、最初のノードがクラスタノードとしてインスタンス化される時点でデフォルトのスイッチを追加します。クラスタにノードを追加すると、スイッチをデフォルトで追加されません。ただし、明示的に追加することはできます。デフォルトのスイッチの名前は switch1 で、そのタイプは switch です。

スイッチのタイプが switch であれば、type サブオプションを指定する必要はありません。この場合、-B *switch-options* を指定するには、次の 2 つの形式のいずれかを使用できます。

```
-B [type=type,]name=name[,
other-options]
-B name
```

指定されたスイッチ *name* にクラスタトランスポートスイッチがすでに構成されている場合、scinstall はメッセージを出力し、-B オプションを無視します。

直接ケーブル接続されたトランスポートアダプタを使用する場合は、トランスポートスイッチを構成する必要はありません。トランスポートスイッチのデフォルト構成を回避するために、次の特別な -B オプションを使用します。

```
-B type=direct
```

[type=*type*]

クラスタトランスポートスイッチのタイプを指定します。type オプションは、-B オプションごとに使用できます。Ethernet スイッチは、スイッチタイプが switch のクラスタトランスポートスイッチの例です。詳細は、119 ページのclinterconnect(1CL) のマニュアルページを参照してください。

type サブオプションを direct に設定すると、すべてのデフォルトスイッチの構成を無効にできます。直接接続方式のトランスポートアダプタだけで転送を行う構成には、スイッチはありません。type サブオプションを direct に設定する場合、name サブオプションを使用する必要はありません。

name=*name*

クラスタトランスポートスイッチの名前を指定します。type が direct である場合を除き、-B オプションごとに name サブオプションを使用して、トランスポートスイッチ *name* を指定する必要があります。*name* は最大 256 文字の英字または数字で構成でき、最初の文字は英字にします。個々のトランスポートスイッチの名前は、クラスタの名前空間全体で一意でなければなりません。

-B でその他のサブオプションが必要なければ、スイッチの *name* を -B のスタンドアロン引数として指定できます (つまり、-B *name* となります)。

[*other-options*]

追加のトランスポートスイッチオプションを指定します。ある特定のスイッチタイプがほかのオプションを提供する場合、これらのオプションは -B オプションで指定することができます。スイッチで使用する可能性のある特殊なオプションについては、119 ページのclinterconnect(1CL) のマニュアルページを参照してください。

-C *clustername*

クラスタの名前を指定します。このオプションが利用できるのは、-F または -N オプションを指定する場合だけです。

- 構成するノードが新しいクラスタ内の最初のノードの場合、デフォルトの *clustername* は、構成しているノードの名前と同じです。
- 構成するノードが既存のクラスタに追加される場合、デフォルトの *clustername* は、*cluster-member* が所属しているクラスタの名前になります。

clustername に対して *cluster-member* が所属していないクラスタ名を指定すると、エラーになります。

-T *authentication-options*

クラスタのノード認証オプションを指定します。このオプションが利用できるのは、-F オプションを指定する場合だけです。

このオプションは、クラスタ構成への追加を試みるノードに対して、認証ポリシーを確立する際に使用します。特に、クラスタにクラスタノードとしてノードを追加するようマシンが要求している場合は、ノードが参加する権利を有しているかどうかの判定が行われます。権利を有している場合、このノードは認証されてクラスタへの参加が許可されます。

-T オプションの使用は、クラスタ内の最初のノードを設定するときに、scinstall コマンドとともに使用する場合に限られます。すでに確立されたクラスタ上で認証リストまたはポリシーを変更する必要がある場合は、claccess コマンドを使用します。

デフォルトでは、任意のマシンがクラスタに追加できます。

-T *authentication-options* は次のように指定します。

```
-T node=nodename[,…][,authtype=authtype][,secureAI=yes]
-T node=archive-source-node::node-to-install[,…][,authtype=authtype][,secureAI=yes]
-T secureAI=yes
```

node=*nodename*[,…]

ノード名を指定してノード認証リストに追加します。レプリケートするすべてのノードについて、ノードのペアを指定する必要があります。-T オプションに追加する node サブオプションは、最低 1 つ指定する必要があります。このオプションは、クラスタ内のノードとして構成が可能なノードリストに、ノード名を追加するために使用されます。認証リストが空であれば、どのノードもクラスタ構成への追加要求を行うことができます。ただし、リストに 1 つでも名前が登録されていれば、こうした要求のすべてに対して、認証リストによる確認が行われます。このノードのリストは、いずれかのアクティブなクラスタノードから claccess コマンドまたは clsetup ユーティリティーを使用していつでも変更またはクリアできます。

node=*archive-source-node*::*node-to-install*[,…]

`node=`*archive-source-node*`::`*node-to-install* オプションは、ノード名のペアを指定します。最初のノード名は、アーカイブが作成されるノードで、2 番目のノード名は、そのアーカイブからインストールする新規クラスタのノードです。この指定は、別のクラスタ上に作成されたアーカイブからクラスタをレプリケートする場合のみ使用し、新規クラスタノードは、アーカイブが作成されるソースクラスタノードと同じハードウェア構成 (またはスーパーセット) を持つ必要があります。

[`authtype=`*authtype*]

scinstall クラスタ構成の初期化 `-i` 形式か、*net-image-source* を使用するときのインストールクライアントの追加 `-c` 形式についてのノード認証のタイプを指定します。現在サポートされている `authtype` は、`des` および `sys` (または `unix`) のみです。`authtype` を指定しない場合、デフォルトで `sys` が使用されます。

`des` (Diffie-Hellman) 認証を指定する場合、`-T` オプションを `scinstall` コマンドで実行する前に、まず追加する各クラスタノードの Unresolved link to " publickey4" データベースにエントリを追加します。

認証タイプは、いずれかのアクティブなクラスタノードから `claccess` コマンドまたは `clsetup` ユーティリティーを使用していつでも変更できます。

[`secureAI=yes`]

AI でセキュアインストールを使用することと、それが AI を使用してクラスタソフトウェアをインストールしている場合にのみ有効であることを指定します。`secureAI=yes` を指定しない場合、デフォルトのアクションでは従来の AI インストールが実行されます。セキュアインストール方式を使用してアーカイブからノードを復元するとき、`-T secureAI=yes` のみを指定する必要があります。`node=nodename[,...]` および `authtype=authtype` は不要です。

`-U` *password-file*

root ユーザーのパスワードを含むファイルの名前を指定します。このオプションは、`-c` オプションも指定されている場合にのみ有効です。

このオプションにより、Oracle Solaris の初期インストールおよび構成時の root パスワードの自動設定が可能になります。ユーザーは、インストールされているシステムの root ユーザーのパスワードとして使用するテキストを含むファイルを作成します。通常、*password-file* は、*nodename* インストールクライアントをインストールするように構成された、すでに確立された AI インストールサーバー上に存在するか、またはそのサーバーからアクセス可能です。`scinstall` ユーティリティーはこのファイルの内容を取得し、それを Oracle Solaris 構成ユーティリティーに提供します。

`-W` *software-specs*

1 つ以上のパブリッシャーとパッケージリポジトリの場所を指定します。また、AI を使用したセキュアインストールに必要な公開鍵と SSL 証明書の情報も指定します。このオプションは、IPS リポジトリからインストールするために、`-c` オプションが指定されている場合にのみ有効です。

-W *software-specs* は次のように指定します。

```
-W publisher=
repo[::key=
key-file::cert=certificate-file] \
::pkg[,…][:::
publisher=repo[::key=key-file::cert=
certificate-file]::pkg[,…]]…
```

読みやすさのために -W オプションが複数行に分かれていますが、1 つの連続した文字列で指定するようにしてください。

-W オプションの構文では、*publisher* はパブリッシャー名 `ha-cluster` または `solaris`、*repo* はリポジトリの場所、*key-file* と *certificate-file* は HTTPS リポジトリからのセキュアインストールに必要な公開鍵と SSL 証明書の情報、*pkg* はソフトウェアパッケージ名です。

セキュアな HTTPS リポジトリを使用して Oracle Solaris または Oracle Solaris Cluster をインストールするには、公開鍵と SSL 証明書の情報を指定する必要があります。公開鍵と SSL 証明書は、`http://pkg-register.oracle.com` のサイトにリクエストし、そこからダウンロードできます。

アップグレードオプション

標準 (順次以外) アップグレード、順次アップグレード、およびデュアルパーティションアップグレードのための -u *upgrade-modes* および *upgrade-options* は次のとおりです。

標準 (順次以外) および順次アップグレード

クラスタノードを標準 (順次以外) または順次アップグレードモードでより新しいリリースの Oracle Solaris Cluster ソフトウェアにアップグレードするには、`-u update` モードを使用します。

- 標準 (または順次以外) アップグレードプロセスでは、クラスタノードが引き続きクラスタ要求に対応している間に、アクティブでないブート環境 (BE) がアップグレードされます。既存のアクティブでない BE を指定しない場合は、`scinstall` ユーティリティーによって新しい BE が自動的に作成されます。アップグレードが完了すると、`scinstall` ユーティリティーはアップグレードされた BE をアクティブにしたあと、そのノードをアップグレードされた BE にリブートすることをユーザーに通知します。
- 順次アップグレードプロセスでは、一度に 1 つのクラスタノードだけが稼働を停止されます。このプロセスは、Oracle Solaris または Oracle Solaris Cluster ソフトウェア、あるいはその両方を、すでにインストールされているバージョンの更新リリースにアップグレードするためにのみ使用できます。1 つのノードをアップグレードしている間、クラスタサービスは残りのクラスタノードで継続されます。ノードをアップグレードしたら、そのノードをクラスタに戻して、次にアップグレードするノードでプロセスを繰り返します。すべてのノードをアップグレードした

ら、1 つのクラスタノードで scversions コマンドを実行して、クラスタをアップグレードされたバージョンにコミットする必要があります。このコマンドを実行するまで、更新リリースで導入された新しい機能の一部を使用できない場合があります。

- オプションで、現在のイメージにすでにインストールされているパッケージ FMRI を指定できます。

標準および順次モード用の -u update の *upgrade-options* は次のとおりです。

```
/usr/cluster/bin/scinstall -u update [-b be-name] [-L {accept | licenses | accept,licenses
 | licenses,accept}] [pkg_fmri_pattern ...]
```

-b *be-name*

新しいブート環境 (BE) に割り当てる名前を指定します。このオプションを指定しない場合は、scinstall によって新しい BE の名前が割り当てられます。この名前は、*currentBE*-*N* という形式の現在の BE の名前に基づいており、接尾辞 -*N* は増分される数字です。最初の新しい BE には *currentBE*-1、次の新しい BE には *currentBE*-2 という名前が付けられ、以降も同様です。BE が削除されたとき、サフィックスの数字がそれより大きな BE 名が存在する場合は、次の新しい BE にその名前は再利用されません。たとえば、sc4.0、sc4.0-1、および sc4.0-2 という BE が存在するときに、sc4.0-1 が削除された場合、次の新しい BE には sc4.0-3 という名前が付けられます。

すでに存在する BE 名を指定した場合、このコマンドはエラーで終了します。

-L {accept | licenses | accept,licenses | licenses,accept }

アップグレードするパッケージのライセンスを受け入れるか表示するか (あるいはその両方) を指定します。

accept 引数は pkg コマンドの --accept オプションに対応し、licenses 引数は --licenses オプションに対応します。

-L accept オプションの指定は、更新されるパッケージのライセンスを受け入れることに同意することを示します。このオプションを指定しないと、パッケージライセンスで受け入れが必要な場合は、更新操作が失敗します。

-L licenses を指定すると、更新されるパッケージのすべてのライセンスが表示されます。

accept および licenses の両方に -L オプションを指定すると、更新されるパッケージのライセンスが表示され、受け入れられます。accept および licenses 引数を指定する順序は、コマンドの動作に影響を与えません。

scinstall -u update コマンドは、更新するパッケージの *pkg_fmri_patterns* を指定する機能をサポートします。

[*pkg_fmri_pattern*...]

更新するパッケージを指定します。これらのパッケージは現在のイメージにインストールする必要があります。指定された *pkg_fmri_pattern* パターンの 1 つがアスタリスク (*) である場合、現在のイメージ内のすべてのインストール済みパッケージを更新できます。

デュアルパーティションアップグレード

デュアルパーティションアップグレードの複数の段階を実行するには、デュアルパーティションアップグレードのための -u *upgrade-modes* および *upgrade-options* を使用します。デュアルパーティションアップグレードでは、まずクラスタノードが 2 つのグループ (つまり、パーティション) に割り当てられます。次に、一方のパーティションがクラスタサービスを提供している間に、もう一方のパーティションをアップグレードします。その後、アップグレード済みのパーティションにサービスを切り替え、残りのパーティションをアップグレードし、アップグレード済みの第 1 パーティションから成るクラスタに第 2 パーティションのアップグレード済みノードを再結合します。デュアルパーティションアップグレードの *upgrade-modes* には、デュアルパーティションアップグレード中に障害が発生したあとの回復のモードも含まれています。

デュアルパーティションアップグレードのモードは、`-u update` アップグレードモードとともに使用されます。詳細は、Unresolved link to " Oracle Solaris Cluster Upgrade Guide "を参照してください。

デュアルパーティションアップグレードのための -u の *upgrade-modes* および *upgrade-options* は次のとおりです。

```
/usr/cluster/bin/scinstall -u begin -h nodelist

/usr/cluster/bin/scinstall -u plan

/usr/cluster/bin/scinstall -u recover

/usr/cluster/bin/scinstall -u status

/usr/cluster/bin/scinstall -u apply

/usr/cluster/bin/scinstall -u status
```

`apply`

パーティションのアップグレードが完了したことを指定します。この形式のコマンドは、パーティションのすべてのノードがアップグレードされたあと、そのアップグレード済みパーティションの任意のノードから実行します。

`apply` アップグレードモードは次のタスクを実行します。

第 1 パーティション

第 1 パーティションのノードから `apply` アップグレードモードを実行すると、第 1 パーティションのすべてのノードが新しいソフトウェアを実行するように準備されます。

第 1 パーティションのノードでクラスタサービスをサポートする準備ができたら、このコマンドは第 2 パーティションのノード上にあるスクリプト `/etc/cluster/ql/cluster_pre_halt_apps` および `/etc/cluster/ql/cluster_post_halt_apps` をリモートで実行します。これらのスクリプトは、Resource Group Manager (RGM) で

管理されていない Oracle Real Application Clusters (Oracle RAC) などのアプリケーションを停止するユーザー製のスクリプトを呼び出すために使用されます。

- `cluster_pre_halt_apps` スクリプトは、RGM で管理されているアプリケーションが停止される前に実行されます。
- `cluster_post_halt_apps` スクリプトは、RGM で管理されているアプリケーションが停止されたあと、ただしノードが停止される前に実行されます。

注記 - `apply` アップグレードモードを実行する前に、必要に応じてスクリプトテンプレートを変更して、ノード上の特定のアプリケーションを停止するために作成したほかのスクリプトを呼び出すようにします。変更したスクリプトとそれらが呼び出すユーザー製のスクリプトを、第 1 パーティションの各ノードに配置します。これらのスクリプトは、第 1 パーティションの任意の 1 つのノードから実行されます。第 1 パーティションの複数のノードで実行されているアプリケーションを停止するには、ユーザー製のスクリプトを適宜変更します。未変更のスクリプトで実行されるデフォルトのアクションはありません。

第 2 パーティションのアプリケーションがすべて停止されたあと、このコマンドは第 2 パーティションのノードを停止します。シャットダウンにより、第 1 パーティションのノードへ、アプリケーションとデータサービスのスイッチオーバーが開始されます。その後、このコマンドは第 2 パーティションのノードをクラスタモードでブートします。

リソースグループのノードリストに第 1 パーティションのメンバーだけが含まれており、そのためリソースグループがオフラインであった場合、リソースグループはオンラインに戻ります。第 1 パーティションに属するノードがリソースグループのノードリストに含まれていない場合、リソースグループはオフラインのままになります。

第 2 パーティション

第 2 パーティションのノードから `apply` アップグレードモードを実行すると、第 2 パーティションのすべてのノードが新しいソフトウェアを実行するように準備されます。その後、このコマンドはノードをクラスタモードでブートします。第 1 パーティションのノードから成るアクティブなクラスタに、第 2 パーティションのノードが再結合されます。

リソースグループのノードリストに第 2 パーティションのメンバーだけが含まれており、そのためリソースグループがオフラインであった場合、リソースグループはオンラインに戻ります。

すべてのノードがクラスタに再結合されたあと、このコマンドは最終処理を実行し、定足数デバイスを再構成し、定足数投票数を復元します。

`begin`

アップグレードする第 1 パーティションに割り当てるノードを指定し、デュアルパーティションアップグレードプロセスを開始します。この形式のコマンドはクラスタの任意のノードから実行します。このアップグレードモードは、`plan` アップグレードモードを使用して可能なパーティションスキームを判定したあとで使用します。

まず、`begin` アップグレードモードは各パーティションに割り当てるノードを記録します。次に、このアップグレードモードは 1 つのノード上のアプリケーションをすべて停止し、そのノードを

シャットダウンします。シャットダウンにより、リソースグループのノードリストに含まれている第 2 パーティションに属するノードへ、ノードの各リソースグループのスイッチオーバーが開始されます。第 2 パーティションに属するノードがリソースグループのノードリストに含まれていない場合、リソースグループはオフラインのままになります。

その後、このコマンドは第 1 パーティションの残りの各ノードで、一度に 1 ノードずつ、この一連のアクションを繰り返します。

第 1 パーティションのアップグレードの間、第 2 パーティションのノードは稼働状態のままです。ノード上の定足数デバイスは一時的に構成解除され、定足数投票数は一時的に変更されます。

`plan`

クラスタのストレージ構成を照会し、共有ストレージの要件を満たす可能なパーティションスキームをすべて表示します。この形式のコマンドはクラスタの任意のノードから実行します。これはデュアルパーティションアップグレードで最初に実行するコマンドです。

デュアルパーティションアップグレードでは、各パーティションの少なくとも 1 つのノードが各共有ストレージアレイに物理的にアクセスする必要があります。

`plan` アップグレードモードは、ゼロ、1、または複数のパーティションソリューションを返します。ソリューションが返されない場合、クラスタ構成はデュアルパーティションアップグレードに適していません。代わりに標準アップグレード方法を使用してください。

どのパーティションソリューションでも、アップグレードする第 1 パーティションとしてどちらかのパーティショングループを選択できます。

`recover`

デュアルパーティションアップグレードの処理中に致命的エラーが発生した場合、ノードのクラスタ構成を回復します。この形式のコマンドはクラスタの各ノードで実行します。

このコマンドを実行する前に、クラスタをシャットダウンし、すべてのノードを非クラスタモードでブートする必要があります。

いったん致命的エラーが発生したら、`recover` アップグレードモードを実行したあとでも、デュアルパーティションアップグレードを再開したり再度開始したりすることはできません。

`recover` アップグレードモードは、デュアルパーティションアップグレードを開始する前に、クラスタ構成リポジトリ (CCR) データベースを元の状態に戻します。

次のリストでは、どのような状況で `recover` アップグレードモードを使用し、どのような状況でほかの手順を使用するかについて説明します。

- `-u begin` 処理中に障害が発生した場合は、`-u recover` アップグレードモードを実行します。
- `-u begin` 処理の完了後、ただし第 2 パーティションのシャットダウン警告が発行される前に障害が発生した場合は、エラーが発生した場所を特定します。
  - 障害が第 1 パーティションのノードで発生した場合は、`-u recover` アップグレードモードを実行します。

- 障害が第 2 パーティションのノードで発生した場合は、回復アクションは必要ありません。
- 第 2 パーティションのシャットダウン警告が発行されたあと、ただし第 2 パーティションに対する -u apply 処理が開始する前に障害が発生した場合は、エラーが発生した場所を特定します。
  - 障害が第 1 パーティションのノードで発生した場合は、-u recover アップグレードモードを実行します。
  - 障害が第 2 パーティションのノードで発生した場合は、障害の発生したノードを非クラスタモードでリブートします。
- 第 2 パーティションに対する -u apply 処理の完了後、ただしアップグレードが完了する前に障害が発生した場合は、エラーが発生した場所を特定します。
  - 障害が第 1 パーティションのノードで発生した場合は、-u recover アップグレードモードを実行します。
  - 障害が第 1 パーティションのノードで発生した場合は、第 1 パーティションが稼働状態のままであれば、障害の発生したノードをリブートします。
  - 障害が第 2 パーティションのノードで発生した場合は、-u recover アップグレードモードを実行します。

どの場合でも、標準アップグレード方法を使用してアップグレードを手動で続行することはでき、その方法ではすべてのクラスタノードをシャットダウンする必要があります。

status

デュアルパーティションアップグレードのステータスを表示します。可能な状態は、次のとおりです。

アップグレードが進行中です

scinstall -u begin コマンドが実行されましたが、デュアルパーティションアップグレードは完了していません。

デュアルパーティションアップグレードの間に致命的エラーが発生した場合も、クラスタはこのステータスを報告します。この場合、回復手順を実行し、標準アップグレード方法を使用してクラスタのアップグレードを完了したあとも、この状態はクリアされません

アップグレードは進行中ではありません

scinstall -u begin コマンドがまだ発行されていないか、デュアルパーティションアップグレードが正常に完了しています。

status アップグレードモードはクラスタの 1 つのノードから実行します。ノードは、クラスタモードでも非クラスタモードでもかまいません。

発行するノードがデュアルパーティションアップグレードのどの段階にあるかにかかわらず、報告される状態はクラスタのすべてのノードについて有効です。

デュアルパーティションアップグレードモードでは次のオプションがサポートされています。

-h *nodelist*
: 第 1 パーティションに割り当てるすべてのノードをスペースで区切ったリストを指定します。`plan` アップグレードモードで、使用するパーティションスキームにおけるパーティションの有効なメンバーとして表示される出力から、これらを選択します。`begin` アップグレードモードに指定しない、クラスタの残りのノードは、第 2 パーティションに割り当てられます。

  このオプションは、`begin` アップグレードモードでのみ有効です。

2 ノードクラスタの確立

次の例では、SPARC ベースのプラットフォーム上に、Oracle Solaris 11 向けの Oracle Solaris Cluster ソフトウェアを使用した標準的な 2 ノードクラスタが確立されます。この例では、Oracle Solaris Cluster ソフトウェアパッケージがすでにノードにインストールされていることを前提とします。

`node1` 上で、次のコマンドを発行します。

```
node1# /usr/cluster/bin/scinstall -i -F
```

`node2` 上で、次のコマンドを発行します。

```
node2# /usr/cluster/bin/scinstall -i -N node1
```

単一ノードクラスタの確立

次のコマンドでは、すべてのデフォルト設定を受け入れて、SPARC ベースのプラットフォーム上に、Oracle Solaris 11 向けの Oracle Solaris Cluster ソフトウェアを使用した単一ノードクラスタが確立されます。この例では、Oracle Solaris Cluster ソフトウェアパッケージがすでにノードにインストールされていることを前提とします。

```
# /usr/cluster/bin/scinstall -i -F -o
```

AI サーバー上のネットイメージ ISO ファイルを使用したインストールクライアントの追加

次の例では、2 ノードクラスタ内の SPARC ベースのプラットフォーム上に、Oracle Solaris 11 向けの Oracle Solaris Cluster ソフトウェアをインストールおよび初期化するための AI インストールサーバーが設定されます。

インストールサーバー上で、次のコマンドを発行します。読みやすさのために `-W` オプションが複数行に分かれていますが、1 つの連続した文字列で指定するようにしてください。

```
# usr/cluster/bin/scinstall –c /export/home/11-ga-ai-x86.iso -h phys-schost-1 \
-U /export/pwdfile \
-C schost \
-F \
-W solaris=http://ipkg.us.oracle.com/solaris11/release::\
entire,server_install:::ha-cluster=cluster-repository::\
ha-cluster-framework-full,ha-cluster-data-services-full,
ha-cluster-geo-full \
-n ip=10.255.85.163/24,mac=12:34:56:78:90:ab \
-T node=phys-schost-1,node=phys-schost-2,authtype=sys \
-w netaddr=172.16.0.0,netmask=255.255.240.0,maxnodes=62,\
maxprivatenets=10,numvirtualclusters=12,numxipvirtualclusters=3 \
-A trtype=dlpi,name=e1000g1 -A trtype=dlpi,name=nxge1 \
-B type=switch,name=switch1 -B type=switch,name=switch2 \
-m endpoint=:e1000g1,endpoint=switch1 \
-m endpoint=:nge1,endpoint=switch2

# usr/cluster/bin/scinstall –c /export/home/11-ga-ai-x86.iso -h phys-schost-2 \
-U /export/pwdfile \
-C schost \
-N phys-schost-1 \
-W solaris=http://ipkg.us.oracle.com/solaris11/release::\
entire,server_install:::ha-cluster=cluster-repository::\
ha-cluster-framework-full,ha-cluster-data-services-full,\
ha-cluster-geo-full \
-n ip=10.255.85.164/24,mac=12:34:56:78:90:ab \
-A trtype=dlpi,name=e1000g1 -A trtype=dlpi,name=nxge1 \
-m endpoint=:e1000g1,endpoint=switch1 \
-m endpoint=:nge1,endpoint=switch2
```

デュアルパーティションアップグレードの実行

次の例では、デュアルパーティション方法を使用して、クラスタのフレームワークおよびデータサービスソフトウェアを、新規リリースの Oracle Solaris Cluster にアップグレードします。この例では、SPARC ベースのプラットフォーム上の Solaris 11 向けの Oracle Solaris Cluster バージョンを使用します。この例では、クラスタに有効なパーティションスキームを照会し、ノードをパーティションに割り当て、第 1 パーティションのノードをリブートし、アップグレード後に第 1 パーティションを稼働状態に戻して第 2 パーティションのノードをリブートし、アップグレード後に第 2 パーティションをクラスタに戻します。

```
# /usr/cluster/bin/scinstall -u plan
  Option 1
    First partition
      phys-schost-1
    Second partition
      phys-schost-2
…
# /usr/cluster/bin/scinstall -u begin -h phys-schost-1 phys-schost-3

ok boot -x
```

(Upgrade the node in the first partition)

```
phys-schost-1# /usr/cluster/bin/scinstall -u apply
ok boot -x
```

(Upgrade the node in the second partition)

```
phys-schost-2# /usr/cluster/bin/scinstall -u apply
```

フレームワークおよびデータサービスソフトウェアのアップグレード (標準または順次アップグレード)

次の例では、クラスタのフレームワークおよびデータサービスソフトウェアを、新規リリースの Oracle Solaris Cluster にアップグレードします。次の操作をそれぞれのクラスタノードで実行します。

注記 - 順次アップグレードの場合、`clnode evacuate` コマンドを使用して、すべてのリソースグループとデバイスグループをクラスタ内に残っているほかのノードに移動させたあと、これらの操作を 1 つのノードで一度に実行します。

```
# /usr/cluster/bin/scinstall -u update
# init 6
```

アーカイブファイルから最初のノードの復元

次の例は、セキュアな AI インストールを使用して、AI サーバーからアクセス可能なファイルシステム上に保存されているアーカイブファイルから、最初のノードを復元します。

```
# /usr/cluster/bin/scinstall -c archive=file:///net/storagenode/export/archive
     /phys-schost-1-recovery-archive,action=restore \
-h phys-schost-1 \
-C schost =\
-F \
-n ip=10.255.85.163/24,mac=12:34:56:78:90:ab \
-T secureAI=yes
```

アーカイブからの他のノードの復元

次の例は、セキュアな AI インストールを使用して、別のノード上で以前作成されたアーカイブから、それらのノードを復元します。

```
# /usr/cluster/bin/scinstall -c archive=file:///net/storagenode/export/archive
     /phys-schost-2-recovery-archive,action=restore \
-h phys-schost-2 \
-C schost =\
-n ip=10.255.85.164/24,mac=12:34:56:78:90:cd \
-T secureAI=yes
```

セキュアでないレプリケーションの実行

次の例は、セキュアでないレプリケーションを実行します。

```
# /usr/cluster/bin/scinstall -c archive=file:///net/storagenode/export/archive
     /source-node-1-archive,action=replicate \
-h phys-schost-1 \
-C schost \
-F \
-n ip=10.255.85.163/24,mac=12:34:56:78:90:ab \
-T node=phys-schost-1,node=phys-schost-2,secureAI=yes \
-U /export/pwdfile

# /usr/cluster/bin/scinstall -c archive=file:///net/pnass3/export/archive
     /vzono1a.clone,action=replicate \
-h phys-schost-2 \
-C schost \
-n ip=10.255.85.164/24,mac=12:34:56:78:90:cd \
-U /export/pwdfile
```

AI サーバー上の IPS リポジトリを使用したインストールクライアントの追加

次の例は、セキュアな AI インストールを使用して、IPS リポジトリから 2 ノードの x86 クラスタをインストールして構成します。

```
# /usr/cluster/bin/scinstall -c solaris=http://ipkg.us.oracle.com/solaris11
     /release::arch=i386 -h phys-schost-1 \
-C schost \
-F \
-W solaris=http://ipkg.us.oracle.com/solaris11/release::entire,server_install:::
     ha-cluster=http://ipkg.us.oracle.com/ha-cluster/release::ha-cluster-framework-full \
-n ip=10.255.85.163/24,mac=12:34:56:78:90:ab \
-T node=phys-schost-1,node=phys-schost-2,authtype=sys,secureAI=yes \
-w netaddr=172.16.0.0,netmask=255.255.240.0,maxnodes=32,maxprivatenets=10,
     numvirtualclusters=12,numxipvirtualclusters=3 \
-A trtype=dlpi,name=net1 -A trtype=dlpi,name=net3 \
-B type=switch,name=switch1 -B type=switch,name=switch2 \
-m endpoint=:net1,endpoint=switch1 \
-m endpoint=:net3,endpoint=switch2 \
-P task=quorum,state=INIT -P task=security,state=SECURE \
-U /export/pwdfile

# /usr/cluster/bin/scinstall -c solaris=http://ipkg.us.oracle.com/solaris11
     /release::arch=i386 -h phys-schost-2 \
-C schost \
-N phys-schost-1 \
-W solaris=http://ipkg.us.oracle.com/solaris11/release::entire,server_install:::
     ha-cluster=http://ipkg.us.oracle.com/ha-cluster/release::ha-cluster-framework-full  \
-n ip=10.255.85.164/24,mac=12:34:56:78:90:ab \
-A trtype=dlpi,name=net1 -A trtype=dlpi,name=net3 \
-m endpoint=:net1,endpoint=switch1 \
```

```
-m endpoint=:net3,endpoint=switch2 \
-U /export/pwdfile
```

次の終了値が返されます。

`0` 正常終了。

`0` 以外 エラーが発生しました。

`/etc/cluster/ql/cluster_post_halt_apps`

`/etc/cluster/ql/cluster_pre_halt_apps`

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/system/install`
インタフェースの安定性	発展中

19 ページの`Intro`(1CL), 37 ページの`claccess`(1CL), 119 ページの`clinterconnect`(1CL), 185 ページの`clnode`(1CL), 487 ページの`clsetup`(1CL), 551 ページの`cluster`(1CL), Unresolved link to " newfs1M", 905 ページの`scversions`(1M), Unresolved link to " netmasks4", Unresolved link to " networks4", Unresolved link to " lofi7D"

Unresolved link to " Oracle Solaris Cluster ソフトウェアのインストール ", Unresolved link to " Oracle Solaris Cluster システム管理 ", Unresolved link to " Oracle Solaris Cluster Upgrade Guide "

名前

scnas — Oracle Solaris Cluster 用のネットワーク接続ストレージ (NAS) デバイス構成データを管理する

```
scnas [-H]

scnas -a [-H] [-n] -h device-name -t device-type -o
     specific-options [-f input-file]

scnas -c [-H] [-n] -h device-name -o specific-options [-f
     input-file]

scnas -p [-H] [-h device-name] [-t device-type]

scnas -r [-H] -h device-name
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

`scnas` コマンドは、Oracle Solaris Cluster 構成内の NAS デバイスを管理します。クラスタ内の NAS ディレクトリを管理する場合は、`scnasdir` コマンドを使用します。

`scnas` コマンドでは、NAS デバイス構成を作成したり、NAS タイプ固有のプロパティーを更新したり、デバイス構成を Oracle Solaris Cluster から削除したりすることができます。このコマンドのオプションは、コマンド行に入力された順に処理されます。

なお `scnas` コマンドは、有効なクラスタノードからのみ実行できます。コマンドを実行して得られる結果は、実行するノードに関係なく、常に同じです。

`-H` オプションは、`scnas` コマンドのすべての形式で使用できます。`-H` を指定すると、ヘルプ情報が表示されます。それ以外のオプションはすべて無視されます。それ以外のオプションはすべて無視されます。ヘルプ情報は、オプションを指定せずに `scnas` を実行した場合も出力されます。

`scnas` コマンドで NAS デバイスを管理する前に、NAS デバイスが設定されている必要があります。デバイスのセットアップ手順については、NAS デバイスのドキュメントを参照してください。

このコマンドは、大域ゾーンだけで使用できます。

基本オプション

次のオプションは、すべての形式の scnas コマンドに共通です。

-H

このオプションをコマンド行のいずれかの場所に指定すると、ヘルプ情報が出力されます。他のすべてのオプションは無視され、実行されません。このヘルプ情報は、オプションを指定せずに scnas を実行した場合も出力されます。

このオプションは大域ゾーンだけで使用できます。

-n

このオプションがコマンド行のどこかに指定されていると、scnas コマンドは使用法をチェックするだけで、構成データは書き込みません。-n オプションが -f オプションとともに指定されると、scnas コマンドは、入力ファイルにパスワードが含まれているかどうかを確認します。

次のオプションは、scnas コマンドの基本的な形式および機能を変更するものです。これらのオプションは、1 つのコマンド行で複数を同時使用できません。

-a

add 形式の scnas コマンドを指定します。

このオプションは大域ゾーンだけで使用できます。

-a オプションは、NAS デバイスを Oracle Solaris Cluster 構成に追加するときに使用できます。

NAS デバイスのタイプによって、追加のプロパティーを設定する必要がある場合があります。このような追加のプロパティーについても、「追加オプション」セクションの -t オプションの説明を参照してください。

-c

change 形式の scnas コマンドを指定します。-c オプションは、NAS デバイスの特定のプロパティーを変更するときに使用します。

このオプションは大域ゾーンだけで使用できます。

-r

remove 形式の scnas コマンドを指定します。-r オプションは、NAS デバイスを Oracle Solaris Cluster 構成から削除するときに使用します。

このオプションは大域ゾーンだけで使用できます。

デバイスを削除する前には、エクスポートされているディレクトリをすべて、scnasdir を使用して削除する必要があります。

-p

print 形式の scnas コマンドを指定します。

このオプションは大域ゾーンだけで使用できます。

オプションがほかに指定されていない場合、-p オプションは、Oracle Solaris Cluster 内に現在構成されているすべての NAS デバイスと関連するすべてのプロパティーを出力します。このオプションをほかのオプションとともに使用すると、特定のデバイスや特定のタイプのデバイスを照会できます。

追加オプション

上で述べた基本オプションの 1 つ以上と次の追加オプションを組み合わせて使用することで、デバイスのすべてのプロパティーを構成できます。これらのオプションを使用するときに、デバイスがオンラインである必要はありません。scnas の各形式で使用できるオプションについては、形式セクションを参照してください。

使用可能なオプションを次に示します。

-h *device-name*

Oracle Solaris Cluster 構成に属する NAS デバイスの名前を指定するときに使用します。デバイス名はデバイスを識別するものであり、このデバイス名を rhs または telnet で使用すると、そのデバイスにリモートからアクセスできます。

このデバイス名は、add、change、および remove形式の scnas コマンドに指定する必要があります。

-t *device-type*

NAS デバイスタイプ。このオプションは、NAS デバイスを Oracle Solaris Cluster 構成に追加するときに指定してください。NAS デバイスタイプは、ベンダー名で指定します。

Oracle の Sun ZFS Storage Appliance には sun_uss を指定できます。

異なるタイプの NAS デバイスには異なるプロパティーがあるか、またはプロパティーがまったくない場合もあります。

-o *specific-options*

NAS デバイスタイプ固有のプロパティーを指定するときに使用します。

userid プロパティーは、クラスタがデバイス上で管理作業をするときに使用します。userid をデバイス構成に追加するときには、対応するパスワードを入力するように求められます。パスワードは、テキストファイルに格納しておき、-f オプションで指定してもかまいません。

-f *input-file*

セキュリティー上の理由により、パスワードはコマンド行オプションには指定できません。パスワードのセキュリティーを強化するには、パスワードをテキストファイルに格納し、-f オプションでこのファイルを指定してください。パスワード用の入力ファイルを指定しないと、パスワードの入力が求められます。

入力ファイルの読み取り可能アクセス権を root だけに与え、このファイルにグループからも一般ユーザーからもアクセスできないようにします。

入力ファイルの複数行に渡ってパスワードを入力できません。行頭にあるブランクやタブは無視されます。コメントは引用されていないシャープ記号 (#) から始まり、次の行に続きます。

すべてのコメントはパーサーによって無視されます。デバイスユーザーのパスワードに入力ファイルを使用する場合は、# 符号をパスワードの一部に使用できません。

例 339 NAS デバイスをクラスタに追加する

次の scnas コマンドは、Oracle Solaris Cluster 構成に NAS デバイスを追加します。

```
# scnas -a -h sun_uss1 -t sun
```

例 340 NAS デバイスをクラスタに追加する

次の scnas コマンドは、Oracle Solaris Cluster 構成にストレージシステムを追加します。

```
# scnas -a -h sun_uss1 -t sun_uss -o userid=root
Please enter password:
```

例 341 NAS デバイスのクラスタからの削除

次の scnas コマンドは、Oracle Solaris Cluster 構成から NAS デバイスを削除します。

```
# scnas -r -h sun_uss1
```

次の終了値が返されます。

0 コマンドは正常に終了。

0 以外 エラーが発生しました。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
安定性	廃止

19 ページのIntro(1CL), 169 ページのclnasdevice(1CL), 239 ページのclquorum(1CL), 551 ページのcluster(1CL), 747 ページのscconf(1M), 849 ページのscnasdir(1M)

名前

scnasdir — では、Oracle Solaris Cluster 構成のネットワーク接続ストレージ (NAS) デバイスにエクスポートされたディレクトリを管理します。

```
scnasdir [-H]

scnasdir [-a] [-H] [-n] -h device-name [-d directory [-d
     directory…]] [-f input-file]

scnasdir -p [-H] [-h device-name] [-t device-type]

scnasdir -r [-H] [-n] -h device-name [-d all | -d directory [-d
     directory…]] [-f input-file]
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページのIntro(1CL) のマニュアルページを参照してください。

---

scnasdir コマンドは、Oracle Solaris Cluster 構成内の NAS デバイスでエクスポートされたディレクトリを管理します。クラスタ内の NAS ディレクトリを管理する場合は、scnas コマンドを使用します。

scnasdir コマンドでは、NAS デバイス構成を作成したり、NAS タイプ固有のプロパティーを更新したり、デバイス構成を Sun Cluster から削除したりすることができます。

このコマンドのオプションは、コマンド行に入力された順に処理されます。なお scnasdir コマンドは、有効なクラスタノードからのみ実行できます。コマンドを実行して得られる結果は、実行するノードに関係なく、常に同じです。

-H オプションは、scnasdir コマンドのすべての形式で使用できます。-H を指定すると、ヘルプ情報が表示されます。それ以外のオプションはすべて無視されます。ヘルプ情報は、オプションを指定せずに scnasdir を実行した場合も出力されます。

このコマンドは、大域ゾーンだけで使用できます。

基本オプション

次のオプションは、すべての形式の scnasdir コマンドに共通です。

-H

このオプションをコマンド行のいずれかの場所に指定すると、ヘルプ情報が出力されます。他のすべてのオプションは無視され、実行されません。このヘルプ情報は、オプションを指定せずに scnasdir を実行した場合も出力されます。

このオプションは大域ゾーンだけで使用できます。

-n

このオプションがコマンド行のどこかに指定されていると、scnasdir コマンドは使用法をチェックするだけで、構成データは書き込みません。-n オプションが -f オプションとともに指定されると、scnasdir コマンドは、入力ファイルにパスワードが含まれているかどうかを確認します。

次のオプションは、scnasdir コマンドの基本的な形式および機能を変更するものです。これらのオプションは、1 つのコマンド行で複数を同時使用できません。

-a

追加形式の scnasdir コマンドを指定します。-a オプションは、Oracle Solaris Cluster 構成にディレクトリを追加するために使用できます。

このオプションは大域ゾーンだけで使用できます。

-p

出力形式の scnasdir コマンドを指定します。オプションがほかに指定されていない場合、-p オプションは、Oracle Solaris Cluster 内に構成されているすべての NAS デバイスのすべてのディレクトリのリストを出力します。このオプションは追加のオプションと併用して、特定のデバイスまたは特定のタイプの NAS デバイスをクエリーできます。

このオプションは大域ゾーンだけで使用できます。

-r

削除形式の scnasdir コマンドを指定します。-r オプションは、Oracle Solaris Cluster 構成から、NAS デバイスのすべてのディレクトリまたは指定したディレクトリを削除するために使用します。

このオプションは大域ゾーンだけで使用できます。

追加オプション

上で述べた基本オプションの 1 つ以上と次の追加オプションを組み合わせて使用することで、デバイスのさまざまなディレクトリを管理できます。

使用可能なオプションを次に示します。

-h *device-name*

Oracle Solaris Cluster 構成に属する NAS デバイスの名前を指定するときに使用します。-h オプションはデバイスを識別するものであり、rhs または telnet で使用すると、そのデバイスにリモートからアクセスできます。

このデバイス名は、追加、変更、および削除の形式の scnasdir コマンドで使用します。

-d *all* | *directory*

このオプションは、NAS デバイスでエクスポートされており、Oracle Solaris Cluster に構成されるディレクトリ (または ボリューム) のリストを出力するために使用します。これらのディレクトリは、scnasdir コマンドを使用する前に、作成およびエクスポートしておく必要があります。ディレクトリのエクスポート手順については、NAS デバイスタイプのドキュメントを参照してください

-d all オプションは、削除オプションの -r でのみ使用できます。

これらのディレクトリは、追加および削除の形式の scnasdir コマンドの -d オプションまたは -f オプションのどちらかを使用して指定する必要があります。

-f *input-file*

ディレクトリは、テキストファイルに 1 行に 1 ディレクトリで登録して、このファイルを -f オプションで指定します。行頭にあるブランクやタブは無視されます。コメントは引用されていないシャープ記号 (#) から始まり、次の行に続きます。すべてのコメントはパーサーによって無視されます。

例 342 2 つの NAS ストレージデバイスディレクトリをクラスタに追加する

次の scnasdir コマンドは、NAS デバイスの 2 つのディレクトリを Oracle Solaris Cluster 構成に追加します。

```
# scnasdir -a -h sunuss1 -d /vol/DB1 -d /vol/DB2
```

例 343 NAS ストレージデバイスのすべてのディレクトリをクラスタから削除する

次の scnasdir コマンドは、NAS デバイスに構成されているすべてのディレクトリを削除します。

```
# scnasdir -r -h sunuss1 -d all
```

次の終了値が返されます。

0 コマンドは正常に終了。

0 以外 エラーが発生しました。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	廃止

19 ページのIntro(1CL), 169 ページのclnasdevice(1CL), 239 ページのclquorum(1CL), 551 ページのcluster(1CL), 747 ページのscconf(1M), 843 ページのscnas(1M)

名前

scprivipadm — プライベート IP アドレス範囲の管理

```
scprivipadm -c netaddr=netaddr[,netmask=netmask]

scprivipadm -c netaddr=netaddr[,maxnodes=nodes,maxprivatenets=
     privnets]

scprivipadm -c netaddr=netaddr[,netmask=netmask,maxnodes=nodes,maxprivatenets=
     privnets]

scprivipadm -p

scprivipadm -R
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページのIntro(1CL) のマニュアルページを参照してください。

---

scprivipadm コマンドは、Oracle Solaris Cluster プライベートインターコネクトに割り当てられている現在の IP アドレス範囲を変更します。

このコマンドの任意の形式を実行する前に、クラスタ内のすべてのノードを非クラスタモードにしなければなりません。このコマンドは、クラスタ内の 1 つのノードから実行します。

scprivipadm コマンドは、入力としてプライベートネットワークアドレスを取ります。また、オプションとして次のいずれか 1 つまたは両方を取ります。

- ネットマスク
- クラスタに予想されるノードの最大数とプライベートネットワークの最大数

次に、このコマンドは、物理アダプタとノードごとの IP アドレスの IP アドレス割り当てを実行します。

このコマンドは、大域ゾーンだけで使用できます。

次のオプションがサポートされています。

-c
: 現在クラスタに割り当てられている IP アドレス範囲を変更します。-c オプションはクラスタの各ノードで実行します。
このオプションは大域ゾーンだけで使用できます。

-c オプションは、次のサブオプションをサポートしています。

`netaddr=`*netaddr*	プライベートネットワークアドレスを指定します。
`netmask=`*netmask*	ネットマスクを指定します。
`maxnodes=`*nodes*	クラスタに予想されるノードの最大数を指定します。
`maxprivatenets=`*privnets*	クラスタに予想されるプライベートネットワークの最大数を指定します。

-c オプションは、サブオプションの各組み合わせに対して次のタスクを実行します。

- `netaddr` サブオプションだけを指定した場合、このコマンドは、デフォルトのネットマスク `255.255.248.0` をプライベートインターコネクトに割り当てます。デフォルトの IP アドレス範囲は、最大 64 のノードと 10 のプライベートネットワークを格納します。
- `netmask` サブオプションも指定する場合は、デフォルトのネットマスク以上の値を指定する必要があります。指定されたネットマスクがデフォルトのネットマスク未満である場合、このコマンドは失敗し、エラーで終了します。指定されたネットマスクがデフォルトのネットマスク以上の場合、このコマンドは指定されたネットマスクをプライベートインターコネクトに割り当てます。結果の IP アドレス範囲は、最大 64 個のノードと最大 10 個のプライベートネットワークをサポートできます。

  デフォルトより小さい IP アドレス範囲を割り当てるには、`maxnodes` および `maxprivatenets` サブオプションを指定します。
- `maxnodes` および `maxprivatenets` サブオプションも指定した場合、このコマンドは、指定された数のノードとプライベートネットワークをサポートするための最小のネットマスクを計算します。このコマンドは次に、計算したネットマスクをプライベートインターコネクトに割り当てます。*nodes* の最大値は 64 です。最小値は 2 です。*privnets* の最大値は 128 です。最小値は 2 です。
- `maxnodes` および `maxprivatenets` サブオプションだけでなく `netmask` サブオプションも指定した場合、このコマンドは、指定された数のノードとプライベートネットワークをサポートする最小のネットマスクを計算します。このコマンドは、計算結果と指定されたネットマスク

を比較します。指定されたネットマスクが計算結果のネットマスク未満である場合、このコマンドは失敗し、エラーで終了します。指定されたネットマスクが計算結果のネットマスク以上の場合、このコマンドは指定されたネットマスクをプライベートインターコネクトに割り当てます。*nodes* の最大値は 64 です。最小値は 2 です。*privnets* の最大値は 128 です。最小値は 2 です。

-c オプションが失敗した場合、-R オプションを各ノードで実行して、構成を修復してから、-c オプションをふたたび実行してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` 役割に基づくアクセス制御 (RBAC) 認証が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

-R

クラスタ構成を修復します。このオプションは、このコマンドがクラスタノード上で IP アドレス範囲を変更しようとして失敗し、結果として、ノード上のクラスタ構成に整合性がなくなったときに使用します。

このオプションは大域ゾーンだけで使用できます。

-R オプションはクラスタの各ノードで実行します。

-R オプションは、すべてのノード上で IP アドレス範囲を変更しようとして失敗した場合に、クラスタ構成を修復して、非整合性を取り除きます。

-R オプションを実行せずに -c オプションを実行し直そうとすると、構成の変更は失敗する可能性があります。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.modify` 役割に基づくアクセス制御 (RBAC) 認証が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

-p

プライベートインターコネクトに割り当てられている現在のプライベートネットワークアドレスを表示します。-p オプションは任意のノードから実行します。

このオプションは大域ゾーンだけで使用できます。

-p オプションは次の情報を出力します。

- プライベートネットワークアドレス
- ネットマスク形式の IP アドレス範囲
- IP アドレス範囲がサポートできるノードの最大数とプライベートネットワークの最大数

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.read` 役割に基づくアクセス制御 (RBAC) 認証

が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

クラスタモードのノードから現在のプライベートネットワークアドレスを表示するには、代わりに、`scconf -p` コマンドまたは `cluster show-netprops` コマンドを実行します。

例 344 独自のプライベート IP アドレス範囲の計算

次のコマンドは、プライベートネットワークアドレス `172.16.0.0` を指定して、ネットマスクを計算します。このコマンドは、計算されたネットマスクが最大 16 個のノードと最大 4 個のプライベートネットワークをクラスタ内でサポートする必要があることを指定します。

```
# scprivipadm -c
 netaddr=172.16.0.0,maxnodes=16,maxprivatenets=4
```

例 345 プライベートネットワークアドレスとネットマスクの指定

次のコマンドは、プライベートネットワークアドレス `172.16.0.0` とネットマスク `255.255.248.0` を指定します。

```
# scprivipadm -c
 netaddr=172.16.0.0,netmask=255.255.248.0
```

次のいずれかの条件が発生した場合、`scprivipadm` コマンドはゼロ以外の値を返します。

- 無効な引数が指定されました。
- このコマンドは、クラスタの一部のノードで、IP アドレス範囲を正常に変更できませんでした。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/system/core`
インタフェースの安定性	廃止

19 ページの`Intro`(1CL), 551 ページの`cluster`(1CL), 747 ページの`scconf`(1M), 819 ページの`scinstall`(1M), Unresolved link to " netmasks4", Unresolved link to " networks4", Unresolved link to " rbac5"

Unresolved link to " Oracle Solaris Cluster ソフトウェアのインストール ", Unresolved link to " Oracle Solaris Cluster システム管理 ",『Oracle Solaris ネットワークの構成と管理』

スーパーユーザーはこのコマンドのすべての形式を実行できます。スーパーユーザー以外のユーザーには、RBAC の承認が必要です。次の表を参照してください。

オプション	RBAC の承認
-c	solaris.cluster.modify
-R	solaris.cluster.modify
-p	solaris.cluster.read

名前

scprivipd — Oracle Solaris Cluster プライベート IP アドレスサービスデーモン

```
/usr/cluster/lib/sc/scprivipd
```

scprivipd デーモンは、システムのブート時に起動されます。これは、ゾーンのブートまたはシャットダウン時、あるいは、scconf 操作の結果として、割り当てられたプライベート IP アドレスを構成または構成解除するのに使用されます。

scprivipd デーモンは、stdin (標準入出)、stdout (標準出力)、または stderr (標準エラー出力) と外部との直接接続はありません。すべての診断メッセージは、syslog 関数を使用して記録されます。

scprivipd デーモンはスーパーユーザーモードで実行してください。

scprivipd デーモンは Service Management Facility (SMF) サービスであり、SMF 経由で起動されます。また、scprivipd デーモンがシグナルによって強制終了された場合、SMF によって自動的に再起動されます。

SIGTERM シグナルを使用することで、scprivipd は正常に終了させることができます。同デーモンを終了させる場合、その他のシグナルは使用しないでください。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	プライベート

185 ページのclnode(1CL), 747 ページのscconf(1M), Unresolved link to " syslog3C", Unresolved link to " attributes5"

名前

scrgadm — リソースタイプ、リソースグループ、およびリソースの登録と登録解除を管理する

```
scrgadm -p[v[v]] [-t resource_type_name] [-g resource_group_name]
     [-j resource_name]

scrgadm -a -t resource_type_name [-h RT_installed_node_list] [-f
     registration_file_path]

scrgadm -a -g RG_name [-h nodelist] [-y property=value…]

scrgadm -a -j resource_name -t resource_type_name -g RG_name [-y
     property=value…] [-x "extension_property[{node_specifier}]=
     value…"]

scrgadm -a -L -g RG_name -l hostnamelist [-j resource_name] [-n
     netiflist] [-y property=value…]

scrgadm -a -S -g RG_name -l hostnamelist [-j resource_name] [-n
     netiflist] [-X auxnodelist] [-y property=value…]

scrgadm -c -t resource_type_name [-h RT_installed_node_list]
     [-y RT_system={TRUE|FALSE}]

scrgadm -c -g RG_name [-h nodelist] -y property=value…

scrgadm -c -j resource_name [-y property…] [-x "
     extension_property[{node_specifier}]=value…"]

scrgadm -r -t resource_type_name

scrgadm -r -g RG_name

scrgadm -r -j resource_name
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページのIntro(1CL) のマニュアルページを参照してください。

リソース型は、その型のすべてのリソースに共通のプロパティーとコールバックメソッドを指定します。ただし、特定のリソース型を作成する場合は、まず次の形式のコマンドを使用してリソースタイプを登録する必要があります。

```
# scrgadm -a -
t resource_type_name
```

リソースグループには、一連のリソースが含まれており、これらすべてのリソースは指定のノードまたはノード群で共にオンラインまたはオフラインになります。リソースを配置する前に、空のリソースグループを作成します。リソースグループを作成するには、次のコマンドを使用します。

```
# scrgadm -a -
g RG_name
```

リソースグループには、2 種類あります。フェイルオーバーとスケーラブルです。

フェイルオーバーリソースグループは、常に単一のノード上でオンラインになります。フェイルオーバーリソースグループには、あらゆるタイプのリソースを含めることができます。ただし、フェイルオーバーリソースグループに構成されたスケーラブルリソースは、常に単一のノード上で実行されます。

MyDatabaseRG という名前のフェイルオーバーリソースグループを作成するには、次のコマンドを使用します。

```
# scrgadm -a -
g MyDatabaseRG
```

スケーラブルリソースグループは、同時に複数のノード上でオンラインになります。スケーラブルリソースグループに含めることができるリソースは、スケーリングをサポートするリソースだけです。スケーラブルリソースグループには、リソースタイプの定義によって使用がフェイルオーバー動作に限定されるリソースを含めることはできません。

MyWebServerRG という名前のスケーラブルリソースグループを作成するには、次のコマンドを使用します。

```
# scrgadm -a -
g MyWebServerRG \
-y Maximum_primaries=integer \
-y Desired_primaries=integer
```

新しく作成したリソースグループの状態は UNMANAGED です。リソースをグループに作成したあとは、scswitch コマンドを使用して、リソースグループを MANAGED 状態にします。

リソースグループ内に指定した型のリソースを作成するには、次のコマンドを使用します。

```
# scrgadm -a -
j resource_name -
t resource_type_name -
g RG_name
```

リソースを作成すると、使用している RGM メカニズムがいくつかのアクションを実行します。まず、配下の RGM メカニズムはリソースに対して VALIDATE メソッドを呼び出して、このリソースのプロパティーの設定が有効であるかどうかを検証します。VALIDATE メソッドが正常に終了

し、リソースグループが MANAGED 状態になったら、RGM はこのリソースに対して INIT メソッドを呼び出し、リソースを初期化します。次に RGM は、このリソースをオンラインにします。ただし、リソースが有効で、かつそのリソースグループがオンラインになっていなければなりません。

管理対象のリソースグループを削除するには、そのリソースグループからをすべてのリソースを削除します。リソースを削除するには、まず、scswitch コマンドでそのリソースを無効にします。リソースを削除すると、RGM がそのリソースに対する FINI メソッドを呼び出して、クリーンアップを行います。

このオプションは大域ゾーンだけで使用できます。

アクションオプション

アクションオプションでは、コマンドによって実行されるアクションを指定します。コマンド行に指定できるアクションオプションは 1 つだけです。

サポートされるアクションオプションには、次のものがあります。

-a

新しい構成を追加します。次のオプションとともに使用します。

-g
: リソースグループを作成します。
このオプションは大域ゾーンだけで使用できます。
このコマンドオプションを、-a、-c、または -r と一緒に使用するには、solaris.cluster.resource.modify の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-j
: リソースを作成します。
このオプションは大域ゾーンだけで使用できます。
このコマンドオプションを、-a、-c、または -r と一緒に使用するには、solaris.cluster.resource.modify の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-t
: リソースタイプを追加します。
このオプションは大域ゾーンだけで使用できます。
このコマンドオプションを、-a、-c、または -r と一緒に使用するには、solaris.cluster.resource.modify の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-c

既存の構成を変更します。指定されたプロパティー値だけが設定されます。その他のプロパティーは現在の値を維持します。次のオプションとともに使用します。

-g
リソースグループを変更します。
このオプションは大域ゾーンだけで使用できます。
このコマンドオプションを、-a、-c、または -r と一緒に使用するには、solaris.cluster.resource.modify の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-j
リソースを変更します。
このオプションは大域ゾーンだけで使用できます。
このコマンドオプションを、-a、-c、または -r と一緒に使用するには、solaris.cluster.resource.modify の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-t
リソースタイプを変更します。
このオプションは大域ゾーンだけで使用できます。
このコマンドオプションを、-a、-c、または -r と一緒に使用するには、solaris.cluster.resource.modify の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-r
構成を削除します。次のオプションとともに使用します。

-g
リソースグループを削除します。
このオプションは大域ゾーンだけで使用できます。
このコマンドオプションを、-a、-c、または -r と一緒に使用するには、solaris.cluster.resource.modify の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-j
リソースを削除します。
このオプションは大域ゾーンだけで使用できます。
このコマンドオプションを、-a、-c、または -r と一緒に使用するには、solaris.cluster.resource.modify の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-t
リソースタイプを削除します。
このオプションは大域ゾーンだけで使用できます。
このコマンドオプションを、-a、-c、または -r と一緒に使用するには、solaris.cluster.resource.modify の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-p
既存の構成情報を表示します。

このオプションは大域ゾーンだけで使用できます。次のオプションとともに使用します:

-g *resource_group_name*

特定のリソースグループの構成情報を表示します。

このコマンドオプションを -p と一緒に使用するには、solaris.cluster.resource.read の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-j *resource_name*

特定のリソースの構成情報を表示します。

このコマンドオプションを -p と一緒に使用するには、solaris.cluster.resource.read の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-t *resource_type_name*

特定のリソースタイプの構成情報を表示します。

このコマンドオプションを -p と一緒に使用するには、solaris.cluster.resource.read の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-v[v]

より詳しい形式で出力を表示します。

このコマンドオプションを -p と一緒に使用するには、solaris.cluster.resource.read の RBAC 承認が必要です。Unresolved link to " rbac5" を参照してください。

-g、-j、-t オプションをどれも指定しない場合、クラスタに現在構成されているすべてのリソースタイプ、リソースグループ、リソースに関する情報がデフォルトで出力されます。

-g、-j、-t オプションは複数の指定が可能で、-v オプションと任意に組み合わせて使用できます。

同じコマンド行に指定できる -v オプションの数は 2 つまでです。

ターゲットオプション

ターゲットオプションでは、ターゲットオブジェクトを特定できます。次のターゲットオプションがサポートされています。

注記 - リソースグループ、リソース、リソースタイプのプロパティー名には、大文字と小文字の区別はありません。プロパティー名を指定する際には、大文字と小文字を任意に組み合わせることができます。

-g *RG_name*

リソースグループ。

-j *resource_name*

リソース。-a オプションとともに使用する場合は、-t および -g ターゲットオプションをコマンドに指定し、インスタンス化するリソースのタイプと、リソースグループの名前を指定する必要があります。

-t *resource_type_name*

リソースタイプ。

リソースタイプ固有のオプション

次のオプションがサポートされています。

-f *registration_file_path*

このオプションは -a と併用可能です。リソースタイプ登録 (RTR) ファイルのパス名を指定します。通常、RTR ファイルは /opt/cluster/lib/rgm/rtreg ディレクトリに存在します。このディレクトリに RTR ファイルがない場合は、このオプションを指定してください。

-h *RT_installed_node_list*

このオプションは-a または -c と併用可能です。このリソースタイプがインストールされるノード名をコンマで区切ったリストで指定します。このタイプのリソースは、ノードリストがこのリストのサブセットであるリソースグループでのみインスタンス化できます。

-a オプションを指定するとき、-h オプションは任意指定です。-h オプションを指定しない場合、リソースタイプはすべてのノードでインストールされていることを意味します。こうすれば、すべてのリソースグループでこのタイプのリソースを初期化できます。

-c オプションと -h オプションを併用する場合、インストールされているノードの新しいリストまたはエスケープ指定されたワイルドカード文字 (*) とともに指定する必要があります。ワイルドカード文字は、そのリソースタイプがすべてのノードにインストールされていることを表します。

注記 - ノード名にコンマを含めることはできません。

-t *resource_type_name*

-a、-c、または -r と併用可能です。リソースタイプはリソースタイプ登録ファイルで定義されます。リソースタイプ登録ファイルには、そのリソースタイプの標準プロパティーと拡張プロパティーの値が指定されています。登録ファイルが通常インストールされる既知のディレクトリ (/opt/cluster/lib/rgm/rtreg) に有効なリソースタイプ登録ファイルを保存すると、次のような短縮形の記述法を使用できるようになります:

```
# scrgadm -a -t SUNW.rt:2.0
```

その場合には、次の表記を使用する必要はありません。

```
# scrgadm -a -t rtn -f full_path_to_SUNW.rt:2.0
```

現在登録されているリソースタイプの名前を表示するには、次のコマンドを使用します。

```
# scrgadm -p
```

Sun Cluster 3.1 以降、リソースタイプの名前は次の構文を取ります。

vendor_id.*resource_type*
:*version*

リソースタイプ名の 3 つのコンポーネントは、RTR ファイル内に指定された 3 つのプロパティー *Vendor_id*、*Resource_type*、および *RT_version* になります。`scrgadm` コマンドでは、区切り文字としてピリオドとコロンを使用します。オプションの接頭辞 *Vendor_id* は、複数のベンダーが同じ名前の登録ファイルを提供している場合、それぞれを区別するために使用します。*RT_version* は、あるバージョンのデータサービスから別のバージョンのデータサービスへのアップグレードに使用されます。

Vendor_id が一意であることを保証するためには、リソースタイプを作成した会社の株式の略号を使用します。`-t` オプションとともに使用する *resource_type_name* には、完全なリソースタイプ名または *Vendor_id* 部分を省略した名前を指定します。たとえば、`-t SUNW.iws` と `-t iws` はどちらも有効な表記法です。ただし、クラスタ内に接頭辞 *Vendor_id* だけが異なるリソースタイプ名が 2 つ存在する場合は、省略方式で名前を指定すると失敗します。

RT_version 文字列に空白、タブ、スラッシュ (`/`)、バックスラッシュ (`\`)、アスタリスク (`*`)、疑問符 (`?`)、左角括弧 (`[`)、または右角括弧 (`]`) 文字が含まれている場合、`scrgadm` コマンドはリソースタイプの登録に失敗します。

`-t` オプションで *resource_type_name* を指定する場合、バージョンが 1 つしか登録されていないなら、バージョンコンポーネントを省略できます。

Sun Cluster 3.1 リリースより前に作成したリソースタイプの名前は、引き続き次の構文に従います。

vendor_id.*resource_type*

`-y RT_system={TRUE|FALSE}`

リソースタイプの `RT_system` プロパティーを `TRUE` または `FALSE` に設定します。`RT_system` プロパティーのデフォルト値は、`FALSE` です。`RT_system` プロパティーについては、1335 ページの`rt_properties(5)` を参照してください。

リソースグループ固有のオプション

次のオプションがサポートされています。

`-h` *nodelist*

このオプションは`-a` または `-c` と併用可能です。このオプションは、`–y Nodelist=` *nodelist* のショートカットです。

`-y` *property*= *value*

このオプションは`-a` または `-c` と併用可能です。`-y` *property=value* は複数回指定できます。*value* の形式は、各 *property* によって指示されます。次の例では、*property1* は単一の文字列を*値* として取り、*property2* はコンマ区切りの文字列を取ります:

-y *property1*=
value1 -y
property2=*value2a*,*value2b*

文字列のプロパティーを空の値にするには、値の部分を空欄にしてこのオプションを指定します。次の例を参照してください。

-y *property*=

-y プロパティー名の評価では、大文字と小文字の区別はありません。

リソースグループプロパティーについては、1319 ページのrg_properties(5) を参照してください。

リソース固有のオプション

次のオプションがサポートされています。

-x *extension_property*= *value*
-x “*extension_property* {*node_specifier*}=*value* ”

このオプションは-a または -c と併用可能です。-x *extension_property*=*value* または -x “*extension_property*{*node_specifier*}=*value*” は複数回指定できます。

node_specifier は*任意指定*の修飾子であり、指定した単数または複数のノード*だけ*で、*extension_property* の値が設定または変更されることを示します。指定したプロパティーの値は、クラスタのほかのノードでは設定または変更されません。*node_specifier* を指定しない場合、指定したプロパティーの値は、クラスタのすべてのノードで設定または変更されます。次に、*node_specifier* の構文の例を示します:

```
-x "myprop{phys-schost-1}=100"
```

中括弧 ({ }) は、プロパティーを設定する特定の単数または複数のノードを示すために指定します。

次の構文の *node_specifier* を使用すると、2 つの異なるノードに異なる値を同時に指定できます:

```
-x "myprop{phys-schost-1}=100" -x "myprop{phys-schost-2}=10"
```

あるいは、次の構文を使用すると、2 つの異なるノードに 1 つの値を同時に設定または変更できます。

```
-x "myprop{phys-schost-1,phys-schost-2}=100"
```

value の形式は、各 *extension_property* によって指示されます。次の例では、*extension_property1* は単一の文字列を *value*として取り、*extension_property2* はコンマ区切りの文字列を取ります:

-x "*extension_property1*{
node_specifier}=*value1*" \
-x "*extension_property2*{

node_specifier}=*value2a*,
value2b"

特定のデータサービスの拡張プロパティーについては、そのデータサービスのマニュアルページを参照してください。

-y *property*= *value*

このオプションは-a または -c と併用可能です。-y *property*=*value* は複数回指定できます。*value* の形式は、各 *property* によって指示されます。次の例では、*property1* は単一の文字列を*値* として取り、*property2* はコンマ区切りの文字列を取ります:

-y *property1*=
value1 -y
property2=*value2a*,*value2b*

プロパティーを空の値に設定するには、値の部分を空欄にしてこのオプションを指定します。次の例を参照してください。

-y *property*=

-y *property* 名の評価では、大文字と小文字の区別はありません。

リソースプロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

LogicalHostname 固有のオプション

これらのオプションは、論理ホスト名リソースに適用されます。LogicalHostname リソースを削除する特別なコマンドは存在しません。

```
# scrgadm -r -
j resource_name
```

resource_name は、LogicalHostname リソースの作成時に任意指定の -j オプションで指定した名前になります。LogicalHostname リソースの作成時に -j オプションと *resource_name* を省略すると、scrgadm によって名前が生成されます。

次のオプションがサポートされています。

-j *resource_name*

-j オプションは、-l *hostnamelist* オプションの最初の引数として、ホスト名ではなく IP アドレスを使用するときに必須です。

-j は、リソースを作成するときに LogicalHostname リソースに明示的に名前を付ける場合は -a とともに、リソースグループからリソースを削除する場合は -r とともに使用します。-j オプションを使用して resource に明示的に名前を付けない場合、scrgadm コマンドは作成したリソースに、*hostnamelist* にある 1 番目のホスト名を割り当てます。

-L

コマンド行に指定されたオプションが論理ホスト名に適用されることを意味します。クラスタノードがアクティブなクラスタメンバーではない時点でこのコマンドを実行する場合は、-n *netiflist* オプションも使用する必要があります。

-l *hostnamelist*

共有する IPv4 アドレスまたは IPv6 アドレスを指定します。IP アドレスを指定できる場合でも、ホスト名を使用してください。*hostnamelist* は、この `LogicalHostname` リソースによって使用可能になるホスト名のコンマ区切りのリストです。

-n *netiflist*

ネットワークインタフェースのリストを指定します。クラスタノードがアクティブなクラスタメンバーではない時点でこのコマンドを実行する場合、-L オプションには -n オプションが必要です。

netiflist は次の形式を取ります。

netif@*node*[,…]

netif は、ネットワークアダプタ名 (`le0` など) または IP ネットワークマルチパス (IPMP) グループ名 (`sc_ipmp` など) で指定します。*node* は、ノード名またはノード識別子で指定します。リソースグループの *nodelist* 内のすべてのノードが *netiflist* に登録されている必要があります。-n *netiflist* が省略されている場合、*nodelist* の各ノードの *hostnamelist* によって識別されたサブネット上のネットアダプタを検出しようとする試みが行われます。発見されたネットワークアダプタが IPMP グループに存在しない場合は、単一アダプタの IPMP グループが作成されます。同様に、指定されたネットワークアダプタが IPMP グループに存在しない場合も、単一アダプタの IPMP グループが作成されます。

詳細は、注 セクションを参照してください。

-y *property= value*

詳細は、「Resource-Specific Options」セクションを参照してください。

SharedAddress 固有のセクション

下記の変更や追加とともに、**論理ホスト名**固有のオプションが**共有アドレス**リソースにも適用されます。

-S

コマンド行に指定されたオプションが共有アドレスに適用されることを意味します。

-X *auxnodelist*

ノード名または識別子のコンマで区切ったリストを指定します。このリストには、クラスタメンバーであるノードの名前または ID を指定します。これらのノードは、指定された共有アドレスのホストとなることはあるが、フェイルオーバーのプライマリノードとして動作することは決してないノードです。

このリストと *nodelist* は相互排他の関係にあります。「Resource-Group Specific Options」の *nodelist* の説明を参照してください。

次の終了値が返されます。

`0`	コマンドは正常に完了しました。 このコマンドが正常に完了した場合でも、警告メッセージが標準エラー出力に書き込まれることがあります。
0 以外	エラーが発生しました。 非ゼロステータスで終了した場合、標準エラーにエラーメッセージが書き出されます。

`RT_System` プロパティーが `TRUE` のリソースタイプでは、一部の処理が実行できません。同様に、`RG_System` プロパティーが `TRUE` のリソースグループ (およびそのリソース) では、一部の処理が実行できません。1335 ページの`rt_properties`(5) および 1319 ページの`rg_properties`(5) を参照してください。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/system/core`
インタフェースの安定性	廃止

19 ページの`Intro`(1CL), 251 ページの`clreslogicalhostname`(1CL), 273 ページの`clresource`(1CL), 305 ページの`clresourcegroup`(1CL), 333 ページの`clresourcetype`(1CL), 347 ページの`clressharedaddress`(1CL), Unresolved link to " ifconfig1M", 877 ページの`scstat`(1M), 883 ページの`scswitch`(1M), 1287 ページの`r_properties`(5), Unresolved link to " rbac5", 1319 ページの`rg_properties`(5), 1335 ページの`rt_properties`(5)

`LogicalHostname` および `SharedAddress` の追加処理中に、使用できるように構成されていないネットワークアダプタを検出したり、IP ネットワークマルチパスグループに追加したりすることはできません。Unresolved link to " ifconfig1M" を参照してください。

`scrgadm` がゼロ以外ステータスで終了し、エラーメッセージ `cluster is reconfiguring` が出力された場合でも (エラーステータスを示している場合でも)、要求した操作は正常に終了してい

ることがあります。結果が疑わしい場合は、再構成の完了後にもう一度同じ引数で `scrgadm` を実行できます。

名前

scsetup — 対話型クラスタ構成ツール

```
scsetup [-f logfile]
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

`scsetup` コマンドが提供する構成機能は、コマンドを実行したときのクラスタの状態によって変わります。

- `scsetup` コマンドをインストール後処理時に実行すると、このコマンドは定足数デバイスの構成や `installmode` プロパティーのリセットなどの初期設定タスクを実行します。クラスタを作成したときに自動定足数構成を使用しなかった場合、クラスタのインストール直後に `scsetup` コマンドを実行します。すべてのノードがクラスタに参加していることを確認してから、`scsetup` コマンドを実行し、`installmode` プロパティーをリセットします。

  クラスタを作成したときに自動定足数構成を使用した場合、クラスタのインストール後に `scsetup` コマンドを実行する必要はありません。自動定足数構成機能は、クラスタの `installmode` プロパティーもリセットします。
- 通常のクラスタ動作中に実行すると、`scsetup` コマンドはメニュー選択方式のユーティリティーを提供します。このユーティリティーを使用すると、現行のクラスタ管理タスクの大部分を実行できます。
- 非クラスタモードのノードから発行すると、`scsetup` ユーティリティーは、プライベート IP アドレス範囲を変更または表示するためのメニュー選択方式のユーティリティーを提供します。この形式の `scsetup` ユーティリティーを開始する前に、すべてのノードを非クラスタモードにリブートしてください。

`scsetup` コマンドは、クラスタの任意のノードから実行できます。

このコマンドは、大域ゾーンだけで使用できます。

次のオプションがサポートされています。

-f *logfile*　　コマンドログを記録するログファイル名を指定します。このオプションを指定した場合、scsetup ユーティリティーが生成するほとんどのコマンドセットは、ユーザーの応答に応じて、実行されてから記録されるか、または単に記録されます。

次の属性の詳細は、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	廃止

19 ページのIntro(1CL), 535 ページのcltelemetryattribute(1CL),
79 ページのcldevicegroup(1CL), 185 ページのclnode(1CL),
239 ページのclquorum(1CL), 251 ページのclreslogicalhostname(1CL),
305 ページのclresourcegroup(1CL), 333 ページのclresourcetype(1CL),
347 ページのclressharedaddress(1CL), 551 ページのcluster(1CL),

名前

scshutdown — クラスタの停止

```
scshutdown [-y] [-g grace-period] [message]
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

`scshutdown` ユーティリティーは、クラスタを正常に停止させる際に使用します。

停止処理を開始するにあたり、`scshutdown` はまず警告メッセージを送信し、次に最終メッセージを送信して停止を確認します。

`scshutdown` コマンドは、1 つのノードだけで実行してください。

`scshutdown` コマンドがクラスタを停止する際には、次の一連のアクションが行われます。

- クラスタ上で稼動中のすべてのリソースグループを、オフライン状態に移行します。移行できないものあった場合、`scshutdown` は完了せずに、エラーメッセージが表示されます。
- すべてのクラスタファイルシステムをアンマウントします。アンマウント処理できないものがあった場合、`scshutdown` は完了せずに、エラーメッセージが表示されます。
- アクティブなすべてのデバイスサービスを停止させます。デバイスのいずれかの移行が失敗した場合は、`scshutdown` が完了せず、エラーメッセージが表示されます。
- すべてのノードで `/usr/sbin/init 0` を実行します。詳細は、Unresolved link to " init1M" を参照してください。

このコマンドは、大域ゾーンだけで使用できます。

このコマンドを使用するためには、`solaris.cluster.system.admin` の承認が必要です。Unresolved link to " rbac5" を参照してください。

次のオプションがサポートされています。

`-g` *grace-period*　　デフォルト値の 60 秒から *grace-period* に指定された時間 (秒) に変更します。

-y　　停止許可の確認を事前に行うことで、停止コマンドをユーザーの介入なしに実行できるようにします。

次のオペランドがサポートされています。

message　　標準的な警告メッセージ `The system will be shut down in …` が出されたあとに出される文字列です。*message* に単語が少なくとも 1 つある場合は、単一引用符 (') または二重引用符 (") で囲む必要があります。標準の警告メッセージおよびユーザー指定の *message* が表示されるタイミングは、`scshutdown` の開始から `7200`、`3600`、`1800`、`1200`、`600`、`300`、`120`、 `60`、`30` 秒前です。

例 346　　クラスタの停止

```
phys-palindrome-1# scshutdown
```

次の終了値が返されます。

`0`　　コマンドは正常に完了しました。

0 以外　　エラーが発生しました。標準出力にエラーメッセージを出力。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	廃止

19 ページの`Intro`(1CL)、551 ページの`cluster`(1CL)、Unresolved link to " shutdown1M"、Unresolved link to " init1M"、Unresolved link to " attributes5"

名前

scstat — Oracle Solaris Cluster 構成のステータスのモニタリング

```
scstat [-DWginpqv [v]] [-h node]
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

`scstat` コマンドは、Oracle Solaris Cluster コンポーネントの現在の状態を表示します。その際、`scstat` コマンドを Oracle Solaris Cluster 構成内のマシン上で 1 回だけ実行します。

オプションを指定しないで実行した場合、`scstat` によってクラスタの全コンポーネントのステータスが表示されます。表示される情報は次のとおりです。

- クラスタメンバーのリスト
- 各クラスタメンバーのステータス
- リソースグループおよびリソースのステータス
- クラスタインターコネクト上の各パスのステータス
- 各ディスクデバイスグループのステータス
- 各定足数デバイスのステータス
- すべての IP ネットワークマルチパス (IPMP) グループとパブリックネットワークのステータス

このコマンドをオプションなしで使用するためには、`solaris.cluster.device.read`、`solaris.cluster.transport.read`、`solaris.cluster.resource.read`、`solar` の RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

リソースとリソースグループ

リソースのステータス、リソースグループのステータス、リソースのステータスは、すべてノード単位で管理されます。たとえば、リソースは、各クラスタノード上で固有のステータス、固有のステータスを持ちます。

リソース状態は、そのリソース上でどのメソッドが呼び出されたかにのみ基づいて、各ノード上の Resource Group Manager (RGM) によって設定されます。たとえば、指定のノード上でリソースに対する `STOP` メソッドを正しく実行した場合、そのリソースのノード上での状態は

OFFLINE になります。STOP メソッドが 0 以外またはタイムアウトで終了した場合、そのリソースの状態は Stop_failed になります。

可能性のあるリソース状態には、Online、Offline、Start_failed、Stop_failed、Monitor_failed、Online_not_monitored、Starting、および Stopping があります。

可能性のあるリソースグループの状態には、Unmanaged、Online、Offline、Pending_online、Pending_offline、Error_stop_failed、Online_faulted、および Pending_online_blocked があります。

RGM は、リソースのステータスだけでなく、リソース自体が API を使って設定するリソースのステータスも維持します。Status Message のフィールドは、実際には、ステータスキーワードとステータスメッセージからなります。ステータスメッセージは、ステータスキーワードのあとに出力される任意のテキスト文字列で、リソースによって任意に設定されます。

リソースステータスの値には、次のものがあります。

DEGRADED　リソースはオンラインですが、何らかの理由でパフォーマンスまたは可用性が低下しています。

FAULTED　リソースの機能を妨げるエラーが検出されました。

OFFLINE　リソースはオフラインです。

ONLINE　リソースはオンラインでサービスを提供します。

UNKNOWN　現在のステータスは不明または遷移中です。

デバイスグループ

デバイスグループのステータスは、そのグループ内のデバイスの可用性を表しています。

次は、指定可能なデバイスグループのステータスの値とその説明です。

DEGRADED　デバイスグループはオンラインですが、潜在的主ノード (二次ノード) の一部が有効になっていません。2 ノード接続の場合、このステータスは基本的に、スタンバイしているプライマリノードが存在しないことを示します。つまり、プライマリノードに障害が発生すると、グループ内のデバイスへのアクセスが失われます。

OFFLINE　デバイスグループはオフラインです。プライマリノードは存在しません。デバイスグループ内のデバイスを使用する前に、デバイスグループをオンラインにする必要があります。

ONLINE　デバイスグループはオンラインです。プライマリノードが存在し、グループ内のデバイスは入出力可能な状態です。

WAIT　デバイスグループは、あるステータスと別のステータスの間になります。たとえば、デバイスグループがオフラインからオンラインに移行している間は、このステータスになります。

IP ネットワークマルチパスグループ

IP ネットワークマルチパス (IPMP) グループのステータスは、バックアップグループとそのアダプタが使用可能かどうかを表しています。

次は、指定可能な IPMP グループのステータスの値とその説明です。

OFFLINE　バックアップグループは異常状態にあります。グループのすべてのアダプタがオフラインになっています。

ONLINE　バックアップグループは機能可能な状態にあります。そのグループの少なくとも 1 つのアダプタがオンラインになっています。

UNKNOWN　上のどの状態でもありません。`if_mpadm` や `ifconfig` コマンドによってアダプタが切り離されたり、停止状態に指定されたりすると、この状態になることがあります。

次は、指定可能な IPMP アダプタのステータスの値とその説明です。

OFFLINE　アダプタが異常状態にあるか、バックアップグループがオフラインになっています。

ONLINE　アダプタは機能可能な状態にあります。

STANDBY　アダプタは待機状態にあります。

UNKNOWN　上のどの状態でもありません。`if_mpadm` や `ifconfig` コマンドによってアダプタが切り離されたり、停止状態に指定されたりすると、この状態になることがあります。

コマンドオプションを指定して、特定のコンポーネントのステータスを要求できます。

複数のオプションを指定した場合、`scstat` コマンドは指定した順番でステータスを出力します。

次のオプションがサポートされています。

-D　すべてのディスクデバイスグループのステータスが表示されます。

このコマンドオプションを使用するために
は、solaris.cluster.device.read の RBAC の承認が必要で
す。Unresolved link to " rbac5" を参照してください。

-g
すべてのリソースグループのステータスが表示されます。

このコマンドオプションを使用するために
は、solaris.cluster.resource.read の RBAC の承認が必要で
す。Unresolved link to " rbac5" を参照してください。

-h *node*
指定のノード (*node*) のステータスと、このノードをプライマリノードとする
ディスクデバイスグループのステータスが表示されます。定足数デバイス
のステータスが表示されます。ただし、このノードは、これらの定足数デバ
イスに対して、このノードが潜在マスターであるリソースグループの予約を
保持し、*node* が接続されているトランスポートパスの予約を保持してい
るものとします。

このコマンドオプションを使用するために
は、solaris.cluster.device.read、solaris.cluster.transport.read、solaris.cluste
RBAC の承認が必要です。Unresolved link to " rbac5" を参照して
ください。

-i
すべての IPMP グループおよびパブリックネットワークアダプタのステー
タスが表示されます。

このオプションは大域ゾーンだけで使用できます。

-n
すべてのノードのステータスが表示されます。

このコマンドオプションを使用するために
は、solaris.cluster.node.read の RBAC の承認が必要で
す。Unresolved link to " rbac5" を参照してください。

-p
クラスタ内のすべてのコンポーネントのステータスが表示されます。-v と
併用すると、より詳しい形式で出力が得られます。

-p を -v と使用するために
は、solaris.cluster.device.read、solaris.cluster.transport.read、solaris.cluste
および solaris.cluster.system.read の RBAC の承認が必要で
す。Unresolved link to " rbac5" を参照してください。

-q
デバイスおよびノードの定足数がすべて表示されます。

このコマンドオプションを使用するために
は、solaris.cluster.quorum.read の RBAC の承認が必要で
す。Unresolved link to " rbac5" を参照してください。

-v[v]
出力が詳細形式で表示されます。

-W
クラスタトランスポートパスのステータスが表示されます。

このコマンドオプションを使用するために
は、`solaris.cluster.transport.read` の RBAC の承認が必要で
す。Unresolved link to " rbac5" を参照してください。

例 347 `scstat` コマンドの使用

次のコマンドは、すべてのリソースグループのステータスと、指定のホストに関連するすべてのコンポーネントのステータスが表示されます。

```
% scstat -g -h host
```

表示される出力は、オプションが指定された順序で表示されます。

出力内容は、次の 2 つのコマンドを入力した場合と同じになります。

```
% scstat -g
```

および

```
% scstat -h host
```

次の終了値が返されます。

`0` コマンドは正常に完了しました。

0 以外 エラーが発生しました。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/system/core`
インタフェースの安定性	廃止

19 ページの`Intro`(1CL)、551 ページの`cluster`(1CL)、Unresolved link to " if_mpadm1M"、Unresolved link to " ifconfig1M"、699 ページの`scha_resource_setstatus`(1HA)、1175 ページの`scha_resource_setstatus`(3HA)、Unresolved link to " attributes5"

オンライン定足数デバイスとは、最後に定足数が満たされた時点で、定足数の構成に貢献可能だったデバイスのことです。定足数アルゴリズムのコンテキストでは、デバイスは定足数の構成に積極的に貢献したため、オンラインになっています。ただし、オンライン定足数デバイスが必ずしも、定足数が再確立されたときの定足数の形成に貢献するのに十分なほど健全な状態を維持できるとは限りません。現在のバージョンの Oracle Solaris Cluster には、ディスクモニタリング機能や定足数デバイスを検証する定期的なプローブは含まれていません。

名前

scswitch — Oracle Solaris Cluster 構成のリソースグループとデバイスグループの所有権および状態の変更の実行

```
scswitch -c -h node[,...] -j resource[,...] -f flag-name

scswitch {-e | -n} [-M] -j resource[,...] [-h node[,...]]

scswitch -F {-g resource-grp[,...] | -D device-group[,...]}

scswitch -m -D device-group[,...]

scswitch -Q [-g resource-grp[,...]] [-k]

scswitch -R -h node[,...] -g resource-grp[,...]

scswitch -r [-g resource-grp[,...]]

scswitch -S -h node[,...] [-K continue_evac]

scswitch -s [-g resource-grp[,...]] [-k]

scswitch {-u | -o} -g resource-grp[,...]

scswitch -Z [-g resource-grp[,...]]

scswitch -z -D device-group[,...] -h node [,...]

scswitch -z [-g resource-grp[,...]] [-h node [,...]]
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

`scswitch` コマンドは、リソースグループまたはデバイスグループ (ディスクデバイスグループとも呼ぶ) を新しいプライマリノードに移動します。このほかに、所有権を変更してすべてのリソースグループおよびデバイスグループを退避する機能、リソースグループやデバイスグループのオンラインとオフラインを切り替える機能、リソースの有効または無効を切り替える機能、リソースグループの状態を `Unmanaged` に切り替える (または元に戻す) 機能、リソースのエラーフラグをオフにする機能などを提供します。

`scswitch` コマンドは、Oracle Solaris Cluster 構成内のどのノードからでも実行できます。デバイスグループがオフラインの場合は、`scswitch` を使って、ノードリストの任意のホストに対してデバイスグループをオンラインにすることができます。ただし、デバイスグループをオンラインに

すると、スペアノードへスイッチオーバーすることができなくなります。scswitch は同時に 1 個だけ実行することができます。

まだ処理が完了していない scswitch を強制終了しないでください。

このコマンドの有効な使用方法の詳細は、個々のオプションの説明を参照してください。管理を容易にするため、このコマンドは大域ゾーンで使用します。

基本オプション

次の基本オプションがサポートされます。これらのいくつかの基本オプションとともに使用できるオプションは、「追加オプション」で説明されています。

-c
: 指定されたノード上で、指定されたリソースセットの -f *flag-name* エラーフラグをクリアします。Oracle Solaris Cluster ソフトウェアの現在のリリースでは、-c オプションが実装されているのは Stop_failed リソース状態だけです。Stop_failed リソース状態をクリアすると、そのリソースは指定されたノード上でオフラインになります。

  このオプションは、大域ゾーンでのみ使用します。

  リソースに対して Stop メソッドが失敗し、そのリソースの Failover_mode プロパティーが Hard に設定されている場合、Resource Group Manager (RGM) はノードを停止またはリブートし、このリソース (およびそのノードがマスターするほかのすべてのリソース) を強制的にオフラインにします。

  Failover_mode プロパティーが Hard 以外の値に設定されているリソースで実行された Stop メソッドが失敗した場合、個々のリソースは Stop_failed リソース状態になり、リソースグループは Error_stop_failed 状態になります。Error_stop_failed 状態になっているリソースグループは、どのノードでもオンラインにすることができず、また、リソースの追加や削除、リソースグループやリソースのプロパティーの変更などの編集も、一切行うことができません。Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド "で説明されている手順を実行して、Stop_failed リソース状態をクリアする必要があります。

注意 - Stop_failed リソース状態をクリアする前に、指定されたノード上でリソースとそのモニターの両方が停止されていることを確認してください。リソースとモニターを完全に終了しないで Stop_failed リソース状態をクリアすると、クラスタ上で複数のリソースインスタンスが同時に実行される可能性があります。共有ストレージを使用している場合、クラスタ上で複数のリソースインスタンスが同時に実行されると、データが破壊される可能性があります。必要な場合は、最後の手段として、関連するプロセスに対して Unresolved link to " kill1" コマンドを実行します。

-e
: 指定された *resource* を有効にします。

このオプションは、大域ゾーンでのみ使用します。

有効にされたリソースは、そのリソースグループがオンラインかオフラインかによって、オンラインまたはオフラインになります。

-h オプションと -e オプションを併用すると、指定されたノードのサブセットだけでリソースを有効にすることができます。-h オプションを省略した場合、指定されたリソースはすべてのノード上で有効になります。

-F

指定されたリソースグループ (-g) またはデバイスグループ (-D) をすべてのノードでオフラインにします。

-F オプションと -D オプションを併用するとき、-F オプションは大域ゾーンだけから実行できます。

-F オプションでデバイスグループをオフラインにすると、関連する Solaris Volume Manager のディスクセットがプライマリノードによりデポートまたは解放されます。デバイスグループをオフラインにする前に、そのデバイスへのアクセスをすべて停止し、依存する全ファイルシステムのマウントを解除してください。明示的な scswitch 呼び出しを発行するか、グループ内のデバイスにアクセスするか、またはこのグループに依存するファイルシステムをマウントすることによって、オフラインのデバイスグループを起動できます。

-m

メンテナンスのため、指定されたデバイスグループをクラスタからオフラインにします。結果は、リブート後も保存されます。

このオプションは、大域ゾーンでのみ使用します。

デバイスグループをメンテナンスモードにする前に、そのデバイスへのアクセスをすべて停止し、依存する全ファイルシステムのマウントを解除してください。デバイスグループがアクセスされている最中である場合、このアクションは失敗し、指定されたデバイスグループはクラスタからオフラインにされません。

デバイスグループをオンラインに戻すには、-z オプションを使用します。デバイスグループのメンテナンスモードを終了するには、scswitch コマンドを明示的に呼び出すしかありません。

-n

指定されたリソースを無効にします。

このオプションは、大域ゾーンでのみ使用します。

現在のマスターでオンラインの無効なリソースは、現在のマスターから即座にオフラインにされます。無効なリソースは、リソースグループの状態にかかわらず、オフラインのままです。

-h オプションを -e オプションとともに指定すると、ノードの指定されたサブセット上のリソースのみを無効にできます。-h オプションを省略した場合、指定されたリソースはすべてのノード上で無効になります。

-o

指定された管理対象外のリソースグループを管理対象にします。

RGM は、管理対象になったリソースグループをオンラインにしようと試みます。

-Q

指定されたリソースグループを休止状態にします。

-g オプションを省略した場合、-Q オプションはすべてのリソースグループに適用されます。

このオプションは、Start メソッドまたは Stop メソッドが失敗した場合に、指定されたリソースグループがあるノードから別のノードに繰り返し切り換えられるのを停止します。この形式の scswitch コマンドは、リソースグループが休止状態になり、どのノード上でも停止したり起動したりしなくなってから終了します。

scswitch -Q コマンドの実行中に、グループ内の任意のリソースに対して実行された Monitor_stop、Stop、Postnet_stop、Start、または Prenet_start メソッドが失敗した場合、リソースは、実際の設定とは関係なく、Failover_mode プロパティーの値が None であるものとして動作します。これらのメソッドのいずれかが失敗した場合、ノードのフェイルオーバーやリブートは実行されず、リソースがエラー状態 (Start_failed または Stop_failed のリソース状態) になります。

scswitch -Q コマンドが終了すると、指定されているリソースグループは、オンラインまたはオフラインもしくは ONLINE_FAULTED または ERROR_STOPPED_FAILED 状態になることがあります。clresourcegroup status コマンドを実行すると、これらリソースグループの現在のステータスを調べることができます。

scswitch -Q コマンドの実行中にノードが停止した場合、実行が中断され、リソースグループが非休止状態になることがあります。割り込みが発生すると、scswitch -Q コマンドは 0 以外の終了コードを返し、標準エラーにエラーメッセージを書き込みます。この場合は、scswitch -Q コマンドを再実行できます。

-k オプションと -Q オプションを併用すると、リソースグループの停止を早めることができます。-k オプションを指定する場合、影響を受けるリソースグループのリソースのために動作しているすべてのメソッドが即座に強制終了されます。-k オプションを指定しないと、終了するか構成されているタイムアウトを超えるまで継続的にメソッドを実行できます。

-R

指定されたプライマリノード上で、指定されたリソースグループをオフラインにしてから、オンラインに戻します。

指定されたノードは、リソースグループの現在のプライマリノードである必要があります。

-r

指定されたりソースグループで、以前 -s オプションで中断されていた自動回復アクションを再開します。

-g オプションを省略した場合、-r オプションはすべてのリソースグループに適用されます。

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

リソースグループの自動回復アクションを中断する方法については、-s オプションの説明を参照してください。

-S

すべてのリソースグループおよびデバイスグループを指定された *node* から切り替えます。

大域ゾーンで実行する場合、このオプションは、クラスタ内の指定された任意のノードを退避できます。

システムは、各グループの構成済みの設定に従って、新しいプライマリノードを選択しようとします。退避されたすべてのグループが、必ずしも同じプライマリノードによって再マスターされるとは限りません。指定されたノードによってマスターされている一部のグループが指定されたノードから正常に退避できなかった場合、このコマンドはエラーで終了します。

リソースグループはまずオフラインになってから、新しいプライマリノードに再配置されます。新しいプライマリノード上でシステムが起動できなかった場合、退避されたリソースグループはオフラインのままになることがあります。

デバイスグループのプライマリ所有権をほかのノードに移すことができない場合、そのデバイスグループのプライマリ所有権は元のノードによって保持されます。

-s

指定されたリソースグループ上の自動復旧アクションを中断し、そのリソースグループを休止します。

-g オプションを省略した場合、-s オプションはすべてのリソースグループに適用されます。

このオプションでリソースグループのモニタリングを明示的に再開するまで、中断されたリソースグループは、自動的に、起動、再起動、またはフェイルオーバーすることはありません。リソースグループのモニタリングが中断された状態にあるとき、データサービスはオンライン状態のままです。指定されたノード上では引き続き、リソースグループを手動でオンラインまたはオフラインに切り替えることができます。また、リソースグループ内の個々のリソースも有効または無効にできます。

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

-k オプションも指定すると、影響を受けるリソースグループのリソースのために動作しているすべてのメソッドを即座に強制終了できます。-k オプションを使用することで、リソースグループの休止を早めることができます。-k オプションを指定しないと、終了するか構成されているタイムアウトを超えるまで継続的にメソッドを実行できます。

リソースグループの自動回復アクションを再開する方法については、-r オプションの説明を参照してください。

-u　指定された管理対象のリソースグループを管理対象外にします。

-u オプションを使用するためには、指定のリソースグループのすべてのリソースをあらかじめ無効にしておく必要があります。

-Z　このオプションを指定すると、次の動作が実行されます。

- 指定されたリソースグループのすべてのリソースを有効にします。
- これらのリソースグループを管理対象にします。
- これらのリソースグループをすべてのデフォルトの主ノードまたはゾーン上でオンラインにします。

-g オプションを省略した場合、-Z オプションはすべてのリソースグループに適用されます。

-g オプションを指定しない場合、scswitch コマンドは、中断されているリソースグループを除いて、すべてのリソースグループをオンラインにしようとします。

-z　指定されたリソースグループまたはデバイスグループの所有権の変更を要求します。

-g オプションを省略した場合、-z オプションはすべてのリソースグループに適用されます。

-D オプションと -z オプションを併用すると、指定した 1 つまたは複数のデバイスグループが指定したノードにスイッチオーバーされます。デバイスグループのスイッチオーバーには、プライマリノード名を 1 個だけ指定できます。複数のデバイスグループを指定した場合、-D オプションはデバイスグループを指定された順番でスイッチオーバーします。-z オプションと -D オプションの併用時にエラーが発生すると、処理は中断され、以降のスイッチオーバーは行われません。

-g オプションだけと併用する場合、-z オプションは、指定されたリソースグループ (すでに管理対象になっている必要があります) を最優先ノードでオンラインにします。この形式の scswitch は、その強い RG_affinities の違反のためにリソースグループをオンラインにすることはなく、いずれかのノード上でリソースグループのアフィニティーを有効にできない場合は警告メッセージを書き込みます。このオプションは、-Z オプションとは異なり、すべてのリソースを有効にしたり、すべてのリソース上

でモニタリングを実行したり、すべての管理対象外のリソースグループを管理対象にしたりすることはありません。

-g および -h オプションとともに使用された場合、-z オプションは、指定されたリソースグループを -h オプションで指定されたノード上ではオンラインにし、その他のすべてのクラスタノード上ではオフラインにします。-h オプションで指定されたノードリストが空 (-h "") の場合、-z オプションは、-g オプションで指定されたリソースグループを、その現在のすべてのマスターからオフラインにします。-h オプションで指定されたすべてのノードは、クラスタの現在のメンバーである必要があり、-g オプションで指定されたすべてのリソースグループの潜在的なプライマリである必要があります。-h オプションで指定されたノード数が、-g オプションで指定された任意のリソースグループの Maximum_primaries プロパティーの設定を超えないようにする必要があります。

単独で使用された場合 (scswitch -z)、-z オプションは、それぞれの最優先ノードで中断されていない管理対象のすべてのリソースグループをオンラインに切り替えます。

1 つ以上のリソースグループの RG_affinities プロパティーを構成し、scswitch -z -g コマンド (-h オプションは指定してもしなくてもよい) を発行すると、-g オプションのあとに指定されたリソースグループ以外のリソースグループも追加で切り替えられる可能性があります。RG_affinities は、1319 ページの rg_properties(5) で説明されています。

追加オプション

上記の基本オプションには、次の追加オプションを組み合わせて使用できます。

-D

1 つまたは複数のデバイスグループの名前を指定します。

-F、-m、-z のいずれかと併用した場合に限り有効です。

このコマンドオプションを、-F、-m、-z の各オプション (-h オプションと併用) と併用するためには、solaris.cluster.device.admin 役割に基づくアクセス制御 (RBAC) の承認が必要です。Unresolved link to "rbac5" を参照してください。

さらにこのコマンドを使用するにあたっては、Oracle Solaris Cluster コマンド権プロファイルを割り当てておく必要があります。認証されたユーザーは、pfsh、pfcsh、または pfksh プロファイルシェルから、Oracle Solaris Cluster の特権コマンドをコマンド行で実行できます。プロファイルシェルは、Oracle Solaris Cluster コマンド権プロファイルに割り当てられた Oracle Solaris Cluster の特権コマンドへのアクセスを可能にする特別なシェルです。プロファイルシェルは、su を実行して役割を引き受けたときに起動されます。pfexec を使用しても、Oracle Solaris Cluster の特権コマンドを実行できます。

-f

エラーの *flag-name* を指定します。

このオプションを併用できるのは -c オプションだけです。

現在サポートされているエラーフラグは、Stop_failed のみです。

このコマンドオプションを -c オプションと併用するためには、solaris.cluster.resource.admin RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

さらにこのコマンドを使用するにあたっては、Oracle Solaris Cluster コマンド権プロファイルを割り当てておく必要があります。認証されたユーザーは、pfsh、pfcsh、または pfksh プロファイルシェルから、Oracle Solaris Cluster の特権コマンドをコマンド行で実行できます。プロファイルシェルは、Oracle Solaris Cluster コマンド権プロファイルに割り当てられた Oracle Solaris Cluster の特権コマンドへのアクセスを可能にする特別なシェルです。プロファイルシェルは、su を実行して役割を引き受けたときに起動されます。pfexec を使用しても、Oracle Solaris Cluster の特権コマンドを実行できます。

-g

1 つまたは複数のリソースグループの名前を指定します。

このオプションは、-F、-o、-Q、-r、-R、-s、-u、-z、および -Z の各オプションと組み合わせた場合に限り有効です。

このコマンドオプションを次のオプションと併用するためには、solaris.cluster.resource.admin RBAC の承認が必要です。

- -F オプション
- -o オプション
- -Q オプション
- -h オプションと併用する -R オプション
- -r オプション
- -s オプション
- -u オプション
- -Z オプション
- -h オプションと併用する -z オプション

Unresolved link to " rbac5" を参照してください。

さらにこのコマンドを使用するにあたっては、Oracle Solaris Cluster コマンド権プロファイルを割り当てておく必要があります。認証されたユーザーは、pfsh、pfcsh、または pfksh プロファイルシェルから、Oracle Solaris Cluster の特権コマンドをコマンド行で実行できます。プロファイルシェルは、Oracle Solaris Cluster コマンド権プロファイルに割り当てられた Oracle Solaris Cluster の特権コマンドへのアクセスを可能にする特別なシェルです。プロファイルシェルは、su を実行して役割を

引き受けたときに起動されます。pfexec を使用しても、Oracle Solaris Cluster の特権コマンドを実行できます。

-h
1 つまたは複数のクラスタノードの名前を指定します。

このオプションは、-c、-e、-n、-R、-S、および -z の各オプションと組み合わせた場合に限り有効です。

-c、-e、-n、-R、または -z オプションとともに使用された場合、-h オプションは、ノードのコンマ区切りリストを受け入れます。

-z オプションに空のノードリストを指定するには、-h オプションへの引数として 2 つの二重引用符 "" を指定します。

複数のプライマリで構成されたリソースグループの場合、-h オプションで一覧表示されるノード名はすべて、-g オプションで指定された各リソースグループの有効な潜在的なプライマリである必要があります。

あるリソースグループが -h オプションで指定されたノード上で正常に起動しなかった場合、そのリソースグループは別のノードにフェイルオーバーすることがあります。この動作は、Failover_mode リソースプロパティーの設定によって決定されます。詳細は、1287 ページのr_properties(5) を参照してください。

-S オプションと併用する場合、-h オプションは、リソースグループとデバイスグループを退避させる単一のノードの名前を指定します。

このコマンドオプションを、-c、-R の各オプション (-g オプションと併用)、-S オプション、または -z オプション (-g オプションと併用) と併用するためには、solaris.cluster.resource.admin RBAC の承認が必要です。また、-z オプション (-D オプションと併用) と併用するためには、solaris.cluster.device.admin RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

さらにこのコマンドを使用するにあたっては、Oracle Solaris Cluster コマンド権プロファイルを割り当てておく必要があります。認証されたユーザーは、pfsh、pfcsh、または pfksh プロファイルシェルから、Oracle Solaris Cluster の特権コマンドをコマンド行で実行できます。プロファイルシェルは、Oracle Solaris Cluster コマンド権プロファイルに割り当てられた Oracle Solaris Cluster の特権コマンドへのアクセスを可能にする特別なシェルです。プロファイルシェルは、su を実行して役割を引き受けたときに起動されます。pfexec を使用しても、Oracle Solaris Cluster の特権コマンドを実行できます。

-j
1 つまたは複数の *リソース* を指定します。

このコマンドオプションは、-c、-e、-n の各オプションと組み合わせた場合に限り有効です。

このコマンドオプションを、-c、-e、または -n オプションと併用するためには、solaris.cluster.resource.admin RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

さらにこのコマンドを使用するにあたっては、Oracle Solaris Cluster コマンド権プロファイルを割り当てておく必要があります。認証されたユーザーは、pfsh、pfcsh、または pfksh プロファイルシェルから、Oracle Solaris Cluster の特権コマンドをコマンド行で実行できます。プロファイルシェルは、Oracle Solaris Cluster コマンド権プロファイルに割り当てられた Oracle Solaris Cluster の特権コマンドへのアクセスを可能にする特別なシェルです。プロファイルシェルは、su を実行して役割を引き受けたときに起動されます。pfexec を使用しても、Oracle Solaris Cluster の特権コマンドを実行できます。

-K

ノードの退避が正常に行われたあと、このノードにリソースグループをスイッチバックしないようにする時間を秒数で指定します。

ノードが退避している間、あるいは退避してから、このオプションで指定した秒数の時間が経過するまで、リソースグループはこのノードにフェイルオーバーまたは自動スイッチオーバーできません。*continue_evac* 秒が経過する前に scswitch -z -g -h コマンドを使用してリソースグループを退避されたノードに切り替えることによって、-K タイマーをオーバーライドできます。このようなスイッチオーバーが行われると、-K タイマは即座に期限切れと見なされます。ただし、-h フラグを指定していない scswitch -z -g や -Z コマンドは引き続き -K タイマーに従い、リソースグループが退避されたノードにスイッチオーバーすることはありません。

このオプションは、-S オプションと組み合わせた場合に限り有効です。0 から 65535 までの整数値を指定してください。値を指定しない場合、デフォルトでは、60 秒が使用されます。

このコマンドオプションを使用するためには、solaris.cluster.resource.admin RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

さらにこのコマンドを使用するにあたっては、Oracle Solaris Cluster コマンド権プロファイルを割り当てておく必要があります。認証されたユーザーは、pfsh、pfcsh、または pfksh プロファイルシェルから、Oracle Solaris Cluster の特権コマンドをコマンド行で実行できます。プロファイルシェルは、Oracle Solaris Cluster コマンド権プロファイルに割り当てられた Oracle Solaris Cluster の特権コマンドへのアクセスを可能にする特別なシェルです。プロファイルシェルは、su を実行して役割を引き受けたときに起動されます。pfexec を使用しても、Oracle Solaris Cluster の特権コマンドを実行できます。

-k

指定されたリソースグループのリソースのために動作している Resource Group Manager (RGM) のリソースメソッドを即座に強制終了します。

このオプションは、-Q オプションおよび -s と併用できます。-k オプションを指定しないと、終了するか構成されているタイムアウトを超えるまで継続的にメソッドを実行できます。

-M

指定されたリソースのモニタリングを有効化 (-e) または無効化 (-n) します。リソースを無効化する場合、リソースとそのモニターはともにオフラインになっているため、該当リソース上のモニタリングを無効化する必要はありません。

このオプションは、-e または -n のオプションと組み合わせた場合に限り有効です。

このコマンドオプションを -e または -n オプションとともに使用するには、solaris.cluster.resource.admin RBAC の承認が必要です。Unresolved link to " rbac5" を参照してください。

さらにこのコマンドを使用するにあたっては、Oracle Solaris Cluster コマンド権プロファイルを割り当てておく必要があります。認証されたユーザーは、pfsh、pfcsh、または pfksh プロファイルシェルから、Oracle Solaris Cluster の特権コマンドをコマンド行で実行できます。プロファイルシェルは、Oracle Solaris Cluster コマンド権プロファイルに割り当てられた Oracle Solaris Cluster の特権コマンドへのアクセスを可能にする特別なシェルです。プロファイルシェルは、su を実行して役割を引き受けたときに起動されます。pfexec を使用しても、Oracle Solaris Cluster の特権コマンドを実行できます。

例 348 リソースグループのスイッチオーバー

次のコマンドは、resource-grp-2 を schost-1 がマスターするようにスイッチオーバーします。

```
schost-1# scswitch –z –h schost-1 –g resource-grp-2
```

例 349 モニタリングまたはリソースを有効化しない管理対象のリソースグループのオンライン化

次のコマンドは、resource-grp-2 がすでに管理対象になっている場合に resource-grp-2 をオンラインにしますが、任意のリソースを有効にしたり、現在無効になっている任意のリソース上でのモニタリングを有効にしたりすることはありません。

```
schost-1# scswitch –z –g resource-grp-2
```

例 350 複数の主ノードまたはゾーンを持つように構成されているリソースグループのスイッチオーバー

次のコマンドは、複数の主ノードまたはゾーンを持つように構成されているリソースグループ resource-grp-3 を、schost-1、schost-2、schost-3 がマスターするようにスイッチオーバーします。

```
schost-1# scswitch –z –h schost-1,schost-2,schost-3 –g resource-grp-3
```

例 351 すべてのリソースグループおよびデバイスグループのノードからの移動

次のコマンドは、すべてのリソースグループとデバイスグループを schost-1 から新しい主ノードまたはゾーンにスイッチオーバーします。

```
schost-1# scswitch –S –h schost-1
```

例 352 すべてのリソースグループおよびデバイスグループのノードからの永続的な移動

次のコマンドは、すべてのリソースグループとデバイスグループを schost-1 から新しい主ノードまたはゾーンにスイッチオーバーします。このコマンドは、リソースグループとデバイスグループがschost-1 にスイッチバックできるようになるまで、120 秒間待つことも指定します。

次のコマンドでは -K オプションを使用しているので、schost-1 が正常に退避できるまで、リソースグループが自動的に schost-1 にスイッチバックすることを防げます。リソースグループが schost-1 にスイッチバックする例としては、リソースグループが新しいマスター上で起動に失敗した場合などがあります。もう 1 つの例としては、リソースグループに強い否定的なアフィニティーが RG_affinities プロパティーで構成されている場合です。

```
schost-1# scswitch –S –h schost-1 -K 120
```

例 353 リソースの無効化

```
schost-1# scswitch –n –j resource-1,resource-2
```

例 354 リソースの有効化

```
schost-1# scswitch –e –j resource-1
```

例 355 リソースグループを管理対象外にする

```
schost-1# scswitch –u –g resource-grp-1,resource-grp-2
```

例 356 リソースグループを管理対象にする

```
schost-1# scswitch –o –g resource-grp-1,resource-grp-2
```

例 357 デバイスグループのスイッチオーバー

次のコマンドは、device-group-1 を schost-2 がマスターするようにスイッチオーバーします。

```
schost-1# scswitch –z –h schost-2 –D device-group-1
```

例 358 デバイスグループをメンテナンスモードに移行

次のコマンドは、device-group-1 をメンテナンスモードにします。

```
schost-1# scswitch –m –D device-group-1
```

例 359 リソースグループの休止

次のコマンドは、リソースグループ RG1 と RG2 を休止状態にします。

```
schost-1# scswitch –Q -g RG1,RG2
```

例 360 リソースグループのスイッチオーバーによる Start_failed リソース状態のクリア

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

Stop_failed リソース状態とは異なり、Start_failed リソース状態は、ユーザーまたは Oracle Solaris Cluster ソフトウェアがリソースグループに対してアクションを実行することを妨げ*ません*。scswitch -c コマンドを実行して Start_failed リソース状態を解除する必要はありません。該当リソースを再起動するコマンドを実行するだけで済みます。

次のコマンドは、resource-grp-2 リソースグループ内のリソース上で発生した Start_failed リソース状態を解除します。このコマンドは、リソースグループを schost-2 ノードに切り替えることで、この状態を解除します。

```
schost-1# scswitch –z –h schost-2 –g resource-grp-2
```

例 361 リソースグループの再起動による Start_failed リソース状態のクリア

次のコマンドは、resource-grp-2 リソースグループ内のリソース上で発生した Start_failed リソース状態を解除します。このコマンドは、この状態をクリアするために、schost-1 ノード上のこのリソースグループを再起動します。

Start_failed リソース状態の詳細は、1287 ページのr_properties(5) のマニュアルページを参照してください。

```
schost-1# scswitch –R –h schost-1 –g resource-grp-2
```

例 362 リソースの無効化および有効化によるリソース状態 Start_failed のクリア

次のコマンドでは、リソース resource-1 上で発生したリソース状態 Start_failed を、リソースをいったん無効化したあと再度有効化することにより、クリアします。

Start_failed リソース状態の詳細は、1287 ページのr_properties(5) のマニュアルページを参照してください。

```
schost-1# scswitch –n –j resource-1
schost-1# scswitch –e –j resource-1
```

このコマンドは、要求されたアクションが完全に終了するまで、あるいは、エラーが発生するまで待機します。

次の終了値が返されます。

0	コマンドは正常に完了しました。
0 以外	エラーが発生しました。scswitch は、エラーメッセージを標準エラーに書き込みます。

scswitch コマンドがゼロ以外の終了ステータスで終了して、「cluster is reconfiguring」というエラーメッセージが表示された場合、エラーにかかわらず、要求された操作は正常に完了している可能性があります。結果が疑わしい場合は、再構成が完了してから、もう一度、scswitch コマンドを同じ引数を指定して実行します。

scswitch コマンドがゼロ以外の終了ステータスで終了して、「Resource group failed to start on chosen node and may fail over to other node(s)」というエラーメッセージが表示された場合、scswitch コマンドが終了したあと、しばらくの間、そのリソースグループは再構成を続けます。リソースグループがすべてのノード上で終了ステータス (つまり、Online、Online_faulted、または Offline の状態) になるまで、そのノード上で scswitch または clresourcegroup 操作を実行しても失敗します。

複数のリソースまたはリソースグループに対して scswitch コマンドを呼び出した結果、複数のエラーが発生した場合、終了値は、複数のエラーのうちの 1 つだけを反映している可能性があります。この可能性を避けるため、scswitch コマンドは、1 度に 1 つのリソースまたはリソースグループだけに実行するようにします。

RG_system プロパティーが True のリソースグループ (および所属リソース) では、一部の処理が実行できません。詳細は、1319 ページのrg_properties(5) を参照してください。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	廃止

Unresolved link to " kill1", Unresolved link to " pfcsh1", Unresolved link to " pfexec1", Unresolved link to " pfksh1", Unresolved link to " pfsh1", 19 ページの`Intro`(1CL), 79 ページの`cldevicegroup`(1CL), 305 ページの`clresourcegroup`(1CL), Unresolved link to " su1M", Unresolved link to " attributes5", Unresolved link to " rbac5", 1287 ページの`r_properties`(5), 1319 ページの`rg_properties`(5)

Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド "

`-z` または `-F` オプションと `-g` オプションを使ってリソースグループをオフラインにしたあと、ノードをリブートした場合、リソースグループのオフライン状態 (`Offline`) は保存されません。ノードが異常終了したり、ノードがクラスタに参加したり、あるいは、ほかのリソースグループがスイッチオーバーしたりした場合、そのリソースグループはオンラインになることかあります。以前、そのリソースグループをオフラインに切り換えていた場合でも、そのリソースグループはノード上でオンラインになります。すべてのリソースを無効にしていても、リソースグループはオンラインになります。

リソースグループが自動的にオンラインになるのを防ぐためには、`-s` オプションを使用して、リソースグループの自動回復アクションを中断します。自動回復アクションを再開するには、`-r` オプションを使用します。

名前

sctelemetry — システムリソースモニタリングの初期化

```
sctelemetry -d

sctelemetry -e

sctelemetry -i -o hasp_rg=rg,hasp_rs=rs [,hasp_mnt_pt=mnt_pt]
     [,db_rg=rg] [,db_rs=rs] [,telemetry_rg=rg] [,telemetry_rs=rs]

sctelemetry -i -o hasp_mnt_pt=mnt_pt,hasp_nodelist=node[:…]
     [,hasp_rs=rs] [,db_rg=rg] [,db_rs=rs]
     [,telemetry_rg=rg,telemetry_rs=rs]

sctelemetry -u
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

`sctelemetry` コマンドは、システムリソースモニタリングを初期化し、モニタリングをオンラインまたはオフラインにします。初期化する場合は、`-o` オプションを `hasp_rg=`*rg*、`hasp_rs=`*rs* パラメータと使用して、`SUNW.HAStoragePlus` タイプの既存のリソースに依存します。`-o` オプションと `hasp_mnt_pt=`*mnt_pt*`,hasp_nodelist=`*node*`[:,...]` パラメータを併用して、`sctelemetry` コマンドで `SUNW.HAStoragePlus` リソースタイプを作成します。リソースタイプの詳細は、1371 ページの`SUNW.derby`(5)、1389 ページの`SUNW.HAStoragePlus`(5)、および 1431 ページの`SUNW.SCTelemetry`(5) のマニュアルページを参照してください。

`SUNW.SCTelemetry` はマルチマスターリソースグループでインスタンス化されており、これはつまり、このリソースグループはすべてのクラスタノードで構成されており、ネットワーク負荷分散を使用しないことを意味します。

このコマンドは、大域ゾーンだけで使用できます。

`sctelemetry` のオプションは次のとおりです。

`-d`

システムリソース使用率データの収集と遠隔測定データが格納されるデータベースを無効にします。

このオプションは大域ゾーンだけで使用できます。

スーパーユーザー以外のユーザーが sctelemetry コマンドの -d オプションを使用するには、solaris.cluster.system.modify RBAC 認証が必要です。詳細は、Unresolved link to " rbac5" マニュアルページを参照してください。

-e

システムリソース使用率データの収集をオンラインにします。sctelemetry コマンドの -i オプションを使用すると、デフォルトでは、システムリソースモニタリングがオンラインになります。

このオプションは大域ゾーンだけで使用できます。

スーパーユーザー以外のユーザーが sctelemetry コマンドの -e オプションを使用するには、solaris.cluster.system.modify RBAC 認証が必要です。詳細は、Unresolved link to " rbac5" マニュアルページを参照してください。

-i

リソースタイプ SUNW.SCTelemetry と SUNW.derby を含むリソースグループを作成します。デフォルトでは、-i オプションを使用してこれらのリソースおよびリソースグループを作成するときに、システムリソースモニタリングがオンラインになっています。

このオプションは大域ゾーンだけで使用できます。

スーパーユーザー以外のユーザーが sctelemetry コマンドの -i オプションを使用するには、solaris.cluster.system.modify RBAC 認証が必要です。詳細は、Unresolved link to " rbac5" マニュアルページを参照してください。

-o hasp_rg=*rg*,hasp_rs=*rs*[,hasp_mnt_pt=*mnt_pt*][,db_rg=*rg*][,db_rs=*rs*] [,telemetry_rg=*rg*] [,telemetry_rs=*rs*]

-i オプションと一緒に使用するときは、データベースで使用されるリソースタイプ SUNW.HAStoragePlus とこのリソースを含むリソースグループを指定します。データ収集機能は、SUNW.HAStoragePlus 用のファイルシステムへのアクセス権が必要です。

パラメータは次のようになります。

hasp_rg=*rg*

システムリソースモニタリングに使用されるリソースタイプ SUNW.HAStoragePlus を含むリソースグループ。*rg*、つまり、このリソースグループの名前を指定してください。

hasp_rs=*rs*

システムリソースモニタリングに使用されるリソースタイプ SUNW.HAStoragePlus 。*rs*、つまり、このリソースの名前を指定してください。

hasp_mnt_pt=*mnt_pt*

sctelemetry がシステムリソースモニタリング用のデータベースファイルを格納するマウントポイント。このマウントポイントは、リソース hasp_rs のプロパティーでなければなりません。hasp_rs に 1 つを超えるマウントポイントがある場合、マウントポイントの指定は必須です。

db_rg=*rg*

`sctelemetry` がリソースタイプ `SUNW.derby` を構成するリソースグループ。*rg*、つまり、このリソースグループの名前を指定できます。

db_rs=*rs*

`SUNW.derby` が構成するリソースタイプ `sctelemetry` 。*rs*、つまり、このリソースの名前を指定できます。

telemetry_rg=*rg*

`sctelemetry` がリソースタイプ `SUNW.SCTelemetry` を構成するリソースグループ。*rg*、つまり、このリソースグループの名前を指定できます。

telemetry_rs=*rs*

`sctelemetry` が構成するリソースタイプ `SUNW.SCTelemetry`。*rs*、つまり、このリソースの名前を指定できます。

`-o` hasp_mnt_pt=*mnt_pt*,hasp_nodelist=*node*[:…][,hasp_rs=*rs*][,db_rg=*rg*] [,db_rs=*rs*] [,telemetry_rg=*rg*][,telemetry_rs=*rs*]

`-i` オプションと一緒に使用すると、データ収集用の `SUNW.HAStoragePlus` ファイルシステムにアクセス可能なノードを指定し、Oracle Solaris Cluster がシステムリソースデータを格納するファイルシステムのマウントポイントを指定します。

パラメータは次のようになります。

hasp_mnt_pt=*mnt_pt*

`sctelemetry` がリソースタイプ `SUNW.HAStoragePlus`を構成するのに使用するマウントポイント。*mnt_pt*、つまり、このマウントポイントの名前を指定してください。`HAStoragePlus` リソースを作成する前に、共有ストレージを構成してくださいこのマウントポイントは共有ストレージを参照するため、`/etc/vfstab` に次のような行が必要です。

```
/dev/md/ddg/dsk/d20 /dev/md/ddg/rdsk/d20  /mntpt ufs  2  no logging
```

hasp_nodelist=*node*[:...]

`sctelemetry` がリソースタイプ `SUNW.HAStoragePlus` を構成するノード。*node*:[...]、つまり、このノードの名前を指定してください。

hasp_rs=*rs*

`sctelemetry` が構成するリソースタイプ `SUNW.HAStoragePlus`。*rs*、つまり、このリソースの名前を指定できます。

db_rg=*rg*

`sctelemetry` がリソースタイプ `SUNW.derby` を構成するリソースグループ。*rg*、つまり、このリソースグループの名前を指定できます。

db_rs=*rs*

SUNW.derby が構成するリソースタイプ sctelemetry 。*rs*、つまり、このリソースの名前を指定できます。

telemetry_rg=*rg*

sctelemetry がリソースタイプ SUNW.SCTelemetry を構成するリソースグループ。*rg*、つまり、このリソースグループの名前を指定できます。

telemetry_rs=*rs*

sctelemetry が構成するリソースタイプ SUNW.SCTelemetry。*rs*、つまり、このリソースの名前を指定できます。

-u

-i オプションを使用して以前作成したリソースとリソースグループを削除します。

このオプションは大域ゾーンだけで使用できます。

スーパーユーザー以外のユーザーが sctelemetry コマンドの -u オプションを使用するには、solaris.cluster.system.modify RBAC 認証が必要です。詳細は、Unresolved link to " rbac5" マニュアルページを参照してください。

HAStoragePlus リソースが存在するときのシステムリソースモニタリングの初期化

この例では、システムリソースモニタリングを初期化して、モニタリングが初期化されていることを確認します。この例では、SUNW.HAStoragePlus リソースがシステムリソースモニタリング用に利用できると仮定します。

この例では、リソース *db_rs* と *telemetry_rs*、またはリソースグループ *db_rg* と *telemetry_rg* の名前は指定しません。sctelemetry コマンドは、これらのリソースおよびリソースグループにデフォルトの名前を付けます。

scstat -g コマンドの出力は、システムリソースモニタリングに関係のあるリソースとリソースグループ間の関係を示しています。この出力はまた、リソース *db_rs* と *hasp_rs*、そして、リソースグループ *db_rg* がそれぞれ 1 つのノード上で online であり、*telemetry_rg* と *telemetry_rs* がすべてのクラスタノードで online であることを示しています。

```
# sctelemetry -i \
-o hasp_mnt_pt=DBDATA,hasp_nodelist=l6-lx-1:l6-lx-4,hasp_rs=anto

# scstat -g
```

システムリソースモニタリングの無効化

この例では、システムリソースモニタリングを無効にして、モニタリングが無効にされていることを確認します。モニタリングを無効にすると、`scstat -g` コマンドの出力は、リソース *db_rs*、*hasp_rs*、および *telemetry_rs* 、そして、リソースグループ *db_rg* と *telemetry_rg* が `offline` であることを示しています。

```
# sctelemetry -d
# scstat -g
```

次の終了値が返されます。

`0`	コマンドは正常に完了しました。
0 以外	エラーが発生しました。

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

19 ページの`Intro`(1CL), 535 ページの`cltelemetryattribute`(1CL), 551 ページの`cluster`(1CL), 877 ページの`scstat`(1M), 899 ページの`sctelemetry`(1M), 1371 ページの`SUNW.derby`(5), 1389 ページの`SUNW.HAStoragePlus`(5), 1431 ページの`SUNW.SCTelemetry`(5)

名前

scversions — Oracle Solaris Cluster のバージョン管理

```
scversions [-c]
```

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページのIntro(1CL) のマニュアルページを参照してください。

`scversions` コマンドは、新しい Oracle Solaris Cluster ソフトウェアへの段階的アップグレードを行なったあと、クラスタの機能を新しいレベルに引き上げます。引数を指定しない場合、`scversions` は、コミットが必要かどうかを示すメッセージを出力します。

次のオペランドがサポートされています。

`-c`

クラスタの現在アクティブなメンバーであるノード全体の機能を可能な最高のレベルにコミットします。

あるノードをアップグレードして (つまり、新しいリリースの製品にアップグレードしたり、パッチを適用して)、そのノードをクラスタに戻したりしたとき、クラスタ内の (まだアップグレードしていない) ほかのノードと正しく協調するためには、そのノードのいくつかの内部プロトコルをバージョンを下げて動作させる必要があります。このような状態のクラスタでは、いくつかの管理アクションが無効になり、アップグレードで導入されるいくつかの新しい機能を利用できない場合があります。

すべてのノードをアップグレードしたあと、このコマンドを任意のノードから実行すると、そのクラスタの内部プロトコルは可能な最高のバージョンに切り替わります。このとき、すべてのノードに同じ Oracle Solaris Cluster ソフトウェアがインストールされていれば、新しい機能がすべて利用できるようになり、管理制限もすべてなくなります。

クラスタのアクティブなメンバーであるノードの 1 つをアップグレードせずに、`-c` オプションを指定して `scversions` を実行した場合、そのクラスタはすでにその時点で可能な最高のレベルの機能で動作しているため、このコマンドは何も影響しません。

クラスタのアクティブなメンバーでない (たとえば、保守のために停止している) ノードの 1 つをアップグレードせずに、`-c` オプションを指定して `scversions` を実行した場合、そのクラスタの内部プロトコルは可能な最高のバージョンに切り替わります。クラスタのアクティブなメンバーでない

ノードでも、そのクラスタに戻す予定がある場合は、アップグレードしておく必要があります。

0	成功
0 以外	失敗

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

819 ページの`scinstall`(1M)

OSC4 3ha

名前

scds_calls — Oracle Solaris Cluster Data Services Development Library (DSDL) 関数

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
```

Data Services Development Library (DSDL) は、scha ライブラリ関数の機能をカプセル化および拡張する、上位のライブラリ関数セットです。scha ライブラリ関数は、1047 ページのscha_calls(3HA) のマニュアルページで説明されています。

DSDL 関数は libdsdev.so ライブラリで実装されています。

DSDL 関数は、一般に次のカテゴリに分類されます。

- 汎用関数

  汎用関数には、初期化関数、取得関数、フェイルオーバーおよび再起動関数、そして実行関数が含まれます。これらの関数では、次の操作を行うことができます。

  - DSDL 環境を初期化します。
  - リソースタイプ、リソース、リソースグループの名前と、拡張プロパティーの値を取得します。
  - リソースグループをフェイルオーバーおよび再起動し、リソースを再起動します。
  - エラー文字列をエラーメッセージに変換します。
  - タイムアウトを適用してコマンドを実行します。
- プロパティー関数

  これらの関数は、関連するリソース、リソースタイプ、およびリソースグループに固有のプロパティー (よく使用される一部の拡張プロパティーも含む) にアクセスするために使用する API を提供します。DSDL は、scds_initialize() 関数を使用してコマンド行引数を解析します。関連するリソース、リソースタイプ、およびリソースグループのさまざまなプロパティーを*キャッシュ*に書き込みます。
- ネットワークリソースアクセス関数

  これらの関数は、リソースおよびリソースグループによって使用されるネットワーク資源を管理します。これらの関数は、ホスト名、ポートリスト、およびネットワークアドレスを処理し、TCP ベースのモニタリングを有効にします。
- プロセスモニター機能 (PMF) 関数

  これらの関数は、Process Monitor Facility (PMF) の機能をカプセル化しています。

- 障害モニター関数

  これらの関数は、障害履歴を保持し、その履歴を `Retry_count` および `Retry_interval` プロパティーと関連付けて評価することにより、障害モニタリングの事前定義モデルを提供します。
- ユーティリティー関数

  これらの関数は、メッセージやデバッグ用メッセージをシステムログに書き込みます。

初期化関数

次の関数は、呼び出しメソッドを初期化します。

- 987 ページの`scds_initialize`(3HA) – リソースを割り当て、DSDL 環境を初期化します。
- 917 ページの`scds_close`(3HA) – `scds_initialize()` 関数によって割り当てられたリソースを解放します。

取得関数

次の関数は、リソースタイプ、リソース、リソースグループ、および拡張プロパティーに関する情報を取得します:

- 947 ページの`scds_free_ext_property`(3HA) – `scds_get_ext_property()` によって割り当てられたメモリーを解放します。
- 961 ページの`scds_get_fullname`(3HA) – ゾーンのノード名を取得します。
- 963 ページの`scds_get_fullname_nodeid`(3HA) – ゾーンのノード名を ASCII ノード ID 番号と一緒に取得します。
- 973 ページの`scds_get_resource_type_name`(3HA) – 呼び出し元プログラム用のリソースタイプの名前を取得します。
- 971 ページの`scds_get_resource_name`(3HA) – 呼び出し元プログラム用のリソースの名前を取得します。
- 969 ページの`scds_get_resource_group_name`(3HA) – 呼び出し元プログラム用のリソースグループの名前を取得します。
- 957 ページの`scds_get_ext_property`(3HA) – 指定した拡張プロパティーの値を取得します。
- 955 ページの`scds_get_current_method_name`(3HA) – データサービスメソッドの呼び出しに使用したパス名の最後の要素を取得します。Unresolved link to "basename3C" のマニュアルページを参照してください。

- 991 ページのscds_is_zone_cluster(3HA) – リソースがゾーンクラスタに構成されているかどうかを示すブール値を返します。

次の関数は、リソースによって使用される SUNW.HAStoragePlus リソースに関するステータス情報を取得します。

983 ページのscds_hasp_check(3HA) – リソースによって使用される SUNW.HAStoragePlus リソースに関するステータス情報を取得します。当該リソース用に定義されている Resource_dependencies または Resource_dependencies_weak のシステム属性を使用することによって、当該リソースが依存しているすべての SUNW.HAStoragePlus リソース状態 (オンラインであるか、オンラインでないか) についての情報が得られます。詳細は、1389 ページのSUNW.HAStoragePlus(5) のマニュアルページを参照してください。

フェイルオーバー関数と再起動関数

次の関数は、リソースまたはリソースグループをフェイルオーバーまたは再起動します。

- 923 ページのscds_failover_rg(3HA) – リソースグループをフェイルオーバーします。
- 1027 ページのscds_restart_rg(3HA) – リソースグループを再起動します。
- 1025 ページのscds_restart_resource(3HA) – リソースを再起動します。

実行関数

次の関数は、タイムアウトを適用してコマンドを実行し、エラーコードをエラーメッセージに変換します。

- 1043 ページのscds_timerun(3HA) – タイムアウト値を適用してコマンドを実行します。
- 919 ページのscds_error_string(3HA) および 921 ページのscds_error_string_i18n(3HA) – エラーコードをエラー文字列に変換します。scds_error_string() から返された文字列は、英語で表示されます。scds_error_string_i18n() から返された文字列は、LC_MESSAGES ロケールカテゴリで指定されているその国および地域の言語で表示されます。
- 1035 ページのscds_svc_wait(3HA) - 指定のタイムアウト時間が経過するまで、モニター対象のプロセスの終了を待機します。

プロパティー関数

これらの関数は、関連するリソースタイプ、リソース、およびリソースグループの特定のプロパティー (よく使用される一部の拡張プロパティーも含む) にアクセスするために使用する簡易 API を提供します。DSDL は、scds_initialize() 関数を使用してコマンド行引数を解析しま

す。関連するリソース、リソースタイプ、およびリソースグループのさまざまなプロパティーを*キャッシュ*に書き込みます。

次の関数を始めとするこれらの関数の説明は、1017 ページの`scds_property_functions(3HA)` のマニュアルページにあります。

- `scds_get_ext_`*property-name*
- `scds_get_rg_`*property-name*
- `scds_get_rs_`*property-name*
- `scds_get_rt_`*property-name*

ネットワークリソースアクセス関数

これらの関数を使用して、ネットワークリソースを管理します。

次の関数はホスト名を扱います。

- 979 ページの`scds_get_rs_hostnames(3HA)` – リソースによって使用されているホスト名のリストを取得します。
- 975 ページの`scds_get_rg_hostnames(3HA )` – リソースグループ内のネットワークリソースによって使用されているホスト名のリストを取得します。
- 1011 ページの`scds_print_net_list(3HA)` – ホスト名リストの内容を Unresolved link to " syslog3C" に書き込みます。通常、この関数はデバッグに使用します。
- 949 ページの`scds_free_net_list(3HA)` – `scds_get_rs_hostnames()` または `scds_get_rg_hostnames()` によって割り当てられたメモリーを解放します。

次の関数はポートリストを扱います。

- 967 ページの`scds_get_port_list(3HA)` – リソースによって使用されているポートとプロトコルのペアのリストを取得します。
- 1015 ページの`scds_print_port_list(3HA)` – ポートとプロトコルのリストの内容を Unresolved link to " syslog3C" に書き込みます。通常、この関数はデバッグに使用します。
- 953 ページの`scds_free_port_list(3HA)` – `scds_get_port_list()` によって割り当てられたメモリーを解放します。

次の関数はネットワークアドレスを扱います。

- 965 ページの`scds_get_netaddr_list(3HA)` – リソースによって使用されているネットワークアドレスのリストを取得します。

- 1013 ページのscds_print_netaddr_list(3HA) – ネットワークアドレスのリストの内容を Unresolved link to " syslog3C" に書き込みます。通常、この関数はデバッグに使用します。
- 951 ページのscds_free_netaddr_list(3HA) – scds_get_netaddr_list() によって割り当てられたメモリーを解放します。

次のカテゴリの関数は、TCP ベースのモニタリングを有効にします。通常、障害モニターはこれらの関数を使用して、サービスとの単純ソケット接続を確立し、サービスのデータを読み書きしてサービスのステータスを確認したあと、サービスとの接続を切断します。

この関数セットには、次の関数が含まれます。

- 939 ページのscds_fm_tcp_connect(3HA) – IPv4 アドレッシングだけを使用するプロセスへの TCP 接続を確立します。
- 929 ページのscds_fm_net_connect(3HA) – IPv4 または IPv6 アドレッシングを使用するプロセスへの TCP 接続を確立します。
- 943 ページのscds_fm_tcp_read(3HA) – TCP 接続を使用して、モニターされているプロセスからデータを読み取ります。
- 945 ページのscds_fm_tcp_write(3HA) – TCP 接続を使用して、モニターされているプロセスにデータを書き込みます。
- 1033 ページのscds_simple_probe(3HA) – プロセスへの TCP 接続を確立および終了することによって、プロセスをプローブします。この関数は IPv4 アドレスだけを扱います。
- 1029 ページのscds_simple_net_probe(3HA) – プロセスへの TCP 接続を確立および終了することによって、プロセスをプローブします。この関数は、IPv4 または IPv6 アドレスを扱います。
- 941 ページのscds_fm_tcp_disconnect(3HA) – モニターされているプロセスへの接続を終了します。この関数は IPv4 アドレスだけを扱います。
- 933 ページのscds_fm_net_disconnect(3HA) – モニターされているプロセスへの接続を終了します。この関数は、IPv4 または IPv6 アドレスを扱います。

PMF 関数

これらの関数は、プロセスモニター機能 (PMF) の機能をカプセル化します。PMF 経由のモニタリングの DSDL モデルは、pmfadm に対して暗黙の *tag* 値を作成し、使用します。詳細は、735 ページのpmfadm(1M) のマニュアルページを参照してください。

また、PMF 機能は、Restart_interval、Retry_count、および action_script 用の暗黙値も使用します (pmfadm の -t、-n、および -a オプション)。もっとも重要な点は、DSDL が PMF によっ

て検出されたプロセス障害履歴を、障害モニターによって検出されたアプリケーション障害履歴に結びつけ、再起動またはフェイルオーバーのどちらを行うかを決定することです。

このカテゴリには次のような関数があります。

- 993 ページの`scds_pmf_get_status`(3HA) – 指定したインスタンスが PMF の制御の下でモニターされているかどうかを判別します。
- 995 ページの`scds_pmf_restart_fm`(3HA) – PMF を使用して障害モニターを再起動します。
- 997 ページの`scds_pmf_signal`(3HA) – PMF の制御の下で実行されているプロセスツリーに指定したシグナルを送信します。
- 999 ページの`scds_pmf_start`(3HA) および 1003 ページの`scds_pmf_start_env`(3HA) – 指定したプログラム (障害モニターを含む) を PMF の制御の下で実行します。`scds_pmf_start_env ()`関数は、`scds_pmf_start()`関数と同じ処理を実行するほか、指定された環境を実行プログラムに渡します。
- 1007 ページの`scds_pmf_stop`(3HA) – PMF の制御の下で実行されているプロセスを終了します。
- 1009 ページの`scds_pmf_stop_monitoring`(3HA) – PMF の制御の下で実行されているプロセスのモニタリングを停止します。

障害モニター関数

これらの関数は、障害履歴を保持し、それを `Retry_count` および `Retry_interval` プロパティーと組み合わせて評価することによって、障害モニタリングの事前定義モデルを提供します。

このカテゴリには次のような関数があります。

- 937 ページの`scds_fm_sleep`(3HA) – 障害モニター制御ソケットでメッセージを待機します。
- 925 ページの`scds_fm_action`(3HA) – プローブ完了のあとでアクションを実行します。
- 935 ページの`scds_fm_print_probes`(3HA) – プローブステータス情報をシステムログに書き込みます。

ユーティリティー関数

次の関数では、メッセージやデバッグメッセージをシステムログに書き込むことができます。

- 1039 ページの`scds_syslog`(3HA) – メッセージをシステムログに書き込みます。

- 1041 ページの`scds_syslog_debug`(3HA) – デバッグメッセージをシステムログに書き込みます。

`/usr/cluster/include/scds.h`	インクルードファイル
`/usr/cluster/lib/libdsdev.so`	ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	発展中

735 ページの`pmfadm`(1M), 917 ページの`scds_close`(3HA),
919 ページの`scds_error_string`(3HA),
921 ページの`scds_error_string_i18n`(3HA), 923 ページの`scds_failover_rg`(3HA),
925 ページの`scds_fm_action`(3HA), 929 ページの`scds_fm_net_connect`(3HA),
933 ページの`scds_fm_net_disconnect`(3HA),
935 ページの`scds_fm_print_probes`(3HA),
937 ページの`scds_fm_sleep`(3HA), 939 ページの`scds_fm_tcp_connect`(3HA),
941 ページの`scds_fm_tcp_disconnect`(3HA),
943 ページの`scds_fm_tcp_read`(3HA), 945 ページの`scds_fm_tcp_write`(3HA),
947 ページの`scds_free_ext_property`(3HA),
949 ページの`scds_free_net_list`(3HA),
951 ページの`scds_free_netaddr_list`(3HA),
953 ページの`scds_free_port_list`(3HA),
957 ページの`scds_get_ext_property`(3HA),
961 ページの`scds_get_fullname`(3HA), 965 ページの`scds_get_netaddr_list`(3HA),
967 ページの`scds_get_port_list`(3HA),
969 ページの`scds_get_resource_group_name`(3HA),
971 ページの`scds_get_resource_name`(3HA),
973 ページの`scds_get_resource_type_name`(3HA),
975 ページの`scds_get_rg_hostnames`(3HA),

979 ページの scds_get_rs_hostnames(3HA), 983 ページの scds_hasp_check(3HA), 987 ページの scds_initialize(3HA), 991 ページの scds_is_zone_cluster(3HA), 993 ページの scds_pmf_get_status(3HA), 995 ページの scds_pmf_restart_fm(3HA), 997 ページの scds_pmf_signal(3HA), 999 ページの scds_pmf_start(3HA), 1007 ページの scds_pmf_stop(3HA), 1009 ページの scds_pmf_stop_monitoring(3HA), 1011 ページの scds_print_net_list(3HA), 1013 ページの scds_print_netaddr_list(3HA), 1015 ページの scds_print_port_list(3HA), 1017 ページの scds_property_functions(3HA), 1025 ページの scds_restart_resource(3HA), 1027 ページの scds_restart_rg(3HA), 1029 ページの scds_simple_net_probe(3HA), 1033 ページの scds_simple_probe(3HA), 1035 ページの scds_svc_wait(3HA), 1039 ページの scds_syslog(3HA), 1041 ページの scds_syslog_debug(3HA), 1043 ページの scds_timerun(3HA), 1047 ページの scha_calls(3HA), 1389 ページの SUNW.HAStoragePlus(5), Unresolved link to " attributes5"

名前

scds_close — DSDL 環境リソースの解放

```
cc [flags…]-I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     void scds_close(scds_handle_t*handle);
```

`scds_close()` 関数は、987 ページの`scds_initialize`(3HA) を使用して、データサービスメソッドの初期化中に割り当てられたリソースを再利用します。この関数は、プログラムの終了前に 1 回呼び出します。

次のパラメータがサポートされます。

`handle` `scds_initialize()` から返されるハンドルです。

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

987 ページの`scds_initialize`(3HA), Unresolved link to " attributes5"

名前

scds_error_string, scds_error_string_i18n — エラーコードからエラー文字列の作成

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     const char *scds_error_string(scha_err_t error_code);

const char *scds_error_string_i18n(scha_err_t error_code);
```

`scds_error_string()` および `scds_error_string_i18n()` 関数は、DSDL 関数から返されたエラーコードのエラーを説明する簡単な文字列を生成します。`scds_error_string()` から返された文字列は、英語で表示されます。`scds_error_string_i18n()` から返された文字列は、`LC_MESSAGES` ロケールカテゴリで指定されているその国および地域の言語で表示されます。Unresolved link to " setlocale3C" を参照してください。エラーコードが無効な場合は、`NULL` が返されます。

この関数から返されるポインタは、DSDL に属するメモリーをポイントします。このメモリーは変更できません。

次のパラメータがサポートされます。

`error_code` DSDL 関数によって返されるエラーコードです。

`/usr/cluster/include/rgm/libdsdev.h`
インクルードファイル

`/usr/cluster/lib/libdsdev.so`
ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

1047 ページの`scha_calls`(3HA), Unresolved link to " setlocale3C", Unresolved link to " attributes5"

名前

scds_error_string, scds_error_string_i18n — エラーコードからエラー文字列の作成

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     const char *scds_error_string(scha_err_t error_code);

const char *scds_error_string_i18n(scha_err_t error_code);
```

`scds_error_string()` および `scds_error_string_i18n()` 関数は、DSDL 関数から返されたエラーコードのエラーを説明する簡単な文字列を生成します。`scds_error_string()` から返された文字列は、英語で表示されます。`scds_error_string_i18n()` から返された文字列は、`LC_MESSAGES` ロケールカテゴリで指定されているその国および地域の言語で表示されます。Unresolved link to " setlocale3C" を参照してください。エラーコードが無効な場合は、`NULL` が返されます。

この関数から返されるポインタは、DSDL に属するメモリーをポイントします。このメモリーは変更できません。

次のパラメータがサポートされます。

`error_code` DSDL 関数によって返されるエラーコードです。

`/usr/cluster/include/rgm/libdsdev.h`
インクルードファイル

`/usr/cluster/lib/libdsdev.so`
ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

1047 ページの`scha_calls`(3HA), Unresolved link to " setlocale3C", Unresolved link to " attributes5"

名前

scds_failover_rg — リソースグループのフェイルオーバー

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>scha_err_t scds_failover_rg(
     scds_handle_t handle);
```

`scds_failover_rg()` 関数は、呼び出し元プログラムに渡されるリソースを含むリソースグループ上で 1101 ページの`scha_control`(3HA) `SCHA_GIVEOVER` 操作を実行します。

この関数は、正常に実行されると戻りません。したがって、この関数を、呼び出し元プログラムで最後に実行されるコードにしてください。

次のパラメータがサポートされます。

`handle` 987 ページの`scds_initialize`(3HA) から返されるハンドルです。

次の戻り値がサポートされています。

`SCHA_ERR_NOERR` 関数が正常に終了。

その他の値 関数の実行に失敗。その他のエラーコードについては、1047 ページの`scha_calls`(3HA) を参照してください。

`/usr/cluster/include/rgm/libdsdev.h`

インクルードファイル

`/usr/cluster/lib/libdsdev.so`

ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

1047 ページの`scha_calls`(3HA), 1101 ページの`scha_control`(3HA), Unresolved link to " attributes5"

名前

scds_fm_action — プローブ完了関数のあとでアクションを実行する

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
  -l dsdev#include <rgm/libdsdev.h> scha_err_t
  scds_fm_action(scds_handle_t handle, int probe_status,
  long elapsed_milliseconds);
```

`scds_fm_action()` 関数は、データサービスの `probe_status` と過去の障害の履歴を元に、次のいずれかのアクションを実行します。

- アプリケーションを再起動します。
- リソースグループのフェイルオーバーを行います。
- 何もしません。

引数 `probe_status` の値で、障害の重要度を表します。たとえば、アプリケーションに接続する処理の失敗を完全な失敗とみなし、アプリケーションとの接続を切断する処理を部分的な失敗とみなすことができます。後者の場合、`probe_status` に 0 から `SCDS_PROBE_COMPLETE_FAILURE` までの値を指定します。

DSDL は `SCDS_PROBE_COMPLETE_FAILURE` を `100` として定義します。成功または失敗の部分的なプローブには、`0` から `SCDS_PROBE_COMPLETE_FAILURE` までの値を使用します。

DSDL は `SCDS_PROBE_IMMEDIATE_FAILOVER` を `201` に定義します。`Failover_mode` プロパティーが `RESTART_ONLY` または `LOG_ONLY` に設定されないかぎり、このプローブステータスはリソースグループの即時フェイルオーバーをトリガーします。最初に再起動を試行することなく即時のフェイルオーバー試行を強制するには、特別な `SCDS_PROBE_IMMEDIATE_FAILOVER` 値を使用します。`Failover_mode` プロパティーの詳細は、1287 ページの`r_properties(5)` を参照してください

その後、`scds_fm_action()` を呼び出すことにより、`probe_status` 入力パラメータの値とリソースの `Retry_interval` プロパティーによって定義された時間間隔が足し合わされ、障害履歴が計算されます。`Retry_interval` よりも古い障害履歴はメモリーからパージされ、再起動やフェイルオーバーの決定には使用されません。

`scds_fm_action()` 関数は、次のアルゴリズムを使って、実行するアクションを選択します。

Restart

累積された障害履歴が SCDS_PROBE_COMPLETE_FAILURE に達すると、scds_fm_action() はリソースの STOP メソッドと START メソッドを順番に呼び出し、リソースを再起動します。リソースタイプに定義された PRENET_START メソッドや POSTNET_STOP メソッドは無視します。

リソースのステータスは、そのリソースがすでに設定されている場合を除き、scha_resource_setstatus() 呼び出しによって SCHA_RSSTATUS_DEGRADED に設定されます。

リソースの START メソッドや STOP メソッドの失敗により再起動の試行に失敗した場合、scha_control() が GIVEOVER オプションで呼び出され、リソースグループが別のノードにフェイルオーバーされます。scha_control() 呼び出しに成功した場合、リソースグループは別のクラスタノードにフェイルオーバーされ、scds_fm_action() 呼び出しは値を返しません。

再起動に成功すると、障害履歴はパージされます。次の再起動は、障害履歴がふたたび SCDS_PROBE_COMPLETE_FAILURE に達するまで試行されません。

Failover

scds_fm_action() 呼び出しが繰り返されることによって、試行される再起動回数がリソースに定義された Retry_count 値に達した場合、scha_control() と GIVEOVER オプションが呼び出され、フェイルオーバーが試行されます。

リソースのステータスは、そのリソースがすでに設定されている場合を除き、scha_resource_setstatus() 呼び出しによって SCHA_RSSTATUS_FAULTED に設定されます。

scha_control() 呼び出しに失敗した場合、scds_fm_action() によって管理されている障害履歴全体がクリアされます。

scha_control() 呼び出しに成功した場合、リソースグループは別のクラスタノードにフェイルオーバーされ、scds_fm_action() 呼び出しは値を返しません。

probe_status 値に SCDS_PROBE_IMMEDIATE_FAILOVER を指定することによって、プローブは再起動を行わずに即時フェイルオーバー試行をトリガーできます。

No Action

障害の蓄積された履歴が SCDS_PROBE_COMPLETE_FAILURE の下に残っている場合、アクションは行われません。さらに、サービスのチェックが正常に行われ、probe_status 値が 0 になっている場合も、障害履歴は考慮されず、アクションは実行されません。

リソースのステータスは、そのリソースがすでに設定されている場合を除き、scha_resource_setstatus() 呼び出しによって SCHA_RSSTATUS_OK に設定されます。

次のパラメータがサポートされます。

handle

987 ページのscds_initialize(3HA) から返されるハンドルです。

probe_status

データサービスのステータスを示す、0 から SCDS_PROBE_COMPLETE_FAILURE または SCDS_PROBE_IMMEDIATE_FAILOVER までの間で指定する数値です。

- 0 は、最近のデータサービスの検査が正常に実行されたことを示します。
- SCDS_PROBE_COMPLETE_FAILURE は、完全な障害を示します。サービスは完全に失敗しました。サービスの部分的な障害を示す、0 から SCDS_PROBE_COMPLETE_FAILURE までの値を指定することもできます。
- 値が SCDS_PROBE_IMMEDIATE_FAILURE のときは、Failover_mode プロパティーが RESTART_ONLY または LOG_ONLY に設定された場合を除き、再起動を行わずにリソースグループのフェイルオーバーがトリガーされます。Failover_mode プロパティーの詳細は、1287 ページの r_properties(5) を参照してください

elapsed_milliseconds

データサービスの検査を完了するまでの時間 (ミリ秒) です。この値は、将来の拡張のために予約されています。

scds_fm_action() 関数の戻り値は次のとおりです。

0	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_NOERR

アクションは実行されませんでした。または、再起動が正常に試行されました。

SCHA_ERR_FAIL

フェイルオーバーが試行されましたが、正常に実行されませんでした。

SCHA_ERR_NOMEM

システムメモリーが不足しています。

/usr/cluster/include/rgm/libdsdev.h

インクルードファイル

/usr/cluster/lib/libdsdev.so

ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

937 ページの`scds_fm_sleep`(3HA), 987 ページの`scds_initialize`(3HA),
1047 ページの`scha_calls`(3HA), 1101 ページの`scha_control`(3HA),
935 ページの`scds_fm_print_probes`(3HA),
1175 ページの`scha_resource_setstatus`(3HA), Unresolved link to " attributes5"

名前

scds_fm_net_connect — アプリケーションとの TCP 接続の確立

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/
libdsdev.h> scha_err_t scds_fm_net_connect(scds_handle_t handle, scds_socket_t
 *socklist, int count, scds_netaddr_t addr, time_t timeout);
```

`scds_fm_net_connect()` 関数は、モニター対象のプロセスとの 1 つまたは複数の TCP 接続を確立します。TCP 接続の数は、`Port_list` にある各アドレスのプロトコル値によって異なります (後述)。

965 ページの`scds_get_netaddr_list(3HA)` を使用して、リソースのネットワークアドレスのリストを取得できます。この関数はまた、リスト内の各アドレスのプロトコル値を設定します。あるアドレスのプロトコルが `tcp6` であると `Port_list` に指定されている場合、プロトコル値は `SCDS_IPPROTO_TCP6` に設定されます。あるアドレスのプロトコルが `tcp` であると `Port_list` に指定されている場合、または `Port_list` にプロトコルが指定されていない場合、プロトコル値は `SCDS_IPPROTO_TCP` に設定されます。

この関数はまた、`addr` に指定された `hostname` を解決して、次のように接続します。

- `addr` に指定されたプロトコルが `SCDS_IPPROTO_TCP` である場合、指定されたポートの `hostname` の IPv4 アドレスに接続します。
- `addr` に指定されたプロトコルが `SCDS_IPPROTO_TCP6` である場合、指定されたポートの `hostname` の IPv4 アドレスと IPv6 アドレスの両方 (それぞれ存在する場合のみ) に接続します。ステータスとファイル記述子は (利用できる場合のみ)、この関数に指定された `scds_socket_t` 配列に格納されます。この配列の 1 番目のメンバーは IPv4 マッピングに使用され、2 番目のメンバーは IPv6 用に使用されます。ステータスに設定できる値は、次のいずれかです。
  - `SCDS_FMSOCK_OK` — 操作は成功しました。関連するソケットファイル記述子は有効です。
  - `SCDS_FMSOCK_NA` — この `hostname` には、アドレスの種類 (IPv4 または IPv6) が適用されません。`hostname` に 1 つまたは複数の IPv4 マッピングしかない場合、この関数に指定された配列の 2 番目のメンバーのステータスは `SCDS_FMSOCK_NA` に設定されます。関連するソケットファイル記述子は未知の値に設定されるため、絶対に使用できません。

- SCDS_FMSOCK_ERR — 操作は失敗またはタイムアウトしました。関連するソケットファイル記述子は未知の値に設定されるため、絶対に使用できません。

次のパラメータがサポートされます。

handle

987 ページのscds_initialize(3HA) から返されるハンドルです。

socklist

メンバー SCDS_MAX_IPADDR_TYPES から構成される型 scds_socket_t の配列です。配列内の各メンバーは、TCP 接続のステータスとソケットファイル記述子を保持します。このパラメータは、この関数が設定する出力引数です。

count

socklist 配列のメンバー数です。このパラメータは SCDS_MAX_IPADDR_TYPES に設定します。

addr

プロセスが待機する場所を指定する hostname、TCP ポート番号、およびプロトコル識別子です。

timeout

タイムアウト値 (秒) です。各ソケットはこれと同じ時間をタイムアウトとして使用して、接続を確立しようとします。これらの時間は並行に進行するため、この値は事実上、関数の実行にかかる最大時間となります。

scds_fm_net_connect() 関数の戻り値は次のとおりです。

0	関数の実行に成功。少なくとも 1 つのソケットが接続されました。
SCHA_ERR_INVAL	この関数は無効なパラメータを使用して呼び出されました。
0 以外	タイムアウトや接続の拒否などのエラーのため、接続は確立されていません。エラーを正確に判断するには、socklist 配列のすべてのメンバーについて、status フィールドが SCDS_FMSOCK_ERR に設定されているかどうかを調査します。

SCHA_ERR_NOERR

関数が正常に終了。

SCHA_ERR_INTERNAL

関数の実行中に内部エラーが発生したことを示します。

SCHA_ERR_STATE

サーバーが接続要求を拒否したことを示します。

SCHA_ERR_TIMEOUT

接続要求がタイムアウトしたことを示します。

例 363 scds_fm_net_connect() 関数の使用法

```
/* this function is called repeatedly,
   after thorough_probe_interval seconds */
int probe(scds_handle_t scds_handle, ...)
{
 scds_socket_t socklist[SCDS_MAX_IPADDR_TYPES];
 ...

 /* for each hostname/port/proto */
 for (i = 0; i < netaddr->num_netaddrs, i++) {
  if (scds_fm_net_connect(scds_handle, socklist,
      SCDS_MAX_IPADDR_TYPES, netaddr[i], timeout) !=
      SCHA_ERR_NOERR)
   {
    /* failed completely */
    ...
  } else {
   /* at least one sock connected */
   for (j = 0, j < SCDS_MAX_IPADDR_TYPES, j++) {
    if (socklist[j].status == SCDS_FM_SOCK_NA)
     continue;

    if (socklist[j].status == SCDS_FMSOCK_ERR) {
     /* this particular connection failed */
     scds_syslog(LOG_ERR, "Failed: %s",
         scds_error_string(socklist[j].err));
     continue;
    }

    /* use socklist[i].fd to perform write/read */
    ...
   }
   (void) scds_fm_net_disconnect(scds_handle, socklist,
       SCDS_MAX_IPADDR_TYPES, remaining_time);
  }

 }
 ...
 return (result);
```

```
}
```

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	発展中

933 ページの`scds_fm_net_disconnect`(3HA),
939 ページの`scds_fm_tcp_connect`(3HA),
965 ページの`scds_get_netaddr_list`(3HA), 987 ページの`scds_initialize`(3HA),
1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5"

名前

scds_fm_net_disconnect — アプリケーションとの TCP 接続の終了

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/
libdsdev.h> scha_err_t scds_fm_net_disconnect(scds_handle_t handle, scds_socket_t
 *socklist, int count, time_t timeout);
```

`scds_fm_net_disconnect()` 関数は、モニターされているプロセスとの TCP 接続を終了します。

指定された `timeout` 間隔で、`socklist` 配列内のすべての有効なソケット接続の終了が試行されます。終了時、`socklist` の各メンバーに `SCDS_FMSOCK_NA` という値が追加されます。

次のパラメータがサポートされます。

`handle`
: 987 ページの`scds_initialize(3HA)` から返されるハンドルです。

`socklist`
: 929 ページの`scds_fm_net_connect(3HA)` から返されるソケットリストです。これは入出力引数です。

`count`
: `socklist` 配列のメンバー数です。このパラメータは `SCDS_MAX_IPADDR_TYPES` に設定します。

`timeout`
: タイムアウト値 (秒) です。各ソケットはこれと同じ時間をタイムアウトとして使用して、接続を切断しようとします。これらの時間は並行に進行するため、この値は事実上、関数の実行にかかる最大時間となります。

`scds_fm_net_disconnect()` 関数の戻り値は次のとおりです。

`0`	関数の実行に成功。
`SCHA_ERR_INVAL`	この関数は無効なパラメータを使用して呼び出されました。
その他の 0 以外の値	関数の実行に失敗。障害コードの意味については、1047 ページの`scha_calls(3HA)` を参照してください。

`/usr/cluster/include/rgm/libdsdev.h`

インクルードファイル

`/usr/cluster/lib/libdsdev.so`

ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	発展中

929 ページの`scds_fm_net_connect`(3HA),
941 ページの`scds_fm_tcp_disconnect`(3HA), 987 ページの`scds_initialize`(3HA),
1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5"

名前

scds_fm_print_probes — プローブデバッグ情報の印刷

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/libdsdev.h>
     void scds_fm_print_probes(scds_handle_t handle, int debug_level);
```

`scds_fm_print_probes()` 関数は、925 ページの`scds_fm_action`(3HA) によって報告されたプローブステータス情報をシステムログに書き込みます。この情報には、DSDL によって管理されるすべてのプローブステータス履歴とタイムスタンプのリストが含まれます。

DSDL の定義では、最大デバッグレベル `SCDS_MAX_DEBUG_LEVEL` は `9` です。

現在使用中のデバッグレベルよりも `debug_level` を上げた場合、情報は書き込まれません。

次のパラメータがサポートされます。

`handle` 987 ページの`scds_initialize`(3HA) から返されるハンドルです。

`debug_level` データが書き込まれるデバッグレベルです。`1` から `SCDS_MAX_DEBUG_LEVEL` (DSDL では `9` と定義) までの整数値です。

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

925 ページの`scds_fm_action`(3HA), 987 ページの`scds_initialize`(3HA), 1041 ページの`scds_syslog_debug`(3HA), Unresolved link to " attributes5"

名前

scds_fm_sleep — 障害モニター制御ソケットでメッセージを待機

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_fm_sleep(scds_handle_t handle, time_t timeout);
```

`scds_fm_sleep()` 関数は、プロセスモニター機能の制御下で実行されているデータサービスアプリケーションプロセスツリーの終了を待機します。指定のタイムアウト時間内に終了しない場合は、`SCHA_ERR_NOERR` を返します。

データサービスアプリケーションのプロセスツリーが終了した場合、`scds_fm_sleep()` は障害履歴内に `SCDS_COMPLETE_FAILURE` を記録し、プロセスツリーを再起動するか、925 ページの`scds_fm_action`(3HA) のマニュアルページに説明されているアルゴリズムに従ってフェイルオーバーを実行します。フェイルオーバーの試行に失敗した場合、アプリケーションの再起動が試行されます。

再起動の試行に失敗した場合、`SCHA_ERR_INTERNAL` が返されます。

障害履歴によってフェイルオーバーを試行し、成功した場合、`scds_fm_sleep()` は値を返しません。

次のパラメータがサポートされます。

`handle` 987 ページの`scds_initialize`(3HA) から返されるハンドルです。

`timeout` タイムアウト時間 (秒) です。

`scds_fm_sleep()` 関数の戻り値は次のとおりです。

`0` 関数の実行に成功。

0 以外 関数の実行に失敗。

`SCHA_ERR_NOERR` プロセスツリーが終了していない。

`SCHA_ERR_INTERNAL` データサービスアプリケーションプロセスツリーは終了したが、再起動に失敗。

その他の値　　関数の実行に失敗。障害コードの意味については、1047 ページのscha_calls(3HA) を参照してください。

/usr/cluster/include/rgm/libdsdev.h

インクルードファイル

/usr/cluster/lib/libdsdev.so

ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

1047 ページのscha_calls(3HA), 925 ページのscds_fm_action(3HA), 987 ページのscds_initialize(3HA), Unresolved link to " attributes5"

名前

scds_fm_tcp_connect — アプリケーションへの TCP 接続の確立

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_fm_tcp_connect(scds_handle_t handle,
     int *sock, const char*hostname, int port, time_t timeout);
```

`scds_fm_tcp_connect()` 関数は、モニター対象のプロセスとの TCP 接続を確立します。

979 ページの`scds_get_rs_hostnames(3HA)` と
975 ページの`scds_get_rg_hostnames(3HA )` のどちらかを使用してホスト名を取得します。

この関数の代わりに 929 ページの`scds_fm_net_connect(3HA)` の使用を検討します。

次のパラメータがサポートされます。

`handle`

987 ページの`scds_initialize(3HA)` から返されるハンドルです。

`sock`

この関数によって確立されるソケットへのハンドルです。このパラメータは、この関数が設定する出力引数です。

`hostname`

プロセスが待機しているホスト名です。`hostname` が IPv4 アドレスだけにマッピングするか、IPv4 と IPv6 アドレスの両方にマッピングする場合、この関数は接続するアドレスとして IPv4 マッピングを使用します。`hostname` が IPv6 アドレスだけにマッピングする場合、この関数は接続するアドレスとして IPv6 マッピングを使用します。

`port`

TCP ポート番号です。

`timeout`

タイムアウト値 (秒) です。

`scds_fm_tcp_connect()` 関数の戻り値は次のとおりです。

`0`　　関数の実行に成功。

0 以外　　関数の実行に失敗。

SCHA_ERR_NOERR
: 関数が正常に終了。

SCHA_ERR_STATE
: ソケットでの接続を初期化する試みがタイムアウト以外の理由で失敗。

SCHA_ERR_TIMEOUT
: 関数がタイムアウト。

その他の値
: 関数の実行に失敗。障害コードの意味については、1047 ページのscha_calls(3HA) を参照してください。

/usr/cluster/include/rgm/libdsdev.h
: インクルードファイル

/usr/cluster/lib/libdsdev.so
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	非推奨

929 ページのscds_fm_net_connect(3HA),
941 ページのscds_fm_tcp_disconnect(3HA),
975 ページのscds_get_rg_hostnames(3HA),
979 ページのscds_get_rs_hostnames(3HA), 987 ページのscds_initialize(3HA),
1047 ページのscha_calls(3HA), Unresolved link to " attributes5"

名前

scds_fm_tcp_disconnect — アプリケーションとの TCP 接続の終了

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_fm_tcp_disconnect(scds_handle_t handle, int sock, time_t timeout);
```

`scds_fm_tcp_disconnect()` 関数は、モニター対象のプロセスとの TCP 接続を終了します。

次のパラメータがサポートされます。

`handle`	987 ページの`scds_initialize`(3HA) から返されるハンドルです。
`sock`	以前の 939 ページの`scds_fm_tcp_connect`(3HA) 呼び出しから返されるソケット番号です。
`timeout`	タイムアウト値 (秒) です。

次の終了値が返されます。

`0`	関数の実行に成功。
0 以外	関数の実行に失敗。

`SCHA_ERR_NOERR`	関数が正常に終了。
`SCHA_ERR_TIMEOUT`	関数がタイムアウト。
その他の値	関数の実行に失敗。障害コードの意味については、1047 ページの`scha_calls`(3HA) を参照してください。

`/usr/cluster/include/rgm/libdsdev.h`

インクルードファイル

`/usr/cluster/lib/libdsdev.so`

ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	非推奨

933 ページのscds_fm_net_disconnect(3HA), 939 ページのscds_fm_tcp_connect(3HA), 987 ページのscds_initialize(3HA), 1047 ページのscha_calls(3HA), Unresolved link to " attributes5"

名前

scds_fm_tcp_read — アプリケーションとの TCP 接続を使用したデータの読み取り

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_fm_tcp_read(scds_handle_t handle, int sock,
     char *buffer, size_t *size, time_t timeout);
```

`scds_fm_tcp_read()` 関数は、モニター対象のプロセスと TCP 接続からデータを読み取ります。

`size` 引数は入力および引数です。入力時にバッファーのサイズをバイト数で指定します。完了時に、この関数は、`buffer` 内にデータを配置し、実際に読み取られたバイト数を `size` に指定します。バッファーのサイズが読み取られるバイト数より小さい場合、`size` バイトの完全なバッファーを返します。この関数は、さらにデータを取得するために再度呼び出すことができます。

タイムアウトになった場合は、`SCHA_ERR_TIMEOUT` を返します。この場合、関数は、要求されたバイト数より少ないバイト数 (`size` で表される) を返します。

次のパラメータがサポートされます。

`handle`
: 987 ページの`scds_initialize(3HA)` から返されるハンドルです。

`sock`
: 以前の 939 ページの`scds_fm_tcp_connect(3HA)` 呼び出しから返されるソケット番号です。

`buffer`
: データバッファーです。

`size`
: データバッファーサイズです。入力時にバッファーのサイズをバイト数で指定します。出力時、関数は実際に読み取られたバイト数を返します。

`timeout`
: タイムアウト値 (秒) です。

`scds_fm_tcp_read()` 関数の戻り値は次のとおりです。

`0` 関数の実行に成功。

0 以外 関数の実行に失敗。

`SCHA_ERR_NOERR`
: 関数が正常に終了。

`SCHA_ERR_TIMEOUT`
: 関数がタイムアウト。

その他の値
: 関数の実行に失敗。障害コードの意味については、1047 ページの`scha_calls`(3HA) を参照してください。

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

941 ページの`scds_fm_tcp_disconnect`(3HA), 945 ページの`scds_fm_tcp_write`(3HA), 987 ページの`scds_initialize`(3HA), 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5"

名前

scds_fm_tcp_write — アプリケーションとの TCP 接続を使用したデータの書き込み

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_fm_tcp_write(scds_handle_t handle, int sock, char *buffer,
     size_t *size, time_t timeout);
```

`scds_fm_tcp_write()` 関数は、モニター対象のプロセスへの TCP 接続を使ってデータを書き込みます。

`size` 引数は入力および出力引数です。入力時には書き込まれるバイト数を指定します。出力時には実際に書き込まれたバイト数を返します。`size` の入力値と出力値が一致しない場合、エラーが発生します。要求されたすべてのデータが書き込まれる前にタイムアウトになった場合、`SCHA_ERR_TIMEOUT` が返されます。

次のパラメータがサポートされます。

`handle`
: 987 ページの`scds_initialize(3HA)` から返されるハンドルです。

`sock`
: 以前の 939 ページの`scds_fm_tcp_connect(3HA)` 呼び出しから返されるソケット番号です。

`buffer`
: データバッファーです。

`size`
: データバッファーサイズです。入力時には書き込まれるバイト数を指定します。出力時には実際に書き込まれたバイト数を返します。

`timeout`
: タイムアウト値 (秒) です。

`scds_fm_tcp_write()` 関数の戻り値は次のとおりです。

`0`	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_NOERR
: 関数が正常に終了。

SCHA_ERR_TIMEOUT
: 関数がタイムアウト。

その他の値
: 関数の実行に失敗。障害コードの意味については、1047 ページのscha_calls(3HA) を参照してください。

/usr/cluster/include/rgm/libdsdev.h
: インクルードファイル

/usr/cluster/lib/libdsdev.so
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

939 ページのscds_fm_tcp_connect(3HA), 943 ページのscds_fm_tcp_read(3HA), 987 ページのscds_initialize(3HA), 1047 ページのscha_calls(3HA), Unresolved link to " attributes5"

名前

scds_free_ext_property — リソース拡張プロパティーメモリーの解放

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/libdsdev.h>
     void scds_free_ext_property(scha_ext_prop_value_t *property_value);
```

`scds_free_ext_property()` 関数は、957 ページの`scds_get_ext_property`(3HA) の呼び出し中に割り当てられたメモリーを再利用します。

次のパラメータがサポートされます。

`property_value`　　プロパティー値のポインタです。

`/usr/cluster/include/rgm/libdsdev.h`
　インクルードファイル

`/usr/cluster/lib/libdsdev.so`
　ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

957 ページの`scds_get_ext_property`(3HA), Unresolved link to " attributes5"

名前

scds_free_net_list — ネットワーク資源メモリーの解放

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/libdsdev.h>
     void scds_free_net_list(scds_net_resource_list_t *net_resource_list);
```

`scds_free_net_list()` 関数は、975 ページの`scds_get_rg_hostnames`(3HA) または 979 ページの`scds_get_rs_hostnames`(3HA) の呼び出し中に割り当てられたメモリーを再利用します。`netresource_list` が指すメモリーの割り当てを解除します。

次のパラメータがサポートされます。

`netresource_list` リソースグループによって使用されるネットワークリソースのリストへのポインタです。

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

975 ページの`scds_get_rg_hostnames`(3HA),
979 ページの`scds_get_rs_hostnames`(3HA), Unresolved link to " attributes5"

名前

scds_free_netaddr_list — ネットワークアドレスメモリーの解放

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/libdsdev.h>
     void scds_free_netaddr_list(scds_netaddr_list_t *netaddr_list);
```

`scds_free_netaddr_list()` 関数は、965 ページの`scds_get_netaddr_list`(3HA) の呼び出し中に割り当てられたメモリーを再利用します。`netaddr_list` が指すメモリーの割り当てを解除します。

次のパラメータがサポートされます。

`netaddr_list`
: リソースグループによって使用される、ホスト名、ポート、プロトコルの組み合わせのリストを指すポインタです。

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

965 ページの`scds_get_netaddr_list`(3HA), Unresolved link to " attributes5"

名前

scds_free_port_list — ポートリストメモリーの解放

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/libdsdev.h>
     void scds_free_port_list(scds_port_list_t *port_list);
```

`scds_free_port_list()` 関数は、967 ページの`scds_get_port_list`(3HA) の呼び出し中に割り当てられたメモリーを再利用します。`port_list` が指すメモリーの割り当てを解除します。

次のパラメータがサポートされます。

`port_list`
: リソースグループによって使用されるポートとプロトコルのペアのリストへのポインタです。

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

967 ページの`scds_get_port_list`(3HA), Unresolved link to " attributes5"

名前

scds_get_current_method_name — 呼び出されたデータサービスメソッドによるパス名の最終要素の取得

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>const char *
     scds_get_current_method_name(scds_handle_t handle);
```

`scds_get_current_method_name()` 関数は、文字列に対するポインタを返します。この文字列には、呼び出されたデータサービスメソッドによるパスの最終要素が含まれています。

詳細は、Unresolved link to " basename3C" のマニュアルページを参照してください。

文字列に対するポインタは、Data Service Development Library (DSDL) に属するメモリーをポイントします。このメモリーは変更できません。このポインタは、`scds_close()` 呼び出しによって無効化されます。

次のパラメータがサポートされます。

`handle` 987 ページの`scds_initialize`(3HA) から返されるハンドルです。

`SCHA_ERR_NOERR` 関数の実行に成功。

その他のエラーコードについては、1047 ページの`scha_calls`(3HA) を参照してください。

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

917 ページの scds_close(3HA), 987 ページの scds_initialize(3HA),
1047 ページの scha_calls(3HA), Unresolved link to " attributes5"

名前

scds_get_ext_property — 拡張プロパティーの取得

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_get_ext_property(scds_handle_t handle,
     const char *property_name, scha_prop_type_t property_type,
     scha_extprop_value_t **property_value);
```

`scds_get_ext_property()` 関数は、指定の拡張プロパティーの値を取得します。

まず、メソッド引数リスト(`argv[]`。`scds_initialize()` によって解析される) 内に指定されたプロパティーリストからプロパティー名を検索します。プロパティー名がメソッド引数リスト内に見つからない場合は、Oracle Solaris Cluster API を使って検索を行ないます。1047 ページの`scha_calls`(3HA) を参照してください。

正常に完了すると、プロパティーの値は `scha_extprop_value_t` 構造の共用体で適切な変数に配置され、この構造へのポインタが `property_value` の呼び出し元へ戻されます。

ユーザーは、`scds_free_ext_property()` を使用してメモリーを解放しなくてはなりません。

データ型 `scha_prop_type_t` および `scha_extprop_value_t` に関する情報は、1047 ページの`scha_calls`(3HA) 内および `scha_types.h` ヘッダーファイル内にあります。

DSDL は、より頻繁に使用されるリソース拡張プロパティーの取得に便利な関数を提供します。1017 ページの`scds_property_functions`(3HA) のマニュアルページを参照してください。

次のパラメータがサポートされます。

`handle`
: 987 ページの`scds_initialize`(3HA) から返されるハンドルです。

`property_name`
: 取得されるプロパティー名です。

`property_type`
: プロパティー値の型です。有効な型は、1047 ページの`scha_calls`(3HA) および 1271 ページの`property_attributes`(5) で定義されています。

property_value

プロパティー値のポインタです。

scds_get_ext_property() 関数の戻り値は次のとおりです。

0　　関数の実行に成功。

0 以外　　関数の実行に失敗。

SCHA_ERR_PROP

RTR ファイルは指定のプロパティーを定義しない。

SCHA_ERR_NOERR

関数の実行に成功。

その他の値

関数の実行に失敗。障害コードの意味については、1047 ページのscha_calls(3HA) を参照してください。

例 364　scds_get_ext_property() の使用法

```
#include <scha_types.h>
#include <libdsdev.h>
#define INT_EXT_PROP “Int_extension_property”
...
int  retCode;
scha_extprop_value_t *intExtProp;
int retrievedValue;
...
    retCode = scds_get_ext_property(handle,
        INT_EXT_PROP, SCHA_PTYPE_INT, &intExtProp);
    if (retCode != SCHA_ERR_NOERR) {
        scds_syslog(LOG_ERR,
            "Failed to retrieve the extension property %s: %s.",
            INT_EXT_PROP, scds_error_string(retCode));
       ...
    } else {
        retrievedValue = intExtProp->val.val_int;
        ...
        scds_free_ext_property(intExtProp);
        ...
    }
   ...
```

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

947 ページの`scds_free_ext_property`(3HA), 987 ページの`scds_initialize`(3HA), 1017 ページの`scds_property_functions`(3HA), 1047 ページの`scha_calls`(3HA), 1251 ページの`rt_reg`(4), Unresolved link to " attributes5", 1271 ページの`property_attributes`(5)

この関数で取得できるのは、RTR ファイル内に定義された拡張プロパティー値だけです。1251 ページの`rt_reg`(4) を参照してください。

名前

scds_get_fullname, scds_get_fullname_nodeid — ゾーンのノード名へのポインタの取得。

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>scha_err_t scds_get_fullname(
     const char *zonename, char **fullname,boolean_t is_zone_cluster);

scha_err_t scds_get_fullname_nodeid(const char *zonename,
     char **fullname,boolean_t is_zone_cluster);
```

`scds_get_fullname()` 関数はゾーンのノード名を `fullname` で返します。`scds_get_fullname_nodeid()` 関数は、ノード名の代わりに ASCII ノード ID 番号を付けてゾーンのノード名を返します。呼び出し元は、`fullname` で指定されたメモリーを解放する責任があります。

`is_zone_cluster` が true の場合、`zonename` はローカルホストのゾーンクラスタの名前を提供します。返される値はローカルホストのゾーンクラスタノード名です。

`is_zone_cluster` を false に設定することは、Oracle Solaris Cluster 3.3 リリースバージョンにのみ適用できます。現時点では、それは Oracle Solaris Cluster 4.x リリースでは使用されません。

`zonename` の値は NULL 以外である必要があり、そうでない場合は `SCHA_ERR_INVAL` が返されて、`fullname` の値は変更されません。

ゾーンクラスタノードに対して (`is_zone_cluster` を true に設定した場合)、`scds_get_fullname` によって返される `fullname` 値の例を次に示します。

```
"zcnode1"
```

zcnode1 のノード ID 番号が 2 の場合、対応する `scds_get_fullname_nodeid` の出力は、次のようになります。

```
"2"
```

次のパラメータがサポートされます。

`zonename` ゾーンクラスタまたはグローバルクラスタの非大域ゾーンの名前を提供します。

`is_zone_cluster` zonename がゾーンクラスタ名かどうかを示します。

`fullname` 返されるノード名文字列を指定する出力パラメータ。

`SCHA_ERR_NOERR` 関数の実行に成功。

その他のエラーコードについては、1047 ページの`scha_calls`(3HA) を参照してください。

`/usr/cluster/include/rgm/libdsdev.h` インクルードファイル

`/usr/cluster/lib/libdsdev.so` ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	SUNWscdev
インタフェースの安定性	発展中

909 ページの`scds_calls`(3HA), 991 ページの`scds_is_zone_cluster`(3HA), 1047 ページの`scha_calls`(3HA), 1239 ページの`scha_strerror`(3HA), Unresolved link to " attributes5"

名前

scds_get_fullname, scds_get_fullname_nodeid — ゾーンのノード名へのポインタの取得。

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>scha_err_t scds_get_fullname(
     const char *zonename, char **fullname,boolean_t is_zone_cluster);

scha_err_t scds_get_fullname_nodeid(const char *zonename,
     char **fullname,boolean_t is_zone_cluster);
```

`scds_get_fullname()` 関数はゾーンのノード名を `fullname` で返します。`scds_get_fullname_nodeid()` 関数は、ノード名の代わりに ASCII ノード ID 番号を付けてゾーンのノード名を返します。呼び出し元は、`fullname` で指定されたメモリーを解放する責任があります。

`is_zone_cluster` が true の場合、`zonename` はローカルホストのゾーンクラスタの名前を提供します。返される値はローカルホストのゾーンクラスタノード名です。

`is_zone_cluster` を false に設定することは、Oracle Solaris Cluster 3.3 リリースバージョンにのみ適用できます。現時点では、それは Oracle Solaris Cluster 4.x リリースでは使用されません。

`zonename` の値は NULL 以外である必要があり、そうでない場合は `SCHA_ERR_INVAL` が返されて、`fullname` の値は変更されません。

ゾーンクラスタノードに対して (`is_zone_cluster` を true に設定した場合)、`scds_get_fullname` によって返される `fullname` 値の例を次に示します。

```
"zcnode1"
```

zcnode1 のノード ID 番号が 2 の場合、対応する `scds_get_fullname_nodeid` の出力は、次のようになります。

```
"2"
```

次のパラメータがサポートされます。

`zonename` ゾーンクラスタまたはグローバルクラスタの非大域ゾーンの名前を提供します。

`is_zone_cluster` zonename がゾーンクラスタ名かどうかを示します。

`fullname` 返されるノード名文字列を指定する出力パラメータ。

`SCHA_ERR_NOERR` 関数の実行に成功。

その他のエラーコードについては、1047 ページの`scha_calls`(3HA) を参照してください。

`/usr/cluster/include/rgm/libdsdev.h` インクルードファイル

`/usr/cluster/lib/libdsdev.so` ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	SUNWscdev
インタフェースの安定性	発展中

909 ページの`scds_calls`(3HA), 991 ページの`scds_is_zone_cluster`(3HA), 1047 ページの`scha_calls`(3HA), 1239 ページの`scha_strerror`(3HA), Unresolved link to " attributes5"

名前

scds_get_netaddr_list — リソースによって使用されるネットワークアドレスの取得

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/
libdsdev.h>scha_err_t scds_get_netaddr_list(scds_handle_t handle, scds_netaddr_list_t
 **netaddr_list);
```

`scds_get_netaddr_list()` 関数は、リソースが使用しているホスト名、ポート、プロトコルのすべての組み合わせを返します。これらの組み合わせは、リソースの `Port_list` プロパティー設定と、リソースが使用しているすべてのホスト名 (`scds_get_rs_hostnames()` 関数が返す値) を組み合わせることによって生成されます。

リソースをモニターしたり、リソースが使用しているホスト名、ポート、およびプロトコルのリストを得たりするには、`scds_get_netaddr_list()` を障害モニターで使用します。

プロトコルタイプの値は、ヘッダーファイル `rgm/libdsdev.h` で定義されます。

この関数が割り当て、返すメモリーを解放するには、`scds_free_netaddr_list()` を使用します。

次のパラメータがサポートされます。

`handle` `scds_initialize()` から返されるハンドルです。

`netaddr_list` リソースグループが使用するホスト名、ポート、およびプロトコルのリストです。

`scds_get_netaddr_list()` 関数の戻り値は次のとおりです。

`0` 関数の実行に成功。

0 以外 関数の実行に失敗。

`SCHA_ERR_NOERR` 関数が正常に終了。

その他の値 関数の実行に失敗。障害コードの意味については、1047 ページの`scha_calls(3HA)` を参照してください。

`/usr/cluster/include/rgm/libdsdev.h`

インクルードファイル

`/usr/cluster/lib/libdsdev.so`

ライブラリ

次の属性の詳細は、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	発展中

951 ページの`scds_free_netaddr_list`(3HA),
979 ページの`scds_get_rs_hostnames`(3HA), 1047 ページの`scha_calls`(3HA),
1287 ページの`r_properties`(5), Unresolved link to " attributes5"

名前

scds_get_port_list — リソースによって使用されるポートリストの取得

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_get_port_list(scds_handle_t handle, scds_port_list_t **port_list);
```

`scds_get_port_list()` 関数は、リソースによって使用されるポートとプロトコルのペアのリストを返します。プロトコル型の値は、ヘッダーファイル `netinet/in.h` に定義されます。

この関数が割り当て、返すメモリーを解放するには、`scds_free_port_list()` を使用します。

次のパラメータがサポートされます。

`handle` `scds_initialize()` から返されるハンドルです。

`port_list` リソースグループによって使用されるポートとプロトコルのペアのリストです。

`scds_get_port_list()` 関数の戻り値は次のとおりです。

`0` 関数の実行に成功。

0 以外 関数の実行に失敗。

`SCHA_ERR_NOERR` 関数が正常に終了。

その他の値 関数の実行に失敗。障害コードの意味については、1047 ページの`scha_calls`(3HA) を参照してください。

`/usr/cluster/include/scha.h`
: インクルードファイル

`/usr/cluster/lib/libscha.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

953 ページの`scds_free_port_list`(3HA), 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5"

名前

scds_get_resource_group_name — リソースグループ名の取得

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/libdsdev.h>
      const char *scds_get_resource_group_name(scds_handle_t handle);
```

`scds_get_resource_group_name()` 関数は、呼び出し元プログラムに渡されるリソースが含まれる、リソースグループ名へのポインタを返します。このポインタは、DSDL に属するメモリーを指定しています。このメモリーは変更できません。このポインタは、`scds_close()` 呼び出しによって無効化されます。

次のパラメータがサポートされます。

`handle`
: `scds_initialize()` から返されるハンドルです。

`NULL`
: 以前に 987 ページの`scds_initialize`(3HA) を呼び出していないなどのエラー条件を示します。

その他のエラーコードについては、1047 ページの`scha_calls`(3HA) を参照してください。

`/usr/cluster/include/scha.h`
: インクルードファイル

`/usr/cluster/lib/libscha.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

917 ページの`scds_close`(3HA), 987 ページの`scds_initialize`(3HA), 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5"

名前

scds_get_resource_name — リソース名の取得

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/libdsdev.h>
     const char *scds_get_resource_name(scds_handle_t handle);
```

`scds_get_resource_name()` 関数は、呼び出し元プログラムに渡されるリソース名が含まれる、文字列へのポインタを返します。このポインタは、DSDL に属するメモリーを指定しています。このメモリーは変更できません。このポインタは、`scds_close()` 呼び出しによって無効化されます。

次のパラメータがサポートされます。

`handle` `scds_initialize()` から返されるハンドルです。

`NULL` 以前に 987 ページの`scds_initialize`(3HA) を呼び出していないなどのエラー条件を示します。

その他のエラーコードについては、1047 ページの`scha_calls`(3HA) を参照してください。

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

917 ページの`scds_close`(3HA), 987 ページの`scds_initialize`(3HA), 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5"

名前

scds_get_resource_type_name — リソースタイプ名の取得

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/libdsdev.h>
     const char *scds_get_resource_type_name(scds_handle_t handle);
```

`scds_get_resource_type_name()` 関数は、呼び出し元プログラムに渡されるリソースのリソースタイプ名が含まれる、文字列へのポインタを返します。このポインタは、DSDL に属するメモリーを指定しています。したがって、このメモリーは変更できません。このポインタは、`scds_close()` 呼び出しによって無効化されます。

次のパラメータがサポートされます。

`handle`
: `scds_initialize()` から返されるハンドルです。

`NULL`
: 以前に `scds_initialize()` を呼び出していないなどのエラー条件を示します。

その他のエラーコードについては、1047 ページの`scha_calls`(3HA) を参照してください。

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

917 ページの`scds_close`(3HA), 987 ページの`scds_initialize`(3HA), 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5"

名前

scds_get_rg_hostnames, scds_get_rg_hostnames_zone — リソースグループで使用されるネットワークリソースの取得

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>scha_err_t
     scds_get_rg_hostnames(char *resourcegroup_name,
     scds_net_resource_list_t **netresource_list);

scha_err_t scds_get_rg_hostnames_zone(char *zone_name, char *
     resourcegroup_name, scds_net_resource_list_t **netresource_list);
```

#1405

`scds_get_rg_hostnames()` および `scds_get_rg_hostnames_zone()` 関数は、リソースグループ内のすべてのネットワークリソースで使用されているホスト名のリストを取得します。`scds_get_rg_hostnames_zone()` 関数を大域ゾーンから実行すると、特定のゾーンクラスタのリソースグループからリストを取得できます。この関数は、`netresource_list` 内のリストへのポインタを返します。リソースグループにネットワークリソースがないか、またはネットワークリソースを使用しないリリソースが含まれる場合があり、これらの関数は `NULL` に設定されている `netresource_list` パラメータを返すことがあります。

システムの任意のリソースグループ名を `scds_get_rg_hostnames()` および `scds_get_rg_hostnames_zone()` に渡すことができます。指定されたリソースグループで実行中のアプリケーションと通信するには、`scds_get_rg_hostnames()` および `scds_get_rg_hostnames_zone()` により返されたホスト名を使用します。

この関数が割り当て、返すメモリーを解放するには、`scds_free_net_list()` を使用します。

次のパラメータがサポートされます。

`resourcegroup_name`	リソースグループ名です。このリソースグループのデータが取得されます。
`netresource_list`	リソースグループによって使用されるネットワーク資源のリスト

`scds_get_rg_hostnames()` および `scds_get_rg_hostnames_zone()` 関数は、次の値を返します:

`0`	関数の実行に成功。

0 以外　　関数の実行に失敗。

SCHA_ERR_NOERR　　関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

/usr/cluster/include/rgm/libdsdev.h
　インクルードファイル

/usr/cluster/lib/libdsdev.so
　ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

949 ページのscds_free_net_list(3HA), 979 ページのscds_get_rs_hostnames(3HA), 1047 ページのscha_calls(3HA), Unresolved link to " attributes5"

名前

scds_get_rg_hostnames, scds_get_rg_hostnames_zone — リソースグループで使用されるネットワークリソースの取得

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>scha_err_t
     scds_get_rg_hostnames(char *resourcegroup_name,
     scds_net_resource_list_t **netresource_list);

scha_err_t scds_get_rg_hostnames_zone(char *zone_name, char *
     resourcegroup_name, scds_net_resource_list_t **netresource_list);
```

#1405

`scds_get_rg_hostnames()` および `scds_get_rg_hostnames_zone()` 関数は、リソースグループ内のすべてのネットワークリソースで使用されているホスト名のリストを取得します。`scds_get_rg_hostnames_zone()` 関数を大域ゾーンから実行すると、特定のゾーンクラスタのリソースグループからリストを取得できます。この関数は、`netresource_list` 内のリストへのポインタを返します。リソースグループにネットワークリソースがないか、またはネットワークリソースを使用しないリリソースが含まれる場合があり、これらの関数は `NULL` に設定されている `netresource_list` パラメータを返すことがあります。

システムの任意のリソースグループ名を `scds_get_rg_hostnames()` および `scds_get_rg_hostnames_zone()` に渡すことができます。指定されたリソースグループで実行中のアプリケーションと通信するには、`scds_get_rg_hostnames()` および `scds_get_rg_hostnames_zone()` により返されたホスト名を使用します。

この関数が割り当て、返すメモリーを解放するには、`scds_free_net_list()` を使用します。

次のパラメータがサポートされます。

`resourcegroup_name`	リソースグループ名です。このリソースグループのデータが取得されます。
`netresource_list`	リソースグループによって使用されるネットワーク資源のリスト

`scds_get_rg_hostnames()` および `scds_get_rg_hostnames_zone()` 関数は、次の値を返します:

`0`	関数の実行に成功。

0 以外　　関数の実行に失敗。

SCHA_ERR_NOERR　　関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

/usr/cluster/include/rgm/libdsdev.h
　インクルードファイル

/usr/cluster/lib/libdsdev.so
　ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

949 ページのscds_free_net_list(3HA), 979 ページのscds_get_rs_hostnames(3HA), 1047 ページのscha_calls(3HA), Unresolved link to " attributes5"

名前

scds_get_rs_hostnames — リソースによって使用されるネットワーク資源の取得

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_get_rs_hostnames(scds_handle_t
     handle,scds_net_resource_list_t **netresource_list);
```

`scds_get_rs_hostnames()` 関数は、リソースによって使用されるホスト名のリストを取得します。リソースプロパティー `Network_resources_used` が設定されている場合、ホスト名は、`Network_resources_used` 内のネットワーク資源に対応しています。それ以外の場合、リソースを含むリソースグループ内のすべてのネットワーク資源に対応しています。

この関数は、`netresource_list` 内のリストへのポインタを返します。リソースグループにネットワーク資源がない場合、またはネットワーク資源を使用しないリソースが含まれる場合、この関数は、`netresource_list` の値を `NULL` に設定して終了します。

949 ページの`scds_free_net_list(3HA)` を使用して、この関数によって割り当てられてから返されるメモリーを解放します。

次のパラメータがサポートされます。

`handle`
: 987 ページの`scds_initialize(3HA)` から返されるハンドルです。

`netresource_list`
: リソースグループによって使用されるネットワークリソースのリストです。

`scds_get_rs_hostnames()` 関数の戻り値は次のとおりです。

`0`	関数の実行に成功
`0 以外`	関数の実行に失敗

`SCHA_ERR_NOERR`
: 関数の実行に成功。

その他のエラーコードについては、1047 ページの`scha_calls(3HA)` を参照してください。

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

949 ページの`scds_free_net_list`(3HA), 975 ページの`scds_get_rg_hostnames`(3HA), 987 ページの`scds_initialize`(3HA), 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5", 1287 ページの`r_properties`(5)

名前

scds_get_zone_name — メソッドの実行対象であるゾーンの名前を取得する

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     const char *scds_get_zone_name(scds_handle_t handle);
```

`scds_get_zone_name()` 関数は、文字列に対するポインタを返します。次の条件が成立する場合、この文字列にはリソースグループが動作するゾーンの名前が含まれています。

- `scds_get_zone_name` 関数は、大域ゾーンから呼び出されます。
- `Global_zone` リソースタイププロパティーが `TRUE` に設定されている。

  `Global_zone` リソースタイププロパティーについては、1335 ページの`rt_properties(5)` のマニュアルページを参照してください。
- リソースはゾーンクラスタに構成されます。

次の場合も含め、その他の場合はすべて、該当する文字列は NULL です。

- リソースグループが大域ゾーンに構成されている。
- `Global_zone` リソースタイププロパティーが `FALSE` に設定されるか、または `Global_zone_override` リソースプロパティーが `FALSE` に設定されます。

メソッドが実際に実行されているゾーンの名前を取得するには、`zonename` コマンドを使用します。Unresolved link to " zonename1" のマニュアルページを参照してください。

文字列に対するポインタは、Data Service Development Library (DSDL) に属するメモリーをポイントします。このメモリーは変更できません。このポインタは、`scds_close()` 呼び出しによって無効化されます。

次のパラメータがサポートされます。

`handle` 987 ページの`scds_initialize(3HA)` から返されるハンドルです。

`SCHA_ERR_NOERR` 関数の実行に成功。

その他のエラーコードについては、1047 ページの`scha_calls(3HA)` を参照してください。

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

Unresolved link to " zonename1", 917 ページの`scds_close`(3HA), 987 ページの`scds_initialize`(3HA), 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5", 1335 ページの`rt_properties`(5)

名前

scds_hasp_check — リソースによって使用される SUNW.HAStoragePlus リソースに関するステータス情報の取得

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_hasp_check(scds_handle_t handle,
     scds_hasp_status_t *hasp_status);
```

scds_hasp_check()() 関数は、リソースによって使用される SUNW.HAStoragePlus リソースに関するステータス情報を取得します。この情報は、リソースの依存先であるすべての SUNW.HAStoragePlus リソースの状態(オンラインまたはその他) から取得されます。この状態は、リソースに対して定義されている Resource_dependencies、Resource_dependencies_weak、Resource_dependencies_restart、または Resource_dependencies_offline_restart プロパティーを使用して取得されます。

SUNW.HAStoragePlus リソースの FileSystemMountPoints プロパティーが空でないときは、FileSystemMountPoints プロパティーにリストされているすべてのファイルシステムが特定のノードに現在マウントされている場合に限り、scds_hasp_check()() 関数はそのリソースがそのノード上でオンラインであるとみなします。グローバルにマウントされているファイルシステムは、SUNW.HAStoragePlus リソースがオフラインのノードにマウントされる場合があります。SUNW.HAStoragePlus リソースが特定のノード上でオンラインであるとみなされるためには、FileSystemMountPoints のすべてが、ローカルにマウントされていてもグローバルにマウントされていても、そのノードにマウントされていなければなりません。このリソースの実際の状態は、オンラインまたはオフラインである可能性があります。これらのいずれかのファイルシステムがノードにマウントされていない場合、リソースはオフラインとみなされ、その実際の状態もオフラインになります。

リソースタイプ実装は、VALIDATE および MONITOR_CHECK メソッドコールバック実装の scds_hasp_check()() を使用して、SUNW.HAStoragePlus リソースによって管理されるファイルシステム固有の検査を実行する必要があるかどうかを確認します。

リソースの依存関係は、関数が実行されるのと同じクラスタコンテキスト (グローバルクラスタまたはゾーンクラスタ) 内でのみ確認されます。フォーム *clustername*: *resourcename* の依存関係 (クラスタ間依存関係) は無視されます。たとえば、HAStoragePlus 依存関係のみがクラスタ間依存関係である場合、この関数はステータスコード SCDS_HASP_NO_RESOURCE を返します。

関数が正常に実行されると、hasp_status パラメータにステータスコードが格納されます。格納されるコードは次のいずれかです。

SCDS_HASP_NO_RESOURCE

リソースが SUNW.HAStoragePlus リソースに依存していないことを示しています。

SCDS_HASP_NOT_ONLINE

リソースの依存先である SUNW.HAStoragePlus リソースが、潜在的なプライマリノードでオンラインでないことを示しています。

SCDS_HASP_ONLINE_NOT_LOCAL

リソースの依存先である SUNW.HAStoragePlus リソースの少なくとも 1 つがこの関数の呼び出し元のノード上でオンラインではありませんが、別のノード上ではオンラインであることを示しています。

SCDS_HASP_ONLINE_LOCAL

リソースの依存先である SUNW.HAStoragePlus リソースがすべて、この関数の呼び出し元のノード上でオンラインであることを示しています。

注記 - 前述のステータスコードは、紹介した順序に従った優先度を持っています。たとえば、SUNW.HAStoragePlus リソースがオンラインでなく、SUNW.HAStoragePlus リソースが別のノード上でオンラインになっている場合、ステータスコードは SCDS_HASP_ONLINE_NOT_LOCAL ではなく SCDS_HASP_NOT_ONLINE に設定されます。

---

scds_hasp_check()() 関数は、GlobalDevicePaths プロパティーが空でなくても、FilesystemMountPoints プロパティーと Zpools プロパティーの両方がデフォルトで空のリストに設定されている SUNW.HAStoragePlus リソースを無視します。

次のパラメータがサポートされます。

handle	scds_initialize から返されるハンドルです。
hasp_status	リソースによって使用される SUNW.HAStoragePlus リソースのステータスです。

SCHA_ERR_NOERR	関数の実行に成功。 この値はまた、hasp_status パラメータに格納されているステータスコードが有効であることも示します。
SCHA_ERR_INTERNAL	関数の実行に失敗。

`hasp_status` パラメータに格納されている値が未定義です。この未定義の値は無視してください。

その他のエラーコードについては、1047 ページの`scha_calls`(3HA) のマニュアルページを参照してください。

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

987 ページの`scds_initialize`(3HA), 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5", 1389 ページの`SUNW.HAStoragePlus`(5)

名前

scds_initialize — DSDL 環境の割り当てと初期化

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_initialize(scds_handle_t *handleint argc, char *argv[]);
```

`scds_initialize()` 関数は、DSDL 環境を初期化します。個々のプログラムまたは DSDL 関数を使用する障害モニターの開始時に、必ずこの関数を呼び出します。

`scds_initialize()` 関数は次のとおりです。

- フレームワークが呼び出し元プログラムに渡すコマンド行引数 (`argc` および `argv[]`) で、`scds_initialize()` にも渡す必要があるコマンド行引数のチェックと処理。呼び出し元プログラムは、コマンド行引数の処理以外を行う必要はありません。例 を参照してください。
- DSDL 内のほかの関数に必要な情報を使って、内部データ構造の初期設定を行います。リソース、リソースタイプ、リソースグループのプロパティー値を取得し、これらのデータ構造に格納します。`argv[]` 引数を使用してコマンド行で指定するプロパティーの値は、RGM から取得される値より優先されます。つまり、コマンド行引数 (`argv[]`) で指定したプロパティーの新しい値がデータサービスメソッドに渡されると、この関数が該当するプロパティー値を取得し、この新しい値を返します。それ以外の場合は、RGM から取得した既存の値を返します。
- データサービス障害モニタリング情報を初期化します。
- ロギング環境を初期化します。すべての syslog メッセージには、次の接頭辞がつきます。`SC[`*<resourceTypeName>*`,`*< resourceGroupName*`>,`*<resourceName*`>,`*<methodName*`>`

  syslog にメッセージを送信する関数は、`scha_cluster_getlogfacility()` から返されるデータを使用します。これらのメッセージは、適切なログファイルとユーザーに転送することができます。詳細は、Unresolved link to " syslog.conf4" を参照してください。
- 障害モニターの検証設定の妥当性を検査します。`Retry_interval` が `Thorough_probe_interval * Retry_count` 以上かどうかを確認し、そうでない場合は syslog に適切なメッセージを送信します。Retry_interval が Thorough_probe_interval * Retry_count 以上でない場合は、syslog に適切なメッセージを送信します。この障害モニタープローブ設定の妥当性検査で、`VALIDATE` メソッド内で `scds_initialize()` および `scds_close()` を呼び出すことができます。`VALIDATE` メソッド内でその他の DSDL 関数を呼び出す必要はありません。

scds_initialize() の実行に成功した場合、呼び出し元プログラムを終了する前に scds_close() を呼び出す必要があります。

scds_initialize() の実行に失敗した場合、クリーンアップを行うために scds_close() を呼び出すことはできません。また、scds_initialize() の実行に失敗した場合、その他の DSDL 関数を呼び出してはいけません。それ以外の場合、SCHA_ERR_INVAL または NULL 値を返します。その代わりに、非ゼロの引数を指定して exit() を呼び出してください。

次のパラメータがサポートされます。

handle　　scds_initialize() によって初期化され、その他の DSDL 関数によって使用されるハンドルです。

argc　　呼び出し元プログラムに渡される引数の数です。

argv　　呼び出し元プログラムに渡される引数配列へのポインタです。

SCHA_ERR_NOERR　　関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

例 365　scds_initialize() の使用法

```
int main(int argc, char *argv[]){
 scds_handle_t handle;

 if (scds_initialize(&handle, argc, argv) !=
 SCHA_ERR_NOERR)
 exit(1);
 ...
 /* data service code */
...
 scds_close(&handle);
}
```

/usr/cluster/include/rgm/libdsdev.h
　インクルードファイル

/usr/cluster/lib/libdsdev.so
　ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

917 ページのscds_close(3HA), 1017 ページのscds_property_functions(3HA), 1047 ページのscha_calls(3HA), 1079 ページのscha_cluster_getlogfacility(3HA), Unresolved link to " syslog.conf4", 1287 ページのr_properties(5)

名前

scds_is_zone_cluster — リソースがゾーンクラスタノードに構成されているかどうかを示すブール値を返す。

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev #include <rgm/libdsdev.h>boolean_t
     scds_is_zone_cluster(scds_handle_t handle);
```

`scds_is_zone_cluster()` 関数は、リソースがゾーンクラスタに構成されている場合は `B_TRUE` を返し、そうでない場合は `B_FALSE` を返します。

次のパラメータがサポートされます。

`handle` 987 ページの`scds_initialize`(3HA) から返されるハンドルです。

`SCHA_ERR_NOERR` 関数の実行に成功。

その他のエラーコードについては、1047 ページの`scha_calls`(3HA) を参照してください。

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	SUNWscdev
インタフェースの安定性	発展中

909 ページの`scds_calls`(3HA), 917 ページの`scds_close`(3HA), 987 ページの`scds_initialize`(3HA), 1047 ページの`scha_calls`(3HA), 613 ページの`clzonecluster`(1CL), Unresolved link to " attributes5"

名前

scds_pmf_get_status — PMF によってモニターされるプロセスツリーが存在するかどうかを判別する

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_pmf_get_status(scds_handle_t handle,
     scds_pmf_type_t program_type, int instance, scds_pmf_status_t*pmf_status);
```

`scds_pmf_get_status()` 関数は、指定のインスタンスが PMF 制御下でモニターされているかどうかを確認します。この関数は、`-q` オプションを指定した 735 ページの`pmfadm`(1M) コマンドと同じです。

次のパラメータがサポートされます。

`handle`

`scds_initialize()` から返されるハンドルです。

`program_type`

実行するプログラムの型です。有効な型は次のとおりです。

`SCDS_PMF_TYPE_SVC`	データサービスアプリケーション
`SCDS_PMF_TYPE_MON`	障害モニター
`SCDS_PMF_TYPE_OTHER`	その他

`instance`

複数のインスタンスを持つリソースの場合、この整数 (0 以上) はインスタンスを一意に識別します。単一のインスタンスの場合、0 を使用します。

`pmf_status`

PMF が指定のインスタンスをモニタリングしている場合、`pmf_status` は `SCDS_PMF_MONITORED` に設定されます。それ以外の場合、`SCDS_PMF_NOT_MONITORED` に設定されます。

`scds_pmf_get_status()` 関数の戻り値は次のとおりです。

`0`	関数の実行に成功。
`0` 以外	関数の実行に失敗。

SCHA_ERR_NOERR
: 関数の実行に成功

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

/usr/cluster/include/rgm/libdsdev.h
: インクルードファイル

/usr/cluster/lib/libdsdev.so
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

735 ページのpmfadm(1M), 987 ページのscds_initialize(3HA),
1047 ページのscha_calls(3HA), Unresolved link to " attributes5"

名前

scds_pmf_restart_fm — PMF を使用して障害モニターを再起動する

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_pmf_restart_fm(scds_handle_t handle, int instance);
```

scds_pmf_restart_fm() 関数は、障害モニターのプロセスツリーに SIGKILL シグナルを送って障害モニターを強制終了したあと、PMF を使って再起動します。この関数はタイムアウト値として MONITOR_STOP_TIMEOUT プロパティーを使用します。つまり、scds_pmf_restart_fm() は、MONITOR_STOP_TIMEOUT プロパティーに指定された時間が経過するまで、プロセスツリーの終了を待機します。

RTR ファイル内に MONITOR_STOP_TIMEOUT プロパティーが明示的に設定されていない場合、デフォルトのタイムアウト値が使用されます。

UPDATE メソッド内でこの関数を呼び出し、モニターを再起動します。このとき、新しいパラメータを使用できます。

次のパラメータがサポートされます。

handle　　scds_initialize() から返されるハンドルです。

instance　　障害モニターのインスタンスを複数持っているリソースの場合、この整数 (0 以上) は障害モニターのインスタンスを一意に識別します。障害モニターのインスタンスが 1 個だけの場合は 0 を使用します。

scds_pmf_restart_fm() 関数の戻り値は次のとおりです。

0　　関数の実行に成功。

0 以外　　関数の実行に失敗。

SCHA_ERR_NOERR　　関数の実行に成功

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

/usr/cluster/include/rgm/libdsdev.h
　インクルードファイル

`/usr/cluster/lib/libdsdev.so`

ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

735 ページの`pmfadm`(1M), 1047 ページの`scha_calls`(3HA), Unresolved link to " signal3HEAD", Unresolved link to " attributes5", 1287 ページの`r_properties`(5)

名前

scds_pmf_signal — PMF の制御下にあるプロセスツリーにシグナルを送信する

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_pmf_signal(scds_handle_t handle, scds_pmf_type_t
     program_type, int instance, int signal, time_t timeout);
```

`scds_pmf_signal()` 関数は、PMF 制御下で実行されているプロセスツリーに指定のシグナルを送信します。この関数は、`-k` オプションを指定した 735 ページの`pmfadm`(1M) コマンドと同じです。

`scds_pmf_signal()` 関数は、シグナルの送信後、指定のタイムアウト時間が経過するまでプロセスツリーの終了を待機してから終了します。`timeout` に `0` が指定されている場合は、プロセスの終了を待たずにただちに終了します。`-1` が指定されている場合は、プロセスが終了するまで無期限に待機します。

次のパラメータがサポートされます。

`handle`

`scds_initialize()` から返されるハンドルです。

`program_type`

実行するプログラムの型です。有効な型は次のとおりです。

`SCDS_PMF_TYPE_SVC`	データサービスアプリケーション
`SCDS_PMF_TYPE_MON`	障害モニター
`SCDS_PMF_TYPE_OTHER`	その他

`instance`

複数のインスタンスを持つリソースの場合、この整数 (0 以上) はインスタンスを一意に識別します。単一のインスタンスの場合、0 を使用します。

`signal`

送信される Solaris シグナルです。Unresolved link to " signal3HEAD" を参照してください。

`timeout`

タイムアウト値 (秒) です。

`scds_pmf_signal()` 関数の戻り値は次のとおりです。

`0`	関数の実行に成功。
0 以外	関数の実行に失敗。

`SCHA_ERR_TIMEOUT`
: シグナルの送信後、タイムアウト時間が経過するまでの間にプロセスツリーが終了しない。

`SCHA_ERR_NOERR`
: 関数の実行に成功。

その他の値
: 関数の実行に失敗。障害コードの意味については、1047 ページの`scha_calls`(3HA) を参照してください。

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

735 ページの`pmfadm`(1M), 987 ページの`scds_initialize`(3HA), 1047 ページの`scha_calls`(3HA), Unresolved link to " signal3HEAD", Unresolved link to " attributes5"

名前

scds_pmf_start, scds_pmf_start_env — PMF の制御下でプログラムを実行する

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>scha_err_t scds_pmf_start(
     scds_handle_t handle, scds_pmf_type_t program_type, int instance,
     const char *command, int child_monitor_level)scha_err_t
     scds_pmf_start_env(scds_handle_t handle, scds_pmf_type_t
     program_type, int instance, const char *command, int
     child_monitor_level, char ** env)
```

`scds_pmf_start()` 関数は、PMF 制御下で、`command` によって指定されたプログラムを実行します。この関数は、`-c` オプションを指定した 735 ページの`pmfadm`(1M) コマンドと同じです。

`command` 引数には、関数に渡されるコマンド行とコマンド行引数が含まれます。

`scds_pmf_start ()` を使って、データサービスアプリケーションやその他のプロセス (プログラム型 `SCDS_PMF_TYPE_SVC`、`SCDS_PMF_TYPE_MON` または `SCDS_PMF_TYPE_OTHER`) を PMF 制御下で起動するとき `child_monitor_level` 引数を使ってモニターする子プロセスのレベルを選択します。`child_monitor_level` 引数は、実行中の子プロセスをレベル `child_monitor_level` でモニターします。元のプロセスはレベル 0、子プロセスはレベル 1、さらにその子プロセスはレベル 2 で実行されます。新しいフォーク操作により、新しい子のレベルが生成されます。すべての子をモニターするには、`-1` を指定します。

たとえば、起動するコマンドがデーモンである場合、適切な `child_monitor_level` は `0` です。起動するコマンドがデーモンを起動するスクリプトである場合、適切な `child_monitor_level` の値は`1` です。

配下のアプリケーションプロセスがすでに実行中の場合、`scds_pmf_start()` は `syslog()` エラーを出力し、 `SCHA_ERR_INTERNAL` を返します。これは、単一ノード上で `START` 関数が 2 回 呼び出された場合、RGM が間に `STOP` 関数を呼び出すことを保証するためです。

次のパラメータがサポートされます。

`handle` 987 ページの`scds_initialize`(3HA) から返されるハンドルです。

`program_type` 実行するプログラムの型です。有効な型は次のとおりです。

`SCDS_PMF_TYPE_SVC` データサービスアプリケーション

SCDS_PMF_TYPE_MON	障害モニター
SCDS_PMF_TYPE_OTHER	その他

`instance` 複数のインスタンスを持つリソースの場合、この整数 (0 以上) はインスタンスを一意に識別します。単一のインスタンスの場合、0 を使用します。

`command` PMF 制御下で実行されるコマンド (コマンド行引数を含む) です。

`child_monitor_level` モニターする子プロセスのレベル(`pmfadm` の `-C` オプションと同等)を指定します。`-1` を使用して、すべてのレベルの子プロセスを指定します。

`env` 環境文字列への文字ポインタの配列を指定します。環境文字列については、Unresolved link to " execve2" のマニュアルページを参照してください。`command` パラメータで指定したプログラムが実行されると、この環境がこのプログラムに渡されます。

`scds_pmf_start()` 関数の戻り値は次のとおりです。

`0`	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_INTERNAL	配下のアプリケーションプロセスがすでに実行中。
SCHA_ERR_NOERR	関数の実行に成功。
その他の値	関数の実行に失敗。その他のエラーコードについては、1047 ページの`scha_calls`(3HA) を参照してください。

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api

属性タイプ	属性値
インタフェースの安定性	発展中

735 ページの`pmfadm`(1M), 987 ページの`scds_initialize`(3HA),
1007 ページの`scds_pmf_stop`(3HA), 1035 ページの`scds_svc_wait`(3HA),
1047 ページの`scha_calls`(3HA), Unresolved link to " execve2", Unresolved link to
" attributes5"

名前

scds_pmf_start, scds_pmf_start_env — PMF の制御下でプログラムを実行する

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>scha_err_t scds_pmf_start(
     scds_handle_t handle, scds_pmf_type_t program_type, int instance,
     const char *command, int child_monitor_level)scha_err_t
     scds_pmf_start_env(scds_handle_t handle, scds_pmf_type_t
     program_type, int instance, const char *command, int
     child_monitor_level, char ** env)
```

scds_pmf_start() 関数は、PMF 制御下で、command によって指定されたプログラムを実行します。この関数は、-c オプションを指定した 735 ページのpmfadm(1M) コマンドと同じです。

command 引数には、関数に渡されるコマンド行とコマンド行引数が含まれます。

scds_pmf_start () を使って、データサービスアプリケーションやその他のプロセス (プログラム型 SCDS_PMF_TYPE_SVC、SCDS_PMF_TYPE_MON または SCDS_PMF_TYPE_OTHER) を PMF 制御下で起動するとき child_monitor_level 引数を使ってモニターする子プロセスのレベルを選択します。child_monitor_level 引数は、実行中の子プロセスをレベル child_monitor_level でモニターします。元のプロセスはレベル 0、子プロセスはレベル 1、さらにその子プロセスはレベル 2 で実行されます。新しいフォーク操作により、新しい子のレベルが生成されます。すべての子をモニターするには、-1 を指定します。

たとえば、起動するコマンドがデーモンである場合、適切な child_monitor_level は 0 です。起動するコマンドがデーモンを起動するスクリプトである場合、適切な child_monitor_level の値は1 です。

配下のアプリケーションプロセスがすでに実行中の場合、scds_pmf_start() は syslog() エラーを出力し、 SCHA_ERR_INTERNAL を返します。これは、単一ノード上で START 関数が 2 回 呼び出された場合、RGM が間に STOP 関数を呼び出すことを保証するためです。

次のパラメータがサポートされます。

handle　　987 ページのscds_initialize(3HA) から返されるハンドルです。

program_type　　実行するプログラムの型です。有効な型は次のとおりです。

SCDS_PMF_TYPE_SVC　　データサービスアプリケーション

	SCDS_PMF_TYPE_MON	障害モニター
	SCDS_PMF_TYPE_OTHER	その他
instance	複数のインスタンスを持つリソースの場合、この整数 (0 以上) はインスタンスを一意に識別します。単一のインスタンスの場合、0 を使用します。	
command	PMF 制御下で実行されるコマンド (コマンド行引数を含む) です。	
child_monitor_level	モニターする子プロセスのレベル(pmfadm の -C オプションと同等)を指定します。-1 を使用して、すべてのレベルの子プロセスを指定します。	
env	環境文字列への文字ポインタの配列を指定します。環境文字列については、Unresolved link to " execve2" のマニュアルページを参照してください。command パラメータで指定したプログラムが実行されると、この環境がこのプログラムに渡されます。	

scds_pmf_start() 関数の戻り値は次のとおりです。

0	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_INTERNAL	配下のアプリケーションプロセスがすでに実行中。
SCHA_ERR_NOERR	関数の実行に成功。
その他の値	関数の実行に失敗。その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

/usr/cluster/include/rgm/libdsdev.h
: インクルードファイル

/usr/cluster/lib/libdsdev.so
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api

属性タイプ	属性値
インタフェースの安定性	発展中

735 ページの`pmfadm`(1M), 987 ページの`scds_initialize`(3HA), 1007 ページの`scds_pmf_stop`(3HA), 1035 ページの`scds_svc_wait`(3HA), 1047 ページの`scha_calls`(3HA), Unresolved link to " execve2", Unresolved link to " attributes5"

名前

scds_pmf_stop — PMF の制御下で実行中のプロセスを終了する

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_pmf_stop(scds_handle_t handle, scds_pmf_type_t
     program_type, int instance, int signal, time_t timeout);
```

scds_pmf_stop() 関数は、PMF の制御下で実行中のプログラムを停止します。この関数は、-s オプションを指定した 735 ページのpmfadm(1M) コマンドと同じです。

要求されたインスタンスが実行中でない場合、scds_pmf_stop() は SCHA_ERR_NOERR を返します。

要求されたインスタンスが実行中の場合、このインスタンスに指定のシグナルが送信されます。タイムアウト値の 80% が経過する間に終了しなかったインスタンスには、SIGKILL が送信されます。このインスタンスが、さらにタイムアウト値の 15% が経過する間に終了しなかった場合、この関数は失敗したとみなされ、SCHA_ERR_TIMEOUT が返されます。タイムアウト値の残りの 5% は、この関数のオーバーヘッドによって消費されたものと見なされます。

次のパラメータがサポートされます。

handle

987 ページのscds_initialize(3HA) から返されるハンドルです。

program_type

実行するプログラムの型です。有効な型は次のとおりです。

SCDS_PMF_TYPE_SVC	データサービスアプリケーション
SCDS_PMF_TYPE_MON	障害モニター
SCDS_PMF_TYPE_OTHER	その他

instance

複数のインスタンスを持つリソースの場合、この整数 (0 以上) はインスタンスを一意に識別します。単一のインスタンスの場合、0 を使用します。

`signal`

インスタンスを強制終了するために送信される Solaris シグナルです。Unresolved link to " signal3HEAD" を参照してください。指定のシグナルでインスタンスを強制終了できない場合は、`SIGKILL` を使用します。

`timeout`

タイムアウト値 (秒) です。

`scds_pmf_stop()` 関数の戻り値は次のとおりです。

`0` 関数の実行に成功。

`0 以外` 関数の実行に失敗。

`SCHA_ERR_TIMEOUT`

関数はタイムアウト。

`SCHA_ERR_NOERR`

関数の実行に成功。

その他の値

関数の実行に失敗。その他のエラーコードについては、1047 ページの`scha_calls`(3HA) を参照してください。

`/usr/cluster/include/rgm/libdsdev.h`

インクルードファイル

`/usr/cluster/lib/libdsdev.so`

ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

735 ページの`pmfadm`(1M), 987 ページの`scds_initialize`(3HA), 999 ページの`scds_pmf_start`(3HA), 1047 ページの`scha_calls`(3HA), Unresolved link to " signal3HEAD", Unresolved link to " attributes5"

名前

scds_pmf_stop_monitoring — PMF の制御下で実行中のプロセスのモニタリングを停止する

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_pmf_stop_monitoring(scds_handle_t handle,
     scds_pmf_type_t program_type, int instance);
```

`scds_pmf_stop_monitoring()` 関数は、PMF 制御下で実行されているプロセスツリーのモニタリングを停止します。PMF は、プロセスを停止するシグナルを送信することはありません。プロセスの再起動も行いません。

要求されたプロセスが PMF 制御下にない場合、`scds_pmf_stop_monitoring ()` は `SCHA_ERR_NOERR` を返して終了します。

次のパラメータがサポートされます。

`handle`

987 ページの`scds_initialize`(3HA) から返されるハンドルです。

`program_type`

実行するプログラムの型です。有効な型は次のとおりです。

`SCDS_PMF_TYPE_SVC`	データサービスアプリケーション
`SCDS_PMF_TYPE_MON`	障害モニター
`SCDS_PMF_TYPE_OTHER`	その他

`instance`

複数のインスタンスを持つリソースの場合、この整数 (0 以上) はインスタンスを一意に識別します。単一のインスタンスの場合、0 を使用します。

`scds_pmf_stop_monitoring()` 関数の戻り値は次のとおりです。

`0`	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_NOERR
: 関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

/usr/cluster/include/rgm/libdsdev.h
: インクルードファイル

/usr/cluster/lib/libdsdev.so
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

735 ページのpmfadm(1M), 987 ページのscds_initialize(3HA), 999 ページのscds_pmf_start(3HA), 1007 ページのscds_pmf_stop(3HA), 1047 ページのscha_calls(3HA), Unresolved link to " attributes5"

名前

scds_print_net_list — ネットワーク資源リストの内容を印刷する

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     void scds_print_net_list(scds_handle_t handle,int
     debug_level,constscds_net_resource_list_t *netresource_list);
```

`scds_print_net_list()` 関数は、`netresource_list` で指定されたネットワーク資源リストの内容を、`debug_level` で指定されたデバッグレベルでシステムログに書き込みます。指定されたデバッグレベルが現在使用されているデバッグレベルより大きい場合、情報は書き込まれません。

次のパラメータがサポートされます。

`handle`

987 ページの`scds_initialize`(3HA) から返されるハンドルです。

`debug_level`

データが書き込まれるデバッグレベルです。

`netresource_list`

初期化されたネットワークリソースリストへのポインタです。975 ページの`scds_get_rg_hostnames`(3HA) と 979 ページの`scds_get_rs_hostnames`(3HA) のどちらかによって取得されます。

`/usr/cluster/include/rgm/libdsdev.h`

インクルードファイル

`/usr/cluster/lib/libdsdev.so`

ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

975 ページの`scds_get_rg_hostnames`(3HA), 979 ページの`scds_get_rs_hostnames`(3HA), 987 ページの`scds_initialize`(3HA), 1041 ページの`scds_syslog_debug`(3HA), Unresolved link to " attributes5"

名前

scds_print_netaddr_list — リソースグループによって使用される、ホスト名、ポート、プロトコルから成るリストの内容を印刷する

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     void scds_print_netaddr_list(scds_handle_t handle, int
     debug_level, constscds_netaddr_list_t *netaddr_list);
```

`scds_print_netaddr_list()` 関数は、`netaddr_list` で指定されたホスト名、ポート、プロトコルの組み合わせのリストの内容を、`debug_level` で指定されたデバッグレベルでシステムログに書き込みます。指定されたデバッグレベルが現在使用されているデバッグレベルより大きい場合、情報は書き込まれません。

次のパラメータがサポートされます。

`handle`

987 ページの`scds_initialize`(3HA) から返されるハンドルです。

`debug_level`

データが書き込まれるデバッグレベルです。

`netaddr_list`

リソースグループによって使用されるホスト名、ポート、プロトコルの組み合わせのリストへのポインタです。965 ページの`scds_get_netaddr_list`(3HA) によって取得されます。

`/usr/cluster/include/rgm/libdsdev.h`

インクルードファイル

`/usr/cluster/lib/libdsdev.so`

ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

965 ページの scds_get_netaddr_list(3HA), 987 ページの scds_initialize(3HA), 1041 ページの scds_syslog_debug(3HA), Unresolved link to " attributes5"

名前

scds_print_port_list — ポートリストの内容を印刷する

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     void scds_print_port_list(scds_handle_t handle,int debug_level,
     constscds_port_list_t *port_list);
```

`scds_print_port_list()` 関数は、`port_list` で指定されたポートリストの内容を、`debug_level` で指定されたデバッグレベルでシステムログに書き込みます。指定されたデバッグレベルが現在使用されているデバッグレベルより大きい場合、情報は書き込まれません。

次のパラメータがサポートされます。

`handle`

987 ページの`scds_initialize`(3HA) から返されるハンドルです。

`debug_level`

データが書き込まれるデバッグレベルです。

`port_list`

リソースグループによって使用されるポートとプロトコルのペアのリストへのポインタです。`scds_get_port_list()` によって取得されます。

`/usr/cluster/include/rgm/libdsdev.h`

インクルードファイル

`/usr/cluster/lib/libdsdev.so`

ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

967 ページの`scds_get_port_list`(3HA), 987 ページの`scds_initialize`(3HA), 1041 ページの`scds_syslog_debug`(3HA), Unresolved link to " attributes5"

名前

scds_property_functions — 一般に使用されるリソースプロパティー、リソースグループプロパティー、リソースタイププロパティー、および拡張プロパティーの値を取得するための関数群

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     return-value-type scds-get-property-name(scds_handle_t handle);
```

Data Service Development Library (DSDL) は、一般に使用されるリソースプロパティー、リソースグループプロパティー、リソースタイププロパティー、および拡張プロパティーの値を取得するために使用できる関数群を提供します。957 ページのscds_get_ext_property(3HA) を使用して、ユーザー定義の拡張プロパティーを取得します。

すべての関数は、次の規則に従うものとします。

- handle 引数だけを取ります。プロパティー取得関数に渡される handle 引数は、以前の 987 ページのscds_initialize(3HA) 呼び出しによって返されます。
- 各関数が特定のプロパティーに対応します。
- 関数の戻り値のタイプは取得するプロパティー値のタイプに一致します。
- これらの関数は、戻り値が 987 ページのscds_initialize(3HA) で事前に計算されているため、エラーを返しません。ポインタを返す関数の場合、エラー条件が発生すると NULL 値が返されます。たとえば、以前に scds_initialize() が呼び出されていない場合などが該当します。
- プロパティーの新しい値が、呼び出し元のプログラム(argv[]) に渡されるコマンド行引数で指定されていると、この新しい値が返されます(Validate メソッドの実装の場合)。この方法で、新しいプロパティーの予想値を実際に設定する前に検証できます。それ以外の場合は、RGM から取得した値が返されます。
- これらの関数の中には、DSDL に属するメモリーへのポインタを返すものもあります。このメモリーは変更できません。このポインタは、917 ページのscds_close(3HA) 呼び出しによって無効化されます。

標準プロパティーについては、1287 ページのr_properties(5)、1319 ページのrg_properties(5)、および 1335 ページのrt_properties(5) のマニュアルページを参照してください。拡張プロパティーについては、個々のデータサービスのマニュアルページを参照してください。

これらの関数によって使用されるデータ型については、1047 ページのscha_calls(3HA) のマニュアルページおよび scha_types.h ヘッダーファイル (scha_prop_type_t、scha_extprop_value_t、scha_initnodes_flag_t、scha_str_array_t、scha_failover_mod など) を参照してください。

これらの関数は、次の命名規則に従うものとします。

リソースプロパティー

scds_get_rs_*property-name*

リソースグループプロパティー

scds_get_rg_*property-name*

リソースタイププロパティー

scds_get_rt_*property-name*

一般に使用される拡張プロパティー

scds_get_ext_*property-name*

注記 - プロパティー名には、大文字と小文字の区別はありません。プロパティー名を指定する際には、大文字と小文字を任意に組み合わせることができます。

リソース固有の関数

この関数は、特定のリソースプロパティーの値を返します。一部のプロパティー値は、RTR ファイル内で、または 273 ページのclresource(1CL) コマンドによって明示的に設定されます。その他のプロパティー値は、RGM によって動的に決定されます。関数は、要求されたプロパティーにふさわしいデータ型を返します。

次のリソースの依存関係のクエリー関数には、それぞれ対応する "Q" または "qualified" バージョンがあります。

scds_get_rs_resource_dependencies

scds_get_rs_resource_dependencies_Q

scds_get_rs_resource_dependencies_offline_restart

scds_get_rs_resource_dependencies_Q_offline_restart

scds_get_rs_resource_dependencies_restart

scds_get_rs_resource_dependencies_Q_restart

scds_get_rs_resource_dependencies_weak

scds_get_rs_resource_dependencies_Q_weak

資格を備えたバージョンは、各リソースの依存関係に宣言された有効範囲または修飾子 (ある場合) を返します。{LOCAL_NODE}、{ANY_NODE}、および {FROM_RG_AFFINITIES} 修飾子は、1287 ページのr_properties(5) のマニュアルページで説明されています。

Cheap_probe_interval

```
int scds_get_rs_cheap_probe_interval(scds_handle_t handle)
```

Failover_mode

```
scha_failover_mode_t scds_get_rs_failover_mode(scds_handle_t handle)
```

Monitor_stop_timeout

```
int scds_get_rs_monitor_stop_timeout(scds_handle_t handle)
```

Monitored_switch

```
scha_switch_t scds_get_rs_monitored_switch(scds_handle_t handle)
```

Network_resources_used

```
scha_str_array_t * scds_get_rs_network_resources_used(scds_handle_t handle)
```

On_off_switch

```
scha_switch_t scds_get_rs_on_off_switch(scds_handle_t handle)
```

Resource_dependencies

```
const scha_str_array_t * scds_get_rs_resource_dependencies(scds_handle_t
handle)
```

Resource_dependencies_Q(qualified)

```
const scha_str_array_t * scds_get_rs_resource_dependencies_Q(scds_handle_t
handle)
```

Resource_dependencies_offline_restart

```
const scha_str_array_t *
scds_get_rs_resource_dependencies_offline_restart(scds_handle_t handle)
```

Resource_dependencies_Q_offline_restart (qualified)

```
const scha_str_array_t *
scds_get_rs_resource_dependencies_Q_offline_restart(scds_handle_t handle)
```

Resource_dependencies_restart

```
const scha_str_array_t *
scds_get_rs_resource_dependencies_restart(scds_handle_t handle)
```

`Resource_dependencies_Q_restart` (qualified)

```
const scha_str_array_t *
scds_get_rs_resource_dependencies_Q_restart(scds_handle_t handle)
```

`Resource_dependencies_weak`

```
const scha_str_array_t *
scds_get_rs_resource_dependencies_weak(scds_handle_t handle)
```

`Resource_dependencies_Q_weak` (qualified)

```
const scha_str_array_t *
scds_get_rs_resource_dependencies_Q_weak(scds_handle_t handle)
```

`Resource_project_name`

```
const char * scds_get_rs_resource_project_name(scds_handle_t handle)
```

`Retry_count`

```
int scds_get_rs_retry_count(scds_handle_t handle)
```

`Retry_interval`

```
int scds_get_rs_retry_interval(scds_handle_t handle)
```

`Scalable`

```
boolean scds_get_rs_scalable(scds_handle_t handle)
```

`Start_timeout`

```
int scds_get_rs_start_timeout(scds_handle_t handle)
```

`Stop_timeout`

```
int scds_get_rs_stop_timeout(scds_handle_t handle)
```

`Thorough_probe_interval`

```
int scds_get_rs_thorough_probe_interval(scds_handle_t handle)
```

リソースグループ固有の関数

この関数は、特定のリソースグループプロパティーの値を返します。一部のプロパティー値は、305 ページの`clresourcegroup`(1CL) コマンドによって明示的に設定されます。その他のプロパティー値は、RGM によって動的に決定されます。関数は、要求されたプロパティーにふさわしいデータ型を返します。

`Desired_primaries`

```
int scds_get_rg_desired_primaries(scds_handle_t handle)
```

Global_resources_used

```
const scha_str_array_t * scds_get_rg_global_resources_used(scds_handle_t
handle)
```

Implicit_network_dependencies

```
boolean_t scds_get_rg_implicit_network_dependencies(scds_handle_t handle)
```

Maximum_primaries

```
int scds_get_rg_maximum_primaries(scds_handle_t handle)
```

Nodelist

```
const scha_str_array_t * scds_get_rg_nodelist (scds_handle_t handle)
```

Pathprefix

```
const char * scds_get_rg_pathprefix(scds_handle_t handle)
```

Pingpong_interval

```
int scds_get_rg_pingpong_interval(scds_handle_t handle)
```

Resource_list

```
const scha_str_array_t * scds_get_rg_resource_list(scds_handle_t handle)
```

RG_affinities

```
const scha_str_array_t * scds_get_rg_rg_affinities(scds_handle_t handle)
```

RG_mode

```
scha_rgmode_t scds_get_rg_rg_mode(scds_handle_t handle)
```

RG_project_name

```
const char * scds_get_rg_rg_project_name(scds_handle_t handle)
```

RG_slm_cpu_shares

```
int scds_get_rg_rg_slm_cpu_shares(scds_handle_t handle)
```

RG_slm_pset_min

```
int scds_get_rg_rg_slm_pset_min(scds_handle_t handle)
```

RG_slm_pset_type

```
const char * scds_get_rg_rg_slm_pset_type(scds_handle_t handle)
```

RG_slm_type

```
const char * scds_get_rg_rg_slm_type(scds_handle_t handle)
```

リソースタイプ固有の関数

この関数は、特定のリソースタイププロパティーの値を返します。一部のプロパティー値は、RTR ファイル内で、または 333 ページの`clresourcetype`(1CL) コマンドによって明示的に設定されます。その他のプロパティー値は、RGM によって動的に決定されます。関数は、要求されたプロパティーにふさわしいデータ型を返します。

```
API_version
    int scds_get_rt_api_version(scds_handle_t handle)

Failover
    boolean_t scds_get_rt_failover(scds_handle_t handle)

Init_nodes
    scha_initnodes_flag_t scds_get_rt_init_nodes(scds_handle_t handle)

Installed_nodes
    const scha_str_array_t * scds_get_rt_installed_nodes(scds_handle_t handle)

RT_basedir
    const char * scds_get_rt_rt_basedir(scds_handle_t handle)

RT_version
    const char * scds_get_rt_rt_version(scds_handle_t handle)

Single_instance
    boolean_t scds_get_rt_single_instance(scds_handle_t handle)

Start_method
    const char * scds_get_rt_start_method(scds_handle_t handle)

Stop_method
    const char * scds_get_rt_stop_method(scds_handle_t handle)
```

拡張プロパティー固有の関数

この関数は、特定のリソース拡張プロパティーの値を返します。これらのプロパティー値は、RTR ファイル内で、または 273 ページの`clresource`(1CL) コマンドによって明示的に設定されます。関数は、要求されたプロパティーにふさわしいデータ型を返します。

リソースタイプでは、ここに記載されている 4 つの拡張プロパティー以外の拡張プロパティーも定義できますが、この 4 つのプロパティーには有用な関数が定義されています。これらの

プロパティーは、これらの有用な関数または 957 ページのscds_get_ext_property(3HA) 関数を使用して取得します。その他の拡張プロパティーを取得する場合は、scds_get_ext_property() を使用する必要があります。

Confdir_list

```
scha_str_array_t * scds_get_ext_confdir_list(scds_handle_t handle)
```

Monitor_retry_count

```
int scds_get_ext_monitor_retry_count(scds_handle_t handle)
```

Monitor_retry_interval

```
int scds_get_ext_monitor_retry_interval(scds_handle_t handle)
```

Probe_timeout

```
int scds_get_ext_probe_timeout(scds_handle_t handle)
```

すべての有用な関数について、次のパラメータがサポートされています。

handle　　987 ページのscds_initialize(3HA) から返されるハンドルです。

関数の戻り値のタイプは取得するプロパティー値のタイプに一致します。

これらの関数は、戻り値が 987 ページのscds_initialize(3HA) で事前に計算されているため、エラーを返しません。ポインタを返す関数の場合、エラー条件が発生すると NULL 値が返されます。たとえば、以前に scds_initialize() が呼び出されていない場合などが該当します。

/usr/cluster/include/rgm/libdsdev.h
: インクルードファイル

/usr/cluster/lib/libdsdev.so
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

273 ページの`clresource`(1CL), 305 ページの`clresourcegroup`(1CL),
333 ページの`clresourcetype`(1CL), 917 ページの`scds_close`(3HA),
957 ページの`scds_get_ext_property`(3HA), 967 ページの`scds_get_port_list`(3HA),
969 ページの`scds_get_resource_group_name`(3HA),
971 ページの`scds_get_resource_name`(3HA),
973 ページの`scds_get_resource_type_name`(3HA), 987 ページの`scds_initialize`(3HA),
1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5",
1287 ページの`r_properties`(5), 1319 ページの`rg_properties`(5), および
1335 ページの`rt_properties`(5)

名前

scds_restart_resource — リソースの再起動

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_restart_resource(scds_handle_t handle);
```

`scds_restart_resource()` 関数は、`SCHA_RESOURCE_RESTART` *tag* 引数を指定して 1101 ページの`scha_control`(3HA) 関数を呼び出し、リソースの再起動をリクエストします。この関数は、障害モニターから呼び出します。

次のパラメータがサポートされます。

`handle` 987 ページの`scds_initialize`(3HA) から返されるハンドルです。

`scha_restart_resource()` 関数の戻り値は次のとおりです。

`0` 関数の実行に成功。

0 以外 関数の実行に失敗。

`SCHA_ERR_NOERR` 関数の実行に成功。

その他のエラーコードについては、1047 ページの`scha_calls`(3HA) を参照してください。

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

645 ページの`rt_callbacks`(1HA), 1027 ページの`scds_restart_rg`(3HA), 1047 ページの`scha_calls`(3HA), 1101 ページの`scha_control`(3HA), Unresolved link to " attributes5"

名前

scds_restart_rg — リソースグループの再起動

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_restart_rg(scds_handle_t handle);
```

`scds_restart_rg()` 関数は、呼び出し元プログラムに渡されるリソースを含むリソースグループ上で 1101 ページの`scha_control`(3HA) `SCHA_RESTART` 操作を実行します。この関数は、障害モニターから呼び出します。

この関数は、正常に実行されると戻りません。したがって、この関数を、呼び出し元プログラムで最後に実行されるコードにしてください。

次のパラメータがサポートされます。

`handle` 987 ページの`scds_initialize`(3HA) から返されるハンドルです。

`scds_restart_rg()` 関数の戻り値は次のとおりです。

`0` 関数の実行に成功。

0 以外 関数の実行に失敗。

`SCHA_ERR_NOERR` 関数の実行に成功。

その他のエラーコードについては、1047 ページの`scha_calls`(3HA) を参照してください。

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api

属性タイプ	属性値
インタフェースの安定性	発展中

1047 ページのscha_calls(3HA), 1101 ページのscha_control(3HA), 987 ページのscds_initialize(3HA), 1025 ページのscds_restart_resource(3HA), Unresolved link to " attributes5"

名前

scds_simple_net_probe — アプリケーションとの TCP 接続を確立し、終了することによる検証

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/
libdsdev.h> scha_err_t scds_simple_net_probe(scds_handle_t handle, scds_netaddr_t addr, time_t timeout, scds_fmsock
 *status, int count);
```

`scds_simple_net_probe()` 関数は、929 ページの`scds_fm_net_connect`(3HA) および 933 ページの`scds_fm_net_disconnect`(3HA) のラッパー関数です。複数のマッピングを持つホストの場合、`scds_simple_net_probe()` は、指定された `hostname` の IPv4 と IPv6 の両方のアドレスを処理します。

965 ページの`scds_get_netaddr_list`(3HA) を使用して、リソースのネットワークアドレスのリストを取得できます。

IPv4 の場合、接続または切断のステータスは `scds_fmsock_status_t` 配列の 1 番目のメンバーに格納されます。IPv6 ターゲットとの接続または切断のステータスは 2 番目のメンバーに格納されます。この関数に指定された `hostname` に IPv4 または IPv6 のマッピングが含まれない場合、対応するステータスは `SCDS_FMSOCK_NA` に設定されます。

次のパラメータがサポートされます。

`handle`

987 ページの`scds_initialize`(3HA) から返されるハンドルです。

`addr`

プロセスが待機する場所を指定する `hostname`、TCP ポート番号、およびプロトコル識別子です。

`timeout`

接続が成功するまで待機するタイムアウト値 (秒)。各ソケット (IPv4 または IPv6) は同じ時間をタイムアウトとして使用します。また、これらのタイムアウトは並行に進行します。

`status`

メンバー `SCDS_MAX_IPADDR_TYPES` から構成される型 `scds_fmsock_status_t` の配列です。配列の各メンバーはステータスを保持します。このパラメータは、この関数が設定する出力引数です。

`count`

`socklist` 配列のメンバー数です。このパラメータは `SCDS_MAX_IPADDR_TYPES` に設定します。

`scds_simple_net_probe()` 関数の戻り値は次のとおりです。

`0`	関数の実行に成功。
`SCHA_ERR_INVAL`	この関数は無効なパラメータを使用して呼び出されました。
その他の 0 以外の値	タイムアウトや接続の拒否などのため、少なくとも 1 つの接続操作が失敗しました。エラーを正確に判断するには、`socklist` 配列のすべてのメンバーについて、`err` フィールドが `SCDS_FMSOCK_ERR` に設定されているかどうかを調査します。
0 以外	少なくとも 1 つの接続操作または切断操作が失敗しました。障害が発生したのが IPv4 ターゲット、IPv6 ターゲット、あるいは両方であるのかを判断するには、`scds_fmsock_status_t` 配列を調査します。

`SCHA_ERR_NOERR`

関数が正常に終了。

`SCHA_ERR_INTERNAL`

関数の実行中に内部エラーが発生したことを示します。

`SCHA_ERR_STATE`

サーバーが接続要求を拒否したことを示します。

`SCHA_ERR_TIMEOUT`

接続要求がタイムアウトしたことを示します。

`/usr/cluster/include/rgm/libdsdev.h`

インクルードファイル

`/usr/cluster/lib/libdsdev.so`

ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`

属性タイプ	属性値
インタフェースの安定性	発展中

929 ページのscds_fm_net_connect(3HA), 933 ページのscds_fm_net_disconnect(3HA), 965 ページのscds_get_netaddr_list(3HA), 987 ページのscds_initialize(3HA), 1033 ページのscds_simple_probe(3HA), 1047 ページのscha_calls(3HA), Unresolved link to " attributes5"

名前

scds_simple_probe — アプリケーションとの TCP 接続を確立し、終了することによる検証

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_simple_probe(scds_handle_t handle, const char *hostname,
     int port, time_t timeout);
```

`scds_simple_probe()` 関数は、Unresolved link to " connect3SOCKET" および Unresolved link to " close2" のラッパー関数で、タイムアウト時間内に実行されます。

975 ページの`scds_get_rg_hostnames`(3HA) と
979 ページの`scds_get_rs_hostnames`(3HA) のどちらかを使用して `hostname` を取得します。

この関数の代わりに 1029 ページの`scds_simple_net_probe`(3HA) の使用を検討します。

次のパラメータがサポートされます。

`handle`
: 987 ページの`scds_initialize`(3HA) から返されるハンドルです。

`hostname`
: 接続先のマシンのインターネットホスト名です。

`port`
: 接続を行うポート番号です。

`timeout`
: 正常に接続が完了するまで待機するタイムアウト値 (秒) です。

`scds_simple_probe()` 関数の戻り値は次のとおりです。

`0`	関数の実行に成功。
0 以外	関数の実行に失敗。

`SCHA_ERR_NOERR`
: 関数が正常に終了。

SCHA_ERR_TIMEOUT
: 関数がタイムアウト。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

/usr/cluster/include/rgm/libdsdev.h
: インクルードファイル

/usr/cluster/lib/libdsdev.so
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	非推奨

Unresolved link to " close2", Unresolved link to " connect3SOCKET", 929 ページのscds_fm_net_connect(3HA), 933 ページのscds_fm_net_disconnect(3HA), 975 ページのscds_get_rg_hostnames(3HA), 979 ページのscds_get_rs_hostnames(3HA), 987 ページのscds_initialize(3HA), 1029 ページのscds_simple_net_probe(3HA), 1047 ページのscha_calls(3HA), Unresolved link to " attributes5"

名前

scds_svc_wait — 指定のタイムアウト時間が経過するまでモニター対象のプロセスの終了を待機する

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_svc_wait(scds_handle_t handle, time_t timeout);
```

`scds_svc_wait()` 関数は、指定のタイムアウト時間が経過するまでモニター対象のプロセスグループの終了を待機します。この関数は、呼び出し元の `START` メソッドに渡されるリソースの 999 ページの`scds_pmf_start`(3HA) によって起動されたすべてのプロセスグループを待機します。`scds_svc_wait()` 関数は、リソースの `Retry_interval` プロパティーと `Retry_count` プロパティーを使用して、待機する終了プロセス数を制限します。`Retry_interval` 内の終了プロセス数が `Retry_count` 個に達した場合、`scds_svc_wait()` は `SCHA_ERR_FAIL` とともに終了します。

失敗プロセス数が `Retry_count` を下回った場合、プロセスは再起動されます。`scds_svc_wait()` は、タイムアウト時間がすべて経過するまで、さらにプロセスの終了を待機します。失敗プロセス数は、そのあとの `scds_svc_wait()` 呼び出しでも計測されます。

次のパラメータがサポートされます。

`handle` 987 ページの`scds_initialize`(3HA) から返されるハンドルです。

`timeout` タイムアウト時間 (秒) です。

`scds_svc_wait()` 関数の戻り値は次のとおりです。

`0` 関数の実行に成功。

0 以外 関数の実行に失敗。

`SCHA_ERR_TIMEOUT` 関数はタイムアウト。

`SCHA_ERR_NOERR` プロセスの終了が行われなかったが、プロセスが正常に再起動された。

`SCHA_ERR_FAIL` 失敗回数が `Retry_count` プロパティーの値に到達。

SCHA_ERR_STATE　　システムエラー、または予期せぬエラーが発生。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

例 366　　START メソッドの scds_svc_wait() の使用法

次の例では、START メソッド内で scds_svc_wait を使用して、サービスの起動に失敗した場合ただちに終了する方法を示します。scds_pmf_start() を使ってアプリケーションプロセスを起動したあと、START メソッドは、処理が正常終了して戻る前にアプリケーションが完全に初期化され、使用可能な状態になるまで待機します。アプリケーションの起動に失敗した場合、START メソッドは、エラーを戻して終了する前に、Start_timeout 時間が経過するまで待機する必要があります。次の例のように、scds_svc_wait() を使用することで、START メソッドによるアプリケーションの再起動回数(Retry_count) を指定できます。サービスを起動できない場合、START メソッドはエラーを出してただちに終了します。

```
/*
 * scds_svc_wait is a subroutine in a START method to
 * check that the service is fully available before returning.
 * Calls svc_probe() to check service availability.
 */
int
svc_wait(scds_handle_t handle)
{
 while (1) {
  /* Wait for 5 seconds */
  if (scds_svc_wait(handle, 5) != SCHA_ERR_NOERR) {
   scds_syslog(LOG_ERR, "Service failed to start.");
   return (1);  /* Start Failure */
  }
  /* Check if service is fully up every 5 seconds */
  if (svc_probe(handle) == 0) {
   scds_syslog(LOG_INFO, "Service started successfully.");
   return (0);
  }
 }
 return (0);
}
```

/usr/cluster/include/rgm/libdsdev.h

　　インクルードファイル

/usr/cluster/lib/libdsdev.so

　　ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

987 ページのscds_initialize(3HA), 999 ページのscds_pmf_start(3HA), 1047 ページのscha_calls(3HA), Unresolved link to " attributes5", 1287 ページのr_properties(5)

- START メソッドがリソース上の Start_timeout 設定を超過した場合、Resource Group Manager (RGM) は START メソッドを強制終了します。この動作は、START メソッドが scds_svc_wait() の終了を待機しているときも変わりません。
- リソース上の Retry_interval の値が Start_timeout の値より大きい場合、START メソッドは RGM によってタイムアウトになります。この動作は、失敗回数が Retry_count に達していない場合も変わりません。
- START メソッドの複数の scds_pmf_start() を呼び出して複数のプロセスを起動した場合、scds_svc_wait() は終了したプロセスグループを起動します。プロセスグループ間の依存関係は強制されません。プロセスグループ間に依存関係があり、失敗時にその他のプロセスグループの再起動を要求するようなプロセスグループが存在する場合は、scds_svc_wait() を使用してはいけません。プロセスグループの健全性検査の間待機する場合は、sleep() を使用してください。

名前

scds_syslog — システムログにメッセージを書き込む

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>void scds_syslog(int priority,
     const char*format...
```

`scds_syslog()` 関数は、システムログにメッセージを書き込みます。この関数は、1079 ページの`scha_cluster_getlogfacility`(3HA) 関数から返される機能を使用します。これらのメッセージは、適切なログファイルとユーザーに転送できます。詳細は、Unresolved link to " syslog.conf4" を参照してください。

すべての syslog メッセージには、次の接頭辞がつきます。

`SC[`<*resourceTypeName* >,<*resourceGroupName*>,< *resourceName*>,<*methodName*>

注意 - システムログに書き込まれるメッセージは多言語化されません。この関数と `gettext()` などのメッセージ変換関数を併用しないでください。

次のパラメータがサポートされます。

`priority`	Unresolved link to " syslog3C" によって指定されるメッセージ優先順位です
`format`	Unresolved link to " printf3C" によって指定されるメッセージ形式の文字列です
`...`	`printf()` によって指定される変数です。`format` パラメータで表されます。

`/usr/cluster/include/rgm/libdsdev.h`

インクルードファイル

`/usr/cluster/lib/libdsdev.so`

ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

Unresolved link to " printf3C", 1041 ページの`scds_syslog_debug`(3HA), 1079 ページの`scha_cluster_getlogfacility`(3HA), Unresolved link to " syslog3C", Unresolved link to " syslog.conf4", Unresolved link to " attributes5"

名前

scds_syslog_debug — システムログにデバッグメッセージを書き込む

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>void scds_syslog_debug(int
     debug_level, constchar *format...
```

`scds_syslog_debug()` 関数は、システムログにデバッグメッセージを書き込みます。この関数は、1079 ページの`scha_cluster_getlogfacility`(3HA) 関数から返される機能を使用します。

すべての syslog メッセージには、次の接頭辞がつきます。

`SC[`<*resourceTypeName* >,<*resourceGroupName*>,< *resourceName*>,<*methodName*>

現在使用中のデバッグレベルよりも `debug_level` を上げた場合、情報は書き込まれません。

DSDL の定義では、最大デバッグレベル `SCDS_MAX_DEBUG_LEVEL` は `9` です。
呼び出し元プログラムが `scds_syslog_debug()` の前に呼び出す必要のある 987 ページの`scds_initialize`(3HA) 関数は、`/var/cluster/rgm/rt/`<*resourceTypeName*>`/loglevel` ファイルから現在のデバッグレベルを取得します。

注意 - システムログに書き込まれるメッセージは多言語化されません。この関数と `gettext()` などのメッセージ変換関数を併用しないでください。

次のパラメータがサポートされます。

`debug_level` メッセージが書き込まれるデバッグレベルです。有効なデバッグレベルは 1 から `SCDS_MAX_DEBUG_LEVEL`(DSDL の定義では 9) です。指定のデバッグレベルが呼び出し元プログラムによって設定されたデバッグレベルより大きい場合、メッセージはシステムログに書き込まれません。

`format` Unresolved link to " printf3C" によって指定されるメッセージ形式の文字列です

`...` Unresolved link to " printf3C" によって指定される変数で、`format` パラメータで表されます。

例 367 すべてのデバッグメッセージの表示

リソースタイプ SUNW.iws のデバッグメッセージをすべて表示するには、全クラスタノード上で次のコマンドを発行します。

```
echo 9 > /var/cluster/rgm/rt/SUNW.iws/loglevel
```

例 368 デバッグメッセージの抑制

リソースタイプ SUNW.iws のデバッグメッセージを抑制するには、全クラスタノード上で次のコマンドを発行します。

```
echo 0 > /var/cluster/rgm/rt/SUNW.iws/loglevel
```

`/usr/cluster/include/rgm/libdsdev.h`
: インクルードファイル

`/usr/cluster/lib/libdsdev.so`
: ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

Unresolved link to " printf3C", 1039 ページの`scds_syslog`(3HA), 1079 ページの`scha_cluster_getlogfacility`(3HA), Unresolved link to " syslog3C", Unresolved link to " syslog.conf4", Unresolved link to " attributes5"

名前

scds_timerun, scds_timerun_delay — 指定されたコマンドを、指定された時間だけ実行する

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l dsdev#include <rgm/libdsdev.h>
     scha_err_t scds_timerun(scds_handle_t handle, const char *command,
     time_t timeout, int signal, int *cmd_exit_code);
     scha_err_t scds_timerun_delay(scds_handle_t handle, const char *command,
     time_t timeout, int signal, int *cmd_exit_code);
```

`scds_timerun()` 関数は、`hatimerun` を使用して、指定されたコマンドを実行します。`scds_timerun()` は、`timeout` 引数で指定された時間内にコマンドが完了しない場合、`signal` 引数で指定されたシグナルを送信してコマンドを強制終了します。

`command` 引数は入出力のリダイレクトをサポートしません。だたし、リダイレクトを実行するスクリプトを作成し、`command` 引数を使って、このスクリプトを `scds_timerun()` が実行するコマンドに指定することは可能です。

`scds_timerun_delay()()` 関数は、`scds_timerun()` と同じように動作します。ただし、`hatimerun` コマンドは `-d` (遅延) コマンド行オプションを付けて呼び出されます。このオプションは、コマンドの実行が開始されるまでタイムアウトクロックの開始を遅延させます。負荷の高いシステムでは、子プロセスがフォークされる時点から指定されたプログラムの実行が開始される時点まで、数秒間の遅延が発生する可能性があります。`-d` オプションを使用すれば、割り当てられたタイムアウト期間に対して追加の実行前時間がカウントされなくなります。

次のパラメータがサポートされます。

`handle`

987 ページの`scds_initialize(3HA)` から返されるハンドルです。

`command`

実行するコマンドを含む文字列です。

`timeout`

コマンドの実行のために割り当てられた時間 (秒) です。

`signal`

タイムアウトに達しても完了しないコマンドを強制終了するシグナルです。`signal = -1` の場合、`SIGKILL` が使用されます。Unresolved link to " signal3HEAD" を参照してください。

`cmd_exit_code`

コマンドの実行によって返されるコードです。

`scds_timerun()` 関数の戻り値は次のとおりです。

`0`	関数の実行に成功。
0 以外	関数の実行に失敗。

`SCHA_ERR_NOERR`

コマンドが実行され、`cmd_exit_code` に子プログラムの終了ステータスを格納。

`SCHA_ERR_INTERNAL`

`scds_timerun()` が、子プログラムでは検出されなかったタイムアウト以外のエラーを検出。または、733 ページの`hatimerun(1M)` がシグナル `SIGTERM` をキャッチしました。

`SCHA_ERR_INVAL`

無効な入力引数を検出。

`SCHA_ERR_TIMEOUT`

`command` 引数によって指定されたコマンドの実行が完了する前にタイムアウト。

その他のエラーコードについては、1047 ページの`scha_calls(3HA)` を参照してください。

`/usr/cluster/include/rgm/libdsdev.h`

インクルードファイル

`/usr/cluster/lib/libdsdev.so`

ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

733 ページの`hatimerun`(1M), 987 ページの`scds_initialize`(3HA), 1047 ページの`scha_calls`(3HA), Unresolved link to " signal3HEAD", Unresolved link to " attributes5"

名前

scha_calls — リソースタイプのコールバックメソッドおよびモニターの実装に使用される Oracle Solaris Cluster ライブラリ関数

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l scha#include <scha.h>scha_err_t scha_get_function(
     handle, const char *tag...);

scha_err_t scha_control(const char *tag…
```

Oracle Solaris Cluster ライブラリ関数である 1127 ページの`scha_resource_get`(3HA)、1215 ページの`scha_resourcetype_get`(3HA)、1185 ページの`scha_r` および 1175 ページの`scha_resource_setstatus`(3HA) は、コールバックメソッドの実装やリソースタイプのモニターで使用されるインタフェースを提供します。リソースタイプは、クラスタの Resource Group Manager (RGM) 機能によって制御されるサービスを表します。

「get」関数はクラスタ構成情報にアクセスします。これらの関数はすべて、同じ汎用シグニチャを持ちます。これらの関数は、以前の「open」関数の呼び出しから返された *handle* 引数を取ります。この *handle* は、クラスタ構成情報内のアクセスされるオブジェクトを示します。*tag* 引数は、アクセスされるオブジェクトのプロパティーを示します。*tag* の値により、引数を追加する必要があるかどうかと、要求された情報を返す最後の「out」引数の型が決定されます。「close」呼び出しが行われるまでは、同じハンドルを使って繰り返し「get」呼び出しを行えます。「close」呼び出しは、ハンドルを無効化し、「get」呼び出しから返された値に割り当てられているメモリーを解放します。

値を返す必要がある場合、各「get」呼び出しでメモリーが割り当てられます。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されたりすることはありません。

1101 ページの`scha_control`(3HA) 関数にはまた、制御操作を示すが、出力引数で情報を返さない *tag* 引数もあります。

699 ページの`scha_resource_setstatus`(1HA) コマンドは、RGM によって管理されるリソースの `Status` および `Status_msg` プロパティーを設定します。

各関数の *tag* 引数値として受け付けられるマクロ値と各 *tag* の引数型については、個々の関数のマニュアルページを参照してください。次のセクションでは、出力引数の型について説明します。

scha 関数には、scha_err_t enum 型の戻り値が 1 セットあります。enum の記号、整数値、および終了コードの意味については、戻り値 を参照してください。

1239 ページのscha_strerror(3HA) 関数は、scha 関数によって返された scha_err_t コードを適切なエラーメッセージに変換します。

出力引数のデータ型

uint_t

符号なし整数型。このタイプは、システムヘッダーファイル sys/types.h で定義されます。

boolean_t

このタイプは、システムヘッダーファイル sys/types.h で定義されます。

```
typedef enum { B_FALSE, B_TRUE } boolean_t;
```

scha_switch_t

リソースプロパティーの値 On_Off_switch または Monitored_switch を示す enum 型です。

```
typedef enum scha_switch {
    SCHA_SWITCH_DISABLED = 0,
    SCHA_SWITCH_ENABLED
} scha_switch_t;
```

scha_rg_preemption_mode_t

Preemption_mode リソースグループプロパティーの値を示す enum 型。

```
typedef enum scha_rg_preemption_mode {
    SCHA_HAS_PREEMPTION_COST = 0,
    SCHA_NO_PREEMPTION_COST,
    SCHA_NEVER_PREEMPT_RG
} scha_rg_preemption_mode_t;
```

scha_rsstate_t

リソースの状態を示す enum 型です。

```
typedef enum scha_rsstate {
    SCHA_RSSTATE_ONLINE = 0,
    SCHA_RSSTATE_OFFLINE,
    SCHA_RSSTATE_START_FAILED,
    SCHA_RSSTATE_STOP_FAILED,
    SCHA_RSSTATE_MONITOR_FAILED,
    SCHA_RSSTATE_ONLINE_NOT_MONITORED,
    SCHA_RSSTATE_STARTING,
    SCHA_RSSTATE_STOPPING
} scha_rsstate_t;
```

scha_rgstate_t

リソースグループの状態を示す enum 型です。

```
typedef enum scha_rgstate {
    SCHA_RGSTATE_UNMANAGED = 0,
    SCHA_RGSTATE_ONLINE,
    SCHA_RGSTATE_OFFLINE,
    SCHA_RGSTATE_PENDING_ONLINE,
    SCHA_RGSTATE_PENDING_OFFLINE,
    SCHA_RGSTATE_ERROR_STOP_FAILED
    SCHA_RGSTATE_ONLINE_FAULTED,
    SCHA_RGSTATE_PENDING_ONLINE_BLOCKED
} scha_rgstate_t;
```

scha_rgmode_t

リソースグループのモードがフェイルオーバーかスケーラブルかを示す enum 型です。

```
typedef enum scha_rgmode {
 RGMODE_NONE = 0,
 RGMODE_FAILOVER,
 RGMODE_SCALABLE
} scha_rgmode_t;
```

scha_failover_mode_t

リソースプロパティー Failover_Mode の値を示す enum 型です。

```
typedef enum scha_failover_mode {
    SCHA_FOMODE_NONE = 0,
    SCHA_FOMODE_HARD,
    SCHA_FOMODE_SOFT,
    SCHA_FOMODE_RESTART_ONLY,
    SCHA_FOMODE_LOG_ONLY
} scha_failover_mode_t;
```

scha_initnodes_flag_t

リソースタイププロパティー Init_nodes の値を示す enum 型です。

```
typedef enum scha_initnodes_flag {
    SCHA_INFLAG_RG_PRIMARIES = 0,
    SCHA_INFLAG_RT_INSTALLED_NODES
} scha_initnodes_flag_t;
```

scha_node_state_t

ノードが起動しているかどうかを示す enum 型です。

```
typedef enum scha_node_state {
    SCHA_NODE_UP = 0,
    SCHA_NODE_DOWN
} scha_node_state_t;
```

scha_str_array_t

文字列のリストの値を格納する構造体です。

```
typedef struct scha_str_array {
```

```
    uint_t       array_cnt;
    boolean_t    is_ALL_value;
    char         **str_array;
} scha_str_array_t;
```

`array_cnt` リスト内の要素数を示します。

`is_ALL_value` プロパティーの値が「all」(ワイルドカードまたはアスタリスク (*) 文字ともいう) に設定されると、`is_ALL_value` は `B_TRUE` に、`str_array` は `NULL` にそれぞれ設定されます。そのため、`str_array` は無視されます。

`str_array` `array_cnt strings` 配列へのポインタです。

`scha_uint_array_t`

符号なし整数のリストの値を格納する構造体です。

```
typedef struct scha_uint_array {
    uint_t array_cnt;
    uint_t *int_array;
} scha_uint_array_t;
```

`array_cnt` リスト内の要素数です。

`int_array` 符号なし整数 `array_cnt` の配列へのポインタです。

`scha_status_value_t`

リソースのステータスとステータスメッセージを返す構造体です。

```
typedef struct scha_status_value {
    scha_rsstatus_t        status;
    char                   *status_msg;
} scha_status_value_t;

typedef enum scha_rsstatus {
    SCHA_RSSTATUS_ONLINE = 0,
    SCHA_RSSTATUS_OFFLINE,
    SCHA_RSSTATUS_FAULTED,
    SCHA_RSSTATUS_DEGRADED,
    SCHA_RSSTATUS_UNKNOWN
} scha_rsstatus_t;
```

`status` リソースモニターの設定に従ってリソースのステータスを示す `enum` の値です。

`scha_extprop_value_t`

拡張プロパティーの値を返すために使用される構造体です。

`prop_type` 構造体メンバーは、拡張プロパティーの型を示し、`prop_type` フィールドと戻り値で使用される共用体要素を決めます。

```
SCHA_PTYPE_STRING        val_str
SCHA_PTYPE_INT           val_int
SCHA_PTYPE_ENUM          val_enum
SCHA_PTYPE_BOOLEAN       val_boolean
SCHA_PTYPE_STRINGARRAY   val_strarray


typedef struct scha_extprop_value {
    scha_prop_type_t  prop_type;
    union {
          char              *val_str;
          int               val_int;
          char              *val_enum;
          boolean_t         val_boolean;
          scha_str_array_t  *val_strarray;
          } val;
} scha_extprop_value_t;
```

次は、1239 ページのscha_strerror(3HA) から返される scha_err_t エラー番号およびエラーコードの一覧です。

0 SCHA_ERR_NOERR	エラーは検出されませんでした。
1 SCHA_ERR_NOMEM	スワップが十分ではありません。
2 SCHA_ERR_HANDLE	無効なリソース管理ハンドルです。
3 SCHA_ERR_INVAL	無効な入力引数です。
4 SCHA_ERR_TAG	無効な API タグです。
5 SCHA_ERR_RECONF	クラスタは再構成されます。
6 SCHA_ERR_ACCESS	アクセス権が拒否されました。
7 SCHA_ERR_SEQID	最後の scha_*_open 呼び出し以降に、リソース、リソースグループまたはリソースタイプが更新されました。
8 SCHA_ERR_DEPEND	オブジェクトの依存の問題です。
9 SCHA_ERR_STATE	オブジェクトの状態が不正です。
10 SCHA_ERR_METHOD	無効なメソッドです。
11 SCHA_ERR_NODE	無効なノードです。

12 SCHA_ERR_RG	無効なリソースグループです。
13 SCHA_ERR_RT	無効なリソースタイプです。
14 SCHA_ERR_RSRC	無効なリソースです。
15 SCHA_ERR_PROP	無効なプロパティーです。
16 SCHA_ERR_CHECKS	妥当性検査に失敗しました。
17 SCHA_ERR_RSTATUS	リソースのステータスが不正です。
18 SCHA_ERR_INTERNAL	内部エラーが発生しました。
19 SCHA_ERR_CLUSTER	ほかのクラスタと通信できません。
20 SCHA_ERR_ZONE_CLUSTER	ゾーンクラスタが無効です。
21 SCHA_ERR_ZC_DOWN	ゾーンクラスタの状態が「実行中」ではありません。
22 SCHA_ERR_LOADLIMIT	負荷制限が無効です。
31 SCHA_ERR_TIMEOUT	操作がタイムアウトしました。
32 SCHA_ERR_FAIL	フェイルオーバーの試みが失敗しました。

/usr/cluster/include/scha.h	インクルードファイル
/usr/cluster/lib/libscha.so	ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

675 ページのscha_cmds(1HA),
699 ページのscha_resource_setstatus(1HA)1063 ページのscha_cluster_get(3HA),
1101 ページのscha_control(3HA), 1127 ページのscha_resource_get(3HA),

1185 ページの`scha_resourcegroup_get`(3HA),
1175 ページの`scha_resource_setstatus`(3HA),
1215 ページの`scha_resourcetype_get`(3HA), 1239 ページの`scha_strerror`(3HA),
Unresolved link to " attributes5"

名前

scha_cluster_open, scha_cluster_open_zone, scha_cluster_get, scha_cluster_get_zone, scha_cluster_close — クラスタに関する情報へのアクセスおよび取得

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l scha#include <scha.h>scha_err_t scha_cluster_open(
     scha_cluster_t *handle);

scha_err_t scha_cluster_open_zone(const char *cluster,
     scha_cluster_t *handlep);

scha_err_t scha_cluster_get(scha_cluster_t handle, const char **
     tag, ...);

scha_err_t scha_cluster_get_zone(const char *cluster,
     scha_cluster_t handlep, const char *cluster_tag, ...);

scha_err_t scha_cluster_close(scha_cluster_t handle);
```

`scha_cluster_open()`、`scha_cluster_get ()`、および `scha_cluster_close()` 関数は、クラスタに関する情報を取得するために一緒に使用します。

`scha_cluster_open()` はクラスタアクセスを初期化し、`scha_cluster_get()` が使用するアクセスハンドルを返します。*handle* 引数は、関数が返す値を格納する変数のアドレスです。

`scha_cluster_get()` 関数は、*tag* 引数に指定されるクラスタの情報にアクセスします。*handle* 引数は、`scha_cluster_open()` の事前呼び出しによって返される値です。*tag* 引数は、`scha_tags.h` ヘッダーファイルのマクロで定義される文字列値です。タグに続く引数は、*tag* 引数の値によって異なります。

情報の取得元となるクラスタノードを指定するために、*tag* 引数のあとに追加の引数を指定する必要がある場合もあります。引数リストの最後の引数は、*tag* 引数で指定される情報の格納に適した変数型にする必要があります。これは出力引数で、クラスタの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。`scha_cluster_get()` 関数が返す情報を格納するために割り当てられたメモリーは、`scha_cluster_get()` 関数に使用したハンドルで `scha_cluster_close()` を呼び出すまで、そのまま残ります。

`scha_cluster_close()` は、以前の `scha_cluster_get()` 関数の呼び出しから返された `handle` 引数を取ります。この関数は、このハンドルを無効にして、このハンドルで行われた

scha_cluster_get() 呼び出しが返した値に割り当てられたメモリーを解放します。値を返す必要が生じるごとに、個々の get 呼び出しでメモリーが割り当てられる点に注意してください。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されることはありません。

scha_cluster_open_zone() 関数と scha_cluster_get_zone() 関数はそれぞれ、scha_cluster_open() および scha_cluster_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_cluster_open_zone() または scha_cluster_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_cluster_open() または scha_cluster_get() と等しくなります)。

scha_cluster_open_zone() から返されたハンドルを閉じるには、scha_cluster_close() を使用します。cluster 引数は不要です。

tag 引数に使用できるマクロ

次に、scha_tags.h に定義されており、*tag* 引数に使用できるマクロを示します。ここでは出力引数および追加引数の型を説明します。構造体と enum 型は、1047 ページのscha_calls(3HA) で説明されています。

SCHA_ALL_LOADLIMITS
: 出力引数の型は scha_str_array_t** です。
: このマクロは、クラスタに定義されているすべての loadlimit 名を返します。

SCHA_ALL_NODEIDS
: 出力引数の型は scha_uint_array_t** です。
: 返される値は、クラスタ内のすべてのノードの数値識別子です。

SCHA_ALL_NODENAMES
: 出力引数の型は scha_str_array_t** です。
: 返される値は、クラスタ内のすべてのノードの名前です。

SCHA_ALL_PRIVATELINK_HOSTNAMES
: 出力引数の型は scha_str_array_t** です。

このマクロは、クラスタインターコネクト上にノードのアドレスを指定するための、すべてのクラスタノードのホスト名を返します。

SCHA_ALL_PSTRINGS

出力引数の型は scha_str_array_t** です。

このマクロは、クラスタに定義されたすべてのプライベート文字列の名前を返します。プライベート文字列の詳細は、217 ページのclpstring(1CL) のマニュアルページを参照してください。

SCHA_ALL_RESOURCEGROUPS

出力引数の型は scha_str_array_t** です。

返される値は、クラスタ上で管理されているすべてのリソースグループの名前です。

SCHA_ALL_RESOURCETYPES

出力引数の型は scha_str_array_t** です。

返される値は、クラスタに登録されているすべてのリソースタイプの名前です。

SCHA_CLUSTERNAME

出力引数の型は char** です。

返される値は、クラスタの名前です。

SCHA_HARD_LOADLIMIT

出力引数の型は scha_str_array_t** です。

このマクロは、指定された範囲名に対し、クラスタのすべてのノードの強い負荷制限値を返します。負荷制限の名前文字列である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d" で、左側の文字列は *nodename*、右側の整数はそのノードで指定されている制限名に対する強い負荷制限値です。強い制限値を指定しない場合、強い制限値の値として -1 が表示されます。

SCHA_LOADLIMIT_PROPS

出力引数の型は scha_str_array_t** です。

このマクロは、指定された範囲名に対し、クラスタのすべてのノードの弱い負荷制限値 (/ 区切り) を返します。負荷制限の名前文字列である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d/%d" で、左側の文字列は *nodename*、最初の整数は弱い制限値、2 番目の整数は強い制限値です。強い制限値を指定しない場合、強い制限値の値として -1 が表示されます。弱い制限値を指定しない場合、弱い制限値の値として 0 が表示されます。

SCHA_LOADLIMITS_NODE

出力引数の型は scha_str_array_t** です。

このマクロは、特定のノードの負荷制限値 (/ 区切り) と制限名を返します。*nodename* である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d/%d" で、文字列は指定したノードに定義されている制限名、最初の整数は弱い制限値、2 番目の整数は強い制限値です。強い制限値を指定しない場合、強い制限値の値として -1 が表示されます。弱い制限値を指定しない場合、弱い制限値の値として 0 が表示されます。

SCHA_NODEID_LOCAL

出力引数の型は uint_t* です。

返される値は、コマンドを実行するノードの数値識別子です。

SCHA_NODEID_NODENAME

出力引数の型は uint_t* です。追加引数の型は char * です。このマクロに必要な追加引数には、クラスタノードの名前を指定します。

返される値は、指定した名前に該当するノードの数値識別子です。

SCHA_NODENAME_LOCAL

出力引数の型は char** です。

返される値は、関数が実行されたクラスタノードの名前です。

SCHA_NODENAME_NODEID

出力引数の型は char** です。追加引数の型は uint_t です。追加引数には、数値式のクラスタノード識別子を指定します。

返される値は、数値識別子で指定されるクラスタノードの名前です。

SCHA_NODESTATE_LOCAL

出力引数の型は scha_node_state_t* です。

返される値は、コマンドが実行されたノードの状態に応じて SCHA_NODE_UP または SCHA_NODE_DOWN となります。

SCHA_NODESTATE_NODE

出力引数の型は scha_node_state_t* です。追加引数の型は char* です。このマクロに必要なフラグ無し追加引数には、クラスタノードの名を指定します。

返される値は、名前付きノードの状態に応じて SCHA_NODE_UP または SCHA_NODE_DOWN となります。

SCHA_PRIVATELINK_HOSTNAME_LOCAL

出力引数の型は char** です。

このマクロは、クラスタインターコネクト上にコマンドを実行するノードのアドレスを指定するためのホスト名を返します。

SCHA_PRIVATELINK_HOSTNAME_NODE

出力引数の型は char** です。追加引数の型は char * です。このマクロに必要なフラグ無し追加引数には、クラスタノードの名前が入ります。

このマクロは、クラスタインターコネクト上に指定のノードのアドレスを指定するためのホスト名を返します。

SCHA_PSTRING

出力引数の型は char** です。

このマクロは、プライベート文字列の平文値を返します。プライベート文字列の名前を指定する、char * 型の追加の引数が必要です。スーパーユーザー以外のユーザーがこのクエリータグを使用するには、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。プライベート文字列の詳細は、217 ページのclpstring(1CL) のマニュアルページを参照してください。

SCHA_RESOURCE_SECURITY

出力引数の型は char** です。

このマクロは、resource_security クラスタプロパティーの現在の設定を返します。

SCHA_SOFT_LOADLIMIT

出力引数の型は scha_str_array_t** です。

このマクロは、指定された範囲名に対し、クラスタのすべてのノードの弱い負荷制限値を返します。負荷制限の名前文字列である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d" で、左側の文字列は *nodename*、右側の整数はそのノードで指定されている制限名に対する弱い負荷制限値です。弱い制限値を指定しない場合、弱い制限値の値として 0 が表示されます。

SCHA_SYSLOG_FACILITY

出力引数の型は int* です。

このマクロは、RGM がログメッセージで使用する Unresolved link to " syslog3C" 関数の番号を返します。返される値は 24 です。これは LOG_DAEMON 機能の値に対応しています。

SCHA_ERR_NOERR　　関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

例 369　scha_cluster_get() 関数の使用

次の例では、scha_cluster_get() 関数を使用して、全クラスタノードの名前を取得します。この関数を使用すると、ノードが稼働しているかどうかもわかります。

このコード例では、制限が構成されている各クラスタノードに対し、**mylimit** という limitname に構成されている弱い負荷制限値と強い負荷制限値も出力されます。各ノードの負荷制限値の出力形式は *nodename=softlimit*/[*hardlimit*] で、強い制限値が設定されていない場合、*hardlimit* の値は無制限 (-1) です。

```
#include <scha.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
        scha_err_t              err;
        scha_node_state_t       node_state;
        scha_str_array_t        *all_nodenames;
        scha_cluster_t          handle;
        int                     ix;
        const char              *str;
        scha_str_array_t        *load_limits;

        err = scha_cluster_open(&handle);
        if (err != SCHA_ERR_NOERR) {
                fprintf(stderr, "FAILED: scha_cluster_open()0);
                exit(err);
        }

        err = scha_cluster_get(handle, SCHA_ALL_NODENAMES, &all_nodenames);
        if (err != SCHA_ERR_NOERR) {
                fprintf(stderr, "FAILED: scha_cluster_get()0);
                exit(err);
        }

        for (ix = 0; ix < all_nodenames->array_cnt; ix++) {
                err = scha_cluster_get(handle, SCHA_NODESTATE_NODE,
                    all_nodenames->str_array[ix], &node_state);
                if (err != SCHA_ERR_NOERR) {
                        fprintf(stderr, "FAILED: scha_cluster_get()"
                            "SCHA_NODESTATE_NODE0);
                        exit(err);
                }

                switch (node_state) {
                case SCHA_NODE_UP:
                        str = "UP";
                        break;
                case SCHA_NODE_DOWN:
                        str = "DOWN";
                        break;
                }

                printf("State of node: %s value: %s\n",
                    all_nodenames->str_array[ix], str);
        }
        err = scha_cluster_get(handle, SCHA_LOADLIMIT_PROPS, "mylimit",
```

```
            &load_limits);

        printf("\n\nLoad limits settings for limitname 'mylimit':\n\n");

        for (ix = 0; ix < load_limits->array_cnt; ix++) {
            printf("%s\n", load_limits->str_array[ix]);
        }
}
```

`/usr/cluster/include/scha.h`	インクルードファイル
`/usr/cluster/lib/libscha.so`	ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	発展中

669 ページの`scha_cluster_get`(1HA), 1047 ページの`scha_calls`(3HA),
1079 ページの`scha_cluster_getlogfacility`(3HA),
1081 ページの`scha_cluster_getnodename`(3HA), 1239 ページの`scha_strerror`(3HA),
Unresolved link to " syslog3C", Unresolved link to " attributes5",
1319 ページの`rg_properties`(5)

名前

scha_cluster_open, scha_cluster_open_zone, scha_cluster_get, scha_cluster_get_zone, scha_cluster_close — クラスタに関する情報へのアクセスおよび取得

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l scha#include <scha.h>scha_err_t scha_cluster_open(
     scha_cluster_t *handle);

scha_err_t scha_cluster_open_zone(const char *cluster,
     scha_cluster_t *handlep);

scha_err_t scha_cluster_get(scha_cluster_t handle, const char **
     tag, ...);

scha_err_t scha_cluster_get_zone(const char *cluster,
     scha_cluster_t handlep, const char *cluster_tag, ...);

scha_err_t scha_cluster_close(scha_cluster_t handle);
```

`scha_cluster_open()`、`scha_cluster_get ()`、および `scha_cluster_close()` 関数は、クラスタに関する情報を取得するために一緒に使用します。

`scha_cluster_open()` はクラスタアクセスを初期化し、`scha_cluster_get()` が使用するアクセスハンドルを返します。*handle* 引数は、関数が返す値を格納する変数のアドレスです。

`scha_cluster_get()` 関数は、*tag* 引数に指定されるクラスタの情報にアクセスします。*handle* 引数は、`scha_cluster_open()` の事前呼び出しによって返される値です。*tag* 引数は、`scha_tags.h` ヘッダーファイルのマクロで定義される文字列値です。タグに続く引数は、*tag* 引数の値によって異なります。

情報の取得元となるクラスタノードを指定するために、*tag* 引数のあとに追加の引数を指定する必要がある場合もあります。引数リストの最後の引数は、*tag* 引数で指定される情報の格納に適した変数型にする必要があります。これは出力引数で、クラスタの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。`scha_cluster_get()` 関数が返す情報を格納するために割り当てられたメモリーは、`scha_cluster_get()` 関数に使用したハンドルで `scha_cluster_close()` を呼び出すまで、そのまま残ります。

`scha_cluster_close()` は、以前の `scha_cluster_get()` 関数の呼び出しから返された `handle` 引数を取ります。この関数は、このハンドルを無効にして、このハンドルで行われた

scha_cluster_get() 呼び出しが返した値に割り当てられたメモリーを解放します。値を返す必要が生じるごとに、個々の get 呼び出しでメモリーが割り当てられる点に注意してください。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されることはありません。

scha_cluster_open_zone() 関数と scha_cluster_get_zone() 関数はそれぞれ、scha_cluster_open() および scha_cluster_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_cluster_open_zone() または scha_cluster_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_cluster_open() または scha_cluster_get() と等しくなります)。

scha_cluster_open_zone() から返されたハンドルを閉じるには、scha_cluster_close() を使用します。cluster 引数は不要です。

tag 引数に使用できるマクロ

次に、scha_tags.h に定義されており、*tag* 引数に使用できるマクロを示します。ここでは出力引数および追加引数の型を説明します。構造体と enum 型は、1047 ページのscha_calls(3HA) で説明されています。

SCHA_ALL_LOADLIMITS
: 出力引数の型は scha_str_array_t** です。
: このマクロは、クラスタに定義されているすべての loadlimit 名を返します。

SCHA_ALL_NODEIDS
: 出力引数の型は scha_uint_array_t** です。
: 返される値は、クラスタ内のすべてのノードの数値識別子です。

SCHA_ALL_NODENAMES
: 出力引数の型は scha_str_array_t** です。
: 返される値は、クラスタ内のすべてのノードの名前です。

SCHA_ALL_PRIVATELINK_HOSTNAMES
: 出力引数の型は scha_str_array_t** です。

このマクロは、クラスタインターコネクト上にノードのアドレスを指定するための、すべてのクラスタノードのホスト名を返します。

SCHA_ALL_PSTRINGS

出力引数の型は scha_str_array_t** です。

このマクロは、クラスタに定義されたすべてのプライベート文字列の名前を返します。プライベート文字列の詳細は、217 ページのclpstring(1CL) のマニュアルページを参照してください。

SCHA_ALL_RESOURCEGROUPS

出力引数の型は scha_str_array_t** です。

返される値は、クラスタ上で管理されているすべてのリソースグループの名前です。

SCHA_ALL_RESOURCETYPES

出力引数の型は scha_str_array_t** です。

返される値は、クラスタに登録されているすべてのリソースタイプの名前です。

SCHA_CLUSTERNAME

出力引数の型は char** です。

返される値は、クラスタの名前です。

SCHA_HARD_LOADLIMIT

出力引数の型は scha_str_array_t** です。

このマクロは、指定された範囲名に対し、クラスタのすべてのノードの強い負荷制限値を返します。負荷制限の名前文字列である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d" で、左側の文字列は *nodename*、右側の整数はそのノードで指定されている制限名に対する強い負荷制限値です。強い制限値を指定しない場合、強い制限値の値として -1 が表示されます。

SCHA_LOADLIMIT_PROPS

出力引数の型は scha_str_array_t** です。

このマクロは、指定された範囲名に対し、クラスタのすべてのノードの弱い負荷制限値 (/ 区切り) を返します。負荷制限の名前文字列である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d/%d" で、左側の文字列は *nodename*、最初の整数は弱い制限値、2 番目の整数は強い制限値です。強い制限値を指定しない場合、強い制限値の値として -1 が表示されます。弱い制限値を指定しない場合、弱い制限値の値として 0 が表示されます。

SCHA_LOADLIMITS_NODE

出力引数の型は scha_str_array_t** です。

このマクロは、特定のノードの負荷制限値 (/ 区切り) と制限名を返します。*nodename* である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d/%d" で、文字列は指定したノードに定義されている制限名、最初の整数は弱い制限値、2 番目の整数は強い制限値です。強い制限値を指定しない場合、強い制限値の値として -1 が表示されます。弱い制限値を指定しない場合、弱い制限値の値として 0 が表示されます。

SCHA_NODEID_LOCAL

出力引数の型は uint_t* です。

返される値は、コマンドを実行するノードの数値識別子です。

SCHA_NODEID_NODENAME

出力引数の型は uint_t* です。追加引数の型は char * です。このマクロに必要な追加引数には、クラスタノードの名前を指定します。

返される値は、指定した名前に該当するノードの数値識別子です。

SCHA_NODENAME_LOCAL

出力引数の型は char** です。

返される値は、関数が実行されたクラスタノードの名前です。

SCHA_NODENAME_NODEID

出力引数の型は char** です。追加引数の型は uint_t です。追加引数には、数値式のクラスタノード識別子を指定します。

返される値は、数値識別子で指定されるクラスタノードの名前です。

SCHA_NODESTATE_LOCAL

出力引数の型は scha_node_state_t* です。

返される値は、コマンドが実行されたノードの状態に応じて SCHA_NODE_UP または SCHA_NODE_DOWN となります。

SCHA_NODESTATE_NODE

出力引数の型は scha_node_state_t* です。追加引数の型は char* です。このマクロに必要なフラグ無し追加引数には、クラスタノードの名を指定します。

返される値は、名前付きノードの状態に応じて SCHA_NODE_UP または SCHA_NODE_DOWN となります。

SCHA_PRIVATELINK_HOSTNAME_LOCAL

出力引数の型は char** です。

このマクロは、クラスタインターコネクト上にコマンドを実行するノードのアドレスを指定するためのホスト名を返します。

SCHA_PRIVATELINK_HOSTNAME_NODE

出力引数の型は char** です。追加引数の型は char * です。このマクロに必要なフラグ無し追加引数には、クラスタノードの名前が入ります。

このマクロは、クラスタインターコネクト上に指定のノードのアドレスを指定するためのホスト名を返します。

SCHA_PSTRING

出力引数の型は char** です。

このマクロは、プライベート文字列の平文値を返します。プライベート文字列の名前を指定する、char * 型の追加の引数が必要です。スーパーユーザー以外のユーザーがこのクエリータグを使用するには、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。プライベート文字列の詳細は、217 ページのclpstring(1CL) のマニュアルページを参照してください。

SCHA_RESOURCE_SECURITY

出力引数の型は char** です。

このマクロは、resource_security クラスタプロパティーの現在の設定を返します。

SCHA_SOFT_LOADLIMIT

出力引数の型は scha_str_array_t** です。

このマクロは、指定された範囲名に対し、クラスタのすべてのノードの弱い負荷制限値を返します。負荷制限の名前文字列である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d" で、左側の文字列は *nodename*、右側の整数はそのノードで指定されている制限名に対する弱い負荷制限値です。弱い制限値を指定しない場合、弱い制限値の値として 0 が表示されます。

SCHA_SYSLOG_FACILITY

出力引数の型は int* です。

このマクロは、RGM がログメッセージで使用する Unresolved link to " syslog3C" 関数の番号を返します。返される値は 24 です。これは LOG_DAEMON 機能の値に対応しています。

SCHA_ERR_NOERR　　関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

例 370　　scha_cluster_get() 関数の使用

次の例では、scha_cluster_get() 関数を使用して、全クラスタノードの名前を取得します。この関数を使用すると、ノードが稼働しているかどうかもわかります。

このコード例では、制限が構成されている各クラスタノードに対し、**mylimit** という limitname に構成されている弱い負荷制限値と強い負荷制限値も出力されます。各ノードの負荷制限値の出力形式は *nodename=softlimit*/[*hardlimit*] で、強い制限値が設定されていない場合、*hardlimit* の値は無制限 (-1) です。

```
#include <scha.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
        scha_err_t              err;
        scha_node_state_t       node_state;
        scha_str_array_t        *all_nodenames;
        scha_cluster_t          handle;
        int                     ix;
        const char              *str;
        scha_str_array_t        *load_limits;

        err = scha_cluster_open(&handle);
        if (err != SCHA_ERR_NOERR) {
                fprintf(stderr, "FAILED: scha_cluster_open()0);
                exit(err);
        }

        err = scha_cluster_get(handle, SCHA_ALL_NODENAMES, &all_nodenames);
        if (err != SCHA_ERR_NOERR) {
                fprintf(stderr, "FAILED: scha_cluster_get()0);
                exit(err);
        }

        for (ix = 0; ix < all_nodenames->array_cnt; ix++) {
                err = scha_cluster_get(handle, SCHA_NODESTATE_NODE,
                    all_nodenames->str_array[ix], &node_state);
                if (err != SCHA_ERR_NOERR) {
                        fprintf(stderr, "FAILED: scha_cluster_get()"
                            "SCHA_NODESTATE_NODE0);
                        exit(err);
                }

                switch (node_state) {
                case SCHA_NODE_UP:
                        str = "UP";
                        break;
                case SCHA_NODE_DOWN:
                        str = "DOWN";
                        break;
                }

                printf("State of node: %s value: %s\n",
                    all_nodenames->str_array[ix], str);
        }
        err = scha_cluster_get(handle, SCHA_LOADLIMIT_PROPS, "mylimit",
```

```
            &load_limits);

        printf("\n\nLoad limits settings for limitname 'mylimit':\n\n");

        for (ix = 0; ix < load_limits->array_cnt; ix++) {
            printf("%s\n", load_limits->str_array[ix]);
        }
}
```

`/usr/cluster/include/scha.h`	インクルードファイル
`/usr/cluster/lib/libscha.so`	ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	発展中

669 ページの`scha_cluster_get`(1HA), 1047 ページの`scha_calls`(3HA),
1079 ページの`scha_cluster_getlogfacility`(3HA),
1081 ページの`scha_cluster_getnodename`(3HA), 1239 ページの`scha_strerror`(3HA),
Unresolved link to " syslog3C", Unresolved link to " attributes5",
1319 ページの`rg_properties`(5)

名前

scha_cluster_open, scha_cluster_open_zone, scha_cluster_get, scha_cluster_get_zone, scha_cluster_close — クラスタに関する情報へのアクセスおよび取得

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l scha#include <scha.h>scha_err_t scha_cluster_open(
     scha_cluster_t *handle);

scha_err_t scha_cluster_open_zone(const char *cluster,
     scha_cluster_t *handlep);

scha_err_t scha_cluster_get(scha_cluster_t handle, const char **
     tag, ...);

scha_err_t scha_cluster_get_zone(const char *cluster,
     scha_cluster_t handlep, const char *cluster_tag, ...);

scha_err_t scha_cluster_close(scha_cluster_t handle);
```

`scha_cluster_open()`、`scha_cluster_get ()`、および `scha_cluster_close()` 関数は、クラスタに関する情報を取得するために一緒に使用します。

`scha_cluster_open()` はクラスタアクセスを初期化し、`scha_cluster_get()` が使用するアクセスハンドルを返します。*handle* 引数は、関数が返す値を格納する変数のアドレスです。

`scha_cluster_get()` 関数は、*tag* 引数に指定されるクラスタの情報にアクセスします。*handle* 引数は、`scha_cluster_open()` の事前呼び出しによって返される値です。*tag* 引数は、`scha_tags.h` ヘッダーファイルのマクロで定義される文字列値です。タグに続く引数は、*tag* 引数の値によって異なります。

情報の取得元となるクラスタノードを指定するために、*tag* 引数のあとに追加の引数を指定する必要がある場合もあります。引数リストの最後の引数は、*tag* 引数で指定される情報の格納に適した変数型にする必要があります。これは出力引数で、クラスタの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。`scha_cluster_get()` 関数が返す情報を格納するために割り当てられたメモリーは、`scha_cluster_get()` 関数に使用したハンドルで `scha_cluster_close()` を呼び出すまで、そのまま残ります。

`scha_cluster_close()` は、以前の `scha_cluster_get()` 関数の呼び出しから返された `handle` 引数を取ります。この関数は、このハンドルを無効にして、このハンドルで行われた

scha_cluster_get() 呼び出しが返した値に割り当てられたメモリーを解放します。値を返す必要が生じるごとに、個々の get 呼び出しでメモリーが割り当てられる点に注意してください。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されることはありません。

scha_cluster_open_zone() 関数と scha_cluster_get_zone() 関数はそれぞれ、scha_cluster_open() および scha_cluster_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_cluster_open_zone() または scha_cluster_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_cluster_open() または scha_cluster_get() と等しくなります)。

scha_cluster_open_zone() から返されたハンドルを閉じるには、scha_cluster_close() を使用します。cluster 引数は不要です。

tag 引数に使用できるマクロ

次に、scha_tags.h に定義されており、*tag* 引数に使用できるマクロを示します。ここでは出力引数および追加引数の型を説明します。構造体と enum 型は、1047 ページのscha_calls(3HA) で説明されています。

SCHA_ALL_LOADLIMITS
: 出力引数の型は scha_str_array_t** です。
: このマクロは、クラスタに定義されているすべての loadlimit 名を返します。

SCHA_ALL_NODEIDS
: 出力引数の型は scha_uint_array_t** です。
: 返される値は、クラスタ内のすべてのノードの数値識別子です。

SCHA_ALL_NODENAMES
: 出力引数の型は scha_str_array_t** です。
: 返される値は、クラスタ内のすべてのノードの名前です。

SCHA_ALL_PRIVATELINK_HOSTNAMES
: 出力引数の型は scha_str_array_t** です。

このマクロは、クラスタインターコネクト上にノードのアドレスを指定するための、すべてのクラスタノードのホスト名を返します。

SCHA_ALL_PSTRINGS

出力引数の型は scha_str_array_t** です。

このマクロは、クラスタに定義されたすべてのプライベート文字列の名前を返します。プライベート文字列の詳細は、217 ページのclpstring(1CL) のマニュアルページを参照してください。

SCHA_ALL_RESOURCEGROUPS

出力引数の型は scha_str_array_t** です。

返される値は、クラスタ上で管理されているすべてのリソースグループの名前です。

SCHA_ALL_RESOURCETYPES

出力引数の型は scha_str_array_t** です。

返される値は、クラスタに登録されているすべてのリソースタイプの名前です。

SCHA_CLUSTERNAME

出力引数の型は char** です。

返される値は、クラスタの名前です。

SCHA_HARD_LOADLIMIT

出力引数の型は scha_str_array_t** です。

このマクロは、指定された範囲名に対し、クラスタのすべてのノードの強い負荷制限値を返します。負荷制限の名前文字列である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d" で、左側の文字列は *nodename*、右側の整数はそのノードで指定されている制限名に対する強い負荷制限値です。強い制限値を指定しない場合、強い制限値の値として -1 が表示されます。

SCHA_LOADLIMIT_PROPS

出力引数の型は scha_str_array_t** です。

このマクロは、指定された範囲名に対し、クラスタのすべてのノードの弱い負荷制限値 (/ 区切り) を返します。負荷制限の名前文字列である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d/%d" で、左側の文字列は *nodename*、最初の整数は弱い制限値、2 番目の整数は強い制限値です。強い制限値を指定しない場合、強い制限値の値として -1 が表示されます。弱い制限値を指定しない場合、弱い制限値の値として 0 が表示されます。

SCHA_LOADLIMITS_NODE

出力引数の型は scha_str_array_t** です。

このマクロは、特定のノードの負荷制限値 (/ 区切り) と制限名を返します。*nodename* である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d/%d" で、文字列は指定したノードに定義されている制限名、最初の整数は弱い制限値、2 番目の整数は強い制限値です。強い制限値を指定しない場合、強い制限値の値として -1 が表示されます。弱い制限値を指定しない場合、弱い制限値の値として 0 が表示されます。

SCHA_NODEID_LOCAL

出力引数の型は uint_t* です。

返される値は、コマンドを実行するノードの数値識別子です。

SCHA_NODEID_NODENAME

出力引数の型は uint_t* です。追加引数の型は char * です。このマクロに必要な追加引数には、クラスタノードの名前を指定します。

返される値は、指定した名前に該当するノードの数値識別子です。

SCHA_NODENAME_LOCAL

出力引数の型は char** です。

返される値は、関数が実行されたクラスタノードの名前です。

SCHA_NODENAME_NODEID

出力引数の型は char** です。追加引数の型は uint_t です。追加引数には、数値式のクラスタノード識別子を指定します。

返される値は、数値識別子で指定されるクラスタノードの名前です。

SCHA_NODESTATE_LOCAL

出力引数の型は scha_node_state_t* です。

返される値は、コマンドが実行されたノードの状態に応じて SCHA_NODE_UP または SCHA_NODE_DOWN となります。

SCHA_NODESTATE_NODE

出力引数の型は scha_node_state_t* です。追加引数の型は char* です。このマクロに必要なフラグ無し追加引数には、クラスタノードの名を指定します。

返される値は、名前付きノードの状態に応じて SCHA_NODE_UP または SCHA_NODE_DOWN となります。

SCHA_PRIVATELINK_HOSTNAME_LOCAL

出力引数の型は char** です。

このマクロは、クラスタインターコネクト上にコマンドを実行するノードのアドレスを指定するためのホスト名を返します。

SCHA_PRIVATELINK_HOSTNAME_NODE

出力引数の型は char** です。追加引数の型は char * です。このマクロに必要なフラグ無し追加引数には、クラスタノードの名前が入ります。

このマクロは、クラスタインターコネクト上に指定のノードのアドレスを指定するためのホスト名を返します。

SCHA_PSTRING

出力引数の型は char** です。

このマクロは、プライベート文字列の平文値を返します。プライベート文字列の名前を指定する、char * 型の追加の引数が必要です。スーパーユーザー以外のユーザーがこのクエリータグを使用するには、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。プライベート文字列の詳細は、217 ページのclpstring(1CL) のマニュアルページを参照してください。

SCHA_RESOURCE_SECURITY

出力引数の型は char** です。

このマクロは、resource_security クラスタプロパティーの現在の設定を返します。

SCHA_SOFT_LOADLIMIT

出力引数の型は scha_str_array_t** です。

このマクロは、指定された範囲名に対し、クラスタのすべてのノードの弱い負荷制限値を返します。負荷制限の名前文字列である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d" で、左側の文字列は *nodename*、右側の整数はそのノードで指定されている制限名に対する弱い負荷制限値です。弱い制限値を指定しない場合、弱い制限値の値として 0 が表示されます。

SCHA_SYSLOG_FACILITY

出力引数の型は int* です。

このマクロは、RGM がログメッセージで使用する Unresolved link to " syslog3C" 関数の番号を返します。返される値は 24 です。これは LOG_DAEMON 機能の値に対応しています。

SCHA_ERR_NOERR　　関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

例 371　　scha_cluster_get() 関数の使用

次の例では、scha_cluster_get() 関数を使用して、全クラスタノードの名前を取得します。この関数を使用すると、ノードが稼働しているかどうかもわかります。

このコード例では、制限が構成されている各クラスタノードに対し、**mylimit** という limitname に構成されている弱い負荷制限値と強い負荷制限値も出力されます。各ノードの負荷制限値の出力形式は *nodename=softlimit*/[*hardlimit*] で、強い制限値が設定されていない場合、*hardlimit* の値は無制限 (-1) です。

```
#include <scha.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
        scha_err_t              err;
        scha_node_state_t       node_state;
        scha_str_array_t        *all_nodenames;
        scha_cluster_t          handle;
        int                     ix;
        const char              *str;
        scha_str_array_t        *load_limits;

        err = scha_cluster_open(&handle);
        if (err != SCHA_ERR_NOERR) {
                fprintf(stderr, "FAILED: scha_cluster_open()0);
                exit(err);
        }

        err = scha_cluster_get(handle, SCHA_ALL_NODENAMES, &all_nodenames);
        if (err != SCHA_ERR_NOERR) {
                fprintf(stderr, "FAILED: scha_cluster_get()0);
                exit(err);
        }

        for (ix = 0; ix < all_nodenames->array_cnt; ix++) {
                err = scha_cluster_get(handle, SCHA_NODESTATE_NODE,
                    all_nodenames->str_array[ix], &node_state);
                if (err != SCHA_ERR_NOERR) {
                        fprintf(stderr, "FAILED: scha_cluster_get()"
                            "SCHA_NODESTATE_NODE0);
                        exit(err);
                }

                switch (node_state) {
                case SCHA_NODE_UP:
                        str = "UP";
                        break;
                case SCHA_NODE_DOWN:
                        str = "DOWN";
                        break;
                }

                printf("State of node: %s value: %s\n",
                    all_nodenames->str_array[ix], str);
        }
        err = scha_cluster_get(handle, SCHA_LOADLIMIT_PROPS, "mylimit",
```

```
            &load_limits);

        printf("\n\nLoad limits settings for limitname 'mylimit':\n\n");

        for (ix = 0; ix < load_limits->array_cnt; ix++) {
            printf("%s\n", load_limits->str_array[ix]);
        }
}
```

`/usr/cluster/include/scha.h`	インクルードファイル
`/usr/cluster/lib/libscha.so`	ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	発展中

669 ページの`scha_cluster_get`(1HA), 1047 ページの`scha_calls`(3HA),
1079 ページの`scha_cluster_getlogfacility`(3HA),
1081 ページの`scha_cluster_getnodename`(3HA), 1239 ページの`scha_strerror`(3HA),
Unresolved link to " syslog3C", Unresolved link to " attributes5",
1319 ページの`rg_properties`(5)

名前

scha_cluster_getlogfacility — クラスタログ機能のアクセス

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l scha#include <scha.h> scha_err_t
     scha_cluster_getlogfacility(int *logfacility);
```

`scha_cluster_getlogfacility()` 関数は、クラスタログとして使用されるシステムログ機能の番号を返します。この値は、Solaris Unresolved link to " syslog3C" 関数とともに、イベントおよびステータスメッセージをクラスタログに記録するリソースタイプ実装によって使用されます。

この関数はエラーステータスを返します。成功した場合、*logfacility* 引数のポイント先の機能番号を返します。

`scha_cluster_getlogfacility()` 関数は、次の値を返します。

`0`	関数の実行に成功。
0 以外	関数の実行に失敗。

`SCHA_ERR_NOERR`	関数の実行に成功。

その他のエラーコードについては、1047 ページの`scha_calls`(3HA) を参照してください。

例 372 `scha_cluster_getlogfacility()` 関数の使用法

```
main()
{
 scha_err_t err_code;
 int logfacility;

 err_code = scha_cluster_getlogfacility(&logfacility);

 if (err_code == SCHA_ERR_NOERR) {
  openlog("test resource", LOG_CONS, logfacility);
  syslog(LOG_INFO, "Access function call succeeded.");
 }
}
```

`/usr/cluster/include/scha.h`	インクルードファイル

`/usr/cluster/lib/libscha.so` ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

Unresolved link to " syslog3C", 1047 ページの`scha_calls`(3HA), 1063 ページの`scha_cluster_get`(3HA), 1239 ページの`scha_strerror`(3HA), Unresolved link to " attributes5"

名前

scha_cluster_getnodename — ローカルクラスタノード名を返す

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l scha#include <scha.h> scha_err_t scha_cluster_getnodename(
     char **nodename);
```

`scha_cluster_getnodename()` 関数を実行すると、この関数を呼び出したクラスタノードの名前が返されます。ノード名は、必ずしも Solaris システム名と一致しているとは限りません。この関数は戻り値にエラーステータスを返すと同時に、実行に成功した場合は、ノード名の文字列を *nodename* 引数の指定位置に収めます。

呼び出しに失敗した場合、*nodename* に `NULL` が設定されます。`scha_cluster_getnodename()` 関数の呼び出し元には、標準 C ライブラリ関数の Unresolved link to " free3C" を使用して、返される文字列に割り当てられたメモリーを解放する責任があります。メモリーの解放が必要になるのは、この関数が成功した場合だけです。

`scha_cluster_getnodename()` 関数の戻り値は次のとおりです。

`0`	関数の実行に成功。
0 以外	関数の実行に失敗。

`SCHA_ERR_NOERR`	関数の実行に成功。

その他のエラーコードについては、1047 ページの`scha_calls`(3HA) を参照してください。

例 373 `scha_cluster_getnodename()` 関数の使用例

```
scha_err_t  err_code;
char *nodename;
err_code = scha_cluster_getnodename(&nodename);
...
if (nodename != NULL) free(nodename);
```

`/usr/cluster/include/scha.h`	インクルードファイル
`/usr/cluster/lib/libscha.so`	ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

Unresolved link to " free3C", 1047 ページの`scha_calls`(3HA), 1063 ページの`scha_cluster_get`(3HA), 1083 ページの`scha_cluster_getzone`(3HA), 1239 ページの`scha_strerror`(3HA), Unresolved link to " attributes5"

名前

scha_cluster_getzone — ゾーン名を返す

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l scha#include <scha.h> scha_err_t scha_cluster_getzone(
     char **zonename);
```

`scha_cluster_getzone()` 関数は、本関数の呼び出し元ゾーンを示す文字列を返します。この関数を非大域ゾーン内で呼び出すと、ゾーン名およびノード名は `nodename:zonename` の形式で返されます。この関数を大域ゾーン内で呼び出すと、ノード名だけが返されます。ノード名は、必ずしも Solaris システム名と一致しているとは限りません。関数はエラーステータスを返します。成功すると、関数はノード名を含む文字列と、*zonename* 引数によってポイントされた場所にあるゾーン名も返します。

呼び出しに失敗した場合、*zonename* 引数に `NULL` が設定されます。`scha_cluster_getzone()` 関数の呼び出し元には、標準 C ライブラリ関数の Unresolved link to " free3C" を使用して、返される文字列に割り当てられたメモリーを解放する責任があります。メモリーの解放が必要になるのは、この関数が成功した場合だけです。

`scha_cluster_getzone()` 関数の戻り値は次のとおりです。

`0`	関数の実行に成功。
0 以外	関数の実行に失敗。

`SCHA_ERR_NOERR`	関数の実行に成功。

その他のエラーコードについては、1047 ページの`scha_calls`(3HA) を参照してください。

例 374 `scha_cluster_getzone()` 関数の使用

```
scha_err_t  err_code;
char *zonename;
err_code = scha_cluster_getzone(&zonename);
...
if (zonename != NULL) free(zonename);
```

`/usr/cluster/include/scha.h`	インクルードファイル

`/usr/cluster/lib/libscha.so` ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

Unresolved link to " free3C", 1047 ページの`scha_calls`(3HA),
1063 ページの`scha_cluster_get`(3HA), 1081 ページの`scha_cluster_getnodename`(3HA),
1239 ページの`scha_strerror`(3HA), Unresolved link to " attributes5"

名前

scha_cluster_open, scha_cluster_open_zone, scha_cluster_get, scha_cluster_get_zone, scha_cluster_close — クラスタに関する情報へのアクセスおよび取得

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l scha#include <scha.h>scha_err_t scha_cluster_open(
     scha_cluster_t *handle);

scha_err_t scha_cluster_open_zone(const char *cluster,
     scha_cluster_t *handlep);

scha_err_t scha_cluster_get(scha_cluster_t handle, const char **
     tag, ...);

scha_err_t scha_cluster_get_zone(const char *cluster,
     scha_cluster_t handlep, const char *cluster_tag, ...);

scha_err_t scha_cluster_close(scha_cluster_t handle);
```

`scha_cluster_open()`、`scha_cluster_get ()`、および `scha_cluster_close()` 関数は、クラスタに関する情報を取得するために一緒に使用します。

`scha_cluster_open()` はクラスタアクセスを初期化し、`scha_cluster_get()` が使用するアクセスハンドルを返します。*handle* 引数は、関数が返す値を格納する変数のアドレスです。

`scha_cluster_get()` 関数は、*tag* 引数に指定されるクラスタの情報にアクセスします。*handle* 引数は、`scha_cluster_open()` の事前呼び出しによって返される値です。*tag* 引数は、`scha_tags.h` ヘッダーファイルのマクロで定義される文字列値です。タグに続く引数は、*tag* 引数の値によって異なります。

情報の取得元となるクラスタノードを指定するために、*tag* 引数のあとに追加の引数を指定する必要がある場合もあります。引数リストの最後の引数は、*tag* 引数で指定される情報の格納に適した変数型にする必要があります。これは出力引数で、クラスタの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。`scha_cluster_get()` 関数が返す情報を格納するために割り当てられたメモリーは、`scha_cluster_get()` 関数に使用したハンドルで `scha_cluster_close()` を呼び出すまで、そのまま残ります。

`scha_cluster_close()` は、以前の `scha_cluster_get()` 関数の呼び出しから返された `handle` 引数を取ります。この関数は、このハンドルを無効にして、このハンドルで行われた

scha_cluster_get() 呼び出しが返した値に割り当てられたメモリーを解放します。値を返す必要が生じるごとに、個々の get 呼び出しでメモリーが割り当てられる点に注意してください。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されることはありません。

scha_cluster_open_zone() 関数と scha_cluster_get_zone() 関数はそれぞれ、scha_cluster_open() および scha_cluster_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_cluster_open_zone() または scha_cluster_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_cluster_open() または scha_cluster_get() と等しくなります)。

scha_cluster_open_zone() から返されたハンドルを閉じるには、scha_cluster_close() を使用します。cluster 引数は不要です。

tag 引数に使用できるマクロ

次に、scha_tags.h に定義されており、*tag* 引数に使用できるマクロを示します。ここでは出力引数および追加引数の型を説明します。構造体と enum 型は、1047 ページのscha_calls(3HA) で説明されています。

SCHA_ALL_LOADLIMITS
: 出力引数の型は scha_str_array_t** です。
: このマクロは、クラスタに定義されているすべての loadlimit 名を返します。

SCHA_ALL_NODEIDS
: 出力引数の型は scha_uint_array_t** です。
: 返される値は、クラスタ内のすべてのノードの数値識別子です。

SCHA_ALL_NODENAMES
: 出力引数の型は scha_str_array_t** です。
: 返される値は、クラスタ内のすべてのノードの名前です。

SCHA_ALL_PRIVATELINK_HOSTNAMES
: 出力引数の型は scha_str_array_t** です。

このマクロは、クラスタインターコネクト上にノードのアドレスを指定するための、すべてのクラスタノードのホスト名を返します。

SCHA_ALL_PSTRINGS

出力引数の型は scha_str_array_t** です。

このマクロは、クラスタに定義されたすべてのプライベート文字列の名前を返します。プライベート文字列の詳細は、217 ページのclpstring(1CL) のマニュアルページを参照してください。

SCHA_ALL_RESOURCEGROUPS

出力引数の型は scha_str_array_t** です。

返される値は、クラスタ上で管理されているすべてのリソースグループの名前です。

SCHA_ALL_RESOURCETYPES

出力引数の型は scha_str_array_t** です。

返される値は、クラスタに登録されているすべてのリソースタイプの名前です。

SCHA_CLUSTERNAME

出力引数の型は char** です。

返される値は、クラスタの名前です。

SCHA_HARD_LOADLIMIT

出力引数の型は scha_str_array_t** です。

このマクロは、指定された範囲名に対し、クラスタのすべてのノードの強い負荷制限値を返します。負荷制限の名前文字列である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d" で、左側の文字列は *nodename*、右側の整数はそのノードで指定されている制限名に対する強い負荷制限値です。強い制限値を指定しない場合、強い制限値の値として -1 が表示されます。

SCHA_LOADLIMIT_PROPS

出力引数の型は scha_str_array_t** です。

このマクロは、指定された範囲名に対し、クラスタのすべてのノードの弱い負荷制限値 (/ 区切り) を返します。負荷制限の名前文字列である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d/%d" で、左側の文字列は *nodename*、最初の整数は弱い制限値、2 番目の整数は強い制限値です。強い制限値を指定しない場合、強い制限値の値として -1 が表示されます。弱い制限値を指定しない場合、弱い制限値の値として 0 が表示されます。

SCHA_LOADLIMITS_NODE

出力引数の型は scha_str_array_t** です。

このマクロは、特定のノードの負荷制限値 (/ 区切り) と制限名を返します。*nodename* である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d/%d" で、文字列は指定したノードに定義されている制限名、最初の整数は弱い制限値、2 番目の整数は強い制限値です。強い制限値を指定しない場合、強い制限値の値として -1 が表示されます。弱い制限値を指定しない場合、弱い制限値の値として 0 が表示されます。

SCHA_NODEID_LOCAL

出力引数の型は uint_t* です。

返される値は、コマンドを実行するノードの数値識別子です。

SCHA_NODEID_NODENAME

出力引数の型は uint_t* です。追加引数の型は char * です。このマクロに必要な追加引数には、クラスタノードの名前を指定します。

返される値は、指定した名前に該当するノードの数値識別子です。

SCHA_NODENAME_LOCAL

出力引数の型は char** です。

返される値は、関数が実行されたクラスタノードの名前です。

SCHA_NODENAME_NODEID

出力引数の型は char** です。追加引数の型は uint_t です。追加引数には、数値式のクラスタノード識別子を指定します。

返される値は、数値識別子で指定されるクラスタノードの名前です。

SCHA_NODESTATE_LOCAL

出力引数の型は scha_node_state_t* です。

返される値は、コマンドが実行されたノードの状態に応じて SCHA_NODE_UP または SCHA_NODE_DOWN となります。

SCHA_NODESTATE_NODE

出力引数の型は scha_node_state_t* です。追加引数の型は char* です。このマクロに必要なフラグ無し追加引数には、クラスタノードの名を指定します。

返される値は、名前付きノードの状態に応じて SCHA_NODE_UP または SCHA_NODE_DOWN となります。

SCHA_PRIVATELINK_HOSTNAME_LOCAL

出力引数の型は char** です。

このマクロは、クラスタインターコネクト上にコマンドを実行するノードのアドレスを指定するためのホスト名を返します。

SCHA_PRIVATELINK_HOSTNAME_NODE

出力引数の型は char** です。追加引数の型は char * です。このマクロに必要なフラグ無し追加引数には、クラスタノードの名前が入ります。

このマクロは、クラスタインターコネクト上に指定のノードのアドレスを指定するためのホスト名を返します。

SCHA_PSTRING

出力引数の型は char** です。

このマクロは、プライベート文字列の平文値を返します。プライベート文字列の名前を指定する、char * 型の追加の引数が必要です。スーパーユーザー以外のユーザーがこのクエリータグを使用するには、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。プライベート文字列の詳細は、217 ページのclpstring(1CL) のマニュアルページを参照してください。

SCHA_RESOURCE_SECURITY

出力引数の型は char** です。

このマクロは、resource_security クラスタプロパティーの現在の設定を返します。

SCHA_SOFT_LOADLIMIT

出力引数の型は scha_str_array_t** です。

このマクロは、指定された範囲名に対し、クラスタのすべてのノードの弱い負荷制限値を返します。負荷制限の名前文字列である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d" で、左側の文字列は *nodename*、右側の整数はそのノードで指定されている制限名に対する弱い負荷制限値です。弱い制限値を指定しない場合、弱い制限値の値として 0 が表示されます。

SCHA_SYSLOG_FACILITY

出力引数の型は int* です。

このマクロは、RGM がログメッセージで使用する Unresolved link to " syslog3C" 関数の番号を返します。返される値は 24 です。これは LOG_DAEMON 機能の値に対応しています。

SCHA_ERR_NOERR　　関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

例 375　scha_cluster_get() 関数の使用

次の例では、scha_cluster_get() 関数を使用して、全クラスタノードの名前を取得します。この関数を使用すると、ノードが稼働しているかどうかもわかります。

このコード例では、制限が構成されている各クラスタノードに対し、**mylimit** という limitname に構成されている弱い負荷制限値と強い負荷制限値も出力されます。各ノードの負荷制限値の出力形式は *nodename=softlimit*/[*hardlimit*] で、強い制限値が設定されていない場合、*hardlimit* の値は無制限 (-1) です。

```
#include <scha.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
        scha_err_t              err;
        scha_node_state_t       node_state;
        scha_str_array_t        *all_nodenames;
        scha_cluster_t          handle;
        int                     ix;
        const char              *str;
        scha_str_array_t        *load_limits;

        err = scha_cluster_open(&handle);
        if (err != SCHA_ERR_NOERR) {
                fprintf(stderr, "FAILED: scha_cluster_open()0);
                exit(err);
        }

        err = scha_cluster_get(handle, SCHA_ALL_NODENAMES, &all_nodenames);
        if (err != SCHA_ERR_NOERR) {
                fprintf(stderr, "FAILED: scha_cluster_get()0);
                exit(err);
        }

        for (ix = 0; ix < all_nodenames->array_cnt; ix++) {
                err = scha_cluster_get(handle, SCHA_NODESTATE_NODE,
                    all_nodenames->str_array[ix], &node_state);
                if (err != SCHA_ERR_NOERR) {
                        fprintf(stderr, "FAILED: scha_cluster_get()"
                            "SCHA_NODESTATE_NODE0);
                        exit(err);
                }

                switch (node_state) {
                case SCHA_NODE_UP:
                        str = "UP";
                        break;
                case SCHA_NODE_DOWN:
                        str = "DOWN";
                        break;
                }

                printf("State of node: %s value: %s\n",
                    all_nodenames->str_array[ix], str);
        }
        err = scha_cluster_get(handle, SCHA_LOADLIMIT_PROPS, "mylimit",
```

```
            &load_limits);

        printf("\n\nLoad limits settings for limitname 'mylimit':\n\n");

        for (ix = 0; ix < load_limits->array_cnt; ix++) {
            printf("%s\n", load_limits->str_array[ix]);
        }
}
```

/usr/cluster/include/scha.h	インクルードファイル
/usr/cluster/lib/libscha.so	ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

669 ページのscha_cluster_get(1HA), 1047 ページのscha_calls(3HA),
1079 ページのscha_cluster_getlogfacility(3HA),
1081 ページのscha_cluster_getnodename(3HA), 1239 ページのscha_strerror(3HA),
Unresolved link to " syslog3C", Unresolved link to " attributes5",
1319 ページのrg_properties(5)

名前

scha_cluster_open, scha_cluster_open_zone, scha_cluster_get, scha_cluster_get_zone, scha_cluster_close — クラスタに関する情報へのアクセスおよび取得

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l scha#include <scha.h>scha_err_t scha_cluster_open(
     scha_cluster_t *handle);

scha_err_t scha_cluster_open_zone(const char *cluster,
     scha_cluster_t *handlep);

scha_err_t scha_cluster_get(scha_cluster_t handle, const char **
     tag, ...);

scha_err_t scha_cluster_get_zone(const char *cluster,
     scha_cluster_t handlep, const char *cluster_tag, ...);

scha_err_t scha_cluster_close(scha_cluster_t handle);
```

`scha_cluster_open()`、`scha_cluster_get ()`、および `scha_cluster_close()` 関数は、クラスタに関する情報を取得するために一緒に使用します。

`scha_cluster_open()` はクラスタアクセスを初期化し、`scha_cluster_get()` が使用するアクセスハンドルを返します。*handle* 引数は、関数が返す値を格納する変数のアドレスです。

`scha_cluster_get()` 関数は、*tag* 引数に指定されるクラスタの情報にアクセスします。*handle* 引数は、`scha_cluster_open()` の事前呼び出しによって返される値です。*tag* 引数は、`scha_tags.h` ヘッダーファイルのマクロで定義される文字列値です。タグに続く引数は、*tag* 引数の値によって異なります。

情報の取得元となるクラスタノードを指定するために、*tag* 引数のあとに追加の引数を指定する必要がある場合もあります。引数リストの最後の引数は、*tag* 引数で指定される情報の格納に適した変数型にする必要があります。これは出力引数で、クラスタの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。`scha_cluster_get()` 関数が返す情報を格納するために割り当てられたメモリーは、`scha_cluster_get()` 関数に使用したハンドルで `scha_cluster_close()` を呼び出すまで、そのまま残ります。

`scha_cluster_close()` は、以前の `scha_cluster_get()` 関数の呼び出しから返された `handle` 引数を取ります。この関数は、このハンドルを無効にして、このハンドルで行われた

scha_cluster_get() 呼び出しが返した値に割り当てられたメモリーを解放します。値を返す必要が生じるごとに、個々の get 呼び出しでメモリーが割り当てられる点に注意してください。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されることはありません。

scha_cluster_open_zone() 関数と scha_cluster_get_zone() 関数はそれぞれ、scha_cluster_open() および scha_cluster_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_cluster_open_zone() または scha_cluster_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_cluster_open() または scha_cluster_get() と等しくなります)。

scha_cluster_open_zone() から返されたハンドルを閉じるには、scha_cluster_close() を使用します。cluster 引数は不要です。

tag 引数に使用できるマクロ

次に、scha_tags.h に定義されており、*tag* 引数に使用できるマクロを示します。ここでは出力引数および追加引数の型を説明します。構造体と enum 型は、1047 ページのscha_calls(3HA) で説明されています。

SCHA_ALL_LOADLIMITS
: 出力引数の型は scha_str_array_t** です。
: このマクロは、クラスタに定義されているすべての loadlimit 名を返します。

SCHA_ALL_NODEIDS
: 出力引数の型は scha_uint_array_t** です。
: 返される値は、クラスタ内のすべてのノードの数値識別子です。

SCHA_ALL_NODENAMES
: 出力引数の型は scha_str_array_t** です。
: 返される値は、クラスタ内のすべてのノードの名前です。

SCHA_ALL_PRIVATELINK_HOSTNAMES
: 出力引数の型は scha_str_array_t** です。

このマクロは、クラスタインターコネクト上にノードのアドレスを指定するための、すべてのクラスタノードのホスト名を返します。

SCHA_ALL_PSTRINGS

出力引数の型は scha_str_array_t** です。

このマクロは、クラスタに定義されたすべてのプライベート文字列の名前を返します。プライベート文字列の詳細は、217 ページのclpstring(1CL) のマニュアルページを参照してください。

SCHA_ALL_RESOURCEGROUPS

出力引数の型は scha_str_array_t** です。

返される値は、クラスタ上で管理されているすべてのリソースグループの名前です。

SCHA_ALL_RESOURCETYPES

出力引数の型は scha_str_array_t** です。

返される値は、クラスタに登録されているすべてのリソースタイプの名前です。

SCHA_CLUSTERNAME

出力引数の型は char** です。

返される値は、クラスタの名前です。

SCHA_HARD_LOADLIMIT

出力引数の型は scha_str_array_t** です。

このマクロは、指定された範囲名に対し、クラスタのすべてのノードの強い負荷制限値を返します。負荷制限の名前文字列である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d" で、左側の文字列は *nodename*、右側の整数はそのノードで指定されている制限名に対する強い負荷制限値です。強い制限値を指定しない場合、強い制限値の値として -1 が表示されます。

SCHA_LOADLIMIT_PROPS

出力引数の型は scha_str_array_t** です。

このマクロは、指定された範囲名に対し、クラスタのすべてのノードの弱い負荷制限値 (/ 区切り) を返します。負荷制限の名前文字列である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d/%d" で、左側の文字列は *nodename*、最初の整数は弱い制限値、2 番目の整数は強い制限値です。強い制限値を指定しない場合、強い制限値の値として -1 が表示されます。弱い制限値を指定しない場合、弱い制限値の値として 0 が表示されます。

SCHA_LOADLIMITS_NODE

出力引数の型は scha_str_array_t** です。

このマクロは、特定のノードの負荷制限値 (/ 区切り) と制限名を返します。*nodename* である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d/%d" で、文字列は指定したノードに定義されている制限名、最初の整数は弱い制限値、2 番目の整数は強い制限値です。強い制限値を指定しない場合、強い制限値の値として -1 が表示されます。弱い制限値を指定しない場合、弱い制限値の値として 0 が表示されます。

SCHA_NODEID_LOCAL

出力引数の型は uint_t* です。

返される値は、コマンドを実行するノードの数値識別子です。

SCHA_NODEID_NODENAME

出力引数の型は uint_t* です。追加引数の型は char * です。このマクロに必要な追加引数には、クラスタノードの名前を指定します。

返される値は、指定した名前に該当するノードの数値識別子です。

SCHA_NODENAME_LOCAL

出力引数の型は char** です。

返される値は、関数が実行されたクラスタノードの名前です。

SCHA_NODENAME_NODEID

出力引数の型は char** です。追加引数の型は uint_t です。追加引数には、数値式のクラスタノード識別子を指定します。

返される値は、数値識別子で指定されるクラスタノードの名前です。

SCHA_NODESTATE_LOCAL

出力引数の型は scha_node_state_t* です。

返される値は、コマンドが実行されたノードの状態に応じて SCHA_NODE_UP または SCHA_NODE_DOWN となります。

SCHA_NODESTATE_NODE

出力引数の型は scha_node_state_t* です。追加引数の型は char* です。このマクロに必要なフラグ無し追加引数には、クラスタノードの名を指定します。

返される値は、名前付きノードの状態に応じて SCHA_NODE_UP または SCHA_NODE_DOWN となります。

SCHA_PRIVATELINK_HOSTNAME_LOCAL

出力引数の型は char** です。

このマクロは、クラスタインターコネクト上にコマンドを実行するノードのアドレスを指定するためのホスト名を返します。

SCHA_PRIVATELINK_HOSTNAME_NODE

出力引数の型は char** です。追加引数の型は char * です。このマクロに必要なフラグ無し追加引数には、クラスタノードの名前が入ります。

このマクロは、クラスタインターコネクト上に指定のノードのアドレスを指定するためのホスト名を返します。

SCHA_PSTRING

出力引数の型は char** です。

このマクロは、プライベート文字列の平文値を返します。プライベート文字列の名前を指定する、char * 型の追加の引数が必要です。スーパーユーザー以外のユーザーがこのクエリータグを使用するには、solaris.cluster.modify 役割に基づくアクセス制御 (RBAC) の承認が必要です。プライベート文字列の詳細は、217 ページのclpstring(1CL) のマニュアルページを参照してください。

SCHA_RESOURCE_SECURITY

出力引数の型は char** です。

このマクロは、resource_security クラスタプロパティーの現在の設定を返します。

SCHA_SOFT_LOADLIMIT

出力引数の型は scha_str_array_t** です。

このマクロは、指定された範囲名に対し、クラスタのすべてのノードの弱い負荷制限値を返します。負荷制限の名前文字列である、char * 型の追加の引数が必要です。

文字列配列の出力の各要素の書式は "%s=%d" で、左側の文字列は *nodename*、右側の整数はそのノードで指定されている制限名に対する弱い負荷制限値です。弱い制限値を指定しない場合、弱い制限値の値として 0 が表示されます。

SCHA_SYSLOG_FACILITY

出力引数の型は int* です。

このマクロは、RGM がログメッセージで使用する Unresolved link to " syslog3C" 関数の番号を返します。返される値は 24 です。これは LOG_DAEMON 機能の値に対応しています。

SCHA_ERR_NOERR　　関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

例 376　scha_cluster_get() 関数の使用

次の例では、scha_cluster_get() 関数を使用して、全クラスタノードの名前を取得します。この関数を使用すると、ノードが稼働しているかどうかもわかります。

このコード例では、制限が構成されている各クラスタノードに対し、**mylimit** という limitname に構成されている弱い負荷制限値と強い負荷制限値も出力されます。各ノードの負荷制限値の出力形式は *nodename=softlimit*/[*hardlimit*] で、強い制限値が設定されていない場合、*hardlimit* の値は無制限 (-1) です。

```
#include <scha.h>
#include <stdio.h>
#include <stdlib.h>

main()
{
        scha_err_t              err;
        scha_node_state_t       node_state;
        scha_str_array_t        *all_nodenames;
        scha_cluster_t          handle;
        int                     ix;
        const char              *str;
        scha_str_array_t        *load_limits;

        err = scha_cluster_open(&handle);
        if (err != SCHA_ERR_NOERR) {
                fprintf(stderr, "FAILED: scha_cluster_open()0);
                exit(err);
        }

        err = scha_cluster_get(handle, SCHA_ALL_NODENAMES, &all_nodenames);
        if (err != SCHA_ERR_NOERR) {
                fprintf(stderr, "FAILED: scha_cluster_get()0);
                exit(err);
        }

        for (ix = 0; ix < all_nodenames->array_cnt; ix++) {
                err = scha_cluster_get(handle, SCHA_NODESTATE_NODE,
                    all_nodenames->str_array[ix], &node_state);
                if (err != SCHA_ERR_NOERR) {
                        fprintf(stderr, "FAILED: scha_cluster_get()"
                            "SCHA_NODESTATE_NODE0);
                        exit(err);
                }

                switch (node_state) {
                case SCHA_NODE_UP:
                        str = "UP";
                        break;
                case SCHA_NODE_DOWN:
                        str = "DOWN";
                        break;
                }

                printf("State of node: %s value: %s\n",
                    all_nodenames->str_array[ix], str);
        }
        err = scha_cluster_get(handle, SCHA_LOADLIMIT_PROPS, "mylimit",
```

```
            &load_limits);

        printf("\n\nLoad limits settings for limitname 'mylimit':\n\n");

        for (ix = 0; ix < load_limits->array_cnt; ix++) {
            printf("%s\n", load_limits->str_array[ix]);
        }
}
```

/usr/cluster/include/scha.h	インクルードファイル
/usr/cluster/lib/libscha.so	ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

669 ページのscha_cluster_get(1HA), 1047 ページのscha_calls(3HA),
1079 ページのscha_cluster_getlogfacility(3HA),
1081 ページのscha_cluster_getnodename(3HA), 1239 ページのscha_strerror(3HA),
Unresolved link to " syslog3C", Unresolved link to " attributes5",
1319 ページのrg_properties(5)

名前

scha_control, scha_control_zone — リソースおよびリソースグループ制御をリクエストする関数

```
cc [flags…] -I/usr/cluster/include file -L/usr/cluster/lib
     -l scha#include <scha.h> scha_err_t scha_control(const char *
     tag, const char *rgname, const char *rname);

 scha_err_t scha_control_zone(const char *tag, const char *
     rgname, const char *rname, const char *zonename);
```

`scha_control()` および `scha_control_zone()` 関数は、それぞれ、Resource Group Manager (RGM) の制御下にあるリソースまたはリソースグループの再起動または再配置をリクエストするためのインタフェースを提供します。これらの関数はリソースモニターで使用します。

`scha_control_zone()` 関数は、`Global_zone` プロパティーが `TRUE` に設定されているリソースタイプにのみ使用します。`Global_zone` プロパティーが `FALSE` に設定されている場合、この機能は必要ありません。詳細は、1335 ページの`rt_properties(5)` のマニュアルページを参照してください。`scha_control_zone()` 関数は、大域ゾーンから呼び出されます。`zonename` 引数は、リソースグループが構成されているゾーンクラスタの名前を指定します。

指定されたリソースの `Failover_mode` プロパティーの設定により、リクエストされた `scha_control()` または `scha_control_zone()` のアクションが抑制される場合もあります。`Failover_mode` が `RESTART_ONLY` である場合、`SCHA_RESOURCE_RESTART` のみが許可されます。`SCHA_GIVEOVER`、`SCHA_CHECK_GIVEOVER`、`SCHA_RESTART`、および `SCHA_CHECK_RESTART` を含む、その他のリクエストは `SCHA_ERR_CHECKS` 終了コードを返し、リクエストされたギブオーバーまたは再起動アクションは実行されず、`syslog` メッセージのみを作成します。`Retry_count` プロパティーと `Retry_interval` プロパティーがリソースで設定されている場合、リソース再起動の回数は `Retry_interval` 内での `Retry_count` の試行回数に制限されます。`Failover_mode` が `LOG_ONLY` の場合、`scha_control()` または `scha_control_zone()` のギブオーバー、再起動、または無効化リクエストは `SCHA_ERR_CHECKS` 終了コードを返し、リクエストされたギブオーバーまたは再起動アクションは実行されず、`syslog` メッセージのみを作成します。

tag 引数

tag 引数には、リソースまたはリソースグループに対して、再配置を行うのか、再起動を行うのかを指定します。この引数は、`scha_tags.h` で定義された次のマクロのうちの 1 つで定義される文字列値にしてください。

SCHA_CHANGE_STATE_OFFLINE

rname 引数で指定したプロキシリソースをローカルノード上でオフラインにするようリクエストします。*プロキシリソース*は、リソースの状態を Oracle Clusterware などの別のクラスタからインポートする Oracle Solaris Cluster リソースです。Oracle Clusterware は、クラスタ環境のための、プラットフォームに依存しない一連のシステムサービスです。Oracle Solaris Cluster ソフトウェアのコンテキスト内での、状態のこの変化には、外部リソースの状態の変化が反映されます。

プロキシリソースの状態をこの *tag* 引数で変更すると、該当するプロキシリソースのメソッドは実行されません。

ノード上にある「依存先の」リソースで障害が発生して、リソースを回復できない場合、モニターはそのノード上のそのリソースをオフラインにします。モニターは scha_control() <! [%zones-deferred; [または scha_control_zone()]]>関数を SCHA_RESOURCE_DISABLE 要求付きで呼び出すことにより、リソースをオフラインにします。また、モニターは、依存先リソースのオフライン、再起動に従属する対象をすべて、それらに対して再起動をトリガーすることにより、オフラインにします。クラスタ管理者が障害を解決し、依存されているリソースを再び有効にすると、モニターは、依存されているリソースのオフライン再起動依存リソースもオンラインに戻します。

SCHA_CHANGE_STATE_ONLINE

rname 引数で指定したプロキシリソースをローカルノード上でオンラインにするようリクエストします。*プロキシリソース*は、リソースの状態を Oracle Clusterware などの別のクラスタからインポートする Oracle Solaris Cluster リソースです。Oracle Solaris Cluster ソフトウェアのコンテキスト内での、状態のこの変化には、外部リソースの状態の変化が反映されます。

プロキシリソースの状態をこの *tag* 引数で変更すると、該当するプロキシリソースのメソッドは実行されません。

SCHA_CHECK_GIVEOVER

rgname 引数に指定したリソースグループに対して *SCHA_GIVEOVER* と同様の検証チェックを実行しますが、実際にリソースグループを再配置することはありません。

SCHA_CHECK_RESTART

rgname 引数に指定したリソースグループに対して *SCHA_RESTART* リクエストと同様の検証チェックを実行しますが、実際にリソースグループを再起動することはありません。

SCHA_CHECK_GIVEOVER および SCHA_CHECK_RESTART リクエストは、scha_control() または scha_control_zone() 関数を起動してギブオーバーまたは再起動を実行するのではなく、リソースに対してアクション (プロセスの終了や再起動など) を直接起こすリソースモニターが使用するためのものです。なお検証に問題が発生した場合、モニターはフェイルオーバー処理を行う代わりに、スリープ状態に置いてからプローブを再開する必要があります。エラーを参照してください。

rgname 引数には、再起動または再配置するリソースグループの名前を指定します。リクエスト元ノードでグループがオンラインになっていない場合、リクエストは拒否されます。

rname 引数には、リソースグループ内のリソースの名前を指定します。これは、モニターが scha_control() または scha_control_zone() のリクエストを実行するリソースと考えられます。指定されたリソースがリソースグループ内に存在しない場合、要求は拒否されます。

コマンドの終了コードは、要求された処理が拒絶されたかどうかを示します。要求が受け入れられた場合、リソースグループまたはリソースに対するオフラインから再オンライン化までの処理が終わった時点で、関数の値が返されます。scha_control() または scha_control_zone() 関数を呼び出した障害モニターは、リソースグループがオフラインになった結果、停止し、成功したリクエストの戻りステータスを受け取らない場合があります。

SCHA_GIVEOVER

rgname 引数に指定したリソースグループをローカルノード上でオフラインにし、RGM が選択する別のノード上で再度オンラインにするようリクエストします。リソースグループが 2 つ以上のノード上で現在オンライン状態にあり、これ以上オンライン化できるノードが存在しない場合、このリソースグループは、ローカルノード上でオフラインになったままオンラインにならない可能性があります。またこうした処理要求は、各種のチェックの結果を受けた結果として、拒絶される場合もあります。たとえば、あるノード上で SCHA_GIVEOVER リクエストにより、グループが Pingpong_interval プロパティーの指定された間隔でオフラインになったために、このノードがホストとして拒絶されることがあります。

クラスタ管理者が 1 つ以上のリソースグループの RG_affinities プロパティーを構成し、1 つのリソースグループで **scha_control GIVEOVER** 要求を発行する場合、複数のリソースグループが再配置される可能性があります。RG_affinities プロパティーについては、1319 ページのrg_properties(5) で説明されています。

MONITOR_CHECK メソッドは、リソースを含むリソースグループが、scha_control() または scha_control_zone() 関数の呼び出しあるいは障害モニターからの scha_control または scha_control_zone() コマンドの発行の結果として新しいノードに再配置される前に呼び出されます。683 ページのscha_control(1HA) のマニュアルページを参照してください。

MONITOR_CHECK メソッドは、リソースグループの新しい潜在マスターである任意のノードから呼び出せます。MONITOR_CHECK メソッドは、リソースを実行するのにノードが十分に健全であるかどうかを確認します。MONITOR_CHECK メソッドは、並行して実行されるほかのメソッドと衝突しない方法で実装する必要があります。

MONITOR_CHECK メソッドが失敗した場合、コールバックが呼び出されたノードへのリソースグループの再配置は拒否されます。

SCHA_IGNORE_FAILED_START

現在実行している Prenet_start メソッドまたは Start メソッドが失敗したリクエストは、Failover_mode プロパティーの設定をしていても、リソースグループのフェイルオーバーの原因にはなりません。

言い換えると、この要求は、Failover_Mode プロパティーが SOFT または HARD に設定されているリソースに対して、通常、そのリソースが起動に失敗したときに行われる回復アクションを無効にします。通常、リソースグループは異なるノードにフェイルオーバーします。代わりに、リソースは Failover_Mode が NONE に設定されているかのように動作します。このリソー

スは START_FAILED 状態になり、その他のエラーが発生していない場合、リソースグループは ONLINE_FAULTED 状態で終了します。

このリクエストが意味を持つのは、あとで 0 以外のステータスで終了するかタイムアウトする Start または Prenet_start メソッドから呼び出された場合のみです。このリクエストは、現在の Start メソッドまたは Prenet_start メソッドの呼び出しに対してのみ有効です。別のノード上でリソースを正常に起動できないことが Start() メソッドで判明した場合は、このリクエストを指定してscha_control() または scha_control_zone 関数を呼び出してください。このリクエストがほかのメソッドから呼び出された場合、エラー SCHA_ERR_INVAL が返されます。この要求は、リソースグループの「ピンポン」フェイルオーバーを防ぐために存在します。SCHA_ERR_INVAL エラーコードについては、1047 ページのscha_calls(3HA) のマニュアルページを参照してください。

SCHA_RESOURCE_DISABLE

scha_control() または scha_control_zone() 関数が呼び出されたノード上で rname 引数によって指定されたリソースを無効にします。

ノード上にある「依存先の」リソースで障害が発生して、リソースを回復できない場合、モニターはそのノード上のそのリソースをオフラインにします。モニターは scha_control() または scha_control_zone() 関数を SCHA_RESOURCE_DISABLE リクエストで呼び出すことにより、リソースをオフラインにします。また、モニターは、依存先リソースのオフライン、再起動に従属する対象をすべて、それらに対して再起動をトリガーすることにより、オフラインにします。クラスタ管理者が障害を解決し、依存されているリソースを再び有効にすると、モニターは、依存されているリソースのオフライン再起動依存リソースもオンラインに戻します。

SCHA_RESOURCE_IS_RESTARTED

rname 引数に指定したリソースの再起動カウンタを、実際にリソースを再起動することなく、ローカルノード上で増分するようリクエストします。

SCHA_RESOURCE_RESTART リクエストで scha_control() または scha_control_zone() 関数を呼び出すことなく、リソースを直接再起動するリソースモニター (たとえば、735 ページのpmfadm(1M) コマンドの使用) は、このリクエストを使用してリソースが再起動されたことを RGM に通知できます。このことは、今後 scha_resource_get() 関数を NUM_RESOURCE_RESTARTS クエリー付きで呼び出す場合に反映されます。

リソースの型が Retry_interval 標準プロパティーの宣言に失敗した場合、scha_control() または scha_control_zone() 関数の SCHA_RESOURCE_IS_RESTARTED リクエストは許可されず、scha_control() または scha_control_zone() 関数はエラーコード 13 (SCHA_ERR_RT) を返します。

SCHA_RESOURCE_RESTART

rname 引数に指定したリソースをオフラインにし、リソースグループ内のほかのリソースを停止させることなく、ローカルノード上で再度オンラインにするようリクエストします。リソースの停止と起動は、ローカルノード上で次の順序でメソッドを適用することで行われます:

MONITOR_STOP
STOP
START

MONITOR_START

リソースタイプが STOP メソッドおよび START メソッドを宣言していない場合は、代わりに POSTNET_STOP および PRENET_START を使用してリソースが再起動されます:

```
MONITOR_STOP
POSTNET_STOP
PRENET_START
MONITOR_START
```

リソースタイプが MONITOR_STOP メソッドおよび MONITOR_START メソッドを宣言していない場合、STOP メソッドと START メソッド、または POSTNET_STOP メソッドと PRENET_START メソッドのみが呼び出され、再起動が実行されます。リソースのタイプは START メソッドと STOP メソッドを宣言する必要があります。SCHA_ERR_RT エラーコードについては、1047 ページのscha_calls(3HA) のマニュアルページを参照してください。

リソースの再起動時にメソッドの呼び出しに失敗した場合は、RGM はリソースの Failover_mode プロパティー設定に応じて、エラー状態の設定、リソースグループの再配置、またはノードのリブートを実行します。詳細は、1287 ページのr_properties(5) の Failover_mode プロパティーを参照してください。

この要求を使用してリソースを再起動するリソースモニターは、scha_resource_get の NUM_RESOURCE_RESTARTS() クエリーを使用して、最近の再起動の試行回数を記録できます。

PRENET_START または POSTNET_STOP メソッドを持つリソースタイプでは、注意して SCHA_RESOURCE_RESTART リクエストを使用する必要があります。リソースに対して使用できるのは MONITOR_STOP、STOP、START、MONITOR_START のメソッドのみです。このリソースが依存するネットワークアドレスリソースは再起動されず、オンライン状態のままになります。

ノード上にある「依存先の」リソースで障害が発生して、リソースを回復できない場合、モニターはそのノード上のそのリソースをオフラインにします。モニターは scha_control() <![%zones-deferred; [または scha_control_zone()]]>関数を SCHA_RESOURCE_DISABLE 要求付きで呼び出すことにより、リソースをオフラインにします。また、モニターは、依存先リソースのオフライン、再起動に従属する対象をすべて、それらに対して再起動をトリガーすることにより、オフラインにします。クラスタ管理者が障害を解決し、依存されているリソースを再び有効にすると、モニターは、依存されているリソースのオフライン再起動依存リソースもオンラインに戻します。

SCHA_RESTART

rgname 引数に指定したリソースグループをオフラインにし、異なるノードに強制的に再配置することなく、再度オンラインにするようリクエストします。ただし、この要求を実行する際にグループ内のリソースの再起動が失敗した場合は、最終的にリソースグループが再配置されることもあります。このリクエストを使用してリソースグループを再起動するリソースモニターは、scha_resource_get の NUM_RG_RESTARTS() クエリーを使用して、最近の再起動の試行回数を記録できます。

これらの関数は、次の戻り値を返します。

`0`	関数の実行に成功。
0 以外	関数の実行に失敗。

`SCHA_ERR_NOERR`	関数の実行に成功。
`SCHA_ERR_CHECKS`	要求は取り消し済み再配置に失敗した箇所のチェックが必要。

その他のエラーコードについては、1047 ページの`scha_calls`(3HA) のマニュアルページを参照してください。

通常、`scha_control()` または `scha_control_zone()` 関数からエラーコードを受信する障害モニターは、しばらくの間スリープ状態になってから、その検証を再起動するようにしてください。これらの関数がこのような動作をするのは、エラー状態の中には、しばらく時間が経過したあと自分自身で解決するものがあるためです。そのようなエラー状態の例としては、広域デバイスサービスのフェイルオーバーがあり、これはディスクリソースが一時的に使用不可能になる原因になります。エラー状態が解決されたあと、リソース自体はふたたび正常な状態に戻ることがあります。そうでない場合、後続の `scha_control()` または `scha_control_zone()` リクエストが成功することがあります。

`/usr/cluster/include/scha.h`	インクルードファイル
`/usr/cluster/lib/libscha.so`	ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	発展中

645 ページの`rt_callbacks`(1HA), 683 ページの`scha_control`(1HA), 735 ページの`pmfadm`(1M), 1047 ページの`scha_calls`(3HA), 1151 ページの`scha_resource_open`(3HA), 1239 ページの`scha_strerror`(3HA), Unresolved link to " attributes5", 1287 ページの`r_properties`(5), 1319 ページの`rg_properties`(5), 1335 ページの`rt_properties`(5)

名前

scha_control, scha_control_zone — リソースおよびリソースグループ制御をリクエストする関数

```
cc [flags…] -I/usr/cluster/include file -L/usr/cluster/lib
     -l scha#include <scha.h> scha_err_t scha_control(const char *
     tag, const char *rgname, const char *rname);

 scha_err_t scha_control_zone(const char *tag, const char *
     rgname, const char *rname, const char *zonename);
```

scha_control() および scha_control_zone() 関数は、それぞれ、Resource Group Manager (RGM) の制御下にあるリソースまたはリソースグループの再起動または再配置をリクエストするためのインタフェースを提供します。これらの関数はリソースモニターで使用します。

scha_control_zone() 関数は、Global_zone プロパティーが TRUE に設定されているリソースタイプにのみ使用します。Global_zone プロパティーが FALSE に設定されている場合、この機能は必要ありません。詳細は、1335 ページのrt_properties(5) のマニュアルページを参照してください。scha_control_zone() 関数は、大域ゾーンから呼び出されます。zonename 引数は、リソースグループが構成されているゾーンクラスタの名前を指定します。

指定されたリソースの Failover_mode プロパティーの設定により、リクエストされた scha_control() または scha_control_zone() のアクションが抑制される場合もあります。Failover_mode が RESTART_ONLY である場合、SCHA_RESOURCE_RESTART のみが許可されます。SCHA_GIVEOVER、SCHA_CHECK_GIVEOVER、SCHA_RESTART、および SCHA_CHECK_RESTART を含む、その他のリクエストは SCHA_ERR_CHECKS 終了コードを返し、リクエストされたギブオーバーまたは再起動アクションは実行されず、syslog メッセージのみを作成します。Retry_count プロパティーと Retry_interval プロパティーがリソースで設定されている場合、リソース再起動の回数は Retry_interval 内での Retry_count の試行回数に制限されます。Failover_mode が LOG_ONLY の場合、scha_control() または scha_control_zone() のギブオーバー、再起動、または無効化リクエストは SCHA_ERR_CHECKS 終了コードを返し、リクエストされたギブオーバーまたは再起動アクションは実行されず、syslog メッセージのみを作成します。

tag 引数

tag 引数には、リソースまたはリソースグループに対して、再配置を行うのか、再起動を行うのかを指定します。この引数は、scha_tags.h で定義された次のマクロのうちの 1 つで定義される文字列値にしてください。

SCHA_CHANGE_STATE_OFFLINE

rname 引数で指定したプロキシリソースをローカルノード上でオフラインにするようリクエストします。*プロキシリソース*は、リソースの状態を Oracle Clusterware などの別のクラスタからインポートする Oracle Solaris Cluster リソースです。Oracle Clusterware は、クラスタ環境のための、プラットフォームに依存しない一連のシステムサービスです。Oracle Solaris Cluster ソフトウェアのコンテキスト内での、状態のこの変化には、外部リソースの状態の変化が反映されます。

プロキシリソースの状態をこの *tag* 引数で変更すると、該当するプロキシリソースのメソッドは実行されません。

ノード上にある「依存先の」リソースで障害が発生して、リソースを回復できない場合、モニターはそのノード上のそのリソースをオフラインにします。モニターは scha_control() <! [%zones-deferred; [または scha_control_zone()]]>関数を SCHA_RESOURCE_DISABLE 要求付きで呼び出すことにより、リソースをオフラインにします。また、モニターは、依存先リソースのオフライン、再起動に従属する対象をすべて、それらに対して再起動をトリガーすることにより、オフラインにします。クラスタ管理者が障害を解決し、依存されているリソースを再び有効にすると、モニターは、依存されているリソースのオフライン再起動依存リソースもオンラインに戻します。

SCHA_CHANGE_STATE_ONLINE

rname 引数で指定したプロキシリソースをローカルノード上でオンラインにするようリクエストします。*プロキシリソース*は、リソースの状態を Oracle Clusterware などの別のクラスタからインポートする Oracle Solaris Cluster リソースです。Oracle Solaris Cluster ソフトウェアのコンテキスト内での、状態のこの変化には、外部リソースの状態の変化が反映されます。

プロキシリソースの状態をこの *tag* 引数で変更すると、該当するプロキシリソースのメソッドは実行されません。

SCHA_CHECK_GIVEOVER

rgname 引数に指定したリソースグループに対して *SCHA_GIVEOVER* と同様の検証チェックを実行しますが、実際にリソースグループを再配置することはありません。

SCHA_CHECK_RESTART

rgname 引数に指定したリソースグループに対して *SCHA_RESTART* リクエストと同様の検証チェックを実行しますが、実際にリソースグループを再起動することはありません。

SCHA_CHECK_GIVEOVER および SCHA_CHECK_RESTART リクエストは、scha_control() または scha_control_zone() 関数を起動してギブオーバーまたは再起動を実行するのではなく、リソースに対してアクション (プロセスの終了や再起動など) を直接起こすリソースモニターが使用するためのものです。なお検証に問題が発生した場合、モニターはフェイルオーバー処理を行う代わりに、スリープ状態に置いてからプローブを再開する必要があります。エラーを参照してください。

rgname 引数には、再起動または再配置するリソースグループの名前を指定します。リクエスト元ノードでグループがオンラインになっていない場合、リクエストは拒否されます。

rname 引数には、リソースグループ内のリソースの名前を指定します。これは、モニターが scha_control() または scha_control_zone() のリクエストを実行するリソースと考えられます。指定されたリソースがリソースグループ内に存在しない場合、要求は拒否されます。

コマンドの終了コードは、要求された処理が拒絶されたかどうかを示します。要求が受け入れられた場合、リソースグループまたはリソースに対するオフラインから再オンライン化までの処理が終わった時点で、関数の値が返されます。scha_control() または scha_control_zone() 関数を呼び出した障害モニターは、リソースグループがオフラインになった結果、停止し、成功したリクエストの戻りステータスを受け取らない場合があります。

SCHA_GIVEOVER

rgname 引数に指定したリソースグループをローカルノード上でオフラインにし、RGM が選択する別のノード上で再度オンラインにするようリクエストします。リソースグループが 2 つ以上のノード上で現在オンライン状態にあり、これ以上オンライン化できるノードが存在しない場合、このリソースグループは、ローカルノード上でオフラインになったままオンラインにならない可能性があります。またこうした処理要求は、各種のチェックの結果を受けた結果として、拒絶される場合もあります。たとえば、あるノード上で SCHA_GIVEOVER リクエストにより、グループが Pingpong_interval プロパティーの指定された間隔でオフラインになったために、このノードがホストとして拒絶されることがあります。

クラスタ管理者が 1 つ以上のリソースグループの RG_affinities プロパティーを構成し、1 つのリソースグループで **scha_control GIVEOVER** 要求を発行する場合、複数のリソースグループが再配置される可能性があります。RG_affinities プロパティーについては、1319 ページのrg_properties(5) で説明されています。

MONITOR_CHECK メソッドは、リソースを含むリソースグループが、scha_control() または scha_control_zone() 関数の呼び出しあるいは障害モニターからの scha_control または scha_control_zone() コマンドの発行の結果として新しいノードに再配置される前に呼び出されます。683 ページのscha_control(1HA) のマニュアルページを参照してください。

MONITOR_CHECK メソッドは、リソースグループの新しい潜在マスターである任意のノードから呼び出せます。MONITOR_CHECK メソッドは、リソースを実行するのにノードが十分に健全であるかどうかを確認します。MONITOR_CHECK メソッドは、並行して実行されるほかのメソッドと衝突しない方法で実装する必要があります。

MONITOR_CHECK メソッドが失敗した場合、コールバックが呼び出されたノードへのリソースグループの再配置は拒否されます。

SCHA_IGNORE_FAILED_START

現在実行している Prenet_start メソッドまたは Start メソッドが失敗したリクエストは、Failover_mode プロパティーの設定をしていても、リソースグループのフェイルオーバーの原因にはなりません。

言い換えると、この要求は、Failover_Mode プロパティーが SOFT または HARD に設定されているリソースに対して、通常、そのリソースが起動に失敗したときに行われる回復アクションを無効にします。通常、リソースグループは異なるノードにフェイルオーバーします。代わりに、リソースは Failover_Mode が NONE に設定されているかのように動作します。このリソー

スは START_FAILED 状態になり、その他のエラーが発生していない場合、リソースグループは ONLINE_FAULTED 状態で終了します。

このリクエストが意味を持つのは、あとで 0 以外のステータスで終了するかタイムアウトする Start または Prenet_start メソッドから呼び出された場合のみです。このリクエストは、現在の Start メソッドまたは Prenet_start メソッドの呼び出しに対してのみ有効です。別のノード上でリソースを正常に起動できないことが Start() メソッドで判明した場合は、このリクエストを指定してscha_control() または scha_control_zone 関数を呼び出してください。このリクエストがほかのメソッドから呼び出された場合、エラー SCHA_ERR_INVAL が返されます。この要求は、リソースグループの「ピンポン」フェイルオーバーを防ぐために存在します。SCHA_ERR_INVAL エラーコードについては、1047 ページのscha_calls(3HA) のマニュアルページを参照してください。

SCHA_RESOURCE_DISABLE

scha_control() または scha_control_zone() 関数が呼び出されたノード上で rname 引数によって指定されたリソースを無効にします。

ノード上にある「依存先の」リソースで障害が発生して、リソースを回復できない場合、モニターはそのノード上のそのリソースをオフラインにします。モニターは scha_control() または scha_control_zone() 関数を SCHA_RESOURCE_DISABLE リクエストで呼び出すことにより、リソースをオフラインにします。また、モニターは、依存先リソースのオフライン、再起動に従属する対象をすべて、それらに対して再起動をトリガーすることにより、オフラインにします。クラスタ管理者が障害を解決し、依存されているリソースを再び有効にすると、モニターは、依存されているリソースのオフライン再起動依存リソースもオンラインに戻します。

SCHA_RESOURCE_IS_RESTARTED

rname 引数に指定したリソースの再起動カウンタを、実際にリソースを再起動することなく、ローカルノード上で増分するようリクエストします。

SCHA_RESOURCE_RESTART リクエストで scha_control() または scha_control_zone() 関数を呼び出すことなく、リソースを直接再起動するリソースモニター (たとえば、735 ページのpmfadm(1M) コマンドの使用) は、このリクエストを使用してリソースが再起動されたことを RGM に通知できます。このことは、今後 scha_resource_get() 関数を NUM_RESOURCE_RESTARTS クエリー付きで呼び出す場合に反映されます。

リソースの型が Retry_interval 標準プロパティーの宣言に失敗した場合、scha_control() または scha_control_zone() 関数の SCHA_RESOURCE_IS_RESTARTED リクエストは許可されず、scha_control() または scha_control_zone() 関数はエラーコード 13 (SCHA_ERR_RT) を返します。

SCHA_RESOURCE_RESTART

rname 引数に指定したリソースをオフラインにし、リソースグループ内のほかのリソースを停止させることなく、ローカルノード上で再度オンラインにするようリクエストします。リソースの停止と起動は、ローカルノード上で次の順序でメソッドを適用することで行われます:

MONITOR_STOP
STOP
START

MONITOR_START

リソースタイプが STOP メソッドおよび START メソッドを宣言していない場合は、代わりに POSTNET_STOP および PRENET_START を使用してリソースが再起動されます:

```
MONITOR_STOP
POSTNET_STOP
PRENET_START
MONITOR_START
```

リソースタイプが MONITOR_STOP メソッドおよび MONITOR_START メソッドを宣言していない場合、STOP メソッドと START メソッド、または POSTNET_STOP メソッドと PRENET_START メソッドのみが呼び出され、再起動が実行されます。リソースのタイプは START メソッドと STOP メソッドを宣言する必要があります。SCHA_ERR_RT エラーコードについては、1047 ページのscha_calls(3HA) のマニュアルページを参照してください。

リソースの再起動時にメソッドの呼び出しに失敗した場合は、RGM はリソースの Failover_mode プロパティー設定に応じて、エラー状態の設定、リソースグループの再配置、またはノードのリブートを実行します。詳細は、1287 ページのr_properties(5) の Failover_mode プロパティーを参照してください。

この要求を使用してリソースを再起動するリソースモニターは、scha_resource_get の NUM_RESOURCE_RESTARTS() クエリーを使用して、最近の再起動の試行回数を記録できます。

PRENET_START または POSTNET_STOP メソッドを持つリソースタイプでは、注意して SCHA_RESOURCE_RESTART リクエストを使用する必要があります。リソースに対して使用できるのは MONITOR_STOP、STOP、START、MONITOR_START のメソッドのみです。このリソースが依存するネットワークアドレスリソースは再起動されず、オンライン状態のままになります。

ノード上にある「依存先の」リソースで障害が発生して、リソースを回復できない場合、モニターはそのノード上のそのリソースをオフラインにします。モニターは scha_control() <![%zones-deferred; [または scha_control_zone()]]>関数を SCHA_RESOURCE_DISABLE 要求付きで呼び出すことにより、リソースをオフラインにします。また、モニターは、依存先リソースのオフライン、再起動に従属する対象をすべて、それらに対して再起動をトリガーすることにより、オフラインにします。クラスタ管理者が障害を解決し、依存されているリソースを再び有効にすると、モニターは、依存されているリソースのオフライン再起動依存リソースもオンラインに戻します。

SCHA_RESTART

rgname 引数に指定したリソースグループをオフラインにし、異なるノードに強制的に再配置することなく、再度オンラインにするようリクエストします。ただし、この要求を実行する際にグループ内のリソースの再起動が失敗した場合は、最終的にリソースグループが再配置されることもあります。このリクエストを使用してリソースグループを再起動するリソースモニターは、scha_resource_get の NUM_RG_RESTARTS() クエリーを使用して、最近の再起動の試行回数を記録できます。

これらの関数は、次の戻り値を返します。

`0`	関数の実行に成功。
0 以外	関数の実行に失敗。

`SCHA_ERR_NOERR`	関数の実行に成功。
`SCHA_ERR_CHECKS`	要求は取り消し済み再配置に失敗した箇所のチェックが必要。

その他のエラーコードについては、1047 ページの`scha_calls`(3HA) のマニュアルページを参照してください。

通常、`scha_control()` または `scha_control_zone()` 関数からエラーコードを受信する障害モニターは、しばらくの間スリープ状態になってから、その検証を再起動するようにしてください。これらの関数がこのような動作をするのは、エラー状態の中には、しばらく時間が経過したあと自分自身で解決するものがあるためです。そのようなエラー状態の例としては、広域デバイスサービスのフェイルオーバーがあり、これはディスクリソースが一時的に使用不可能になる原因になります。エラー状態が解決されたあと、リソース自体はふたたび正常な状態に戻ることがあります。そうでない場合、後続の `scha_control()` または `scha_control_zone()` リクエストが成功することがあります。

`/usr/cluster/include/scha.h`	インクルードファイル
`/usr/cluster/lib/libscha.so`	ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	発展中

645 ページの`rt_callbacks`(1HA), 683 ページの`scha_control`(1HA), 735 ページの`pmfadm`(1M), 1047 ページの`scha_calls`(3HA), 1151 ページの`scha_resource_open`(3HA), 1239 ページの`scha_strerror`(3HA), Unresolved link to " attributes5", 1287 ページの`r_properties`(5), 1319 ページの`rg_properties`(5), 1335 ページの`rt_properties`(5)

名前

scha_get_fullname — リソースグループが実行されるクラスタノード名を返す

```
scha_err_tscha_get_fullname(const char *zonename, char **fullname)
```

`scha_get_fullname()` 関数は、ローカルノード上のゾーンコンテキストの論理ノード名を返します。出力パラメータの `fullname` は、クラスタノード名を含む文字列に設定されます。この文字列は、呼び出し元が解放する必要があります。

`zonename` が NULL である場合、ゾーンコンテキストは、この呼び出しが実行されている場所に依存します。

- 大域ゾーンで実行されている場合は、ローカルノード名を返します。
- ゾーンクラスタで実行されている場合は、グローバルクラスタノード名ではなく、ゾーンクラスタノード名を返します。

`zonename` が null 以外である場合、これは `zonename` で指定された非大域ゾーンで構成されているリソースの代わりに、大域ゾーンで呼び出されることが想定されています。`zonename` がゾーンクラスタの名前 (これは、基本となる非大域ゾーンの名前でもあります) である場合は、ゾーンクラスタノード名を返します。

この関数がリソースコールバックメソッドから呼び出され、`zonename` パラメータが `scds_get_zone_name()` 関数の出力に設定されている場合は、いずれの場合も、結果として得られる `fullname` はリソースグループのノードリスト内の現在のエントリに一致します。

- グローバルクラスタまたはゾーンクラスタ
- `global_zone` リソースタイプまたは通常のリソースタイプ

この関数は、次の値を返します。

`0`	関数の実行に成功。
0 以外	関数の実行に失敗。

`SCHA_ERR_NOERR`	関数の実行に成功。

その他のエラーコードについては、1047 ページの`scha_calls(3HA)` を参照してください。

例 377 scha_get_fullname() 関数の使用法

次の例では、scha_get_fullname() を使用して、現在のプログラムが実行されている nodename コンテキストを表す fullname 文字列を取得し、この名前をリソースグループのノードリスト内で検索します。

```
#include <scha.h>
#include <libdsdev.h>

main(int argc, char *argv[])

{
        scha_err_t        err;
        scds_handle_t     handle;
        char              *myzonename;
        char              *fullname;
        const scha_str_array_t  *rgnodelist;
        uint_t            ix;

        if (scds_initialize(&handle) != SCHA_ERR_NOERR) {
                /* handle the error */
                ...
        }
        myzonename = scds_get_zone_name(handle);
        rgnodelist = scds_get_rg_nodelist(handle);
        err = scha_get_fullname(myzonename, &fullname);
        ...
        for (ix = 0; ix < rgnodelist->array_cnt; ix++) {
                if (strcmp(fullname, rgnodelist->str_array[ix]) == 0) {
                        /* found this node in the node list */
                        ...
                }
        }
        ...
}
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

909 ページのscds_calls(3HA), 987 ページのscds_initialize(3HA), 1081 ページのscha_cluster_getnodename(3HA), Unresolved link to " attributes5"

名前

scha_resource_open, scha_resource_open_zone, scha_resource_get, scha_resource_get_zone, scha_resource_close — リソース情報アクセス関数

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l scha#include
 <scha.h>scha_err_t scha_resource_open( const char *rname, const char *rgname, scha_resource_t
 *handle);

scha_err_t scha_resource_open_zone(const char *cluster, const char *rs_name, const char
 *rg_name, scha_resource_t * handlep);

scha_err_t scha_resource_get(scha_resource_t handle, const char *tag,...);

scha_err_t scha_resource_get_zone(const char *cluster,
     scha_resource_t handlep, const char *rs_tag, ...);

scha_err_t scha_resource_close(scha_resource_t handle);
```

scha_resource_open()、scha_resource_get ()、および scha_resource_close() 関数は、Resource Group Manager (RGM) クラスタ機能で管理されるリソースに関する情報にアクセスするために一緒に使用します。

scha_resource_open() は、リソースへのアクセスを初期化し、scha_resource_get() が使用するハンドルを返します。

scha_resource_open の rname() 引数には、アクセスするリソースの名前を指定します。*rgname* 引数には、該当するリソースを割り当てたリソースグループの名前を指定します。グループ名が不明な場合、*rgname* 引数を NULL にすることができます。ただし、グループ名を指定する方が、関数の処理をより効率的に進めることができます。*handle* 引数の値は、関数の戻り値を格納する変数のアドレスとなります。

scha_resource_get() 関数は、*tag* 引数に指定されるリソースの情報にアクセスします。*tag* 引数には、scha_tags.h ヘッダーファイルのマクロで定義される文字列値を指定してください。タグ以降の引数は、*tag* の値に依存します。tag より後に追加する引数は、情報を取り出すクラスタノードや、tag 固有の他の情報を指定する際に必要となることがあります。引数リストの最後の引数は、*tag* で指定される情報の格納に適した変数型にする必要があります。これは出力引数で、リソースの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。

scha_resource_get() で使用されたハンドルで scha_resource_close() が呼び出されるまで、scha_resource_get() の返す情報用に割り当てられたメモリーは維持されます。同じ

handle と tag を指定して scha_resource_get() の呼び出しを繰り返すと、新規にメモリーが割り当てられます。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されたりすることはありません。

scha_resource_close() 関数は、以前の *scha_resource_open* の呼び出しで返された handle() 引数を取ります。これは、ハンドルを使用して得た scha_resource_get() 呼び出しの戻り値に割り当てられたメモリーを解放するとともに、このハンドルを無効化します。

scha_tags.h に定義されたマクロは、*scha_resource_get* の tag() 引数に使用される場合があります。

ここでは出力引数および追加引数の型を説明します。構造体と enum 型は、1047 ページのscha_calls(3HA) で説明されています。

scha_resource_open_zone() 関数および scha_resource_get_zone() 関数はそれぞれ、scha_resource_open() および scha_resource_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_resource_open_zone() または scha_resource_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_resource_open() または scha_resource_get() と等しくなります)。

scha_resource_open_zone() から返されたハンドルを閉じるには、scha_resource_close() を使用します。cluster 引数は不要です。

tag 引数

次に、リソースプロパティーを指定するマクロを示します。リソースプロパティーの値が出力されます。SCHA_RESOURCE_STATE、SCHA_STATUS、SCHA_NUM_RG_RESTARTS、SCHA_NUM_RESOURCE_RESTARTS の各プロパティーは、コマンドが実行されるノード上の値を参照します (1287 ページのr_properties(5) を参照)。

ここでは出力引数および追加引数の型を示します。

拡張プロパティー

これらのプロパティーは、リソースタイプの Resource Type Registration (RTR) ファイル内で宣言します。リソースタイプの実装によって、これらのプロパティーを定義します。

SCHA_AFFINITY_TIMEOUT

出力引数の型は int* です。

SCHA_ALL_EXTENSIONS

出力引数の型は scha_str_array_t** です。リソースのすべての拡張プロパティーの名前を返します。

SCHA_APPLICATION_USER

出力引数の型は char** です。

SCHA_BOOT_TIMEOUT

出力引数の型は int* です。

SCHA_CHEAP_PROBE_INTERVAL

出力引数の型は int* です。

SCHA_EXTENSION

出力引数の型は scha_extprop_value_t* です。拡張プロパティーの名前を提供する、char* 型の追加の引数が必要です。ローカルノードのプロパティーのタイプとその値を返します。

明示的な値が割り当てられていないノード上でこのプロパティーの値をユーザーがリクエストすると、RTR ファイルで宣言されているデフォルト値が返されます。1251 ページのrt_reg(4) のマニュアルページを参照してください。

SCHA_EXTENSION_NODE

出力引数の型は scha_extprop_value_t* です。char* 型の追加の引数が必要です。最初の引数では、拡張プロパティーの名前を指定し、2 番目の引数ではクラスタノードを指定します。指定のノードのプロパティーのタイプとその値を返します。

明示的な値が割り当てられていないノード上でこのプロパティーの値をユーザーがリクエストすると、RTR ファイルで宣言されているデフォルト値が返されます。1251 ページのrt_reg(4) のマニュアルページを参照してください。

SCHA_FAILOVER_MODE

出力引数の型は scha_failover_mode_t* です。

SCHA_FINI_TIMEOUT

出力引数の型は int* です。

SCHA_GLOBAL_ZONE_OVERRIDE

出力引数の型は boolean_t* です。

SCHA_GROUP

出力引数の型は char** です。リソースを構成するリソースグループの名前を返します。

SCHA_INIT_TIMEOUT

出力引数の型は int* です。

SCHA_LOAD_BALANCING_POLICY

出力引数の型は char** です。

SCHA_LOAD_BALANCING_WEIGHTS

出力引数の型は scha_str_array_t** です。

SCHA_MONITOR_CHECK_TIMEOUT

出力引数の型は int* です。

SCHA_MONITOR_START_TIMEOUT

出力引数の型は int* です。

SCHA_MONITOR_STOP_TIMEOUT

出力引数の型は int* です。

SCHA_MONITORED_SWITCH

出力引数の型は scha_switch_t* です。戻り値は、ローカルノードでリソースがモニターされるかどうかを示します。

SCHA_MONITORED_SWITCH_NODE

出力引数の型は scha_switch_t* です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードでリソースがモニターされるかどうかを示します。

SCHA_NETWORK_RESOURCES_USED

出力引数の型は scha_str_array_t** です。

SCHA_NUM_RESOURCE_RESTARTS

出力引数の型は int* です。クエリーが実行されるゾーンのこのリソースで発生した、リソースの再起動のリクエスト数を返します。詳細は、1287 ページのr_properties(5) のマニュアルページを参照してください。

SCHA_NUM_RG_RESTARTS

出力引数の型は int* です。クエリーが実行されるゾーンのこのリソースで発生した、リソースグループの再起動のリクエスト数を返します。詳細は、1287 ページのr_properties(5) のマニュアルページを参照してください。

SCHA_ON_OFF_SWITCH

出力引数の型は scha_switch_t* です。戻り値は、ローカルノードでリソースが有効かどうかを示します。

SCHA_ON_OFF_SWITCH_NODE

出力引数の型は scha_switch_t* です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードでリソースが有効かどうかを示します。

SCHA_PORT_LIST

出力引数の型は scha_str_array_t** です。

SCHA_POSTNET_STOP_TIMEOUT

出力引数の型は int* です。

SCHA_PRENET_START_TIMEOUT

出力引数の型は int* です。

SCHA_PRE_EVICT

出力引数の型は boolean_t* です。

SCHA_R_DESCRIPTION

出力引数の型は char** です。

SCHA_RESOURCE_DEPENDENCIES

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_OFFLINE_RESTART

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_OFFLINE_RESTART_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_Q

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_Q_OFFLINE_RESTART

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES_OFFLINE_RESTART タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_Q_RESTART

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES_RESTART タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_Q_WEAK

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES_WEAK タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_RESTART

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_RESTART_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_WEAK

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_WEAK_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_PROJECT_NAME

出力引数の型は char** です。

SCHA_RESOURCE_STATE

出力引数の型は scha_rsstate_t* です。ローカルノード用に、リソースの RESOURCE_STATE プロパティー値を返します。

SCHA_RESOURCE_STATE_NODE

出力引数の型は scha_rsstate_t* です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。指定のノードの、リソースの RESOURCE_STATE プロパティー値を返します。

SCHA_RETRY_COUNT

出力引数の型は int* です。

SCHA_RETRY_INTERVAL

出力引数の型は int* です。

SCHA_SCALABLE

出力引数の型は boolean_t* です。

SCHA_START_TIMEOUT

出力引数の型は int* です。

SCHA_STATUS

出力引数の型は scha_status_value_t** です。ローカルノード用に、リソースの STATUS プロパティー値を返します。

SCHA_STATUS_NODE

出力引数の型は scha_status_value_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。指定のノードの、リソースの STATUS プロパティー値を返します。

SCHA_STOP_TIMEOUT

出力引数の型は int* です。

SCHA_THOROUGH_PROBE_INTERVAL

出力引数の型は int* です。

SCHA_TYPE

出力引数の型は char** です。

SCHA_TYPE_VERSION

出力引数の型は char** です。

SCHA_UDP_AFFINITY

出力引数の型は boolean_t* です。

SCHA_UPDATE_TIMEOUT
: 出力引数の型は int* です。

SCHA_VALIDATE_TIMEOUT
: 出力引数の型は int* です。

SCHA_WEAK_AFFINITY
: 出力引数の型は boolean_t* です。

次に、リソースタイププロパティーを指定するマクロを示します。出力は、リソースタイプのプロパティーの値です。リソースタイププロパティーについては、1335 ページのrt_properties(5) を参照してください。

SCHA_API_VERSION
: 出力引数の型は int* です。

SCHA_BOOT
: 出力引数の型は char** です。

SCHA_FAILOVER
: 出力引数の型は boolean_t* です。

SCHA_FINI
: 出力引数の型は char** です。

SCHA_GLOBAL_ZONE
: 出力引数の型は boolean_t* です。

SCHA_INIT
: 出力引数の型は char** です。

SCHA_INIT_NODES
: 出力引数の型は scha_initnodes_flag_t* です。

SCHA_INSTALLED_NODES
: 出力引数の型は scha_str_array_t** です。

SCHA_MONITOR_CHECK
: 出力引数の型は char** です。

SCHA_MONITOR_START
: 出力引数の型は char** です。

SCHA_MONITOR_STOP
: 出力引数の型は char** です。

SCHA_PKGLIST
: 出力引数の型は scha_str_array_t** です。

SCHA_POSTNET_STOP
: 出力引数の型は char** です。

SCHA_PRENET_START
: 出力引数の型は char** です。

SCHA_PROXY
: 出力引数の型は boolean_t* です。

SCHA_RT_BASEDIR
: 出力引数の型は char** です。

SCHA_RT_DESCRIPTION
: 出力引数の型は char** です。

SCHA_RT_SYSTEM
: 出力引数の型は boolean_t* です。

SCHA_RT_VERSION
: 出力引数の型は char** です。

SCHA_SINGLE_INSTANCE
: 出力引数の型は boolean_t* です。

SCHA_START
: 出力引数の型は char** です。

SCHA_STOP
: 出力引数の型は char** です。

SCHA_UPDATE
: 出力引数の型は char** です。

SCHA_VALIDATE
: 出力引数の型は char** です。

このリソースのタイプが GLOBAL_ZONE_OVERRIDE リソースプロパティーを宣言すると、SCHA_GLOBAL_ZONE *optag* で取得される値は GLOBAL_ZONE プロパティーの

値ではなく、GLOBAL_ZONE_OVERRIDE プロパティーの現在の値になります。詳細は、1335 ページのrt_properties(5) のマニュアルページの Global_zone プロパティー、および 1287 ページのr_properties(5) のマニュアルページの Global_zone_override プロパティーの説明を参照してください。

これらの関数は、次の戻り値を返します。

0	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_NOERR	関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

例 378 scha_resource_get() 関数の使用例

次の例では、scha_resource_get() を使用して、あるリソースの Retry_count プロパティーの値と Loglevel という拡張プロパティーの値を取得します。

```
main() {
  #include <scha.h>

  scha_err_t err;
  int retry_count_out;
  scha_extprop_value_t *loglevel_out;
  scha_resource_t handle;

/* a configured resource */
  char * resource_name = "example_R";
/* resource group containing example_R */
  char * group_name = "example_RG";

  err = scha_resource_open(resource_name, group_name, &handle);

  err = scha_resource_get(handle, SCHA_RETRY_COUNT, &retry_count_out);

/* Given extension property must be defined in resourcetype RTR file. */
  err = scha_resource_get(handle, SCHA_EXTENSION, "LogLevel", &loglevel_out);

  err = scha_resource_close(handle);

  printf("The retry count for resource %s is %d\n", resource_name,
         retry_count_out);

  printf("The log level for resource %s is %d\n", resource_name,
         loglevel_out->val.val_int);
```

```
}
```

`/usr/cluster/include/scha.h`	インクルードファイル
`/usr/cluster/lib/libscha.so`	ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	発展中

691 ページの`scha_resource_get`(1HA), 1047 ページの`scha_calls`(3HA), 1239 ページの`scha_strerror`(3HA), Unresolved link to " attributes5", 1287 ページの`r_properties`(5), 1335 ページの`rt_properties`(5), 1251 ページの`rt_reg`(4)

名前

scha_resource_open, scha_resource_open_zone, scha_resource_get, scha_resource_get_zone, scha_resource_close — リソース情報アクセス関数

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l scha#include
 <scha.h>scha_err_t scha_resource_open( const char *rname, const char *rgname, scha_resource_t
 *handle);

scha_err_t scha_resource_open_zone(const char *cluster, const char *rs_name, const char
 *rg_name, scha_resource_t * handlep);

scha_err_t scha_resource_get(scha_resource_t handle, const char *tag,...);

scha_err_t scha_resource_get_zone(const char *cluster,
     scha_resource_t handlep, const char *rs_tag, ...);

scha_err_t scha_resource_close(scha_resource_t handle);
```

scha_resource_open()、scha_resource_get ()、および scha_resource_close() 関数は、Resource Group Manager (RGM) クラスタ機能で管理されるリソースに関する情報にアクセスするために一緒に使用します。

scha_resource_open() は、リソースへのアクセスを初期化し、scha_resource_get() が使用するハンドルを返します。

scha_resource_open の rname() 引数には、アクセスするリソースの名前を指定します。*rgname* 引数には、該当するリソースを割り当てたリソースグループの名前を指定します。グループ名が不明な場合、*rgname* 引数を NULL にすることができます。ただし、グループ名を指定する方が、関数の処理をより効率的に進めることができます。*handle* 引数の値は、関数の戻り値を格納する変数のアドレスとなります。

scha_resource_get() 関数は、*tag* 引数に指定されるリソースの情報にアクセスします。*tag* 引数には、scha_tags.h ヘッダーファイルのマクロで定義される文字列値を指定してください。タグ以降の引数は、*tag* の値に依存します。tag より後に追加する引数は、情報を取り出すクラスタノードや、tag 固有の他の情報を指定する際に必要となることがあります。引数リストの最後の引数は、*tag* で指定される情報の格納に適した変数型にする必要があります。これは出力引数で、リソースの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。

scha_resource_get() で使用されたハンドルで scha_resource_close() が呼び出されるまで、scha_resource_get() の返す情報用に割り当てられたメモリーは維持されます。同じ

handle と tag を指定して scha_resource_get() の呼び出しを繰り返すと、新規にメモリーが割り当てられます。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されたりすることはありません。

scha_resource_close() 関数は、以前の *scha_resource_open* の呼び出しで返された handle() 引数を取ります。これは、ハンドルを使用して得た scha_resource_get() 呼び出しの戻り値に割り当てられたメモリーを解放するとともに、このハンドルを無効化します。

scha_tags.h に定義されたマクロは、*scha_resource_get* の tag() 引数に使用される場合があります。

ここでは出力引数および追加引数の型を説明します。構造体と enum 型は、1047 ページのscha_calls(3HA) で説明されています。

scha_resource_open_zone() 関数および scha_resource_get_zone() 関数はそれぞれ、scha_resource_open() および scha_resource_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_resource_open_zone() または scha_resource_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_resource_open() または scha_resource_get() と等しくなります)。

scha_resource_open_zone() から返されたハンドルを閉じるには、scha_resource_close() を使用します。cluster 引数は不要です。

tag 引数

次に、リソースプロパティーを指定するマクロを示します。リソースプロパティーの値が出力されます。SCHA_RESOURCE_STATE、SCHA_STATUS、SCHA_NUM_RG_RESTARTS、SCHA_NUM_RESOURCE_RESTARTS の各プロパティーは、コマンドが実行されるノード上の値を参照します (1287 ページのr_properties(5) を参照)。

ここでは出力引数および追加引数の型を示します。

拡張プロパティー

これらのプロパティーは、リソースタイプの Resource Type Registration (RTR) ファイル内で宣言します。リソースタイプの実装によって、これらのプロパティーを定義します。

SCHA_AFFINITY_TIMEOUT

出力引数の型は int* です。

SCHA_ALL_EXTENSIONS

出力引数の型は scha_str_array_t** です。リソースのすべての拡張プロパティーの名前を返します。

SCHA_APPLICATION_USER

出力引数の型は char** です。

SCHA_BOOT_TIMEOUT

出力引数の型は int* です。

SCHA_CHEAP_PROBE_INTERVAL

出力引数の型は int* です。

SCHA_EXTENSION

出力引数の型は scha_extprop_value_t* です。拡張プロパティーの名前を提供する、char* 型の追加の引数が必要です。ローカルノードのプロパティーのタイプとその値を返します。

明示的な値が割り当てられていないノード上でこのプロパティーの値をユーザーがリクエストすると、RTR ファイルで宣言されているデフォルト値が返されます。1251 ページのrt_reg(4) のマニュアルページを参照してください。

SCHA_EXTENSION_NODE

出力引数の型は scha_extprop_value_t* です。char* 型の追加の引数が必要です。最初の引数では、拡張プロパティーの名前を指定し、2 番目の引数ではクラスタノードを指定します。指定のノードのプロパティーのタイプとその値を返します。

明示的な値が割り当てられていないノード上でこのプロパティーの値をユーザーがリクエストすると、RTR ファイルで宣言されているデフォルト値が返されます。1251 ページのrt_reg(4) のマニュアルページを参照してください。

SCHA_FAILOVER_MODE

出力引数の型は scha_failover_mode_t* です。

SCHA_FINI_TIMEOUT

出力引数の型は int* です。

SCHA_GLOBAL_ZONE_OVERRIDE

出力引数の型は boolean_t* です。

SCHA_GROUP

出力引数の型は char** です。リソースを構成するリソースグループの名前を返します。

SCHA_INIT_TIMEOUT

出力引数の型は int* です。

SCHA_LOAD_BALANCING_POLICY

出力引数の型は char** です。

SCHA_LOAD_BALANCING_WEIGHTS

出力引数の型は scha_str_array_t** です。

SCHA_MONITOR_CHECK_TIMEOUT

出力引数の型は int* です。

SCHA_MONITOR_START_TIMEOUT

出力引数の型は int* です。

SCHA_MONITOR_STOP_TIMEOUT

出力引数の型は int* です。

SCHA_MONITORED_SWITCH

出力引数の型は scha_switch_t* です。戻り値は、ローカルノードでリソースがモニターされるかどうかを示します。

SCHA_MONITORED_SWITCH_NODE

出力引数の型は scha_switch_t* です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードでリソースがモニターされるかどうかを示します。

SCHA_NETWORK_RESOURCES_USED

出力引数の型は scha_str_array_t** です。

SCHA_NUM_RESOURCE_RESTARTS

出力引数の型は int* です。クエリーが実行されるゾーンのこのリソースで発生した、リソースの再起動のリクエスト数を返します。詳細は、1287 ページのr_properties(5) のマニュアルページを参照してください。

SCHA_NUM_RG_RESTARTS

出力引数の型は int* です。クエリーが実行されるゾーンのこのリソースで発生した、リソースグループの再起動のリクエスト数を返します。詳細は、1287 ページのr_properties(5) のマニュアルページを参照してください。

SCHA_ON_OFF_SWITCH

出力引数の型は scha_switch_t* です。戻り値は、ローカルノードでリソースが有効かどうかを示します。

SCHA_ON_OFF_SWITCH_NODE

出力引数の型は scha_switch_t* です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードでリソースが有効かどうかを示します。

SCHA_PORT_LIST

出力引数の型は scha_str_array_t** です。

SCHA_POSTNET_STOP_TIMEOUT

出力引数の型は int* です。

SCHA_PRENET_START_TIMEOUT

出力引数の型は int* です。

SCHA_PRE_EVICT

出力引数の型は boolean_t* です。

SCHA_R_DESCRIPTION

出力引数の型は char** です。

SCHA_RESOURCE_DEPENDENCIES

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_OFFLINE_RESTART

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_OFFLINE_RESTART_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_Q

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_Q_OFFLINE_RESTART

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES_OFFLINE_RESTART タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_Q_RESTART

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES_RESTART タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_Q_WEAK

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES_WEAK タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_RESTART

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_RESTART_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_WEAK

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_WEAK_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_PROJECT_NAME

出力引数の型は char** です。

SCHA_RESOURCE_STATE

出力引数の型は scha_rsstate_t* です。ローカルノード用に、リソースの RESOURCE_STATE プロパティー値を返します。

SCHA_RESOURCE_STATE_NODE

出力引数の型は scha_rsstate_t* です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。指定のノードの、リソースの RESOURCE_STATE プロパティー値を返します。

SCHA_RETRY_COUNT

出力引数の型は int* です。

SCHA_RETRY_INTERVAL

出力引数の型は int* です。

SCHA_SCALABLE

出力引数の型は boolean_t* です。

SCHA_START_TIMEOUT

出力引数の型は int* です。

SCHA_STATUS

出力引数の型は scha_status_value_t** です。ローカルノード用に、リソースの STATUS プロパティー値を返します。

SCHA_STATUS_NODE

出力引数の型は scha_status_value_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。指定のノードの、リソースの STATUS プロパティー値を返します。

SCHA_STOP_TIMEOUT

出力引数の型は int* です。

SCHA_THOROUGH_PROBE_INTERVAL

出力引数の型は int* です。

SCHA_TYPE

出力引数の型は char** です。

SCHA_TYPE_VERSION

出力引数の型は char** です。

SCHA_UDP_AFFINITY

出力引数の型は boolean_t* です。

SCHA_UPDATE_TIMEOUT
: 出力引数の型は int* です。

SCHA_VALIDATE_TIMEOUT
: 出力引数の型は int* です。

SCHA_WEAK_AFFINITY
: 出力引数の型は boolean_t* です。

次に、リソースタイププロパティーを指定するマクロを示します。出力は、リソースタイプのプロパティーの値です。リソースタイププロパティーについては、1335 ページのrt_properties(5) を参照してください。

SCHA_API_VERSION
: 出力引数の型は int* です。

SCHA_BOOT
: 出力引数の型は char** です。

SCHA_FAILOVER
: 出力引数の型は boolean_t* です。

SCHA_FINI
: 出力引数の型は char** です。

SCHA_GLOBAL_ZONE
: 出力引数の型は boolean_t* です。

SCHA_INIT
: 出力引数の型は char** です。

SCHA_INIT_NODES
: 出力引数の型は scha_initnodes_flag_t* です。

SCHA_INSTALLED_NODES
: 出力引数の型は scha_str_array_t** です。

SCHA_MONITOR_CHECK
: 出力引数の型は char** です。

SCHA_MONITOR_START
: 出力引数の型は char** です。

SCHA_MONITOR_STOP
: 出力引数の型は char** です。

SCHA_PKGLIST
: 出力引数の型は scha_str_array_t** です。

SCHA_POSTNET_STOP
: 出力引数の型は char** です。

SCHA_PRENET_START
: 出力引数の型は char** です。

SCHA_PROXY
: 出力引数の型は boolean_t* です。

SCHA_RT_BASEDIR
: 出力引数の型は char** です。

SCHA_RT_DESCRIPTION
: 出力引数の型は char** です。

SCHA_RT_SYSTEM
: 出力引数の型は boolean_t* です。

SCHA_RT_VERSION
: 出力引数の型は char** です。

SCHA_SINGLE_INSTANCE
: 出力引数の型は boolean_t* です。

SCHA_START
: 出力引数の型は char** です。

SCHA_STOP
: 出力引数の型は char** です。

SCHA_UPDATE
: 出力引数の型は char** です。

SCHA_VALIDATE
: 出力引数の型は char** です。

このリソースのタイプが GLOBAL_ZONE_OVERRIDE リソースプロパティーを宣言すると、SCHA_GLOBAL_ZONE *optag* で取得される値は GLOBAL_ZONE プロパティーの

値ではなく、GLOBAL_ZONE_OVERRIDE プロパティーの現在の値になります。詳細は、1335 ページのrt_properties(5) のマニュアルページの Global_zone プロパティー、および 1287 ページのr_properties(5) のマニュアルページの Global_zone_override プロパティーの説明を参照してください。

これらの関数は、次の戻り値を返します。

0	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_NOERR	関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

例 379 scha_resource_get() 関数の使用例

次の例では、scha_resource_get() を使用して、あるリソースの Retry_count プロパティーの値と Loglevel という拡張プロパティーの値を取得します。

```
main() {
  #include <scha.h>

  scha_err_t err;
  int retry_count_out;
  scha_extprop_value_t *loglevel_out;
  scha_resource_t handle;

/* a configured resource */
  char * resource_name = "example_R";
/* resource group containing example_R */
  char * group_name = "example_RG";

  err = scha_resource_open(resource_name, group_name, &handle);

  err = scha_resource_get(handle, SCHA_RETRY_COUNT, &retry_count_out);

/* Given extension property must be defined in resourcetype RTR file. */
  err = scha_resource_get(handle, SCHA_EXTENSION, "LogLevel", &loglevel_out);

  err = scha_resource_close(handle);

  printf("The retry count for resource %s is %d\n", resource_name,
         retry_count_out);

  printf("The log level for resource %s is %d\n", resource_name,
         loglevel_out->val.val_int);
```

```
}
```

`/usr/cluster/include/scha.h`	インクルードファイル
`/usr/cluster/lib/libscha.so`	ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	発展中

691 ページの`scha_resource_get`(1HA), 1047 ページの`scha_calls`(3HA), 1239 ページの`scha_strerror`(3HA), Unresolved link to " attributes5", 1287 ページの`r_properties`(5), 1335 ページの`rt_properties`(5), 1251 ページの`rt_reg`(4)

名前

scha_resource_open, scha_resource_open_zone, scha_resource_get, scha_resource_get_zone, scha_resource_close — リソース情報アクセス関数

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l scha#include
 <scha.h>scha_err_t scha_resource_open( const char *rname, const char *rgname, scha_resource_t
 *handle);

scha_err_t scha_resource_open_zone(const char *cluster, const char *rs_name, const char
 *rg_name, scha_resource_t * handlep);

scha_err_t scha_resource_get(scha_resource_t handle, const char *tag,...);

scha_err_t scha_resource_get_zone(const char *cluster,
     scha_resource_t handlep, const char *rs_tag, ...);

scha_err_t scha_resource_close(scha_resource_t handle);
```

scha_resource_open()、scha_resource_get ()、および scha_resource_close() 関数は、Resource Group Manager (RGM) クラスタ機能で管理されるリソースに関する情報にアクセスするために一緒に使用します。

scha_resource_open() は、リソースへのアクセスを初期化し、scha_resource_get() が使用するハンドルを返します。

scha_resource_open の rname() 引数には、アクセスするリソースの名前を指定します。*rgname* 引数には、該当するリソースを割り当てたリソースグループの名前を指定します。グループ名が不明な場合、*rgname* 引数を NULL にすることができます。ただし、グループ名を指定する方が、関数の処理をより効率的に進めることができます。*handle* 引数の値は、関数の戻り値を格納する変数のアドレスとなります。

scha_resource_get() 関数は、*tag* 引数に指定されるリソースの情報にアクセスします。*tag* 引数には、scha_tags.h ヘッダーファイルのマクロで定義される文字列値を指定してください。タグ以降の引数は、*tag* の値に依存します。tag より後に追加する引数は、情報を取り出すクラスタノードや、tag 固有の他の情報を指定する際に必要となることがあります。引数リストの最後の引数は、*tag* で指定される情報の格納に適した変数型にする必要があります。これは出力引数で、リソースの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。

scha_resource_get() で使用されたハンドルで scha_resource_close() が呼び出されるまで、scha_resource_get() の返す情報用に割り当てられたメモリーは維持されます。同じ

handle と tag を指定して scha_resource_get() の呼び出しを繰り返すと、新規にメモリーが割り当てられます。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されたりすることはありません。

scha_resource_close() 関数は、以前の *scha_resource_open* の呼び出しで返された handle() 引数を取ります。これは、ハンドルを使用して得た scha_resource_get() 呼び出しの戻り値に割り当てられたメモリーを解放するとともに、このハンドルを無効化します。

scha_tags.h に定義されたマクロは、*scha_resource_get* の tag() 引数に使用される場合があります。

ここでは出力引数および追加引数の型を説明します。構造体と enum 型は、1047 ページのscha_calls(3HA) で説明されています。

scha_resource_open_zone() 関数および scha_resource_get_zone() 関数はそれぞれ、scha_resource_open() および scha_resource_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_resource_open_zone() または scha_resource_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_resource_open() または scha_resource_get() と等しくなります)。

scha_resource_open_zone() から返されたハンドルを閉じるには、scha_resource_close() を使用します。cluster 引数は不要です。

tag 引数

次に、リソースプロパティーを指定するマクロを示します。リソースプロパティーの値が出力されます。SCHA_RESOURCE_STATE、SCHA_STATUS、SCHA_NUM_RG_RESTARTS、SCHA_NUM_RESOURCE_RESTARTS の各プロパティーは、コマンドが実行されるノード上の値を参照します (1287 ページのr_properties(5) を参照)。

ここでは出力引数および追加引数の型を示します。

拡張プロパティー

これらのプロパティーは、リソースタイプの Resource Type Registration (RTR) ファイル内で宣言します。リソースタイプの実装によって、これらのプロパティーを定義します。

SCHA_AFFINITY_TIMEOUT

出力引数の型は int* です。

SCHA_ALL_EXTENSIONS

出力引数の型は scha_str_array_t** です。リソースのすべての拡張プロパティーの名前を返します。

SCHA_APPLICATION_USER

出力引数の型は char** です。

SCHA_BOOT_TIMEOUT

出力引数の型は int* です。

SCHA_CHEAP_PROBE_INTERVAL

出力引数の型は int* です。

SCHA_EXTENSION

出力引数の型は scha_extprop_value_t* です。拡張プロパティーの名前を提供する、char* 型の追加の引数が必要です。ローカルノードのプロパティーのタイプとその値を返します。

明示的な値が割り当てられていないノード上でこのプロパティーの値をユーザーがリクエストすると、RTR ファイルで宣言されているデフォルト値が返されます。1251 ページのrt_reg(4) のマニュアルページを参照してください。

SCHA_EXTENSION_NODE

出力引数の型は scha_extprop_value_t* です。char* 型の追加の引数が必要です。最初の引数では、拡張プロパティーの名前を指定し、2 番目の引数ではクラスタノードを指定します。指定のノードのプロパティーのタイプとその値を返します。

明示的な値が割り当てられていないノード上でこのプロパティーの値をユーザーがリクエストすると、RTR ファイルで宣言されているデフォルト値が返されます。1251 ページのrt_reg(4) のマニュアルページを参照してください。

SCHA_FAILOVER_MODE

出力引数の型は scha_failover_mode_t* です。

SCHA_FINI_TIMEOUT

出力引数の型は int* です。

SCHA_GLOBAL_ZONE_OVERRIDE

出力引数の型は boolean_t* です。

SCHA_GROUP

出力引数の型は char** です。リソースを構成するリソースグループの名前を返します。

SCHA_INIT_TIMEOUT

出力引数の型は int* です。

SCHA_LOAD_BALANCING_POLICY

出力引数の型は char** です。

SCHA_LOAD_BALANCING_WEIGHTS

出力引数の型は scha_str_array_t** です。

SCHA_MONITOR_CHECK_TIMEOUT

出力引数の型は int* です。

SCHA_MONITOR_START_TIMEOUT

出力引数の型は int* です。

SCHA_MONITOR_STOP_TIMEOUT

出力引数の型は int* です。

SCHA_MONITORED_SWITCH

出力引数の型は scha_switch_t* です。戻り値は、ローカルノードでリソースがモニターされるかどうかを示します。

SCHA_MONITORED_SWITCH_NODE

出力引数の型は scha_switch_t* です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードでリソースがモニターされるかどうかを示します。

SCHA_NETWORK_RESOURCES_USED

出力引数の型は scha_str_array_t** です。

SCHA_NUM_RESOURCE_RESTARTS

出力引数の型は int* です。クエリーが実行されるゾーンのこのリソースで発生した、リソースの再起動のリクエスト数を返します。詳細は、1287 ページのr_properties(5) のマニュアルページを参照してください。

SCHA_NUM_RG_RESTARTS

出力引数の型は int* です。クエリーが実行されるゾーンのこのリソースで発生した、リソースグループの再起動のリクエスト数を返します。詳細は、1287 ページのr_properties(5) のマニュアルページを参照してください。

SCHA_ON_OFF_SWITCH

出力引数の型は scha_switch_t* です。戻り値は、ローカルノードでリソースが有効かどうかを示します。

SCHA_ON_OFF_SWITCH_NODE

出力引数の型は scha_switch_t* です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードでリソースが有効かどうかを示します。

SCHA_PORT_LIST

出力引数の型は scha_str_array_t** です。

SCHA_POSTNET_STOP_TIMEOUT

出力引数の型は int* です。

SCHA_PRENET_START_TIMEOUT

出力引数の型は int* です。

SCHA_PRE_EVICT

出力引数の型は boolean_t* です。

SCHA_R_DESCRIPTION

出力引数の型は char** です。

SCHA_RESOURCE_DEPENDENCIES

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_OFFLINE_RESTART

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_OFFLINE_RESTART_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_Q

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_Q_OFFLINE_RESTART

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES_OFFLINE_RESTART タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_Q_RESTART

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES_RESTART タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_Q_WEAK

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES_WEAK タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_RESTART

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_RESTART_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_WEAK

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_WEAK_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_PROJECT_NAME

出力引数の型は char** です。

SCHA_RESOURCE_STATE

出力引数の型は scha_rsstate_t* です。ローカルノード用に、リソースの RESOURCE_STATE プロパティー値を返します。

SCHA_RESOURCE_STATE_NODE

出力引数の型は scha_rsstate_t* です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。指定のノードの、リソースの RESOURCE_STATE プロパティー値を返します。

SCHA_RETRY_COUNT

出力引数の型は int* です。

SCHA_RETRY_INTERVAL

出力引数の型は int* です。

SCHA_SCALABLE

出力引数の型は boolean_t* です。

SCHA_START_TIMEOUT

出力引数の型は int* です。

SCHA_STATUS

出力引数の型は scha_status_value_t** です。ローカルノード用に、リソースの STATUS プロパティー値を返します。

SCHA_STATUS_NODE

出力引数の型は scha_status_value_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。指定のノードの、リソースの STATUS プロパティー値を返します。

SCHA_STOP_TIMEOUT

出力引数の型は int* です。

SCHA_THOROUGH_PROBE_INTERVAL

出力引数の型は int* です。

SCHA_TYPE

出力引数の型は char** です。

SCHA_TYPE_VERSION

出力引数の型は char** です。

SCHA_UDP_AFFINITY

出力引数の型は boolean_t* です。

SCHA_UPDATE_TIMEOUT
: 出力引数の型は int* です。

SCHA_VALIDATE_TIMEOUT
: 出力引数の型は int* です。

SCHA_WEAK_AFFINITY
: 出力引数の型は boolean_t* です。

次に、リソースタイププロパティーを指定するマクロを示します。出力は、リソースタイプのプロパティーの値です。リソースタイププロパティーについては、1335 ページのrt_properties(5) を参照してください。

SCHA_API_VERSION
: 出力引数の型は int* です。

SCHA_BOOT
: 出力引数の型は char** です。

SCHA_FAILOVER
: 出力引数の型は boolean_t* です。

SCHA_FINI
: 出力引数の型は char** です。

SCHA_GLOBAL_ZONE
: 出力引数の型は boolean_t* です。

SCHA_INIT
: 出力引数の型は char** です。

SCHA_INIT_NODES
: 出力引数の型は scha_initnodes_flag_t* です。

SCHA_INSTALLED_NODES
: 出力引数の型は scha_str_array_t** です。

SCHA_MONITOR_CHECK
: 出力引数の型は char** です。

SCHA_MONITOR_START
: 出力引数の型は char** です。

SCHA_MONITOR_STOP
: 出力引数の型は char** です。

SCHA_PKGLIST
: 出力引数の型は scha_str_array_t** です。

SCHA_POSTNET_STOP
: 出力引数の型は char** です。

SCHA_PRENET_START
: 出力引数の型は char** です。

SCHA_PROXY
: 出力引数の型は boolean_t* です。

SCHA_RT_BASEDIR
: 出力引数の型は char** です。

SCHA_RT_DESCRIPTION
: 出力引数の型は char** です。

SCHA_RT_SYSTEM
: 出力引数の型は boolean_t* です。

SCHA_RT_VERSION
: 出力引数の型は char** です。

SCHA_SINGLE_INSTANCE
: 出力引数の型は boolean_t* です。

SCHA_START
: 出力引数の型は char** です。

SCHA_STOP
: 出力引数の型は char** です。

SCHA_UPDATE
: 出力引数の型は char** です。

SCHA_VALIDATE
: 出力引数の型は char** です。

このリソースのタイプが GLOBAL_ZONE_OVERRIDE リソースプロパティーを宣言すると、SCHA_GLOBAL_ZONE *optag* で取得される値は GLOBAL_ZONE プロパティーの

値ではなく、GLOBAL_ZONE_OVERRIDE プロパティーの現在の値になります。詳細は、1335 ページのrt_properties(5) のマニュアルページの Global_zone プロパティー、および 1287 ページのr_properties(5) のマニュアルページの Global_zone_override プロパティーの説明を参照してください。

これらの関数は、次の戻り値を返します。

0	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_NOERR	関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

例 380 scha_resource_get() 関数の使用例

次の例では、scha_resource_get() を使用して、あるリソースの Retry_count プロパティーの値と Loglevel という拡張プロパティーの値を取得します。

```
main() {
  #include <scha.h>

  scha_err_t err;
  int retry_count_out;
  scha_extprop_value_t *loglevel_out;
  scha_resource_t handle;

/* a configured resource */
  char * resource_name = "example_R";
/* resource group containing example_R */
  char * group_name = "example_RG";

  err = scha_resource_open(resource_name, group_name, &handle);

  err = scha_resource_get(handle, SCHA_RETRY_COUNT, &retry_count_out);

/* Given extension property must be defined in resourcetype RTR file. */
  err = scha_resource_get(handle, SCHA_EXTENSION, "LogLevel", &loglevel_out);

  err = scha_resource_close(handle);

  printf("The retry count for resource %s is %d\n", resource_name,
         retry_count_out);

  printf("The log level for resource %s is %d\n", resource_name,
         loglevel_out->val.val_int);
```

```
}
```

`/usr/cluster/include/scha.h`	インクルードファイル
`/usr/cluster/lib/libscha.so`	ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	発展中

691 ページの`scha_resource_get`(1HA), 1047 ページの`scha_calls`(3HA), 1239 ページの`scha_strerror`(3HA), Unresolved link to " attributes5", 1287 ページの`r_properties`(5), 1335 ページの`rt_properties`(5), 1251 ページの`rt_reg`(4)

名前

scha_resource_open, scha_resource_open_zone, scha_resource_get, scha_resource_get_zone, scha_resource_close — リソース情報アクセス関数

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l scha#include
 <scha.h>scha_err_t scha_resource_open( const char *rname, const char *rgname, scha_resource_t
 *handle);

scha_err_t scha_resource_open_zone(const char *cluster, const char *rs_name, const char
 *rg_name, scha_resource_t * handlep);

scha_err_t scha_resource_get(scha_resource_t handle, const char *tag,...);

scha_err_t scha_resource_get_zone(const char *cluster,
     scha_resource_t handlep, const char *rs_tag, ...);

scha_err_t scha_resource_close(scha_resource_t handle);
```

scha_resource_open()、scha_resource_get ()、および scha_resource_close() 関数は、Resource Group Manager (RGM) クラスタ機能で管理されるリソースに関する情報にアクセスするために一緒に使用します。

scha_resource_open() は、リソースへのアクセスを初期化し、scha_resource_get() が使用するハンドルを返します。

scha_resource_open の rname() 引数には、アクセスするリソースの名前を指定します。*rgname* 引数には、該当するリソースを割り当てたリソースグループの名前を指定します。グループ名が不明な場合、*rgname* 引数を NULL にすることができます。ただし、グループ名を指定する方が、関数の処理をより効率的に進めることができます。*handle* 引数の値は、関数の戻り値を格納する変数のアドレスとなります。

scha_resource_get() 関数は、*tag* 引数に指定されるリソースの情報にアクセスします。*tag* 引数には、scha_tags.h ヘッダーファイルのマクロで定義される文字列値を指定してください。タグ以降の引数は、*tag* の値に依存します。tag より後に追加する引数は、情報を取り出すクラスタノードや、tag 固有の他の情報を指定する際に必要となることがあります。引数リストの最後の引数は、*tag* で指定される情報の格納に適した変数型にする必要があります。これは出力引数で、リソースの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。

scha_resource_get() で使用されたハンドルで scha_resource_close() が呼び出されるまで、scha_resource_get() の返す情報用に割り当てられたメモリーは維持されます。同じ

handle と tag を指定して scha_resource_get() の呼び出しを繰り返すと、新規にメモリーが割り当てられます。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されたりすることはありません。

scha_resource_close() 関数は、以前の *scha_resource_open* の呼び出しで返された handle() 引数を取ります。これは、ハンドルを使用して得た scha_resource_get() 呼び出しの戻り値に割り当てられたメモリーを解放するとともに、このハンドルを無効化します。

scha_tags.h に定義されたマクロは、*scha_resource_get* の tag() 引数に使用される場合があります。

ここでは出力引数および追加引数の型を説明します。構造体と enum 型は、1047 ページのscha_calls(3HA) で説明されています。

scha_resource_open_zone() 関数および scha_resource_get_zone() 関数はそれぞれ、scha_resource_open() および scha_resource_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_resource_open_zone() または scha_resource_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_resource_open() または scha_resource_get() と等しくなります)。

scha_resource_open_zone() から返されたハンドルを閉じるには、scha_resource_close() を使用します。cluster 引数は不要です。

tag 引数

次に、リソースプロパティーを指定するマクロを示します。リソースプロパティーの値が出力されます。SCHA_RESOURCE_STATE、SCHA_STATUS、SCHA_NUM_RG_RESTARTS、SCHA_NUM_RESOURCE_RESTARTS の各プロパティーは、コマンドが実行されるノード上の値を参照します (1287 ページのr_properties(5) を参照)。

ここでは出力引数および追加引数の型を示します。

拡張プロパティー

これらのプロパティーは、リソースタイプの Resource Type Registration (RTR) ファイル内で宣言します。リソースタイプの実装によって、これらのプロパティーを定義します。

SCHA_AFFINITY_TIMEOUT

出力引数の型は int* です。

SCHA_ALL_EXTENSIONS

出力引数の型は scha_str_array_t** です。リソースのすべての拡張プロパティーの名前を返します。

SCHA_APPLICATION_USER

出力引数の型は char** です。

SCHA_BOOT_TIMEOUT

出力引数の型は int* です。

SCHA_CHEAP_PROBE_INTERVAL

出力引数の型は int* です。

SCHA_EXTENSION

出力引数の型は scha_extprop_value_t* です。拡張プロパティーの名前を提供する、char* 型の追加の引数が必要です。ローカルノードのプロパティーのタイプとその値を返します。

明示的な値が割り当てられていないノード上でこのプロパティーの値をユーザーがリクエストすると、RTR ファイルで宣言されているデフォルト値が返されます。1251 ページのrt_reg(4) のマニュアルページを参照してください。

SCHA_EXTENSION_NODE

出力引数の型は scha_extprop_value_t* です。char* 型の追加の引数が必要です。最初の引数では、拡張プロパティーの名前を指定し、2 番目の引数ではクラスタノードを指定します。指定のノードのプロパティーのタイプとその値を返します。

明示的な値が割り当てられていないノード上でこのプロパティーの値をユーザーがリクエストすると、RTR ファイルで宣言されているデフォルト値が返されます。1251 ページのrt_reg(4) のマニュアルページを参照してください。

SCHA_FAILOVER_MODE

出力引数の型は scha_failover_mode_t* です。

SCHA_FINI_TIMEOUT

出力引数の型は int* です。

SCHA_GLOBAL_ZONE_OVERRIDE

出力引数の型は boolean_t* です。

SCHA_GROUP

出力引数の型は char** です。リソースを構成するリソースグループの名前を返します。

SCHA_INIT_TIMEOUT

出力引数の型は int* です。

SCHA_LOAD_BALANCING_POLICY

出力引数の型は char** です。

SCHA_LOAD_BALANCING_WEIGHTS

出力引数の型は scha_str_array_t** です。

SCHA_MONITOR_CHECK_TIMEOUT

出力引数の型は int* です。

SCHA_MONITOR_START_TIMEOUT

出力引数の型は int* です。

SCHA_MONITOR_STOP_TIMEOUT

出力引数の型は int* です。

SCHA_MONITORED_SWITCH

出力引数の型は scha_switch_t* です。戻り値は、ローカルノードでリソースがモニターされるかどうかを示します。

SCHA_MONITORED_SWITCH_NODE

出力引数の型は scha_switch_t* です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードでリソースがモニターされるかどうかを示します。

SCHA_NETWORK_RESOURCES_USED

出力引数の型は scha_str_array_t** です。

SCHA_NUM_RESOURCE_RESTARTS

出力引数の型は int* です。クエリーが実行されるゾーンのこのリソースで発生した、リソースの再起動のリクエスト数を返します。詳細は、1287 ページのr_properties(5) のマニュアルページを参照してください。

SCHA_NUM_RG_RESTARTS

出力引数の型は int* です。クエリーが実行されるゾーンのこのリソースで発生した、リソースグループの再起動のリクエスト数を返します。詳細は、1287 ページのr_properties(5) のマニュアルページを参照してください。

SCHA_ON_OFF_SWITCH

出力引数の型は scha_switch_t* です。戻り値は、ローカルノードでリソースが有効かどうかを示します。

SCHA_ON_OFF_SWITCH_NODE

出力引数の型は scha_switch_t* です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードでリソースが有効かどうかを示します。

SCHA_PORT_LIST

出力引数の型は scha_str_array_t** です。

SCHA_POSTNET_STOP_TIMEOUT

出力引数の型は int* です。

SCHA_PRENET_START_TIMEOUT

出力引数の型は int* です。

SCHA_PRE_EVICT

出力引数の型は boolean_t* です。

SCHA_R_DESCRIPTION

出力引数の型は char** です。

SCHA_RESOURCE_DEPENDENCIES

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_OFFLINE_RESTART

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_OFFLINE_RESTART_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_Q

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_Q_OFFLINE_RESTART

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES_OFFLINE_RESTART タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_Q_RESTART

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES_RESTART タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_Q_WEAK

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES_WEAK タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_RESTART

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_RESTART_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_WEAK

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_WEAK_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_PROJECT_NAME

出力引数の型は char** です。

SCHA_RESOURCE_STATE

出力引数の型は scha_rsstate_t* です。ローカルノード用に、リソースの RESOURCE_STATE プロパティー値を返します。

SCHA_RESOURCE_STATE_NODE

出力引数の型は scha_rsstate_t* です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。指定のノードの、リソースの RESOURCE_STATE プロパティー値を返します。

SCHA_RETRY_COUNT

出力引数の型は int* です。

SCHA_RETRY_INTERVAL

出力引数の型は int* です。

SCHA_SCALABLE

出力引数の型は boolean_t* です。

SCHA_START_TIMEOUT

出力引数の型は int* です。

SCHA_STATUS

出力引数の型は scha_status_value_t** です。ローカルノード用に、リソースの STATUS プロパティー値を返します。

SCHA_STATUS_NODE

出力引数の型は scha_status_value_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。指定のノードの、リソースの STATUS プロパティー値を返します。

SCHA_STOP_TIMEOUT

出力引数の型は int* です。

SCHA_THOROUGH_PROBE_INTERVAL

出力引数の型は int* です。

SCHA_TYPE

出力引数の型は char** です。

SCHA_TYPE_VERSION

出力引数の型は char** です。

SCHA_UDP_AFFINITY

出力引数の型は boolean_t* です。

SCHA_UPDATE_TIMEOUT

出力引数の型は int* です。

SCHA_VALIDATE_TIMEOUT

出力引数の型は int* です。

SCHA_WEAK_AFFINITY

出力引数の型は boolean_t* です。

次に、リソースタイププロパティーを指定するマクロを示します。出力は、リソースタイプのプロパティーの値です。リソースタイププロパティーについては、1335 ページのrt_properties(5) を参照してください。

SCHA_API_VERSION

出力引数の型は int* です。

SCHA_BOOT

出力引数の型は char** です。

SCHA_FAILOVER

出力引数の型は boolean_t* です。

SCHA_FINI

出力引数の型は char** です。

SCHA_GLOBAL_ZONE

出力引数の型は boolean_t* です。

SCHA_INIT

出力引数の型は char** です。

SCHA_INIT_NODES

出力引数の型は scha_initnodes_flag_t* です。

SCHA_INSTALLED_NODES

出力引数の型は scha_str_array_t** です。

SCHA_MONITOR_CHECK

出力引数の型は char** です。

SCHA_MONITOR_START

出力引数の型は char** です。

SCHA_MONITOR_STOP
: 出力引数の型は char** です。

SCHA_PKGLIST
: 出力引数の型は scha_str_array_t** です。

SCHA_POSTNET_STOP
: 出力引数の型は char** です。

SCHA_PRENET_START
: 出力引数の型は char** です。

SCHA_PROXY
: 出力引数の型は boolean_t* です。

SCHA_RT_BASEDIR
: 出力引数の型は char** です。

SCHA_RT_DESCRIPTION
: 出力引数の型は char** です。

SCHA_RT_SYSTEM
: 出力引数の型は boolean_t* です。

SCHA_RT_VERSION
: 出力引数の型は char** です。

SCHA_SINGLE_INSTANCE
: 出力引数の型は boolean_t* です。

SCHA_START
: 出力引数の型は char** です。

SCHA_STOP
: 出力引数の型は char** です。

SCHA_UPDATE
: 出力引数の型は char** です。

SCHA_VALIDATE
: 出力引数の型は char** です。

このリソースのタイプが GLOBAL_ZONE_OVERRIDE リソースプロパティーを宣言すると、SCHA_GLOBAL_ZONE *optag* で取得される値は GLOBAL_ZONE プロパティーの

値ではなく、GLOBAL_ZONE_OVERRIDE プロパティーの現在の値になります。詳細は、1335 ページのrt_properties(5) のマニュアルページの Global_zone プロパティー、および 1287 ページのr_properties(5) のマニュアルページの Global_zone_override プロパティーの説明を参照してください。

これらの関数は、次の戻り値を返します。

0	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_NOERR	関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

例 381 scha_resource_get() 関数の使用例

次の例では、scha_resource_get() を使用して、あるリソースの Retry_count プロパティーの値と Loglevel という拡張プロパティーの値を取得します。

```
main() {
  #include <scha.h>

  scha_err_t err;
  int retry_count_out;
  scha_extprop_value_t *loglevel_out;
  scha_resource_t handle;

/* a configured resource */
  char * resource_name = "example_R";
/* resource group containing example_R */
  char * group_name = "example_RG";

  err = scha_resource_open(resource_name, group_name, &handle);

  err = scha_resource_get(handle, SCHA_RETRY_COUNT, &retry_count_out);

/* Given extension property must be defined in resourcetype RTR file. */
  err = scha_resource_get(handle, SCHA_EXTENSION, "LogLevel", &loglevel_out);

  err = scha_resource_close(handle);

  printf("The retry count for resource %s is %d\n", resource_name,
         retry_count_out);

  printf("The log level for resource %s is %d\n", resource_name,
         loglevel_out->val.val_int);
```

```
}
```

`/usr/cluster/include/scha.h`	インクルードファイル
`/usr/cluster/lib/libscha.so`	ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	発展中

691 ページの`scha_resource_get`(1HA), 1047 ページの`scha_calls`(3HA), 1239 ページの`scha_strerror`(3HA), Unresolved link to " attributes5", 1287 ページの`r_properties`(5), 1335 ページの`rt_properties`(5), 1251 ページの`rt_reg`(4)

名前

scha_resource_open, scha_resource_open_zone, scha_resource_get, scha_resource_get_zone, scha_resource_close — リソース情報アクセス関数

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l scha#include
 <scha.h>scha_err_t scha_resource_open( const char *rname, const char *rgname, scha_resource_t
 *handle);

scha_err_t scha_resource_open_zone(const char *cluster, const char *rs_name, const char
 *rg_name, scha_resource_t * handlep);

scha_err_t scha_resource_get(scha_resource_t handle, const char *tag,...);

scha_err_t scha_resource_get_zone(const char *cluster,
     scha_resource_t handlep, const char *rs_tag, ...);

scha_err_t scha_resource_close(scha_resource_t handle);
```

scha_resource_open()、scha_resource_get ()、および scha_resource_close() 関数は、Resource Group Manager (RGM) クラスタ機能で管理されるリソースに関する情報にアクセスするために一緒に使用します。

scha_resource_open() は、リソースへのアクセスを初期化し、scha_resource_get() が使用するハンドルを返します。

scha_resource_open の rname() 引数には、アクセスするリソースの名前を指定します。*rgname* 引数には、該当するリソースを割り当てたリソースグループの名前を指定します。グループ名が不明な場合、*rgname* 引数を NULL にすることができます。ただし、グループ名を指定する方が、関数の処理をより効率的に進めることができます。*handle* 引数の値は、関数の戻り値を格納する変数のアドレスとなります。

scha_resource_get() 関数は、*tag* 引数に指定されるリソースの情報にアクセスします。*tag* 引数には、scha_tags.h ヘッダーファイルのマクロで定義される文字列値を指定してください。タグ以降の引数は、*tag* の値に依存します。tag より後に追加する引数は、情報を取り出すクラスタノードや、tag 固有の他の情報を指定する際に必要となることがあります。引数リストの最後の引数は、*tag* で指定される情報の格納に適した変数型にする必要があります。これは出力引数で、リソースの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。

scha_resource_get() で使用されたハンドルで scha_resource_close() が呼び出されるまで、scha_resource_get() の返す情報用に割り当てられたメモリーは維持されます。同じ

handle と tag を指定して scha_resource_get() の呼び出しを繰り返すと、新規にメモリーが割り当てられます。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されたりすることはありません。

scha_resource_close() 関数は、以前の *scha_resource_open* の呼び出しで返された handle() 引数を取ります。これは、ハンドルを使用して得た scha_resource_get() 呼び出しの戻り値に割り当てられたメモリーを解放するとともに、このハンドルを無効化します。

scha_tags.h に定義されたマクロは、*scha_resource_get* の tag() 引数に使用される場合があります。

ここでは出力引数および追加引数の型を説明します。構造体と enum 型は、1047 ページのscha_calls(3HA) で説明されています。

scha_resource_open_zone() 関数および scha_resource_get_zone() 関数はそれぞれ、scha_resource_open() および scha_resource_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_resource_open_zone() または scha_resource_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_resource_open() または scha_resource_get() と等しくなります)。

scha_resource_open_zone() から返されたハンドルを閉じるには、scha_resource_close() を使用します。cluster 引数は不要です。

tag 引数

次に、リソースプロパティーを指定するマクロを示します。リソースプロパティーの値が出力されます。SCHA_RESOURCE_STATE、SCHA_STATUS、SCHA_NUM_RG_RESTARTS、SCHA_NUM_RESOURCE_RESTARTS の各プロパティーは、コマンドが実行されるノード上の値を参照します (1287 ページのr_properties(5) を参照)。

ここでは出力引数および追加引数の型を示します。

拡張プロパティー

これらのプロパティーは、リソースタイプの Resource Type Registration (RTR) ファイル内で宣言します。リソースタイプの実装によって、これらのプロパティーを定義します。

SCHA_AFFINITY_TIMEOUT

出力引数の型は int* です。

SCHA_ALL_EXTENSIONS

出力引数の型は scha_str_array_t** です。リソースのすべての拡張プロパティーの名前を返します。

SCHA_APPLICATION_USER

出力引数の型は char** です。

SCHA_BOOT_TIMEOUT

出力引数の型は int* です。

SCHA_CHEAP_PROBE_INTERVAL

出力引数の型は int* です。

SCHA_EXTENSION

出力引数の型は scha_extprop_value_t* です。拡張プロパティーの名前を提供する、char* 型の追加の引数が必要です。ローカルノードのプロパティーのタイプとその値を返します。

明示的な値が割り当てられていないノード上でこのプロパティーの値をユーザーがリクエストすると、RTR ファイルで宣言されているデフォルト値が返されます。1251 ページのrt_reg(4) のマニュアルページを参照してください。

SCHA_EXTENSION_NODE

出力引数の型は scha_extprop_value_t* です。char* 型の追加の引数が必要です。最初の引数では、拡張プロパティーの名前を指定し、2 番目の引数ではクラスタノードを指定します。指定のノードのプロパティーのタイプとその値を返します。

明示的な値が割り当てられていないノード上でこのプロパティーの値をユーザーがリクエストすると、RTR ファイルで宣言されているデフォルト値が返されます。1251 ページのrt_reg(4) のマニュアルページを参照してください。

SCHA_FAILOVER_MODE

出力引数の型は scha_failover_mode_t* です。

SCHA_FINI_TIMEOUT

出力引数の型は int* です。

SCHA_GLOBAL_ZONE_OVERRIDE

出力引数の型は boolean_t* です。

SCHA_GROUP

出力引数の型は char** です。リソースを構成するリソースグループの名前を返します。

SCHA_INIT_TIMEOUT

出力引数の型は int* です。

SCHA_LOAD_BALANCING_POLICY

出力引数の型は char** です。

SCHA_LOAD_BALANCING_WEIGHTS

出力引数の型は scha_str_array_t** です。

SCHA_MONITOR_CHECK_TIMEOUT

出力引数の型は int* です。

SCHA_MONITOR_START_TIMEOUT

出力引数の型は int* です。

SCHA_MONITOR_STOP_TIMEOUT

出力引数の型は int* です。

SCHA_MONITORED_SWITCH

出力引数の型は scha_switch_t* です。戻り値は、ローカルノードでリソースがモニターされるかどうかを示します。

SCHA_MONITORED_SWITCH_NODE

出力引数の型は scha_switch_t* です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードでリソースがモニターされるかどうかを示します。

SCHA_NETWORK_RESOURCES_USED

出力引数の型は scha_str_array_t** です。

SCHA_NUM_RESOURCE_RESTARTS

出力引数の型は int* です。クエリーが実行されるゾーンのこのリソースで発生した、リソースの再起動のリクエスト数を返します。詳細は、1287 ページのr_properties(5) のマニュアルページを参照してください。

SCHA_NUM_RG_RESTARTS

出力引数の型は int* です。クエリーが実行されるゾーンのこのリソースで発生した、リソースグループの再起動のリクエスト数を返します。詳細は、1287 ページのr_properties(5) のマニュアルページを参照してください。

SCHA_ON_OFF_SWITCH

出力引数の型は scha_switch_t* です。戻り値は、ローカルノードでリソースが有効かどうかを示します。

SCHA_ON_OFF_SWITCH_NODE

出力引数の型は scha_switch_t* です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードでリソースが有効かどうかを示します。

SCHA_PORT_LIST

出力引数の型は scha_str_array_t** です。

SCHA_POSTNET_STOP_TIMEOUT

出力引数の型は int* です。

SCHA_PRENET_START_TIMEOUT

出力引数の型は int* です。

SCHA_PRE_EVICT

出力引数の型は boolean_t* です。

SCHA_R_DESCRIPTION

出力引数の型は char** です。

SCHA_RESOURCE_DEPENDENCIES

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_OFFLINE_RESTART

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_OFFLINE_RESTART_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_Q

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_Q_OFFLINE_RESTART

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES_OFFLINE_RESTART タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_Q_RESTART

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES_RESTART タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_Q_WEAK

出力引数の型は scha_str_array_t** です。

SCHA_RESOURCE_DEPENDENCIES_WEAK タグの「Q」バージョンも、リソースの依存関係のために宣言されたスコープまたは修飾子があれば、それを返します。{LOCAL_NODE}、{ANY_NODE}、{FROM_RG_AFFINITIES}、および @*node* 修飾子については、Unresolved link to " r_properties(5)" のマニュアルページで説明されています。

SCHA_RESOURCE_DEPENDENCIES_RESTART

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_RESTART_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_WEAK

出力引数の型は scha_str_array_t** です。戻り値は、ローカルノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_DEPENDENCIES_WEAK_NODE

出力引数の型は scha_str_array_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。戻り値は、指定したノードで適用可能な依存関係を一覧表示します。

SCHA_RESOURCE_PROJECT_NAME

出力引数の型は char** です。

SCHA_RESOURCE_STATE

出力引数の型は scha_rsstate_t* です。ローカルノード用に、リソースの RESOURCE_STATE プロパティー値を返します。

SCHA_RESOURCE_STATE_NODE

出力引数の型は scha_rsstate_t* です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。指定のノードの、リソースの RESOURCE_STATE プロパティー値を返します。

SCHA_RETRY_COUNT

出力引数の型は int* です。

SCHA_RETRY_INTERVAL

出力引数の型は int* です。

SCHA_SCALABLE

出力引数の型は boolean_t* です。

SCHA_START_TIMEOUT

出力引数の型は int* です。

SCHA_STATUS

出力引数の型は scha_status_value_t** です。ローカルノード用に、リソースの STATUS プロパティー値を返します。

SCHA_STATUS_NODE

出力引数の型は scha_status_value_t** です。クラスタノードに名前を付ける、char * 型の追加の引数が必要です。指定のノードの、リソースの STATUS プロパティー値を返します。

SCHA_STOP_TIMEOUT

出力引数の型は int* です。

SCHA_THOROUGH_PROBE_INTERVAL

出力引数の型は int* です。

SCHA_TYPE

出力引数の型は char** です。

SCHA_TYPE_VERSION

出力引数の型は char** です。

SCHA_UDP_AFFINITY

出力引数の型は boolean_t* です。

SCHA_UPDATE_TIMEOUT
: 出力引数の型は int* です。

SCHA_VALIDATE_TIMEOUT
: 出力引数の型は int* です。

SCHA_WEAK_AFFINITY
: 出力引数の型は boolean_t* です。

次に、リソースタイププロパティーを指定するマクロを示します。出力は、リソースタイプのプロパティーの値です。リソースタイププロパティーについては、1335 ページのrt_properties(5) を参照してください。

SCHA_API_VERSION
: 出力引数の型は int* です。

SCHA_BOOT
: 出力引数の型は char** です。

SCHA_FAILOVER
: 出力引数の型は boolean_t* です。

SCHA_FINI
: 出力引数の型は char** です。

SCHA_GLOBAL_ZONE
: 出力引数の型は boolean_t* です。

SCHA_INIT
: 出力引数の型は char** です。

SCHA_INIT_NODES
: 出力引数の型は scha_initnodes_flag_t* です。

SCHA_INSTALLED_NODES
: 出力引数の型は scha_str_array_t** です。

SCHA_MONITOR_CHECK
: 出力引数の型は char** です。

SCHA_MONITOR_START
: 出力引数の型は char** です。

SCHA_MONITOR_STOP
: 出力引数の型は char** です。

SCHA_PKGLIST
: 出力引数の型は scha_str_array_t** です。

SCHA_POSTNET_STOP
: 出力引数の型は char** です。

SCHA_PRENET_START
: 出力引数の型は char** です。

SCHA_PROXY
: 出力引数の型は boolean_t* です。

SCHA_RT_BASEDIR
: 出力引数の型は char** です。

SCHA_RT_DESCRIPTION
: 出力引数の型は char** です。

SCHA_RT_SYSTEM
: 出力引数の型は boolean_t* です。

SCHA_RT_VERSION
: 出力引数の型は char** です。

SCHA_SINGLE_INSTANCE
: 出力引数の型は boolean_t* です。

SCHA_START
: 出力引数の型は char** です。

SCHA_STOP
: 出力引数の型は char** です。

SCHA_UPDATE
: 出力引数の型は char** です。

SCHA_VALIDATE
: 出力引数の型は char** です。

このリソースのタイプが GLOBAL_ZONE_OVERRIDE リソースプロパティーを宣言すると、SCHA_GLOBAL_ZONE *optag* で取得される値は GLOBAL_ZONE プロパティーの

値ではなく、GLOBAL_ZONE_OVERRIDE プロパティーの現在の値になります。詳細は、1335 ページのrt_properties(5) のマニュアルページの Global_zone プロパティー、および 1287 ページのr_properties(5) のマニュアルページの Global_zone_override プロパティーの説明を参照してください。

これらの関数は、次の戻り値を返します。

0	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_NOERR	関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

例 382 scha_resource_get() 関数の使用例

次の例では、scha_resource_get() を使用して、あるリソースの Retry_count プロパティーの値と Loglevel という拡張プロパティーの値を取得します。

```
main() {
  #include <scha.h>

  scha_err_t err;
  int retry_count_out;
  scha_extprop_value_t *loglevel_out;
  scha_resource_t handle;

/* a configured resource */
  char * resource_name = "example_R";
/* resource group containing example_R */
  char * group_name = "example_RG";

  err = scha_resource_open(resource_name, group_name, &handle);

  err = scha_resource_get(handle, SCHA_RETRY_COUNT, &retry_count_out);

/* Given extension property must be defined in resourcetype RTR file. */
  err = scha_resource_get(handle, SCHA_EXTENSION, "LogLevel", &loglevel_out);

  err = scha_resource_close(handle);

  printf("The retry count for resource %s is %d\n", resource_name,
         retry_count_out);

  printf("The log level for resource %s is %d\n", resource_name,
         loglevel_out->val.val_int);
```

```
}
```

`/usr/cluster/include/scha.h`	インクルードファイル
`/usr/cluster/lib/libscha.so`	ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	`ha-cluster/developer/api`
インタフェースの安定性	発展中

691 ページの`scha_resource_get`(1HA), 1047 ページの`scha_calls`(3HA), 1239 ページの`scha_strerror`(3HA), Unresolved link to " attributes5", 1287 ページの`r_properties`(5), 1335 ページの`rt_properties`(5), 1251 ページの`rt_reg`(4)

名前

scha_resource_setstatus, scha_resource_setstatus_zone — リソースステータスの設定関数

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l scha#include <scha.h>scha_err_t scha_resource_setstatus(
     const char *rname, const char *rgname, scha_rsstatus_t status,
     const char *status_msg);

scha_err_t scha_resource_setstatus_zone(const char *rname,
     const char *rgname, const char *zonename, scha_rsstatus_t
     status, const char *status_msg);
```

`scha_resource_setstatus()` および `scha_resource_setstatus_zone()` 関数は、Resource Group Manager (RGM) によって管理されているリソースの `Status` および `Status_msg` プロパティーを設定します。リソースのモニターは、これらの関数を使用して、モニターが感知しているリソースの状態を示します。

`scha_resource_setstatus_zone()` 関数は、`Global_zone` プロパティーが `TRUE` に設定されているリソースタイプにのみ使用します。`Global_zone` プロパティーが `FALSE` に設定されている場合、この関数は必要ありません。詳細は、1335 ページの`rt_properties(5)` のマニュアルページを参照してください。

rname 引数には、ステータスを設定するリソースの名前を指定します。

rgname 引数には、該当するリソースが属するリソースグループの名前を指定します。

zonename 引数は、リソースグループが実行するよう構成されているゾーンクラスタの名前を指定します。`Global_zone` プロパティーが `TRUE` に設定されている場合、リソースを含むリソースグループがゾーンクラスタで動作しているときでも、メソッドは大域ゾーンで実行されます。

status 引数には、`scha_rsstatus_t` 型の `scha_rsstatus_t` 値を指定します。これに該当するの
は、`SCHA_RSSTATUS_OK`、`SCHA_RSSTATUS_OFFLINE`、`SCHA_RSSTATUS_FAULTED`、`SCHA_RSSTATUS_DEGRADED`、`SCHA_RSSTA`
です。

status-msg 引数は、`Status_msg` プロパティーの新しい値です。*status-msg* 引数は `NULL` にすることができます。

`scha_resource_setstatus()` または `scha_resource_setstatus_zone()` 関数の呼び出しに成功すると、リソースの `Status` および `Status_msg` プロパティーが、指定した値で更新されま

す。リソースステータスの更新内容は、クラスタシステムログに記録され、クラスタ管理ツールによって管理できます。

`scha_resource_setstatus()` および `scha_resource_setstatus_zone()` 関数は次の値を返します:

`0`	関数の実行に成功。
0 以外	関数の実行に失敗。

`SCHA_ERR_NOERR`	関数の実行に成功。

その他のエラーコードについては、1047 ページの`scha_calls`(3HA) を参照してください。

例 383 `scha_resource_setstatus()` 関数の使用

```
#include <scha.h>

scha_err_t err_code;
const char *rname = "example_R";
const char *rgname = "example_RG";

err_code = scha_resource_setstatus(rname, rgname,
           SCHA_RSSTATUS_OK, "No problems");
```

`/usr/cluster/include/scha.h`	インクルードファイル
`/usr/cluster/lib/libscha.so`	ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

699 ページの`scha_resource_setstatus`(1HA), 1047 ページの`scha_calls`(3HA), 1239 ページの`scha_strerror`(3HA), Unresolved link to " attributes5", 1335 ページの`rt_properties`(5)

名前

scha_resource_setstatus, scha_resource_setstatus_zone — リソースステータスの設定関数

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib
     -l scha#include <scha.h>scha_err_t scha_resource_setstatus(
     const char *rname, const char *rgname, scha_rsstatus_t status,
     const char *status_msg);

scha_err_t scha_resource_setstatus_zone(const char *rname,
     const char *rgname, const char *zonename, scha_rsstatus_t
     status, const char *status_msg);
```

`scha_resource_setstatus()` および `scha_resource_setstatus_zone()` 関数は、Resource Group Manager (RGM) によって管理されているリソースの `Status` および `Status_msg` プロパティーを設定します。リソースのモニターは、これらの関数を使用して、モニターが感知しているリソースの状態を示します。

`scha_resource_setstatus_zone()` 関数は、`Global_zone` プロパティーが `TRUE` に設定されているリソースタイプにのみ使用します。`Global_zone` プロパティーが `FALSE` に設定されている場合、この関数は必要ありません。詳細は、1335 ページの`rt_properties(5)` のマニュアルページを参照してください。

rname 引数には、ステータスを設定するリソースの名前を指定します。

rgname 引数には、該当するリソースが属するリソースグループの名前を指定します。

zonename 引数は、リソースグループが実行するよう構成されているゾーンクラスタの名前を指定します。`Global_zone` プロパティーが `TRUE` に設定されている場合、リソースを含むリソースグループがゾーンクラスタで動作しているときでも、メソッドは大域ゾーンで実行されます。

status 引数には、`scha_rsstatus_t` 型の `scha_rsstatus_t` 値を指定します。これに該当するの
は、`SCHA_RSSTATUS_OK`、`SCHA_RSSTATUS_OFFLINE`、`SCHA_RSSTATUS_FAULTED`、`SCHA_RSSTATUS_DEGRADED`、`SCHA_RSSTA`
です。

status-msg 引数は、`Status_msg` プロパティーの新しい値です。*status-msg* 引数は `NULL` にすることができます。

`scha_resource_setstatus()` または `scha_resource_setstatus_zone()` 関数の呼び出しに成功すると、リソースの `Status` および `Status_msg` プロパティーが、指定した値で更新されま

す。リソースステータスの更新内容は、クラスタシステムログに記録され、クラスタ管理ツールによって管理できます。

`scha_resource_setstatus()` および `scha_resource_setstatus_zone()` 関数は次の値を返します:

`0`	関数の実行に成功。
0 以外	関数の実行に失敗。

`SCHA_ERR_NOERR`	関数の実行に成功。

その他のエラーコードについては、1047 ページの`scha_calls`(3HA) を参照してください。

例 384 `scha_resource_setstatus()` 関数の使用

```
#include <scha.h>

scha_err_t err_code;
const char *rname = "example_R";
const char *rgname = "example_RG";

err_code = scha_resource_setstatus(rname, rgname,
           SCHA_RSSTATUS_OK, "No problems");
```

`/usr/cluster/include/scha.h`	インクルードファイル
`/usr/cluster/lib/libscha.so`	ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

699 ページの`scha_resource_setstatus`(1HA), 1047 ページの`scha_calls`(3HA), 1239 ページの`scha_strerror`(3HA), Unresolved link to " attributes5", 1335 ページの`rt_properties`(5)

名前

scha_resourcegroup_open, scha_resourcegroup_open_zone, scha_resourcegroup_get, scha_resourcegroup_get_zone, scha_resourcegroup_close — リソース情報アクセス関数

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l scha#include
 <scha.h>scha_err_t scha_resourcegroup_open( const char *rgname, scha_resourcegroup_t
 *handle);

scha_err_t scha_resourcegroup_open_zone( const char *cluster, const char
 *rg_name, scha_resourcegroup_t * handlep);

scha_err_t scha_resourcegroup_close(scha_resourcegroup_t handle);

scha_err_t scha_resourcegroup_get(scha_resourcegroup_t handle, const char *tag...);

scha_err_t scha_resourcegroup_get_zone(const char *cluster,
     scha_resourcegroup_t handlep, const char *rg_tag, ...);
```

`scha_resourcegroup_open()`、`scha_resourcegroup_get()`、`scha_resourcegroup_close()` の 3 つの関数を同時に使用することで Resource Group Manager (RGM) クラスタ機能の管理するリソースグループに関する情報を入手できます。

`scha_resourcegroup_open()` は、リソースグループへのアクセスを初期し、`scha_resourcegroup_get()` が使用するためのアクセスハンドルを返します。

rgname 引数には、アクセスするリソースグループの名前を指定します。

handle 引数は、関数が返す値を格納する変数のアドレスです。

`scha_resourcegroup_get()` 関数は、*tag* 引数に指定されるリソースグループの情報にアクセスします。*tag* は、`scha_tags.h` ヘッダーファイルのマクロで定義される文字列値であるはずです。タグ以降の引数は、*tag* の値に依存します。タグ以降に追加する引数は、情報を取り出すクラスタノードを指定する際に必要です。

引数リストの最後の引数は、*tag* で指定される情報の格納に適した変数型にする必要があります。このパラメータは出力引数で、取得したリソースグループの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。`scha_resourcegroup_get()` から返される情報を格納するために割り当てられたメモリーは、`scha_resourcegroup_get()` に使用されたハンドルで `scha_resourcegroup_close()` が呼び出されるまで、そのまま残ります。

scha_resourcegroup_close() は、以前の scha_resourcegroup_open() の呼び出しから返された *handle* 引数を取ります。それは、このハンドルを無効にして、このハンドルで行われた scha_resourcegroup_get() 呼び出しの戻り値に割り当てられているメモリーを解放します。値を返す必要が生じるごとに、個々の get 呼び出しでメモリーが割り当てられる点に注意してください。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されたりすることはありません。

scha_resourcegroup_open_zone() 関数および scha_resourcegroup_get_zone() 関数はそれぞれ、scha_resourcegroup_open() および scha_resourcegroup_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_resourcegroup_open_zone() または scha_resourcegroup_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_resourcegroup_open() または scha_resourcegroup_get() と等しくなります)。

scha_resourcegroup_open_zone() から返されたハンドルを閉じるには、scha_resourcegroup_close() を使用します。cluster 引数は不要です。

tag 引数に使用できるマクロ

scha_tags.h に定義されている次のマクロを scha_resourcegroup_get() 関数の *tag* 引数として使用できます。これらのマクロはリソースグループプロパティーに名前を付けます。リソースグループのプロパティー値が生成されます。RG_STATE プロパティーは、関数を呼び出したノード上の値を示します。

ここでは出力引数および追加引数の型を説明します。構造体と enum 型については 1047 ページのscha_calls(3HA) のマニュアルページで説明されています。

SCHA_ALL_LOAD_FACTORS
: 出力引数の型は scha_str_array_t** です。

SCHA_ALL_LOAD_FACTOR_NAMES
: 出力引数の型は scha_str_array_t** です。

SCHA_DESIRED_PRIMARIES
: 出力引数の型は int* です。

SCHA_FAILBACK

出力引数の型は boolean_t* です。

SCHA_LOAD_FACTOR

出力引数の型は int* です。

SCHA_GLOBAL_RESOURCES_USED

出力引数の型は scha_str_array_t** です。

SCHA_IMPL_NET_DEPEND

出力引数の型は boolean_t* です。

SCHA_MAXIMUM_PRIMARIES

出力引数の型は int* です。

SCHA_NODELIST

出力引数の型は scha_str_array_t** です。

SCHA_PATHPREFIX

出力引数の型は char** です。

SCHA_PINGPONG_INTERVAL

出力引数の型は int* です。

SCHA_PREEMPTION_MODE

出力引数の型は scha_rg_preemption_mode_t* です。

SCHA_PRIORITY

出力引数の型は int* です。

SCHA_RESOURCE_LIST

出力引数の型は scha_str_array_t** です。

SCHA_RG_AFFINITIES

出力引数の型は char** です。

SCHA_RG_AUTO_START

出力引数の型は boolean_t* です。

SCHA_RG_DEPENDENCIES

出力引数の型は scha_str_array_t** です。

SCHA_RG_DESCRIPTION

出力引数の型は char** です。

SCHA_RG_IS_FROZEN
: 出力引数の型は boolean_t* です。

SCHA_RG_MODE
: 出力引数の型は scha_rgmode_t* です。

SCHA_RG_PROJECT_NAME
: 出力引数の型は char** です。

SCHA_RG_SLM_CPU
: 出力引数の型は char** です。

SCHA_RG_SLM_CPU_MIN
: 出力引数の型は char** です。

SCHA_RG_SLM_PSET_TYPE
: 出力引数の型は char** です。

SCHA_RG_SLM_TYPE
: 出力引数の型は char** です。

SCHA_RG_STATE
: 出力引数の型は scha_rgstate_t* です。

SCHA_RG_STATE_NODE
: 出力引数の型は scha_rgstate_t* です。追加引数の型は char* です。追加引数はクラスタノードを指定し、そのノード上のリソースグループの状態を返します。

SCHA_RG_SUSP_AUTO_RECOVERY
: 出力引数の型は boolean_t* です。

SCHA_RG_SYSTEM
: 出力引数の型は boolean_t* です。

SCHA_TARGET_NODES
: 出力引数の型は scha_str_array_t** です。

これらの関数は、次の戻り値を返します。

0	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_NOERR　　関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

例 385　　scha_resourcegroup_get() 関数の使用例

次の例では、scha_resourcegroup_get() を用いて、example_RG 内のリソースリストを取得します。

```
main() {
   #include <scha.h>

   scha_err_t err;
   scha_str_array_t *resource_list;
   scha_resourcegroup_t handle;
   int ix;

   char * rgname = "example_RG";

   err = scha_resourcegroup_open(rgname, &handle);

   err = scha_resourcegroup_get(handle, SCHA_RESOURCE_LIST, \
         &resource_list);

   if (err == SCHA_ERR_NOERR) {
    for (ix = 0; ix < resource_list->array_cnt; ix++) {
        printf("Group: %s contains resource %s\n", rgname,
                resource_list->str_array[ix]);
        }
    }

/* resource_list memory freed */
 err = scha_resourcegroup_close(handle);
}
```

/usr/cluster/include/scha.h　　インクルードファイル

/usr/cluster/lib/libscha.so　　ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

185 ページの`clnode`(1CL), 703 ページの`scha_resourcegroup_get`(1HA) , 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5"

名前

scha_resourcegroup_open, scha_resourcegroup_open_zone, scha_resourcegroup_get, scha_resourcegroup_get_zone, scha_resourcegroup_close — リソース情報アクセス関数

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l scha#include
 <scha.h>scha_err_t scha_resourcegroup_open( const char *rgname, scha_resourcegroup_t
 *handle);

scha_err_t scha_resourcegroup_open_zone( const char *cluster, const char
 *rg_name, scha_resourcegroup_t * handlep);

scha_err_t scha_resourcegroup_close(scha_resourcegroup_t handle);

scha_err_t scha_resourcegroup_get(scha_resourcegroup_t handle, const char *tag...);

scha_err_t scha_resourcegroup_get_zone(const char *cluster,
     scha_resourcegroup_t handlep, const char *rg_tag, ...);
```

`scha_resourcegroup_open()`、`scha_resourcegroup_get()`、`scha_resourcegroup_close()` の 3 つの関数を同時に使用することで Resource Group Manager (RGM) クラスタ機能の管理するリソースグループに関する情報を入手できます。

`scha_resourcegroup_open()` は、リソースグループへのアクセスを初期し、`scha_resourcegroup_get()` が使用するためのアクセスハンドルを返します。

rgname 引数には、アクセスするリソースグループの名前を指定します。

handle 引数は、関数が返す値を格納する変数のアドレスです。

`scha_resourcegroup_get()` 関数は、*tag* 引数に指定されるリソースグループの情報にアクセスします。*tag* は、`scha_tags.h` ヘッダーファイルのマクロで定義される文字列値であるはずです。タグ以降の引数は、*tag* の値に依存します。タグ以降に追加する引数は、情報を取り出すクラスタノードを指定する際に必要です。

引数リストの最後の引数は、*tag* で指定される情報の格納に適した変数型にする必要があります。このパラメータは出力引数で、取得したリソースグループの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。`scha_resourcegroup_get()` から返される情報を格納するために割り当てられたメモリーは、`scha_resourcegroup_get()` に使用されたハンドルで `scha_resourcegroup_close()` が呼び出されるまで、そのまま残ります。

scha_resourcegroup_close() は、以前の scha_resourcegroup_open() の呼び出しから返された *handle* 引数を取ります。それは、このハンドルを無効にして、このハンドルで行われた scha_resourcegroup_get() 呼び出しの戻り値に割り当てられているメモリーを解放します。値を返す必要が生じるごとに、個々の get 呼び出しでメモリーが割り当てられる点に注意してください。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されたりすることはありません。

scha_resourcegroup_open_zone() 関数および scha_resourcegroup_get_zone() 関数はそれぞれ、scha_resourcegroup_open() および scha_resourcegroup_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_resourcegroup_open_zone() または scha_resourcegroup_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_resourcegroup_open() または scha_resourcegroup_get() と等しくなります)。

scha_resourcegroup_open_zone() から返されたハンドルを閉じるには、scha_resourcegroup_close() を使用します。cluster 引数は不要です。

tag 引数に使用できるマクロ

scha_tags.h に定義されている次のマクロを scha_resourcegroup_get() 関数の *tag* 引数として使用できます。これらのマクロはリソースグループプロパティーに名前を付けます。リソースグループのプロパティー値が生成されます。RG_STATE プロパティーは、関数を呼び出したノード上の値を示します。

ここでは出力引数および追加引数の型を説明します。構造体と enum 型については 1047 ページのscha_calls(3HA) のマニュアルページで説明されています。

SCHA_ALL_LOAD_FACTORS
: 出力引数の型は scha_str_array_t** です。

SCHA_ALL_LOAD_FACTOR_NAMES
: 出力引数の型は scha_str_array_t** です。

SCHA_DESIRED_PRIMARIES
: 出力引数の型は int* です。

SCHA_FAILBACK

出力引数の型は boolean_t* です。

SCHA_LOAD_FACTOR

出力引数の型は int* です。

SCHA_GLOBAL_RESOURCES_USED

出力引数の型は scha_str_array_t** です。

SCHA_IMPL_NET_DEPEND

出力引数の型は boolean_t* です。

SCHA_MAXIMUM_PRIMARIES

出力引数の型は int* です。

SCHA_NODELIST

出力引数の型は scha_str_array_t** です。

SCHA_PATHPREFIX

出力引数の型は char** です。

SCHA_PINGPONG_INTERVAL

出力引数の型は int* です。

SCHA_PREEMPTION_MODE

出力引数の型は scha_rg_preemption_mode_t* です。

SCHA_PRIORITY

出力引数の型は int* です。

SCHA_RESOURCE_LIST

出力引数の型は scha_str_array_t** です。

SCHA_RG_AFFINITIES

出力引数の型は char** です。

SCHA_RG_AUTO_START

出力引数の型は boolean_t* です。

SCHA_RG_DEPENDENCIES

出力引数の型は scha_str_array_t** です。

SCHA_RG_DESCRIPTION

出力引数の型は char** です。

SCHA_RG_IS_FROZEN
: 出力引数の型は boolean_t* です。

SCHA_RG_MODE
: 出力引数の型は scha_rgmode_t* です。

SCHA_RG_PROJECT_NAME
: 出力引数の型は char** です。

SCHA_RG_SLM_CPU
: 出力引数の型は char** です。

SCHA_RG_SLM_CPU_MIN
: 出力引数の型は char** です。

SCHA_RG_SLM_PSET_TYPE
: 出力引数の型は char** です。

SCHA_RG_SLM_TYPE
: 出力引数の型は char** です。

SCHA_RG_STATE
: 出力引数の型は scha_rgstate_t* です。

SCHA_RG_STATE_NODE
: 出力引数の型は scha_rgstate_t* です。追加引数の型は char* です。追加引数はクラスタノードを指定し、そのノード上のリソースグループの状態を返します。

SCHA_RG_SUSP_AUTO_RECOVERY
: 出力引数の型は boolean_t* です。

SCHA_RG_SYSTEM
: 出力引数の型は boolean_t* です。

SCHA_TARGET_NODES
: 出力引数の型は scha_str_array_t** です。

これらの関数は、次の戻り値を返します。

0	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_NOERR　　関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

例 386　　scha_resourcegroup_get() 関数の使用例

次の例では、scha_resourcegroup_get() を用いて、example_RG 内のリソースリストを取得します。

```
main() {
   #include <scha.h>

   scha_err_t err;
   scha_str_array_t *resource_list;
   scha_resourcegroup_t handle;
   int ix;

   char * rgname = "example_RG";

   err = scha_resourcegroup_open(rgname, &handle);

   err = scha_resourcegroup_get(handle, SCHA_RESOURCE_LIST, \
         &resource_list);

   if (err == SCHA_ERR_NOERR) {
    for (ix = 0; ix < resource_list->array_cnt; ix++) {
        printf("Group: %s contains resource %s\n", rgname,
                resource_list->str_array[ix]);
        }
    }

/* resource_list memory freed */
 err = scha_resourcegroup_close(handle);
}
```

/usr/cluster/include/scha.h　　インクルードファイル

/usr/cluster/lib/libscha.so　　ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

185 ページの`clnode`(1CL), 703 ページの`scha_resourcegroup_get`(1HA) , 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5"

名前

scha_resourcegroup_open, scha_resourcegroup_open_zone, scha_resourcegroup_get, scha_resourcegroup_get_zone, scha_resourcegroup_close — リソース情報アクセス関数

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l scha#include
 <scha.h>scha_err_t scha_resourcegroup_open( const char *rgname, scha_resourcegroup_t
 *handle);

scha_err_t scha_resourcegroup_open_zone( const char *cluster, const char
 *rg_name, scha_resourcegroup_t * handlep);

scha_err_t scha_resourcegroup_close(scha_resourcegroup_t handle);

scha_err_t scha_resourcegroup_get(scha_resourcegroup_t handle, const char *tag...);

scha_err_t scha_resourcegroup_get_zone(const char *cluster,
     scha_resourcegroup_t handlep, const char *rg_tag, ...);
```

`scha_resourcegroup_open()`、`scha_resourcegroup_get()`、`scha_resourcegroup_close()` の 3 つの関数を同時に使用することで Resource Group Manager (RGM) クラスタ機能の管理するリソースグループに関する情報を入手できます。

`scha_resourcegroup_open()` は、リソースグループへのアクセスを初期し、`scha_resourcegroup_get()` が使用するためのアクセスハンドルを返します。

rgname 引数には、アクセスするリソースグループの名前を指定します。

handle 引数は、関数が返す値を格納する変数のアドレスです。

`scha_resourcegroup_get()` 関数は、*tag* 引数に指定されるリソースグループの情報にアクセスします。*tag* は、`scha_tags.h` ヘッダーファイルのマクロで定義される文字列値であるはずです。タグ以降の引数は、*tag* の値に依存します。タグ以降に追加する引数は、情報を取り出すクラスタノードを指定する際に必要です。

引数リストの最後の引数は、*tag* で指定される情報の格納に適した変数型にする必要があります。このパラメータは出力引数で、取得したリソースグループの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。`scha_resourcegroup_get()` から返される情報を格納するために割り当てられたメモリーは、`scha_resourcegroup_get()` に使用されたハンドルで `scha_resourcegroup_close()` が呼び出されるまで、そのまま残ります。

scha_resourcegroup_close() は、以前の scha_resourcegroup_open() の呼び出しから返された *handle* 引数を取ります。それは、このハンドルを無効にして、このハンドルで行われた scha_resourcegroup_get() 呼び出しの戻り値に割り当てられているメモリーを解放します。値を返す必要が生じるごとに、個々の get 呼び出しでメモリーが割り当てられる点に注意してください。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されたりすることはありません。

scha_resourcegroup_open_zone() 関数および scha_resourcegroup_get_zone() 関数はそれぞれ、scha_resourcegroup_open() および scha_resourcegroup_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_resourcegroup_open_zone() または scha_resourcegroup_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_resourcegroup_open() または scha_resourcegroup_get() と等しくなります)。

scha_resourcegroup_open_zone() から返されたハンドルを閉じるには、scha_resourcegroup_close() を使用します。cluster 引数は不要です。

tag 引数に使用できるマクロ

scha_tags.h に定義されている次のマクロを scha_resourcegroup_get() 関数の *tag* 引数として使用できます。これらのマクロはリソースグループプロパティーに名前を付けます。リソースグループのプロパティー値が生成されます。RG_STATE プロパティーは、関数を呼び出したノード上の値を示します。

ここでは出力引数および追加引数の型を説明します。構造体と enum 型については 1047 ページのscha_calls(3HA) のマニュアルページで説明されています。

SCHA_ALL_LOAD_FACTORS
: 出力引数の型は scha_str_array_t** です。

SCHA_ALL_LOAD_FACTOR_NAMES
: 出力引数の型は scha_str_array_t** です。

SCHA_DESIRED_PRIMARIES
: 出力引数の型は int* です。

SCHA_FAILBACK

出力引数の型は boolean_t* です。

SCHA_LOAD_FACTOR

出力引数の型は int* です。

SCHA_GLOBAL_RESOURCES_USED

出力引数の型は scha_str_array_t** です。

SCHA_IMPL_NET_DEPEND

出力引数の型は boolean_t* です。

SCHA_MAXIMUM_PRIMARIES

出力引数の型は int* です。

SCHA_NODELIST

出力引数の型は scha_str_array_t** です。

SCHA_PATHPREFIX

出力引数の型は char** です。

SCHA_PINGPONG_INTERVAL

出力引数の型は int* です。

SCHA_PREEMPTION_MODE

出力引数の型は scha_rg_preemption_mode_t* です。

SCHA_PRIORITY

出力引数の型は int* です。

SCHA_RESOURCE_LIST

出力引数の型は scha_str_array_t** です。

SCHA_RG_AFFINITIES

出力引数の型は char** です。

SCHA_RG_AUTO_START

出力引数の型は boolean_t* です。

SCHA_RG_DEPENDENCIES

出力引数の型は scha_str_array_t** です。

SCHA_RG_DESCRIPTION

出力引数の型は char** です。

SCHA_RG_IS_FROZEN
: 出力引数の型は boolean_t* です。

SCHA_RG_MODE
: 出力引数の型は scha_rgmode_t* です。

SCHA_RG_PROJECT_NAME
: 出力引数の型は char** です。

SCHA_RG_SLM_CPU
: 出力引数の型は char** です。

SCHA_RG_SLM_CPU_MIN
: 出力引数の型は char** です。

SCHA_RG_SLM_PSET_TYPE
: 出力引数の型は char** です。

SCHA_RG_SLM_TYPE
: 出力引数の型は char** です。

SCHA_RG_STATE
: 出力引数の型は scha_rgstate_t* です。

SCHA_RG_STATE_NODE
: 出力引数の型は scha_rgstate_t* です。追加引数の型は char* です。追加引数はクラスタノードを指定し、そのノード上のリソースグループの状態を返します。

SCHA_RG_SUSP_AUTO_RECOVERY
: 出力引数の型は boolean_t* です。

SCHA_RG_SYSTEM
: 出力引数の型は boolean_t* です。

SCHA_TARGET_NODES
: 出力引数の型は scha_str_array_t** です。

これらの関数は、次の戻り値を返します。

0	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_NOERR　　関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

例 387　　scha_resourcegroup_get() 関数の使用例

次の例では、scha_resourcegroup_get() を用いて、example_RG 内のリソースリストを取得します。

```
main() {
   #include <scha.h>

   scha_err_t err;
   scha_str_array_t *resource_list;
   scha_resourcegroup_t handle;
   int ix;

   char * rgname = "example_RG";

   err = scha_resourcegroup_open(rgname, &handle);

   err = scha_resourcegroup_get(handle, SCHA_RESOURCE_LIST, \
         &resource_list);

   if (err == SCHA_ERR_NOERR) {
    for (ix = 0; ix < resource_list->array_cnt; ix++) {
        printf("Group: %s contains resource %s\n", rgname,
                resource_list->str_array[ix]);
        }
    }

/* resource_list memory freed */
 err = scha_resourcegroup_close(handle);
}
```

/usr/cluster/include/scha.h　　インクルードファイル

/usr/cluster/lib/libscha.so　　ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

185 ページの`clnode`(1CL), 703 ページの`scha_resourcegroup_get`(1HA) , 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5"

名前

scha_resourcegroup_open, scha_resourcegroup_open_zone, scha_resourcegroup_get, scha_resourcegroup_get_zone, scha_resourcegroup_close — リソース情報アクセス関数

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l scha#include
 <scha.h>scha_err_t scha_resourcegroup_open( const char *rgname, scha_resourcegroup_t
 *handle);

scha_err_t scha_resourcegroup_open_zone( const char *cluster, const char
 *rg_name, scha_resourcegroup_t * handlep);

scha_err_t scha_resourcegroup_close(scha_resourcegroup_t handle);

scha_err_t scha_resourcegroup_get(scha_resourcegroup_t handle, const char *tag...);

scha_err_t scha_resourcegroup_get_zone(const char *cluster,
     scha_resourcegroup_t handlep, const char *rg_tag, ...);
```

`scha_resourcegroup_open()`、`scha_resourcegroup_get()`、`scha_resourcegroup_close()` の 3 つの関数を同時に使用することで Resource Group Manager (RGM) クラスタ機能の管理するリソースグループに関する情報を入手できます。

`scha_resourcegroup_open()` は、リソースグループへのアクセスを初期し、`scha_resourcegroup_get()` が使用するためのアクセスハンドルを返します。

rgname 引数には、アクセスするリソースグループの名前を指定します。

handle 引数は、関数が返す値を格納する変数のアドレスです。

`scha_resourcegroup_get()` 関数は、*tag* 引数に指定されるリソースグループの情報にアクセスします。*tag* は、`scha_tags.h` ヘッダーファイルのマクロで定義される文字列値であるはずです。タグ以降の引数は、*tag* の値に依存します。タグ以降に追加する引数は、情報を取り出すクラスタノードを指定する際に必要です。

引数リストの最後の引数は、*tag* で指定される情報の格納に適した変数型にする必要があります。このパラメータは出力引数で、取得したリソースグループの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。`scha_resourcegroup_get()` から返される情報を格納するために割り当てられたメモリーは、`scha_resourcegroup_get()` に使用されたハンドルで `scha_resourcegroup_close()` が呼び出されるまで、そのまま残ります。

scha_resourcegroup_close() は、以前の scha_resourcegroup_open() の呼び出しから返された *handle* 引数を取ります。それは、このハンドルを無効にして、このハンドルで行われた scha_resourcegroup_get() 呼び出しの戻り値に割り当てられているメモリーを解放します。値を返す必要が生じるごとに、個々の get 呼び出しでメモリーが割り当てられる点に注意してください。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されたりすることはありません。

scha_resourcegroup_open_zone() 関数および scha_resourcegroup_get_zone() 関数はそれぞれ、scha_resourcegroup_open() および scha_resourcegroup_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_resourcegroup_open_zone() または scha_resourcegroup_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_resourcegroup_open() または scha_resourcegroup_get() と等しくなります)。

scha_resourcegroup_open_zone() から返されたハンドルを閉じるには、scha_resourcegroup_close() を使用します。cluster 引数は不要です。

tag 引数に使用できるマクロ

scha_tags.h に定義されている次のマクロを scha_resourcegroup_get() 関数の *tag* 引数として使用できます。これらのマクロはリソースグループプロパティーに名前を付けます。リソースグループのプロパティー値が生成されます。RG_STATE プロパティーは、関数を呼び出したノード上の値を示します。

ここでは出力引数および追加引数の型を説明します。構造体と enum 型については 1047 ページのscha_calls(3HA) のマニュアルページで説明されています。

SCHA_ALL_LOAD_FACTORS
: 出力引数の型は scha_str_array_t** です。

SCHA_ALL_LOAD_FACTOR_NAMES
: 出力引数の型は scha_str_array_t** です。

SCHA_DESIRED_PRIMARIES
: 出力引数の型は int* です。

SCHA_FAILBACK

出力引数の型は boolean_t* です。

SCHA_LOAD_FACTOR

出力引数の型は int* です。

SCHA_GLOBAL_RESOURCES_USED

出力引数の型は scha_str_array_t** です。

SCHA_IMPL_NET_DEPEND

出力引数の型は boolean_t* です。

SCHA_MAXIMUM_PRIMARIES

出力引数の型は int* です。

SCHA_NODELIST

出力引数の型は scha_str_array_t** です。

SCHA_PATHPREFIX

出力引数の型は char** です。

SCHA_PINGPONG_INTERVAL

出力引数の型は int* です。

SCHA_PREEMPTION_MODE

出力引数の型は scha_rg_preemption_mode_t* です。

SCHA_PRIORITY

出力引数の型は int* です。

SCHA_RESOURCE_LIST

出力引数の型は scha_str_array_t** です。

SCHA_RG_AFFINITIES

出力引数の型は char** です。

SCHA_RG_AUTO_START

出力引数の型は boolean_t* です。

SCHA_RG_DEPENDENCIES

出力引数の型は scha_str_array_t** です。

SCHA_RG_DESCRIPTION

出力引数の型は char** です。

SCHA_RG_IS_FROZEN
: 出力引数の型は boolean_t* です。

SCHA_RG_MODE
: 出力引数の型は scha_rgmode_t* です。

SCHA_RG_PROJECT_NAME
: 出力引数の型は char** です。

SCHA_RG_SLM_CPU
: 出力引数の型は char** です。

SCHA_RG_SLM_CPU_MIN
: 出力引数の型は char** です。

SCHA_RG_SLM_PSET_TYPE
: 出力引数の型は char** です。

SCHA_RG_SLM_TYPE
: 出力引数の型は char** です。

SCHA_RG_STATE
: 出力引数の型は scha_rgstate_t* です。

SCHA_RG_STATE_NODE
: 出力引数の型は scha_rgstate_t* です。追加引数の型は char* です。追加引数はクラスタノードを指定し、そのノード上のリソースグループの状態を返します。

SCHA_RG_SUSP_AUTO_RECOVERY
: 出力引数の型は boolean_t* です。

SCHA_RG_SYSTEM
: 出力引数の型は boolean_t* です。

SCHA_TARGET_NODES
: 出力引数の型は scha_str_array_t** です。

これらの関数は、次の戻り値を返します。

0	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_NOERR　　関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

例 388　　scha_resourcegroup_get() 関数の使用例

次の例では、scha_resourcegroup_get() を用いて、example_RG 内のリソースリストを取得します。

```
main() {
   #include <scha.h>

   scha_err_t err;
   scha_str_array_t *resource_list;
   scha_resourcegroup_t handle;
   int ix;

   char * rgname = "example_RG";

   err = scha_resourcegroup_open(rgname, &handle);

   err = scha_resourcegroup_get(handle, SCHA_RESOURCE_LIST, \
         &resource_list);

   if (err == SCHA_ERR_NOERR) {
    for (ix = 0; ix < resource_list->array_cnt; ix++) {
        printf("Group: %s contains resource %s\n", rgname,
                resource_list->str_array[ix]);
        }
    }

/* resource_list memory freed */
 err = scha_resourcegroup_close(handle);
}
```

/usr/cluster/include/scha.h　　インクルードファイル

/usr/cluster/lib/libscha.so　　ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

185 ページの`clnode`(1CL), 703 ページの`scha_resourcegroup_get`(1HA) , 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5"

名前

scha_resourcegroup_open, scha_resourcegroup_open_zone, scha_resourcegroup_get, scha_resourcegroup_get_zone, scha_resourcegroup_close — リソース情報アクセス関数

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l scha#include
 <scha.h>scha_err_t scha_resourcegroup_open( const char *rgname, scha_resourcegroup_t
 *handle);

scha_err_t scha_resourcegroup_open_zone( const char *cluster, const char
 *rg_name, scha_resourcegroup_t * handlep);

scha_err_t scha_resourcegroup_close(scha_resourcegroup_t handle);

scha_err_t scha_resourcegroup_get(scha_resourcegroup_t handle, const char *tag...);

scha_err_t scha_resourcegroup_get_zone(const char *cluster,
     scha_resourcegroup_t handlep, const char *rg_tag, ...);
```

`scha_resourcegroup_open()`、`scha_resourcegroup_get()`、`scha_resourcegroup_close()` の 3 つの関数を同時に使用することで Resource Group Manager (RGM) クラスタ機能の管理するリソースグループに関する情報を入手できます。

`scha_resourcegroup_open()` は、リソースグループへのアクセスを初期し、`scha_resourcegroup_get()` が使用するためのアクセスハンドルを返します。

rgname 引数には、アクセスするリソースグループの名前を指定します。

handle 引数は、関数が返す値を格納する変数のアドレスです。

`scha_resourcegroup_get()` 関数は、*tag* 引数に指定されるリソースグループの情報にアクセスします。*tag* は、`scha_tags.h` ヘッダーファイルのマクロで定義される文字列値であるはずです。タグ以降の引数は、*tag* の値に依存します。タグ以降に追加する引数は、情報を取り出すクラスタノードを指定する際に必要です。

引数リストの最後の引数は、*tag* で指定される情報の格納に適した変数型にする必要があります。このパラメータは出力引数で、取得したリソースグループの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。`scha_resourcegroup_get()` から返される情報を格納するために割り当てられたメモリーは、`scha_resourcegroup_get()` に使用されたハンドルで `scha_resourcegroup_close()` が呼び出されるまで、そのまま残ります。

scha_resourcegroup_close() は、以前の scha_resourcegroup_open() の呼び出しから返された *handle* 引数を取ります。それは、このハンドルを無効にして、このハンドルで行われた scha_resourcegroup_get() 呼び出しの戻り値に割り当てられているメモリーを解放します。値を返す必要が生じるごとに、個々の get 呼び出しでメモリーが割り当てられる点に注意してください。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されたりすることはありません。

scha_resourcegroup_open_zone() 関数および scha_resourcegroup_get_zone() 関数はそれぞれ、scha_resourcegroup_open() および scha_resourcegroup_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_resourcegroup_open_zone() または scha_resourcegroup_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_resourcegroup_open() または scha_resourcegroup_get() と等しくなります)。

scha_resourcegroup_open_zone() から返されたハンドルを閉じるには、scha_resourcegroup_close() を使用します。cluster 引数は不要です。

tag 引数に使用できるマクロ

scha_tags.h に定義されている次のマクロを scha_resourcegroup_get() 関数の *tag* 引数として使用できます。これらのマクロはリソースグループプロパティーに名前を付けます。リソースグループのプロパティー値が生成されます。RG_STATE プロパティーは、関数を呼び出したノード上の値を示します。

ここでは出力引数および追加引数の型を説明します。構造体と enum 型については 1047 ページのscha_calls(3HA) のマニュアルページで説明されています。

SCHA_ALL_LOAD_FACTORS
: 出力引数の型は scha_str_array_t** です。

SCHA_ALL_LOAD_FACTOR_NAMES
: 出力引数の型は scha_str_array_t** です。

SCHA_DESIRED_PRIMARIES
: 出力引数の型は int* です。

SCHA_FAILBACK

出力引数の型は boolean_t* です。

SCHA_LOAD_FACTOR

出力引数の型は int* です。

SCHA_GLOBAL_RESOURCES_USED

出力引数の型は scha_str_array_t** です。

SCHA_IMPL_NET_DEPEND

出力引数の型は boolean_t* です。

SCHA_MAXIMUM_PRIMARIES

出力引数の型は int* です。

SCHA_NODELIST

出力引数の型は scha_str_array_t** です。

SCHA_PATHPREFIX

出力引数の型は char** です。

SCHA_PINGPONG_INTERVAL

出力引数の型は int* です。

SCHA_PREEMPTION_MODE

出力引数の型は scha_rg_preemption_mode_t* です。

SCHA_PRIORITY

出力引数の型は int* です。

SCHA_RESOURCE_LIST

出力引数の型は scha_str_array_t** です。

SCHA_RG_AFFINITIES

出力引数の型は char** です。

SCHA_RG_AUTO_START

出力引数の型は boolean_t* です。

SCHA_RG_DEPENDENCIES

出力引数の型は scha_str_array_t** です。

SCHA_RG_DESCRIPTION

出力引数の型は char** です。

SCHA_RG_IS_FROZEN
: 出力引数の型は boolean_t* です。

SCHA_RG_MODE
: 出力引数の型は scha_rgmode_t* です。

SCHA_RG_PROJECT_NAME
: 出力引数の型は char** です。

SCHA_RG_SLM_CPU
: 出力引数の型は char** です。

SCHA_RG_SLM_CPU_MIN
: 出力引数の型は char** です。

SCHA_RG_SLM_PSET_TYPE
: 出力引数の型は char** です。

SCHA_RG_SLM_TYPE
: 出力引数の型は char** です。

SCHA_RG_STATE
: 出力引数の型は scha_rgstate_t* です。

SCHA_RG_STATE_NODE
: 出力引数の型は scha_rgstate_t* です。追加引数の型は char* です。追加引数はクラスタノードを指定し、そのノード上のリソースグループの状態を返します。

SCHA_RG_SUSP_AUTO_RECOVERY
: 出力引数の型は boolean_t* です。

SCHA_RG_SYSTEM
: 出力引数の型は boolean_t* です。

SCHA_TARGET_NODES
: 出力引数の型は scha_str_array_t** です。

これらの関数は、次の戻り値を返します。

0	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_NOERR　　関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) を参照してください。

例 389　　scha_resourcegroup_get() 関数の使用例

次の例では、scha_resourcegroup_get() を用いて、example_RG 内のリソースリストを取得します。

```
main() {
   #include <scha.h>

   scha_err_t err;
   scha_str_array_t *resource_list;
   scha_resourcegroup_t handle;
   int ix;

   char * rgname = "example_RG";

   err = scha_resourcegroup_open(rgname, &handle);

   err = scha_resourcegroup_get(handle, SCHA_RESOURCE_LIST, \
         &resource_list);

   if (err == SCHA_ERR_NOERR) {
    for (ix = 0; ix < resource_list->array_cnt; ix++) {
        printf("Group: %s contains resource %s\n", rgname,
                resource_list->str_array[ix]);
        }
    }

/* resource_list memory freed */
 err = scha_resourcegroup_close(handle);
}
```

/usr/cluster/include/scha.h　　インクルードファイル

/usr/cluster/lib/libscha.so　　ライブラリ

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

185 ページの`clnode`(1CL), 703 ページの`scha_resourcegroup_get`(1HA) , 1047 ページの`scha_calls`(3HA), Unresolved link to " attributes5"

名前

scha_resourcetype_open, scha_resourcetype_open_zone, scha_resourcetype_get, scha_resourcetype_get_zone, scha_resourcetype_close — リソースタイプ情報アクセス関数。

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l scha#include
 <scha.h>scha_err_t scha_resourcetype_open( const char *rtname, scha_resourcetype_t *handle);

scha_err_t scha_resourcetype_open_zone(const char *cluster, const char
 *rt_name, scha_resourcetype_t *handlep);

scha_err_t scha_resourcetype_close(scha_resourcetype_t handle);

scha_err_t scha_resourcetype_get(scha_resourcetype_t handle, const char *tag...);

scha_err_t scha_resourcetype_get_zone(const char *cluster,
     scha_resourcetype_t handlep,const char *rt_tag, ...);
```

`scha_resourcetype_open()`、`scha_resourcetype_get ()`、および`scha_resourcetype_close()` 関数を使用すると、Resource Group Manager (RGM) クラスタ機能が利用するリソースタイプの情報を入手できます。

`scha_resourcetype_open()` は、リソースタイプへのアクセスを初期化し、`scha_resourcetype_get()` が使用するアクセスハンドルを返します。

`scha_resourcetype_open()` の *rtname* 引数には、アクセスするリソースタイプの名前を指定します。

handle 引数の値は、関数の戻り値を格納する変数のアドレスとなります。

`scha_resourcetype_get()` 関数は、*tag* 引数に指定されるリソースタイプの情報にアクセスします。*tag* 引数には、`scha_tags.h` ヘッダーファイルのマクロで定義される文字列値が入ります。タグ以降の引数は、*tag* の値に依存します。

タグ以降に追加する引数は、情報を取り出すクラスタノードや、タグ固有の他の情報を指定する際に必要となることがあります。引数リストの最後の引数は、*tag* で指定される情報の格納に適した型にする必要があります。これは out 引数で、リソースタイプの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。`scha_resourcetype_get()` で使用されたハンドルで `scha_resourcetype_close()` が呼び出されるまで、`scha_resourcetype_get()` の返す情報用に割り当てられたメモリーは維持されます。

scha_resourcetype_close() には、事前に *scha_resourcetype_open* 関数を使って得た handle() 引数の値を指定します。この関数は、該当するハンドルを使用して得た scha_resourcegroup_get() の戻り値用割り当てメモリーを解放するとともに、このハンドルを無効化します。値を返す必要が生じると、get 呼び出しごとにメモリー割り当てが発生する点に注意してください。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されることはありません。

scha_tags.h に定義されるマクロには、*scha_resourcetype_get* の tag() 引数に使用されるものがあります。ここでは出力引数および追加引数の型を説明します。構造体と enum 型は、1047 ページのscha_calls(3HA) で説明されています。

scha_resourcetype_open_zone() 関数および scha_resourcetype_get_zone() 関数はそれぞれ、scha_resourcetype_open() および scha_resourcetype_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_resourcetype_open_zone() または scha_resourcetype_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_resourcetype_open() または scha_resourcetype_get() と等しくなります)。

scha_resourcetype_open_zone() から返されたハンドルを閉じるには、scha_resourcetype_close() を使用します。cluster 引数は不要です。

optag 引数

次のマクロによって、リソースタイプのプロパティーを指定します。出力は、リソースタイプの名前付きプロパティーの値です。

注記 - *optag* 引数 (SCHA_API_VERSION や SCHA_BOOT など) には大文字と小文字の区別は*ありません*。*optag* 引数を指定するときには、大文字と小文字の任意の組み合わせを使用できます。

SCHA_API_VERSION

出力引数の型は int* です。

SCHA_BOOT

出力引数の型は char ** です。

SCHA_FAILOVER

出力引数の型は boolean_t * です。

SCHA_FINI

出力引数の型は char ** です。

SCHA_GLOBALZONE

出力引数の型は boolean_t * です。

SCHA_INIT

出力引数の型は char ** です。

SCHA_INIT_NODES

出力引数の型は scha_initnodes_flag_t * です。

SCHA_INSTALLED_NODES

出力引数の型は scha_str_array_t ** です。

SCHA_IS_LOGICAL_HOSTNAME

出力引数の型は boolean_t * です。

SCHA_IS_SHARED_ADDRESS

出力引数の型は boolean_t * です。

SCHA_MONITOR_CHECK

出力引数の型は char ** です。

SCHA_MONITOR_START

出力引数の型は char ** です。

SCHA_MONITOR_STOP

出力引数の型は char ** です。

SCHA_PER_NODE

出力引数の型は boolean_t * です。

SCHA_PKGLIST

出力引数の型は scha_str_array_t ** です。

SCHA_POSTNET_STOP

出力引数の型は char ** です。

SCHA_PRENET_START

出力引数の型は char ** です。

SCHA_PROXY
: 出力引数の型は boolean_t * です。

SCHA_RESOURCE_LIST
: 出力引数の型は scha_str_array_t ** です。

SCHA_RT_BASEDIR
: 出力引数の型は char ** です。

SCHA_RT_DESCRIPTION
: 出力引数の型は char ** です。

SCHA_RT_SYSTEM
: 出力引数の型は boolean_t * です。

SCHA_RT_VERSION
: 出力引数の型は char ** です。

SCHA_SINGLE_INSTANCE
: 出力引数の型は boolean_t * です。

SCHA_START
: 出力引数の型は char ** です。

SCHA_STOP
: 出力引数の型は char ** です。

SCHA_UPDATE
: 出力引数の型は char ** です。

SCHA_VALIDATE
: 出力引数の型は char ** です。

これらの関数は、次の戻り値を返します。

0	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_NOERR	関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) のマニュアルページを参照してください。

`/usr/cluster/include/scha.h`	インクルードファイル
`/usr/cluster/lib/libscha.so`	ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

691 ページの`scha_resource_get`(1HA), 1047 ページの`scha_calls`(3HA), 1239 ページの`scha_strerror`(3HA), 1241 ページの`scha_strerror_i18n`(3HA), Unresolved link to " attributes5", 1335 ページの`rt_properties`(5)

名前

scha_resourcetype_open, scha_resourcetype_open_zone, scha_resourcetype_get, scha_resourcetype_get_zone, scha_resourcetype_close — リソースタイプ情報アクセス関数。

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l scha#include
 <scha.h>scha_err_t scha_resourcetype_open( const char *rtname, scha_resourcetype_t *handle);

scha_err_t scha_resourcetype_open_zone(const char *cluster, const char
 *rt_name, scha_resourcetype_t *handlep);

scha_err_t scha_resourcetype_close(scha_resourcetype_t handle);

scha_err_t scha_resourcetype_get(scha_resourcetype_t handle, const char *tag...);

scha_err_t scha_resourcetype_get_zone(const char *cluster,
     scha_resourcetype_t handlep,const char *rt_tag, ...);
```

`scha_resourcetype_open()`、`scha_resourcetype_get ()`、および`scha_resourcetype_close()` 関数を使用すると、Resource Group Manager (RGM) クラスタ機能が利用するリソースタイプの情報を入手できます。

`scha_resourcetype_open()` は、リソースタイプへのアクセスを初期化し、`scha_resourcetype_get()` が使用するアクセスハンドルを返します。

`scha_resourcetype_open()` の *rtname* 引数には、アクセスするリソースタイプの名前を指定します。

handle 引数の値は、関数の戻り値を格納する変数のアドレスとなります。

`scha_resourcetype_get()` 関数は、*tag* 引数に指定されるリソースタイプの情報にアクセスします。*tag* 引数には、`scha_tags.h` ヘッダーファイルのマクロで定義される文字列値が入ります。タグ以降の引数は、*tag* の値に依存します。

タグ以降に追加する引数は、情報を取り出すクラスタノードや、タグ固有の他の情報を指定する際に必要となることがあります。引数リストの最後の引数は、*tag* で指定される情報の格納に適した型にする必要があります。これは out 引数で、リソースタイプの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。`scha_resourcetype_get()` で使用されたハンドルで `scha_resourcetype_close()` が呼び出されるまで、`scha_resourcetype_get()` の返す情報用に割り当てられたメモリーは維持されます。

scha_resourcetype_close() には、事前に *scha_resourcetype_open* 関数を使って得た handle() 引数の値を指定します。この関数は、該当するハンドルを使用して得た scha_resourcegroup_get() の戻り値用割り当てメモリーを解放するとともに、このハンドルを無効化します。値を返す必要が生じると、get 呼び出しごとにメモリー割り当てが発生する点に注意してください。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されることはありません。

scha_tags.h に定義されるマクロには、*scha_resourcetype_get* の tag() 引数に使用されるものがあります。ここでは出力引数および追加引数の型を説明します。構造体と enum 型は、1047 ページのscha_calls(3HA) で説明されています。

scha_resourcetype_open_zone() 関数および scha_resourcetype_get_zone() 関数はそれぞれ、scha_resourcetype_open() および scha_resourcetype_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_resourcetype_open_zone() または scha_resourcetype_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_resourcetype_open() または scha_resourcetype_get() と等しくなります)。

scha_resourcetype_open_zone() から返されたハンドルを閉じるには、scha_resourcetype_close() を使用します。cluster 引数は不要です。

optag 引数

次のマクロによって、リソースタイプのプロパティーを指定します。出力は、リソースタイプの名前付きプロパティーの値です。

注記 - *optag* 引数 (SCHA_API_VERSION や SCHA_BOOT など) には大文字と小文字の区別は*ありません*。*optag* 引数を指定するときには、大文字と小文字の任意の組み合わせを使用できます。

SCHA_API_VERSION

出力引数の型は int* です。

SCHA_BOOT

出力引数の型は char ** です。

SCHA_FAILOVER

出力引数の型は boolean_t * です。

SCHA_FINI

出力引数の型は char ** です。

SCHA_GLOBALZONE

出力引数の型は boolean_t * です。

SCHA_INIT

出力引数の型は char ** です。

SCHA_INIT_NODES

出力引数の型は scha_initnodes_flag_t * です。

SCHA_INSTALLED_NODES

出力引数の型は scha_str_array_t ** です。

SCHA_IS_LOGICAL_HOSTNAME

出力引数の型は boolean_t * です。

SCHA_IS_SHARED_ADDRESS

出力引数の型は boolean_t * です。

SCHA_MONITOR_CHECK

出力引数の型は char ** です。

SCHA_MONITOR_START

出力引数の型は char ** です。

SCHA_MONITOR_STOP

出力引数の型は char ** です。

SCHA_PER_NODE

出力引数の型は boolean_t * です。

SCHA_PKGLIST

出力引数の型は scha_str_array_t ** です。

SCHA_POSTNET_STOP

出力引数の型は char ** です。

SCHA_PRENET_START

出力引数の型は char ** です。

SCHA_PROXY
: 出力引数の型は boolean_t * です。

SCHA_RESOURCE_LIST
: 出力引数の型は scha_str_array_t ** です。

SCHA_RT_BASEDIR
: 出力引数の型は char ** です。

SCHA_RT_DESCRIPTION
: 出力引数の型は char ** です。

SCHA_RT_SYSTEM
: 出力引数の型は boolean_t * です。

SCHA_RT_VERSION
: 出力引数の型は char ** です。

SCHA_SINGLE_INSTANCE
: 出力引数の型は boolean_t * です。

SCHA_START
: 出力引数の型は char ** です。

SCHA_STOP
: 出力引数の型は char ** です。

SCHA_UPDATE
: 出力引数の型は char ** です。

SCHA_VALIDATE
: 出力引数の型は char ** です。

これらの関数は、次の戻り値を返します。

0	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_NOERR	関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) のマニュアルページを参照してください。

`/usr/cluster/include/scha.h`	インクルードファイル
`/usr/cluster/lib/libscha.so`	ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

691 ページの`scha_resource_get`(1HA), 1047 ページの`scha_calls`(3HA), 1239 ページの`scha_strerror`(3HA), 1241 ページの`scha_strerror_i18n`(3HA), Unresolved link to " attributes5", 1335 ページの`rt_properties`(5)

名前

scha_resourcetype_open, scha_resourcetype_open_zone, scha_resourcetype_get, scha_resourcetype_get_zone, scha_resourcetype_close — リソースタイプ情報アクセス関数。

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l scha#include
 <scha.h>scha_err_t scha_resourcetype_open( const char *rtname, scha_resourcetype_t *handle);

scha_err_t scha_resourcetype_open_zone(const char *cluster, const char
 *rt_name, scha_resourcetype_t *handlep);

scha_err_t scha_resourcetype_close(scha_resourcetype_t handle);

scha_err_t scha_resourcetype_get(scha_resourcetype_t handle, const char *tag...);

scha_err_t scha_resourcetype_get_zone(const char *cluster,
     scha_resourcetype_t handlep,const char *rt_tag, ...);
```

`scha_resourcetype_open()`、`scha_resourcetype_get ()`、および`scha_resourcetype_close()` 関数を使用すると、Resource Group Manager (RGM) クラスタ機能が利用するリソースタイプの情報を入手できます。

`scha_resourcetype_open()` は、リソースタイプへのアクセスを初期化し、`scha_resourcetype_get()` が使用するアクセスハンドルを返します。

`scha_resourcetype_open()` の *rtname* 引数には、アクセスするリソースタイプの名前を指定します。

handle 引数の値は、関数の戻り値を格納する変数のアドレスとなります。

`scha_resourcetype_get()` 関数は、*tag* 引数に指定されるリソースタイプの情報にアクセスします。*tag* 引数には、`scha_tags.h` ヘッダーファイルのマクロで定義される文字列値が入ります。タグ以降の引数は、*tag* の値に依存します。

タグ以降に追加する引数は、情報を取り出すクラスタノードや、タグ固有の他の情報を指定する際に必要となることがあります。引数リストの最後の引数は、*tag* で指定される情報の格納に適した型にする必要があります。これは out 引数で、リソースタイプの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。`scha_resourcetype_get()` で使用されたハンドルで `scha_resourcetype_close()` が呼び出されるまで、`scha_resourcetype_get()` の返す情報用に割り当てられたメモリーは維持されます。

scha_resourcetype_close() には、事前に *scha_resourcetype_open* 関数を使って得た handle() 引数の値を指定します。この関数は、該当するハンドルを使用して得た scha_resourcegroup_get() の戻り値用割り当てメモリーを解放するとともに、このハンドルを無効化します。値を返す必要が生じると、get 呼び出しごとにメモリー割り当てが発生する点に注意してください。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されることはありません。

scha_tags.h に定義されるマクロには、*scha_resourcetype_get* の tag() 引数に使用されるものがあります。ここでは出力引数および追加引数の型を説明します。構造体と enum 型は、1047 ページのscha_calls(3HA) で説明されています。

scha_resourcetype_open_zone() 関数および scha_resourcetype_get_zone() 関数はそれぞれ、scha_resourcetype_open() および scha_resourcetype_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_resourcetype_open_zone() または scha_resourcetype_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_resourcetype_open() または scha_resourcetype_get() と等しくなります)。

scha_resourcetype_open_zone() から返されたハンドルを閉じるには、scha_resourcetype_close() を使用します。cluster 引数は不要です。

optag 引数

次のマクロによって、リソースタイプのプロパティーを指定します。出力は、リソースタイプの名前付きプロパティーの値です。

注記 - *optag* 引数 (SCHA_API_VERSION や SCHA_BOOT など) には大文字と小文字の区別は*ありません*。*optag* 引数を指定するときには、大文字と小文字の任意の組み合わせを使用できます。

SCHA_API_VERSION

出力引数の型は int* です。

SCHA_BOOT

出力引数の型は char ** です。

SCHA_FAILOVER

出力引数の型は boolean_t * です。

SCHA_FINI

出力引数の型は char ** です。

SCHA_GLOBALZONE

出力引数の型は boolean_t * です。

SCHA_INIT

出力引数の型は char ** です。

SCHA_INIT_NODES

出力引数の型は scha_initnodes_flag_t * です。

SCHA_INSTALLED_NODES

出力引数の型は scha_str_array_t ** です。

SCHA_IS_LOGICAL_HOSTNAME

出力引数の型は boolean_t * です。

SCHA_IS_SHARED_ADDRESS

出力引数の型は boolean_t * です。

SCHA_MONITOR_CHECK

出力引数の型は char ** です。

SCHA_MONITOR_START

出力引数の型は char ** です。

SCHA_MONITOR_STOP

出力引数の型は char ** です。

SCHA_PER_NODE

出力引数の型は boolean_t * です。

SCHA_PKGLIST

出力引数の型は scha_str_array_t ** です。

SCHA_POSTNET_STOP

出力引数の型は char ** です。

SCHA_PRENET_START

出力引数の型は char ** です。

SCHA_PROXY
: 出力引数の型は boolean_t * です。

SCHA_RESOURCE_LIST
: 出力引数の型は scha_str_array_t ** です。

SCHA_RT_BASEDIR
: 出力引数の型は char ** です。

SCHA_RT_DESCRIPTION
: 出力引数の型は char ** です。

SCHA_RT_SYSTEM
: 出力引数の型は boolean_t * です。

SCHA_RT_VERSION
: 出力引数の型は char ** です。

SCHA_SINGLE_INSTANCE
: 出力引数の型は boolean_t * です。

SCHA_START
: 出力引数の型は char ** です。

SCHA_STOP
: 出力引数の型は char ** です。

SCHA_UPDATE
: 出力引数の型は char ** です。

SCHA_VALIDATE
: 出力引数の型は char ** です。

これらの関数は、次の戻り値を返します。

0	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_NOERR	関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) のマニュアルページを参照してください。

`/usr/cluster/include/scha.h`	インクルードファイル
`/usr/cluster/lib/libscha.so`	ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

691 ページの`scha_resource_get`(1HA), 1047 ページの`scha_calls`(3HA), 1239 ページの`scha_strerror`(3HA), 1241 ページの`scha_strerror_i18n`(3HA), Unresolved link to " attributes5", 1335 ページの`rt_properties`(5)

名前

scha_resourcetype_open, scha_resourcetype_open_zone, scha_resourcetype_get, scha_resourcetype_get_zone, scha_resourcetype_close — リソースタイプ情報アクセス関数。

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l scha#include
 <scha.h>scha_err_t scha_resourcetype_open( const char *rtname, scha_resourcetype_t *handle);

scha_err_t scha_resourcetype_open_zone(const char *cluster, const char
 *rt_name, scha_resourcetype_t *handlep);

scha_err_t scha_resourcetype_close(scha_resourcetype_t handle);

scha_err_t scha_resourcetype_get(scha_resourcetype_t handle, const char *tag...);

scha_err_t scha_resourcetype_get_zone(const char *cluster,
     scha_resourcetype_t handlep,const char *rt_tag, ...);
```

`scha_resourcetype_open()`、`scha_resourcetype_get ()`、および`scha_resourcetype_close()` 関数を使用すると、Resource Group Manager (RGM) クラスタ機能が利用するリソースタイプの情報を入手できます。

`scha_resourcetype_open()` は、リソースタイプへのアクセスを初期化し、`scha_resourcetype_get()` が使用するアクセスハンドルを返します。

`scha_resourcetype_open()` の *rtname* 引数には、アクセスするリソースタイプの名前を指定します。

handle 引数の値は、関数の戻り値を格納する変数のアドレスとなります。

`scha_resourcetype_get()` 関数は、*tag* 引数に指定されるリソースタイプの情報にアクセスします。*tag* 引数には、`scha_tags.h` ヘッダーファイルのマクロで定義される文字列値が入ります。タグ以降の引数は、*tag* の値に依存します。

タグ以降に追加する引数は、情報を取り出すクラスタノードや、タグ固有の他の情報を指定する際に必要となることがあります。引数リストの最後の引数は、*tag* で指定される情報の格納に適した型にする必要があります。これは out 引数で、リソースタイプの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。`scha_resourcetype_get()` で使用されたハンドルで `scha_resourcetype_close()` が呼び出されるまで、`scha_resourcetype_get()` の返す情報用に割り当てられたメモリーは維持されます。

scha_resourcetype_close() には、事前に *scha_resourcetype_open* 関数を使って得た handle() 引数の値を指定します。この関数は、該当するハンドルを使用して得た scha_resourcegroup_get() の戻り値用割り当てメモリーを解放するとともに、このハンドルを無効化します。値を返す必要が生じると、get 呼び出しごとにメモリー割り当てが発生する点に注意してください。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されることはありません。

scha_tags.h に定義されるマクロには、*scha_resourcetype_get* の tag() 引数に使用されるものがあります。ここでは出力引数および追加引数の型を説明します。構造体と enum 型は、1047 ページのscha_calls(3HA) で説明されています。

scha_resourcetype_open_zone() 関数および scha_resourcetype_get_zone() 関数はそれぞれ、scha_resourcetype_open() および scha_resourcetype_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_resourcetype_open_zone() または scha_resourcetype_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_resourcetype_open() または scha_resourcetype_get() と等しくなります)。

scha_resourcetype_open_zone() から返されたハンドルを閉じるには、scha_resourcetype_close() を使用します。cluster 引数は不要です。

optag 引数

次のマクロによって、リソースタイプのプロパティーを指定します。出力は、リソースタイプの名前付きプロパティーの値です。

注記 - *optag* 引数 (SCHA_API_VERSION や SCHA_BOOT など) には大文字と小文字の区別は*ありません*。*optag* 引数を指定するときには、大文字と小文字の任意の組み合わせを使用できます。

SCHA_API_VERSION

出力引数の型は int* です。

SCHA_BOOT

出力引数の型は char ** です。

SCHA_FAILOVER

出力引数の型は boolean_t * です。

SCHA_FINI

出力引数の型は char ** です。

SCHA_GLOBALZONE

出力引数の型は boolean_t * です。

SCHA_INIT

出力引数の型は char ** です。

SCHA_INIT_NODES

出力引数の型は scha_initnodes_flag_t * です。

SCHA_INSTALLED_NODES

出力引数の型は scha_str_array_t ** です。

SCHA_IS_LOGICAL_HOSTNAME

出力引数の型は boolean_t * です。

SCHA_IS_SHARED_ADDRESS

出力引数の型は boolean_t * です。

SCHA_MONITOR_CHECK

出力引数の型は char ** です。

SCHA_MONITOR_START

出力引数の型は char ** です。

SCHA_MONITOR_STOP

出力引数の型は char ** です。

SCHA_PER_NODE

出力引数の型は boolean_t * です。

SCHA_PKGLIST

出力引数の型は scha_str_array_t ** です。

SCHA_POSTNET_STOP

出力引数の型は char ** です。

SCHA_PRENET_START

出力引数の型は char ** です。

SCHA_PROXY
: 出力引数の型は boolean_t * です。

SCHA_RESOURCE_LIST
: 出力引数の型は scha_str_array_t ** です。

SCHA_RT_BASEDIR
: 出力引数の型は char ** です。

SCHA_RT_DESCRIPTION
: 出力引数の型は char ** です。

SCHA_RT_SYSTEM
: 出力引数の型は boolean_t * です。

SCHA_RT_VERSION
: 出力引数の型は char ** です。

SCHA_SINGLE_INSTANCE
: 出力引数の型は boolean_t * です。

SCHA_START
: 出力引数の型は char ** です。

SCHA_STOP
: 出力引数の型は char ** です。

SCHA_UPDATE
: 出力引数の型は char ** です。

SCHA_VALIDATE
: 出力引数の型は char ** です。

これらの関数は、次の戻り値を返します。

0	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_NOERR	関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) のマニュアルページを参照してください。

`/usr/cluster/include/scha.h`	インクルードファイル
`/usr/cluster/lib/libscha.so`	ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

691 ページの`scha_resource_get`(1HA), 1047 ページの`scha_calls`(3HA), 1239 ページの`scha_strerror`(3HA), 1241 ページの`scha_strerror_i18n`(3HA), Unresolved link to " attributes5", 1335 ページの`rt_properties`(5)

名前

scha_resourcetype_open, scha_resourcetype_open_zone, scha_resourcetype_get, scha_resourcetype_get_zone, scha_resourcetype_close — リソースタイプ情報アクセス関数。

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l scha#include
 <scha.h>scha_err_t scha_resourcetype_open( const char *rtname, scha_resourcetype_t *handle);

scha_err_t scha_resourcetype_open_zone(const char *cluster, const char
 *rt_name, scha_resourcetype_t *handlep);

scha_err_t scha_resourcetype_close(scha_resourcetype_t handle);

scha_err_t scha_resourcetype_get(scha_resourcetype_t handle, const char *tag...);

scha_err_t scha_resourcetype_get_zone(const char *cluster,
     scha_resourcetype_t handlep,const char *rt_tag, ...);
```

`scha_resourcetype_open()`、`scha_resourcetype_get ()`、および`scha_resourcetype_close()` 関数を使用すると、Resource Group Manager (RGM) クラスタ機能が利用するリソースタイプの情報を入手できます。

`scha_resourcetype_open()` は、リソースタイプへのアクセスを初期化し、`scha_resourcetype_get()` が使用するアクセスハンドルを返します。

`scha_resourcetype_open()` の *rtname* 引数には、アクセスするリソースタイプの名前を指定します。

handle 引数の値は、関数の戻り値を格納する変数のアドレスとなります。

`scha_resourcetype_get()` 関数は、*tag* 引数に指定されるリソースタイプの情報にアクセスします。*tag* 引数には、`scha_tags.h` ヘッダーファイルのマクロで定義される文字列値が入ります。タグ以降の引数は、*tag* の値に依存します。

タグ以降に追加する引数は、情報を取り出すクラスタノードや、タグ固有の他の情報を指定する際に必要となることがあります。引数リストの最後の引数は、*tag* で指定される情報の格納に適した型にする必要があります。これは out 引数で、リソースタイプの情報を格納します。関数の実行に失敗した場合、出力引数に値は返されません。`scha_resourcetype_get()` で使用されたハンドルで `scha_resourcetype_close()` が呼び出されるまで、`scha_resourcetype_get()` の返す情報用に割り当てられたメモリーは維持されます。

scha_resourcetype_close() には、事前に *scha_resourcetype_open* 関数を使って得た handle() 引数の値を指定します。この関数は、該当するハンドルを使用して得た scha_resourcegroup_get() の戻り値用割り当てメモリーを解放するとともに、このハンドルを無効化します。値を返す必要が生じると、get 呼び出しごとにメモリー割り当てが発生する点に注意してください。ある呼び出しで値を返すために割り当てられたメモリーが、以降の呼び出しによって上書きされたり、再利用されることはありません。

scha_tags.h に定義されるマクロには、*scha_resourcetype_get* の tag() 引数に使用されるものがあります。ここでは出力引数および追加引数の型を説明します。構造体と enum 型は、1047 ページのscha_calls(3HA) で説明されています。

scha_resourcetype_open_zone() 関数および scha_resourcetype_get_zone() 関数はそれぞれ、scha_resourcetype_open() および scha_resourcetype_get() と同じ目的で使用され、追加の cluster 引数で、リソースグループが存在し、操作の対象となるゾーンクラスタの名前を指定します。これらの関数は、大域ゾーンで実行されるコードを特定のゾーンクラスタで動作させる必要がある場合に便利です。ゾーンクラスタ内で、異なるゾーンクラスタにアクセスするために使用することはできません。

scha_resourcetype_open_zone() または scha_resourcetype_get_zone() の cluster 引数が NULL の場合、クエリーは、呼び出しが実行されるクラスタで実行されます (つまり、NULL 引数での呼び出しは、それぞれ scha_resourcetype_open() または scha_resourcetype_get() と等しくなります)。

scha_resourcetype_open_zone() から返されたハンドルを閉じるには、scha_resourcetype_close() を使用します。cluster 引数は不要です。

optag 引数

次のマクロによって、リソースタイプのプロパティーを指定します。出力は、リソースタイプの名前付きプロパティーの値です。

注記 - *optag* 引数 (SCHA_API_VERSION や SCHA_BOOT など) には大文字と小文字の区別は*ありません*。*optag* 引数を指定するときには、大文字と小文字の任意の組み合わせを使用できます。

SCHA_API_VERSION

出力引数の型は int* です。

SCHA_BOOT

出力引数の型は char ** です。

SCHA_FAILOVER

出力引数の型は boolean_t * です。

SCHA_FINI

出力引数の型は char ** です。

SCHA_GLOBALZONE

出力引数の型は boolean_t * です。

SCHA_INIT

出力引数の型は char ** です。

SCHA_INIT_NODES

出力引数の型は scha_initnodes_flag_t * です。

SCHA_INSTALLED_NODES

出力引数の型は scha_str_array_t ** です。

SCHA_IS_LOGICAL_HOSTNAME

出力引数の型は boolean_t * です。

SCHA_IS_SHARED_ADDRESS

出力引数の型は boolean_t * です。

SCHA_MONITOR_CHECK

出力引数の型は char ** です。

SCHA_MONITOR_START

出力引数の型は char ** です。

SCHA_MONITOR_STOP

出力引数の型は char ** です。

SCHA_PER_NODE

出力引数の型は boolean_t * です。

SCHA_PKGLIST

出力引数の型は scha_str_array_t ** です。

SCHA_POSTNET_STOP

出力引数の型は char ** です。

SCHA_PRENET_START

出力引数の型は char ** です。

SCHA_PROXY
: 出力引数の型は boolean_t * です。

SCHA_RESOURCE_LIST
: 出力引数の型は scha_str_array_t ** です。

SCHA_RT_BASEDIR
: 出力引数の型は char ** です。

SCHA_RT_DESCRIPTION
: 出力引数の型は char ** です。

SCHA_RT_SYSTEM
: 出力引数の型は boolean_t * です。

SCHA_RT_VERSION
: 出力引数の型は char ** です。

SCHA_SINGLE_INSTANCE
: 出力引数の型は boolean_t * です。

SCHA_START
: 出力引数の型は char ** です。

SCHA_STOP
: 出力引数の型は char ** です。

SCHA_UPDATE
: 出力引数の型は char ** です。

SCHA_VALIDATE
: 出力引数の型は char ** です。

これらの関数は、次の戻り値を返します。

0	関数の実行に成功。
0 以外	関数の実行に失敗。

SCHA_ERR_NOERR	関数の実行に成功。

その他のエラーコードについては、1047 ページのscha_calls(3HA) のマニュアルページを参照してください。

`/usr/cluster/include/scha.h`	インクルードファイル
`/usr/cluster/lib/libscha.so`	ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

691 ページの`scha_resource_get`(1HA), 1047 ページの`scha_calls`(3HA), 1239 ページの`scha_strerror`(3HA), 1241 ページの`scha_strerror_i18n`(3HA), Unresolved link to " attributes5", 1335 ページの`rt_properties`(5)

名前

scha_strerror, scha_strerror_i18n — エラーコードからエラーメッセージの作成

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l scha#include <scha.h>char
 *scha_strerror(scha_err_t error_code);

char *scha_strerror_i18n(scha_err_t error_code);
```

scha_strerror() および scha_strerror_i18n() 関数は、与えられたエラーコード scha_err_t のエラーを説明する簡単な文字列を生成します。scha_strerror() から返された文字列は、英語で表示されます。scha_strerror_i18n() から返された文字列は、LC_MESSAGESロケールカテゴリで指定されているその国および地域の言語で表示されます。Unresolved link to " setlocale3C" を参照してください。

次のパラメータがサポートされます。

error_code　　エラーを説明する簡単な文字列の生成元となるエラーコード。

例 390　　scha_strerror_i18n() 関数の使用

```
sample()
{
      scha_err_t  err;

      /* resource group containing example_R */
      char * resource_group = "example_RG";

      /* a configured resource */
      char * resource_name = "example_R";

      err = scha_control(SCHA_GIVEOVER, resource_group, resource_name);

      if (err != SCHA_ERR_NOERR) {
          syslog(LOG_ERR, "scha_control GIVEOVER failed: %s",
             scha_strerror_i18n(err));
      }
}
```

/usr/cluster/include/scha.h　　インクルードファイル

/usr/cluster/lib/libscha.so　　ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

1047 ページの`scha_calls`(3HA), Unresolved link to " setlocale3C", Unresolved link to " syslog3C", Unresolved link to " attributes5"

名前

scha_strerror, scha_strerror_i18n — エラーコードからエラーメッセージの作成

```
cc [flags…] -I /usr/cluster/include file -L /usr/cluster/lib -l scha#include <scha.h>char
 *scha_strerror(scha_err_t error_code);

char *scha_strerror_i18n(scha_err_t error_code);
```

scha_strerror() および scha_strerror_i18n() 関数は、与えられたエラーコード scha_err_t のエラーを説明する簡単な文字列を生成します。scha_strerror() から返された文字列は、英語で表示されます。scha_strerror_i18n() から返された文字列は、LC_MESSAGESロケールカテゴリで指定されているその国および地域の言語で表示されます。Unresolved link to " setlocale3C" を参照してください。

次のパラメータがサポートされます。

error_code　　エラーを説明する簡単な文字列の生成元となるエラーコード。

例 391　　scha_strerror_i18n() 関数の使用

```
sample()
{
      scha_err_t  err;

      /* resource group containing example_R */
      char * resource_group = "example_RG";

      /* a configured resource */
      char * resource_name = "example_R";

      err = scha_control(SCHA_GIVEOVER, resource_group, resource_name);

      if (err != SCHA_ERR_NOERR) {
          syslog(LOG_ERR, "scha_control GIVEOVER failed: %s",
             scha_strerror_i18n(err));
      }
}
```

/usr/cluster/include/scha.h　　インクルードファイル

/usr/cluster/lib/libscha.so　　ライブラリ

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

1047 ページの`scha_calls`(3HA), Unresolved link to " setlocale3C", Unresolved link to " syslog3C", Unresolved link to " attributes5"

OSC4 4

名前

clusters — クラスタ名のデータベース

/etc/clusters

clusters ファイルには、ローカルネーミングドメイン内にある既知のクラスタに関する情報を記述します。クラスタごとに、1 行形式で次の情報を指定する必要があります。

clustername whitespace-delimited list of hosts

一覧中の右側の名前で拡張マーカー「*」記号が付けられているものは、再帰的に展開されます。

項目間の区切りには、任意の数の半角スペース、タブ記号が使用できます。「#」記号以降の記述は、コメント文と見なされます。該当行の末尾までの記述は、ファイルの検索時に無視され対象外となります。

クラスタ名には、印字可能なすべての文字が使用できますが、大文字、フィールド区切り記号、改行文字、コメント記号は使用できません。クラスタ名の最大長は 32 文字です。

この情報は、管理するノードのグループを指定するために、pconsole コマンドなどの Oracle Solaris Cluster システム管理ツールによって使用されます。このデータベースで使用する名前は、ホストデータベースで使用されるホスト名と一致している必要があります。

データベースは、NIS または NIS+マップ、あるいはローカルファイルから使用できます。検索の順番は /etc/nsswitch.conf ファイルに指定できます。デフォルトの順番は nis files になっています。

例 392　　/etc/clusters のサンプルファイル

通常 /etc/clusters ファイルは次のように記述されます。

```
bothclusters  *planets *wine
planets   mercury venus
wine    zinfandel merlot chardonnay riesling
```

次の行は、通常の /etc/nsswitch.conf エントリです。

```
clusters: nis files
```

/etc/clusters

/etc/nsswitch.conf

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

1261 ページのserialports(4), Unresolved link to " nsswitch.conf4", Unresolved link to " attributes5"

名前

commandlog — コマンドログファイル

```
/var/cluster/logs/commandlog
```

ASCII テキストファイル `commandlog` には、クラスタ内で実行された、選択された Oracle Solaris Cluster コマンドが記録されています。コマンドのロギングは、ユーザーがクラスタを設定したときに自動的に開始され、ユーザーがクラスタをシャットダウンしたときに終了します。

クラスタの構成や現在の状態を表示するようなコマンドは、このファイルにロギングされません。次のような、クラスタの現在の状態の構成や変更を行うコマンドは、このファイルにロギングされます。

- `claccess`
- `cldevice`
- `cldevicegroup`
- `clinterconnect`
- `clnasdevice`
- `clnode`
- `clquorum`
- `clreslogicalhostname`
- `clresource`
- `clresourcegroup`
- `clresourcetype`
- `clressharedaddress`
- `clsnmphost`
- `clsnmpmib`
- `clsnmpuser`
- `cltelemetryattribute`
- `cluster`
- `clzonecluster`
- `scconf`

- scdidadm
- scdpm
- scgdevs
- scrgadm
- scshutdown
- scswitch

commandlog ファイル内の各レコードには、次の情報が含まれています。

- 日付とタイムスタンプ
- コマンドが実行されたホスト名
- コマンドのプロセス ID
- コマンドを実行したユーザーの ID
- ユーザーが実行したコマンド (すべてのオプションとオペランドを含む)

注記 - コマンドオプションは、commandlog ファイルに含まれているので、シェル内でコピー、貼り付け、および実行することができます。

- 実行されたコマンドの終了ステータスまたはシグナル

commandlog ファイルはデフォルトでは、週末ごとに定期的にアーカイブされます。Oracle Solaris Cluster は任意の時点で、アーカイブ済みの commandlog ファイルを、クラスタノードごとに最大 8 個保持します。

例 393 /var/cluster/logs/commandlog ファイル

/var/cluster/logs/commandlog ファイルの典型的な内容の例を次に示します。

```
11/11/2011 09:43:36 phys-schost-1 5758 root START - clrg add "app-sa-1"
11/11/2011 09:43:36 phys-schost-1 5758 root END 0
11/11/2011 09:43:36 phys-schost-1 5760 root START - clrg set -y
"RG_description=Department Shared Address RG" "app-sa-1"
11/11/2011 09:43:37 phys-schost-1 5760 root END 0
11/11/2011 09:44:15 phys-schost-1 5810 root START - clrg online "app-sa-1"
11/11/2011 09:44:15 phys-schost-1 5810 root END 0
11/11/2011 09:44:19 phys-schost-1 5222 root END -20988320
12/02/2011 14:37:21 phys-schost-1 5542 jbloggs START - clrg -c -g "app-sa-1"
-y "RG_description=Joe Bloggs Shared Address RG"
12/02/2011 14:37:22 phys-schost-1 5542 jbloggs END 0
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core
インタフェースの安定性	発展中

683 ページの`scha_control`(1HA), 699 ページの`scha_resource_setstatus`(1HA), 747 ページの`scconf`(1M), 793 ページの`scdidadm`(1M), 803 ページの`scdpm`(1M), 815 ページの`scgdevs`(1M), 861 ページの`scrgadm`(1M), 875 ページの`scshutdown`(1M), 883 ページの`scswitch`(1M), Unresolved link to " attributes5"

名前

rt_reg — リソースタイプ登録 (RTR) ファイル

リソースタイプ登録 (RTR) ファイルには、リソースタイプの内容を記述します。リソースタイプは、Resource Group Manager (RGM) クラスタの制御下で動作するスケーラブルまたは高可用性サービスを表します。このファイルは、リソースタイプ実装の一部であり、クラスタ構成にリソースタイプを登録するための 861 ページのscrgadm(1M) コマンドの入力ファイルとして使用されます。特定の型のリソースをクラスタ内で運用するには、事前に該当するリソースタイプを登録しておく必要があります。通常、RTR ファイルは /opt/cluster/lib/rgm/rtreg ディレクトリに存在します。

RTR ファイルに宣言する内容は、リソースタイプのリソースタイププロパティーおよびリソースプロパティーです。ファイルは、リソースタイププロパティーの宣言およびリソースプロパティーの宣言の 2 つのパートに分かれています。プロパティー名では、大文字と小文字が区別されません。

リソースタイププロパティーの宣言では、リソースタイプの実装情 (報該当する型のリソースを制御するため RGM が呼び出すコールバックメソッドのパスなど) を登録します。こうしたリソースタイププロパティーの大部分は、rt_reg ファイルに固定値として設定されます。これらのプロパティーは、該当する型のすべてのリソースで継承されます。

リソースタイプを実装する際には、リソースプロパティーに対する管理範囲を拡張するよう、カスタマイズすることも可能です。rt_reg ファイルの 2 番目のパートにエントリを記載するリリースプロパティーには、システム定義プロパティーと拡張プロパティーの 2 種類があります。

システム定義のリソースプロパティーは、型と意味が事前に定義されています。rt_reg ファイルを使うことで、システム定義のリソースプロパティーのデフォルト値、最大値、最小値などの属性を設定できます。rt_reg ファイルは、リソースタイプの実装によって完全に定義される拡張プロパティーを宣言するためにも使用できます。拡張プロパティーは、クラスタシステムが管理維持するリソースの構成データに追加する情報のリソースタイプを提供します。

rt_reg ファイルはリソースプロパティーのデフォルト値を設定しますが、実際に用いる値は個々のリソースごとに設定します。rt_reg ファイルのプロパティーに変数を使用する場合は、クラスタ管理者がその値を設定します。

リソースタイププロパティーの宣言

リソースタイププロパティーの宣言では、多数のプロパティー値を指定します。

```
PROPERTY_NAME = "Value";
```

rt_reg ファイルで宣言できるリソースタイププロパティーのリストについては、1335 ページのrt_properties(5) のマニュアルページを参照してください。ほとんどのプロパティーはデフォルト値を持っているか、またはオプションであるため、RTR ファイルで重要なのは、タイプ名、START へのパス、STOP コールバックメソッド、および RT_version だけです。

ファイル内に記載するプロパティーには、Resource_type プロパティーを最初に設定する必要があります。

リソースタイプ名の形式は *vendor-id.RT-name:version* です。

リソースタイプの名前は、RTR ファイル内に指定された 3 つのプロパティー *vendor_id*、*resource_type* 、*RT_version* で構成されます。scrgadm コマンドでは、区切り文字としてピリオドとコロンを使用します。*vendor-id* 接頭辞はオプションですが、異なるベンダーの登録ファイル名が重複する場合は、ファイル名を区別するために使用することをお勧めします。*vendor-id* が一意であることを保証するためには、リソースタイプを作成した会社の株式の略号を使用します。

リソースプロパティーの宣言

リソースプロパティーの宣言は多数のエントリから構成されます。個々のエントリは、列挙した属性値を括弧で囲みます。エントリ内に記載する属性については、リソースプロパティー名を最初に設定する必要があります。

システム定義プロパティーには、事前に定義されたタイプ属性と記述属性が含まれているため、これらの属性を rt_reg ファイルで宣言し直すことはできません。管理者が値を設定するときの範囲の制限、デフォルト値、および制約事項をシステム定義プロパティーで宣言できます。

システム定義プロパティーに設定できる属性は、1271 ページのproperty_attributes(5) のマニュアルページに一覧表示されています。システム定義プロパティーに指定できない属性については、表中にその旨記載されています。

rt_reg ファイル内にエントリを含めることができるシステム定義プロパティーは、1287 ページのr_properties(5) のマニュアルページに一覧表示されています。次のサンプルエントリでは、システム定義のリソースプロパティーである RETRY_COUNT を宣言しています。

```
{
  PROPERTY = RETRY_COUNT;
  MIN=0;
  MAX=10;
  DEFAULT=2;
  TUNABLE = ANYTIME;
}
```

拡張プロパティーのエントリには、プロパティーの型を指定する必要があります。拡張プロパティーに設定できる属性は、1271 ページのproperty_attributes(5) のマニュアルページに一覧表示されています。

次のサンプルエントリでは、文字列型の拡張プロパティーである「ConfigDir」を宣言しています。TUNABLE 属性では、リソースの作成時にクラスタ管理者がプロパティー値を設定できることを示します。

```
{
  PROPERTY = ConfigDir;
  EXTENSION;
  STRING;
  DEFAULT="/";
  TUNABLE = AT_CREATION;
}
```

使用法

rt_reg ファイルは、ASCII テキストファイルです。この中にはファイルの内容に関するコメントを記入しておくことができます。先に述べたように、このファイル内容は、リソースタイププロパティーの一覧記述とそのあとのリソースプロパティーの宣言という2 つのパートに分かれます。

空白には、半角スペース、タブ、改行、およびコメントが該当します。こうした空白は、トークンの前後に記入できます。ただし、引用符で囲まれたトークン内部にある半角スペースおよび記号 (#) については、空白とはみなされません。空白はトークン間の区切り記号の役割を果たし、それ以外は無視されます。

記号から直後の改行までの記述は、すべてコメントと見なされます。

#$ 記号から直後の改行までの記述は、すべて指令 とみなされます。指令は、RTR ファイルのリソースタイププロパティーの宣言セクションとリソースプロパティーの宣言セクションの間に記入する必要があります。指令を RTR ファイルのどこか別の場所に挿入すると、パーサーエラーが発生します。有効な指令は #$upgrade および #$upgrade_from のみです。これ以外の指令を記載した場合、構文解析の際にエラーとなります。

トークンを構成するものは、プロパティー名、プロパティー値および次の記号類です。

{ }	パラメータテーブルプロパティーを囲みます。
;	プロパティーおよび属性の終端を示します。
=	プロパティー名とプロパティー値、または属性名と属性値を区切ります。
,	値リストの値を区切ります。

ファイル内のプロパティー名キーワードは、大文字と小文字が区別されません。

プロパティーおよび属性の記述は、次の 3 つのいずれかの形式を取ります。

property-name = *property-value*;
property-name;
property-name = *property-value* [, *property-value*];

上記の表記形式で、角括弧 [ ] で囲まれている部分はオプションの項目です。つまり、プロパティー値は 1 つの *property-value* またはコンマで区切った 2 つ以上の *property-value* になります。

プロパティーリストの最初のプロパティーには、リソースタイプの名前のみを記述する必要があります。

ブール型のプロパティーおよび属性は、次の構文で記述します。

boolean-property-name;
boolean-property-name = TRUE;
boolean-property-name = FALSE;

上の 2 つの形式は、ともに *boolean-property-name* を TRUE に設定します。

リソースタイププロパティー名は、1335 ページのrt_properties(5) のマニュアルページに一覧表示されています。システム定義プロパティーは、1287 ページのr_properties(5) のマニュアルページに一覧表示されています。

リソース宣言では任意の数のエントリを指定できます。個々のエントリは、列挙したリソースプロパティーの属性を括弧で囲みます。

{*attribute-value-list*}

各属性値リストは 1 つのリソースプロパティーに対する属性値で構成されます。その書式はプロパティー値の場合と同じですが、さらに型と属性を組み合わせた 2 つの形式が追加されます。

type-attribute-value;
enum-type-attribute { *enum-value* [, *enum-value*] };

type-attribute-value の構文では、値 *type-attribute-value* を持つ拡張プロパティーのデータ型を宣言します。これは、*boolean-property-name* によって指定されたプロパティーが TRUE の値を持つように定義する最初の形式の *boolean-property-name* とは異なります。

たとえば TUNABLE 属性の場合、次のいずれかの値をとることができます。FALSE または NONE、AT_CREATION、TRUE または ANYTIME、および WHEN_DISABLED。TUNABLE 属性は、下記のように記述することもできます。

```
TUNABLE;
```

この場合の値は ANYTIME となります。

文法

次は、BNF 文法に準拠した rt_reg ファイルの構文解説です。実際の rt_reg ファイルではキーワードの大文字と小文字は区別されませんが、ここでは非終端キーワードを小文字、終端キーワードを大文字で表記しています。行の先頭にある非終端記号に続くコロン (:) は、文法生成規則の始まりを示します。文法生成規則の右側にくる文字列は、縦棒 (|) で始まる行に表記しています。変数の終端トークンは山括弧 (< >) で囲み、コメントを丸括弧 (( )) で囲んでいます。セミコロン (;)、イコール記号 (=)、山括弧 ({}) など、文法生成の右側にあるその他の区切り文字はリテラルになります。

コメントは次の形式で表記されます。

```
COMMENT : # anything but NEWLINE NEWLINE
```

コメントは、トークンの後に記載される場合もあります。コメントは空白記号と同様に扱われます。

```
    rt_reg_file :  Resource_type = value ; proplist upgradesect paramtable

    proplist :  (NONE: empty)
 | proplist rtproperty

    rtproperty : rtboolean_prop ;
 | rtvalue_prop ;

    rtboolean_prop : SINGLE_INSTANCE
 | FAILOVER | RT_SYSTEM

    rtvalue_prop : rtprop = value
 | PKGLIST = valuelist

    rtprop : RT_BASEDIR
 | RT_VERSION
 | API_VERSION
 | INIT_NODES
 | START
 | STOP
 | VALIDATE
 | UPDATE
 | INIT
 | FINI
```

```
| BOOT
| MONITOR_START
| MONITOR_STOP
| MONITOR_CHECK
| PRENET_START
| POSTNET_STOP
| RT_DESCRIPTION
| VENDOR_ID
| rtboolean_prop (booleans may have explicit assignments.)

   value : contiguous-non-ws-non-;-characters
| "anything but quote"
| TRUE
| FALSE
| ANYTIME
| WHEN_DISABLED
| AT_CREATION
| RG_PRIMARIES
| RT_INSTALLED_NODES
|  (NONE: Empty value)

   valuelist : value
| valuelist , value

   upgradesect : (empty)
| #$UPGRADE upgradelist

   upgradelist :  (empty)
| upgradelist #$UPGRADE_FROM rt_version upgtunability

   upgtunability : ANYTIME
| AT_CREATION
| WHEN_DISABLED
| WHEN_OFFLINE
| WHEN_UNMANAGED
| WHEN_UNMONITORED

   paramtable :  (empty)
| paramtable parameter

   parameter :  { pproplist }

   pproplist : PROPERTY = value ;  (property name must come first)
| pproplist pproperty

   pproperty : pboolean_prop ;
| pvalue_prop ;
| typespec ;

   pvalue_prop : tunable_prop
| pprop = value
| pprop =  (NONE: no value setting)
| DEFAULT = valuelist
```

```
   pprop : DESCRIPTION
 | MIN
 | MAX
 | MINLENGTH
 | MAXLENGTH
 | ARRAY_MINSIZE
 | ARRAY_MAXSIZE
 | pboolean_prop

   tunable_prop : TUNABLE
 | TUNABLE = AT_CREATION
 | TUNABLE = ANYTIME
 | TUNABLE = WHEN_DISABLED
 | TUNABLE = TRUE
 | TUNABLE = FALSE
 | TUNABLE = NONE

   typespec : INT
 | BOOLEAN
 | STRING
 | STRINGARRAY
 | ENUM { valuelist }
```

例 394 サンプル登録ファイル

次の登録ファイルは、簡単なリソースタイプのサンプルです。

```
#
# Registration information for example resource type
#

Resource_type = example_RT;
Vendor_id = SUNW;
RT_Version = 2.0
RT_Basedir= /opt/SUNWxxx;
START = bin/example_service_start;
STOP  = bin/example_service_stop;

#$upgrade
#$upgrade_from "1.0" when_unmonitored

#
# Set range and defaults for method timeouts and Retry_count.
#
{ Property = START_TIMEOUT;  Tunable; MIN=60; DEFAULT=300; }
{ Property = STOP_TIMEOUT;  Tunable; MIN=60; DEFAULT=300; }
{ Property = Retry_count;  Tunable; MIN=1; MAX=20; DEFAULT=10; }

#
# An extension property that can be set at resource creation
#
{ Property = LogLevel;
```

```
  Extension;
  enum { OFF, TERSE, VERBOSE };
  Default = TERSE;
  Tunable = AT_CREATION;
  Description = "Controls the detail of example_service logging";
}
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

861 ページの`scrgadm`(1M), Unresolved link to " attributes5",
1335 ページの`rt_properties`(5), 1287 ページの`r_properties`(5),
1271 ページの`property_attributes`(5)

Unresolved link to " Oracle Solaris Cluster Data Services Developer's Guide "

名前

scdpmd.conf — ディスクパスモニタリングデーモン構成ファイル

```
/etc/cluster/scdpm/scdpmd.conf
```

scdpmd デーモンは、ディスクパスをモニターし、パスに障害がある場合に、適切なアクションを実行します。調整可能プロパティーを持つ構成ファイル /etc/cluster/scdpm/scdpmd.conf を作成または変更してこのデーモンを調整し、SIGHUP 信号を scdpmd デーモンに送り、構成ファイルを読み取ることができます。

```
# pkill -HUP scdpmd
```

scdpmd.conf ファイルの次のプロパティーを調整できます。

Ping_interval

説明
: ディスクパスステータスチェックの間隔 (秒)

デフォルト
: 600

最小
: 20

最大
: 3600

Ping_retry

説明
: 障害時にディスクパスステータスをクエリーする再試行回数

デフォルト
: 3

最小
: 2

最大
: 10

Ping_timeout

説明

ディスクパスステータスをクエリーするタイムアウト(秒)

デフォルト

30

最小

1

最大

1800

次は有効な`scdpmd.conf`ファイルの例です:

```
Ping_interval = 120
Ping_retry = 5
Ping_timeout = 10
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
インタフェースの安定性	発展中

61 ページの`cldevice`(1CL), 185 ページの`clnode`(1CL)

名前

serialports — ホスト名とシリアルポートの対応付けデータベース

```
/etc/serialportsserialports NIS or NIS+ maps
```

serialports データベースは、指定された端末サーバーホストに接続したシリアルポートを示す TCP ポート番号およびサーバー名を、単一の名前に関連付けたものです。このデータベースは、通常、コンソールにホスト名を割り当てるために使用しますが、プリンタ、モデムなどへのアクセスを可能にするために使用する場合もあります。割り当ては、サービスがネットワークベースの端末集配信装置によって提供される場合に使用します。個々の名前については、1 行形式で次の情報を指定する必要があります。

```
host-name concentrator-hostname tcp-port-number
```

項目間の区切りには、任意の数の半角スペース、TAB 記号が使用できます。ポンド記号 (#) は、コメントの始まりを示します。ポンド記号と行の終端の間の文字は、ファイルを検索するルーチンによって解釈されません。

コマンド行から並列コンソールアクセス (pconsole) ユーティリティーを使用して、リモートからクラスタにログインできます。pconsole ユーティリティーは、Oracle Solaris terminal/pconsole パッケージの一部です。このパッケージは、pkg install terminal/pconsole を実行することによってインストールします。pconsole ユーティリティーは、コマンド行で指定されたリモートホストごとにホスト端末ウィンドウを作成します。このユーティリティーはまた、そこで入力された内容を、ユーザーによって開かれている各接続に伝えるための中央 (またはマスター) コンソールウィンドウも開きます。

```
/etc/serialports
```

```
/etc/nsswitch.conf
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/developer/api
インタフェースの安定性	発展中

1245 ページの`clusters`(4), Unresolved link to " nsswitch.conf4", Unresolved link to " attributes5"

OSC4 5

名前

SUNW.crs_framework, crs_framework — Oracle Clusterware のシャットダウンを調整するリソースタイプ実装

SUNW.crs_framework リソースタイプは、Oracle Solaris Cluster Support for Oracle Real Application Clusters (Oracle RAC) 構成で、Oracle Clusterware リソースと Oracle Solaris Cluster リソースのシャットダウンを調整します。このリソースタイプは、Oracle Solaris Cluster が Oracle Clusterware を停止できるようにすることで、Oracle Solaris Cluster と Oracle Clusterware の相互運用を可能にします。

注記 - このリソースタイプは、Oracle Solaris Cluster 管理コマンドを使用して Oracle Clusterware を起動できるようにするものでは*ありません*。Oracle Clusterware を起動するには、Oracle コマンドを使用するか、ノードをブートするかの方法しかありません。

Oracle Clusterware 投票ディスクと Oracle クラスタレジストリ (OCR) ファイルは、SUNW.ScalDeviceGroup または SUNW.ScalMountPoint タイプのリソースが表すストレージに存在することがあります。この状況では、SUNW.ScalDeviceGroup または SUNW.ScalMountPoint タイプのリソースをオフラインにする前に、Oracle Clusterware を停止する必要があります。この要件に適合するため、SUNW.crs_framework タイプのリソースは、次の状況になると、ノードにある Oracle Clusterware プロセスを停止します:

- そのノードでタイプ SUNW.ScalDeviceGroup または SUNW.ScalMountPoint のリソースがオフラインになるとき。Oracle Clusterware プロセスを次の理由のため停止する必要があります。
  - SUNW.ScalDeviceGroup または SUNW.ScalMountPointタイプのリソースが正しく停止するのを確認するため。
  - Oracle Clusterware プロセスまたは Oracle RAC プロセスがストレージにアクセスしているときに、SUNW.ScalDeviceGroup または SUNW.ScalMountPoint タイプのリソースがオフラインになった場合に、データベースまたはノードに障害が発生するのを防ぐため。
- ノードが停止するとき。Oracle Clusterware プロセスが停止しない場合、ノードはシャットダウンできません。

SUNW.crs_framework リソースタイプは、シングルインスタンスリソースタイプです。したがって、クラスタに作成可能なリソースは 1 個だけです。

Oracle Solaris Cluster が正しい順序でリソースを停止するようにするには、SUNW.crs_framework タイプのリソースを次のように構成します。

- SUNW.crs_framework リソースを含むリソースグループに対して、SUNW.ScalDeviceGroup または SUNW.ScalMountPoint タイプのリソースを含むすべてのリソースグループが、強いポジティブなアフィニティーを宣言することを確認します。
- Oracle Clusterware 投票ディスクと OCR ファイルのストレージを表すすべてのリソースに対して、SUNW.crs_framework リソースによるオフライン再起動の依存関係を設定します。これらのリソースのタイプは、SUNW.ScalDeviceGroup または SUNW.ScalMountPoint です。各依存関係のスコープを、SUNW.ScalDeviceGroup リソースまたは SUNW.ScalMountPoint リソースが動作しているノードだけに限定します。
- SUNW.rac_framework タイプのリソースに対して、SUNW.crs_framework タイプのリソースによる強い依存関係を設定します。

これらの依存関係およびアフィニティーは、Oracle RAC 用 Oracle Solaris Cluster サポートデータサービスのデータベースリソースを構成するときに作成します。詳細は、Unresolved link to " Oracle Solaris Cluster Data Service for Oracle Real Application Clusters ガイド のOracle RAC のサポート データベースインスタンスのリソースの構成"を参照してください。

このリソースタイプを登録し、リソースタイプのインスタンスを作成するには、次のいずれかの方法を使用します。

- Oracle Solaris Cluster Manager
- Oracle Solaris Cluster Support for Oracle Real Application Clusters を構成するためのオプションを指定する 487 ページのclsetup(1CL) ユーティリティー
- 次の一連の Oracle Solaris Cluster 保守コマンド。

  1. このリソースタイプを登録するには、333 ページのclresourcetype(1CL) コマンドを使用します。
  2. このリソースタイプのインスタンスを作成するには、273 ページのclresource(1CL) コマンドを使用します。

標準プロパティー

すべての標準リソースプロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

標準のリソースタイププロパティーは、次のようにこのリソースタイプに対して無効にされます。

Monitor_start_timeout

最小 60

デフォルト　300

Monitor_stop_timeout

最小　60

デフォルト　300

Start_timeout

最小　60

デフォルト　300

Stop_timeout

最小　60

デフォルト　1200

Update_timeout

最小　60

デフォルト　300

Validate_timeout

最小　60

デフォルト　300

拡張プロパティー

SUNW.crs_framework リソースタイプには拡張プロパティーはありません。

例 395　SUNW.crs_framework リソースの作成

この例では、SUNW.crs_framework リソースタイプを登録し、crs_framework-rs という名前の SUNW.crs_framework リソースタイプのインスタンスを作成する方法を示します。この例では、次のように仮定します。

- C シェルを使用します。
- crs-framework-rg という名前のリソースグループが存在します。
- 次のリソースが存在します。

- `rac_framework-rs` という名前のタイプ `SUNW.rac_framework` のリソースで、Oracle RAC フレームワークを表します。
- `db-storage-rs` という名前のタイプ `SUNW.ScalDeviceGroup` のリソースで、Oracle Clusterware 投票ディスクと OCR ファイルを格納するスケーラブルデバイスグループを表します。

```
phys-schost-1# clresourcetype register SUNW.crs_framework

phys-schost-1# clresource create -g crs-framework-rg \
-t SUNW.crs_framework \
-p resource_dependencies=rac_framework-rs \
-p resource_dependencies_offline_restart=db-storage-rs\{local_node\} \
crs_framework-rs
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	`ha-cluster/library/ucmm`

273 ページの`clresource`(1CL), 333 ページの`clresourcetype`(1CL), 487 ページの`clsetup`(1CL), Unresolved link to " attributes5", 1411 ページの`SUNW.rac_framework`(5), 1415 ページの`SUNWct.ScalDeviceGroup`(5), 1423 ページの`SUNW.ScalMountPoint`(5)

Unresolved link to " Oracle Solaris Cluster Data Service for Oracle Real Application Clusters ガイド "

名前

SUNW.derby, derby — Java DBデータベースのリソースタイプの実装

SUNW.derby は、Oracle Solaris Cluster と一緒に Java DB データベースを使用できるようにするフェイルオーバーリソースタイプです。Java DB データベースは Derby データベースに基づきます。このデータベースについては、http://db.apache.org/derby/ を参照してください。

SUNW.derby リソースタイプに関連付けられている拡張プロパティーは、次のとおりです。

DB_path(string)

Java DB データベースのデータファイルの位置を指定します。

DB_path の値は PATH を指定する文字列です。指定されたパスは選択されたストレージ、たとえば、HAStoragePlus によって制御されたパスにする必要があります。

カテゴリ	必須
調整可能	無効時

DB_port(integer)

Java DB データベースのポートを指定します。

カテゴリ	必須
デフォルト	1527
調整可能	無効時

DB_probe_port(integer)

Oracle Solaris Cluster が Java DB データベースのサーバーの健全性をテストするために使用するポートを指定します。

カテゴリ	必須
デフォルト	1528
調整可能	無効時

Monitor_retry_count(integer)

障害モニターの再起動を制御します。このプロパティーには、障害モニターがプロセスモニター機能により再起動される回数を示しています。このプロパティーは、735 ページのpmfadm(1M) コマンドに渡される -n オプションに相当します。Resource

Group Manager(RGM) は、指定された時間枠の再起動回数をカウントします (`Monitor_retry_interval` を参照)。なお、`Monitor_retry_count` は、タイプが `SUNW.HADerby` のリソースではなく、障害モニター自体の再起動を指します。

カテゴリ	任意
デフォルト	4
調整可能	すべての時間

`Monitor_retry_interval`(integer)

RGM が障害モニターで問題が発見された回数をカウントする時間を分単位で指定します。このプロパティーは、735 ページの`pmfadm`(1M) コマンドに渡される `-t` オプションに相当します。障害モニターで問題が発見された回数が拡張プロパティー `Monitor_retry_count` の値を超過すると、プロセスモニター機能は障害モニターを再起動しません。

カテゴリ	任意
デフォルト	2 分
調整可能	すべての時間

`Probe_timeout`(integer)

プローブコマンドのタイムアウト値を秒単位で指定します。

カテゴリ	任意
デフォルト	120 秒
デフォルト	2 秒
調整可能	すべての時間

735 ページの`pmfadm`(1M)

名前

property_attributes — リソースプロパティーの属性

次では、システム定義プロパティーの変更や拡張プロパティーの作成に使用できるリソースプロパティーの属性を説明します。

NULL または空の文字列 ("") は、Boolean、Enum、または (Int) タイプのデフォルト値として指定できません。

Array_maxsize

Stringarray タイプの場合に設定できる配列要素数の最大値です。

Array_minsize

Stringarray タイプの場合に設定できる配列要素の最小値です。

デフォルト

プロパティーのデフォルト値を示します。

説明

プロパティーを簡潔に記述した注記 (文字列)。RTR ファイル内でシステム定義プロパティーに対する Description 属性を設定できません。

enumList

Enum タイプの場合、プロパティーに設定できる文字列値のセットです。

EXTENSION

リソースタイプの実装によって定義された拡張プロパティーが RTR ファイルのエントリで宣言されていることを示します。拡張プロパティーが使用されていない場合、そのエントリはシステム定義プロパティーです。

Max

Int タイプの場合にプロパティーに設定できる最大値です。メソッドタイムアウトに最大値は指定できません。

maxLength

String および Stringarray タイプの場合、設定できる文字列の長さの最大値です。

Min

Int タイプの場合にプロパティーに設定できる最小値です。メソッドタイムアウトに Min=0 を指定できません。

Minlength

String および Stringarray タイプの場合、設定できる文字列の長さの最小値です。

Per_node

使用した場合、ノード単位で拡張プロパティーを設定できることを示します。

Per_node プロパティー属性をタイプ定義で指定する場合は、Default プロパティー属性でデフォルト値も指定してください。デフォルト値を指定すると、明示的な値が割り当てられていないノード上でノード単位のプロパティー値をユーザーがリクエストした場合に、確実に値が返されます。

Property

リソースプロパティーの名前。

プロパティーのタイプ

指定できるタイプは、次のとおりです。String、Boolean、Int、Enum、および Stringarray。RTR ファイル内で、システム定義プロパティーのタイプの属性を設定できません。タイプは、RTR ファイルのエントリに登録できる、指定可能なプロパティー値とタイプ固有の属性を決定します。Enum タイプは、文字列値のセットです。

調整可能

クラスタ管理者がリソースのプロパティー値をいつ設定できるかを示します。管理者にプロパティーの設定を許可しない場合は、None または False に設定します。管理者にプロパティーの調整を許可する属性値は、次のとおりです。True または Anytime (任意の時点)、At_creation (リソースの作成時のみ)、When_disabled (リソースがオフラインのとき)。

デフォルトは TrueAnytime です。

例 396 Int タイプの定義

Int タイプは次のように定義されます。

```
{
        Property = Probe_timeout;
        Extension;
        Int;
        Default = 30;
        Tunable = Anytime;
        Description = "Time out value for the probe (seconds)";
}
```

例 397 Per_node タイプの定義

Per_node タイプは次のように定義されます。

```
{
```

```
        Property = LogLevel;
        Extension;
        Enum { Off, Terse, Verbose };
        Default = Terse;
        Per_node;
        Tunable = At_creation;
        Description ="Controls the level of detail for logging";
}
```

PER_NODE プロパティー属性をタイプ定義で指定する場合は、DEFAULT プロパティー属性でデフォルト値も指定してください。

273 ページのclresource(1CL), 305 ページのclresourcegroup(1CL), 333 ページのclresourcetype(1CL), 1287 ページのr_properties(5), 1319 ページのrg_properties(5), 1335 ページのrt_properties(5)

名前

SUNW.Proxy_SMF_failover, Proxy_SMF_failover — フェイルオーバー SMF サービスのプロキシ作成のためのリソースタイプ

SUNW.Proxy_SMF_failover リソースタイプは、サービス管理機能 (SMF) サービスのフェイルオーバーのプロキシを表します。

SUNW.proxysmffailover リソースタイプに対して定義される標準プロパティーと拡張プロパティーについては、これ以降のサブセクションで説明します。これらのプロパティーを SUNW.Proxy_SMF_failover リソースタイプのインスタンスに設定するには、clresource コマンド (273 ページのclresource(1CL)) を使用します。

標準プロパティー

次のリソースプロパティーについては、1287 ページのr_properties(5) を参照してください。

Start_timeout
: デフォルト: 3600
: 最小: 60

Stop_timeout
: デフォルト: 3600
: 最小: 60

Init_timeout
: デフォルト: 3600
: 最小: 60

Boot_timeout
: デフォルト: 3600
: 最小: 60

Fini_timeout
: デフォルト: 3600
: 最小: 60

Validate_timeout
: デフォルト: 3600
: 最小: 60

`Failover_mode`

デフォルト: `SOFT`

調整可能: `Anytime`

`R_description`

デフォルト: `""`

調整可能: `Anytime`

`Retry_count`

デフォルト: `2`

最小: `0`

最大: `10`

調整可能: `Anytime`

`Retry_interval`

デフォルト: `300`

最大: `3600`

調整可能: `Anytime`

`Through_probe_interval`

デフォルト: `60`

調整可能: `Anytime`

拡張プロパティー

`Proxied_service_instances`

リソースによってプロキシされる SMF サービスに関する情報を含みます。値はプロキシされるすべての SMF サービスを含むファイルのパスです。ファイル内の各行は 1 つの SMF サービス専用で、`svc fmri` および対応するサービスマニフェストファイルのパスを指定します。たとえば、リソースが `restarter_svc_test_1:default` と `restarter_svc_test_2:default` の 2 つのサービスを管理する必要がある場合、このファイルには次の 2 行を含めるようにしてください。

```
<svc:/system/cluster/restarter_svc_test_1:default>,
  svc:/system/cluster/restarter_svc_test_1:default>,
  </var/svc/manifest/system/cluster/restarter_svc_test_1.xml>

<svc:/system/cluster/restarter_svc_test_2:default>,
  </var/svc/manifest/system/cluster/restarter_svc_test_2.xml>
```

注記 - 上記エントリはそれぞれ、本来 1 行になるべきものです。ここでは読みやすさを考慮して複数の行に分けています。

デフォルト: ""

調整可能: When disabled

例

この例では、SUNW.Proxy_SMF_failover リソースタイプの登録、アプリケーション用のリソースグループの作成、フェイルオーバーアプリケーションリソースの作成、リソースグループの管理、およびリソースを有効にする方法を示します。

リソースタイプを登録します。

```
# clresourcetype -f <path-to-rtrfile> SUNW.Proxy_SMF_failover
```

アプリケーション用の rg1 というリソースグループを作成します。

```
# clresourcegroup create rg1
```

myfailoverres というフェイルオーバーアプリケーションリソースを作成します。

```
# clresource create -t SUNW.Proxy_SMF_failover -g rg1 \
-x proxied_service_instances="/usr/local/app/svc myfailoverres"
```

/usr/local/app/svc はテキストファイルです。

リソースグループ rg1 の管理:

```
# clresourcegroup manage rg1
```

myfailoverres リソースを有効にします。

```
# clresource enable myfailoverres
```

次のコマンドを使用して、アプリケーションのステータスをチェックします。

```
# clresource status
```

735 ページのpmfadm(1M), 691 ページのscha_resource_get(1HA),
333 ページのclresourcetype(1CL), 273 ページのclresource(1CL),
305 ページのclresourcegroup(1CL), Unresolved link to " attributes5",
1287 ページのr_properties(5)

Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド "

名前

SUNW.Proxy_SMF_multimaster, Proxy_SMF_multimaster — マルチマスター SMF サービスのプロキシ作成のためのリソースタイプ

SUNW.Proxy_SMF_multimaster リソースタイプは、マルチマスターサービス管理機能 (SMF) サービスのプロキシを表します。

SUNW.proxysmfmultimaster リソースタイプに対して定義される標準プロパティーと拡張プロパティーについては、これ以降のサブセクションで説明します。これらのプロパティーを SUNW.Proxy_SMF_multimaster リソースタイプのインスタンスに設定するには、clresource コマンド (273 ページのclresource(1CL)) を使用します。

標準プロパティー

次のリソースプロパティーについては、1287 ページのr_properties(5) を参照してください。

Start_timeout
: デフォルト: 3600
: 最小: 60

Stop_timeout
: デフォルト: 3600
: 最小: 60

Init_timeout
: デフォルト: 3600
: 最小: 60

Boot_timeout
: デフォルト: 3600
: 最小: 60

Fini_timeout
: デフォルト: 3600
: 最小: 60

Validate_timeout
: デフォルト: 3600
: 最小: 60

`Failover_mode`

デフォルト: `SOFT`

調整可能: `Anytime`

`R_description`

デフォルト: `""`

調整可能: `Anytime`

`Retry_count`

デフォルト: `2`

最小: `0`

最大: `3`

調整可能: `Anytime`

`Retry_interval`

デフォルト: `300`

最大: `3600`

調整可能: `Anytime`

`Through_probe_interval`

デフォルト: `60`

調整可能: `Anytime`

拡張プロパティー

`Proxied_service_instances`

リソースによってプロキシされる SMF サービスに関する情報を含みます。値はプロキシされるすべての SMF サービスを含むファイルのパスです。ファイル内の各行は 1 つの SMF サービス専用で、`svc fmri` および対応するサービスマニフェストファイルのパスを指定します。たとえば、リソースが `restarter_svc_test_1:default` と `restarter_svc_test_2:default` の 2 つのサービスを管理する必要がある場合、このファイルには次の 2 行を含めるようにしてください。

```
<svc:/system/cluster/restarter_svc_test_1:default>,
  svc:/system/cluster/restarter_svc_test_1:default>,
  </var/svc/manifest/system/cluster/restarter_svc_test_1.xml>

<svc:/system/cluster/restarter_svc_test_2:default>,
  </var/svc/manifest/system/cluster/restarter_svc_test_2.xml>
```

注記 - 上記エントリはそれぞれ、本来 1 行になるべきものです。ここでは読みやすさを考慮して複数の行に分けています。

デフォルト: ""

調整可能: When disabled

例

この例では、SUNW.Proxy_SMF_multimaster リソースタイプを登録して、アプリケーション用のリソースグループを作成して、マルチマスターアプリケーションリソースを作成して、リソースグループを管理状態にして、リソースを有効にする方法を示します。

リソースタイプを登録します。

```
# clresourcetype -f <path-to-rtrfile> SUNW.Proxy_SMF_multimaster
```

アプリケーション用の rg1 というリソースグループを作成します。

```
# clresourcegroup create rg1
```

mymultimasterres というマルチマスターアプリケーションリソースを作成します。

```
# clresource create -t SUNW.Proxy_SMF_multimaster -g rg1 \
-x proxied_service_instances="/usr/local/app/svc" mymultimasterres
```

/usr/local/app/svc はテキストファイルです。

リソースグループ rg1 の管理:

```
# clresourcegroup manage rg1
```

mymultimasterres リソースを有効にします。

```
# clresource enable mymultimasterres
```

次のコマンドを使用して、アプリケーションのステータスをチェックします。

```
# clresource status
```

735 ページのpmfadm(1M), 691 ページのscha_resource_get(1HA),
333 ページのclresourcetype(1CL), 273 ページのclresource(1CL),
305 ページのclresourcegroup(1CL), Unresolved link to " attributes5",
1287 ページのr_properties(5)

Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド "

名前

SUNW.Proxy_SMF_scalable, Proxy_SMF_scalable — スケーラブル SMF サービスのプロキシ作成のためのリソースタイプ

SUNW.Proxy_SMF_scalable リソースタイプは、スケーラブルなサービス管理機能 (Service Management Facility、SMF) サービスのプロキシを表します。

SUNW.proxysmfscalable リソースタイプに対して定義される標準プロパティーと拡張プロパティーについて、これ以降のサブセクションで説明します。SUNW.Proxy_SMF_scalable リソースタイプのインスタンスに対してこれらのプロパティーを設定するには、clresource コマンドを使用します。

標準プロパティー

次のリソースプロパティーについては、1287 ページのr_properties(5) を参照してください。

Start_timeout
: デフォルト: 3600
: 最小: 60

Stop_timeout
: デフォルト: 3600
: 最小: 60

Init_timeout
: デフォルト: 3600
: 最小: 60

Boot_timeout
: デフォルト: 3600
: 最小: 60

Fini_timeout
: デフォルト: 3600
: 最小: 60

Validate_timeout
: デフォルト: 3600
: 最小: 60

`Failover_mode`

デフォルト: `SOFT`

調整可能: 常時

`R_description`

デフォルト: `""`

調整可能: 常時

`Retry_count`

デフォルト: `2`

最小: `0`

最大: `3`

調整可能: 常時

`Retry_interval`

デフォルト: `300`

最大: `3600`

調整可能: 常時

`Through_probe_interval`

デフォルト: `60`

調整可能: 常時

拡張プロパティー

`Proxied_service_instances`

リソースによってプロキシされる SMF サービスに関する情報を含みます。値はプロキシされるすべての SMF サービスを含むファイルのパスです。ファイル内の各行は 1 つの SMF サービス専用で、`svc fmri` および対応するサービスマニフェストファイルのパスを指定します。たとえば、リソースが `restarter_svc_test_1:default` と `restarter_svc_test_2:default` の 2 つのサービスを管理する必要がある場合、このファイルには次の 2 行を含めるようにしてください。

```
<svc:/system/cluster/restarter_svc_test_1:default>,
  svc:/system/cluster/restarter_svc_test_1:default>,
  </var/svc/manifest/system/cluster/restarter_svc_test_1.xml>

<svc:/system/cluster/restarter_svc_test_2:default>,
  </var/svc/manifest/system/cluster/restarter_svc_test_2.xml>
```

注記 - 上記エントリはそれぞれ、本来 1 行になるべきものです。ここでは読みやすさを考慮して複数の行に分けています。

デフォルト: ""

調整可能: When disabled

例

この例では、SUNW.Proxy_SMF_scalable リソースタイプを登録して、アプリケーション用のリソースグループを作成して、負荷分散アプリケーションリソースを作成して、リソースグループを管理状態にして、そのリソースをすべて有効にして、そのリソースをオンラインにする方法を示します。

リソースタイプを登録します。

```
# clresourcetype -f <path-to-rtrfile> SUNW.Proxy_SMF_scalable
```

アプリケーション用の rg1 というリソースグループを作成します。

```
# clresourcegroup create rg1
```

myloadbalancedres という負荷分散アプリケーションリソースを作成します。

```
# clresource create -t SUNW.Proxy_SMF_scalable -g rg1 \
-x proxied_service_instances="/usr/local/app/svc myloadbalancedres"
```

/usr/local/app/svc はテキストファイルです。

リソースグループ rg1 の管理:

```
# clresourcegroup manage rg1
```

myloadbalancedres リソースを有効にします。

```
# clresource enable myloadbalancedres
```

次のコマンドを使用して、アプリケーションのステータスをチェックします。

```
# clresource status
```

735 ページのpmfadm(1M), 691 ページのscha_resource_get(1HA),
333 ページのclresourcetype(1CL), 273 ページのclresource(1CL),
305 ページのclresourcegroup(1CL), Unresolved link to " attributes5",
1287 ページのr_properties(5)

Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド "

名前

r_properties — リソースプロパティー

次の情報では、Oracle Solaris Cluster ソフトウェアが定義する標準リソースプロパティーについて説明します。標準プロパティーは、使用されるリソースタイプ全体で共通する意味を持っています。この説明は、データサービスの開発者を対象としたものです。特定のデータサービスについては、そのデータサービスのマニュアルページを参照してください。

各リソースタイプは、標準プロパティーに加えて、拡張プロパティーと呼ばれる独自のタイプ固有のリソースプロパティーを定義できます。リソースタイプ登録 (RTR) ファイルでは、標準プロパティーと拡張プロパティーの両方が宣言されます。RTR ファイルは、クラスタ管理者が Oracle Solaris Cluster ソフトウェアでデータサービスを登録するときのデータサービスの初期構成を定義します。

クラスタ管理者がノード単位またはクラスタ全体に対して拡張プロパティーを設定できるように指定できます。ただし、クラスタ管理者が標準プロパティーで実行できる同じ操作を (RTR ファイルで) 指定することはできません。標準プロパティーは、すべてのノードまたは特定のノードに暗黙に適用される場合があります。標準プロパティーがすべてのノードに適用されるのか、または特定のノードにのみ適用されるのかは、各標準プロパティーの特定の定義によって異なります。

RTR ファイルの詳細は、1251 ページの`rt_reg(4)` のマニュアルページを参照してください。リソースプロパティーに設定できる個々の属性については、1271 ページの`property_attributes(5)` のマニュアルページを参照してください。

注記 - Oracle Solaris Cluster ソフトウェアのネットワーク負荷分散機能を使用する前に、リソースでスケーラブルリソースプロパティーを TRUE に設定する必要があります。スケーラブルリソースでは、`Affinity_timeout`、`Generic_affinity`、`Load_balancing_policy`、`Load_balancing_weights`、`Conn_threshold`、および `Weak_affinity` プロパティーを使用できます。

リソースタイプの中には、ネットワーク負荷分散を使用せずに複数のノードで動作できるものがあります。このようなリソースの `Scalable` プロパティーは `FALSE` に設定され、前述の追加プロパティーを使用しません。

標準リソースプロパティーのカテゴリ

必須　　クラスタ管理者は、管理ユーティリティーを使ってリソースを作成するとき、必ず値を指定しなければなりません。

任意　　クラスタ管理者がリソースグループの作成時に値を指定しない場合、システムのデフォルト値が使用されます。

条件付き　　Resource Group Manager (RGM) は、RTR (Resource Type Registration) ファイルで宣言されている場合だけプロパティーを作成します。宣言されていない場合プロパティーは存在せず、クラスタ管理者はこれを利用できません。RTR ファイルで宣言されている条件付きのプロパティーは、デフォルト値が RTR ファイル内で指定されているかどうかによって、必須または任意になります。詳細については、各条件付きプロパティーの説明を参照してください。

照会のみ　　管理ツールで直接設定できません。

クラスタ管理者は、次のコマンドを使用して、すべての調整可能なプロパティーを編集できます。

```
# clresource set -p property=
new-value  resource
```

標準リソースプロパティーの説明

注記 - プロパティー名 (`Affinity_timeout` や `Cheap_probe_interval` など) には、大小文字の区別はありません。プロパティー名を指定する際には、大文字と小文字を任意に組み合わせることができます。

`Affinity_timeout` (`integer`)

時間の長さ (秒) で、リソース内のサービスのクライアント IP アドレスからの接続はこの間に同じサーバーノードに送信されます。

このプロパティーに `-1` を設定した場合、すべての接続が同じノードに送信されます。このプロパティーに **`0`** を設定した場合、オープンされているすべての接続が同じノードに送信されます。このプロパティーに *n* を設定した場合、最後の接続がクローズされてから *n* 秒間の間、新しい接続はすべて、最後の接続と同じノードに送信されます。

どのような場合も、障害が発生した結果、サーバーノードがクラスタに対して何の措置も取らない場合、新しいサーバーノードが選択されます。

このプロパティーは、`Load_balancing_policy` が `Lb_sticky` または `Lb_sticky_wild` の場合にかぎり有効です。さらに、`Weak_affinity` を `False` (デフォルト値) に設定する必要があります。

このプロパティーは、スケーラブルサービス専用です。

カテゴリ　　条件付き/任意

デフォルト　　0

調整可能　　すべての時間

`Application_user` (`string`)

リソースに関連するアプリケーションプログラムの実行に使用する Oracle Solaris ユーザー名。

リソースメソッドまたはモニターで実行されるアプリケーションプログラムは、特定のエージェントの実装方法に応じて、root または root 以外のユーザー (「アプリケーションユーザー」) として実行される場合があります。`application_user` リソースプロパティーはすべてのリソースに存在するわけではなく、このプロパティーを設定できると宣言するリソースタイプにのみ存在します。

`application_user` リソースプロパティーを宣言するリソースタイプは通常、663 ページの`scha_check_app_user`(1HA) インタフェースを使用して、アプリケーションプログラムの実行可能ファイルの所有権およびアクセス権の追加チェックを実行するエージェントです。実行可能なアプリケーションプログラムが root で所有されていないが、root で実行される場合、または実行可能ファイルに group または world 書き込み権限がある場合、セキュアでない状態になります。この場合、`resource_security` プロパティーが `SECURE` に設定されていると、アプリケーションプログラムの実行は実行時に失敗し、エラーが返されます。`resource_security` がその他の設定であれば、アプリケーションプログラムは警告メッセージとともに実行を許可されます。

`application_user` プロパティーを宣言するリソースタイプは、`resource_security` クラスタプロパティーの設定に従って、アプリケーションプログラムの実行のユーザー ID を設定します。`resource_security` が `COMPATIBILITY` に設定されている場合、`application_user` リソースプロパティーの設定は無視され、アプリケーションユーザーは呼び出し元 (通常は root) の実効ユーザー ID になります。この動作は以前のリリースの Oracle Solaris Cluster と互換性があります。

`resource_security` が `OVERRIDE` に設定されている場合、`application_user` プロパティーは無視され、アプリケーションユーザーはアプリケーションプログラムの実行可能ファイルの所有者になります。

`resource_security` が `SECURE` または `WARN` に設定されている場合、アプリケーションユーザーは `application_user` リソースプロパティーの値になりますが、`application_user` が未設定または空の場合、アプリケーションユーザーはアプリケーションプログラムの実行可能ファイルの所有者になります。

RTR ファイル内で `Tunable` 属性が指定されていない場合は、プロパティーの調整機能は、`When_disabled` になります。

カテゴリ	条件付き/任意
デフォルト	空の文字列
調整可能	無効時

`Cheap_probe_interval` (`integer`)

リソースの即時障害検証の呼び出しの間隔 (秒数)。このプロパティーは、RGM のみが作成でき、RTR ファイル内で宣言されている場合は、クラスタ管理者がこのプロパティーを利用できます。

RTR ファイル内でデフォルト値が指定されている場合、このプロパティーは任意です。リソースタイプファイル内で Tunable 属性が指定されていない場合は、プロパティーの Tunable 値は、When_disabled (無効にするとき) になります。

カテゴリ	条件付き
デフォルト	上記を参照
調整可能	無効時

CheckActivePortInstances (boolean)

Port_list プロパティーで指定されるポートすべてにアクティブなリスニング (待機) プロセスがあるわけではない場合は、ノードがスケーラブルなサービスに参加するかどうかを決定し、ロードバランサからのクライアントリクエストを受信します。このプロパティーは、スケーラブルなサービス専用です。

次の値がサポートされています:

- FALSE (デフォルト) – 少なくとも 1 つのポートにアクティブなリスニングプロセスがある場合、ノードはスケーラブルなサービスに参加します。
- TRUE – すべてのポートにアクティブなリスニングプロセスがある場合、ノードはスケーラブルなサービスに参加します。

カテゴリ	条件付き/任意
デフォルト	False
調整可能	無効時

Conn_threshold (integer)

Round_robin 負荷分散が有効な場合にサポートされるアクティブな接続またはクライアントの最大数。TCP 接続は、接続のエンドポイントがサーバーノード上に存続している場合にアクティブとみなされます。トラフィックフローが UDP セッションのアクティブなタイムアウト枠の設定内にある場合 (udp_session_timeout クラスタプロパティーを参照)、UDP セッションはアクティブとみなされます。

カテゴリ	任意
デフォルト	10000
調整可能	無効時

Failover_mode (enum)

リソースが正常に開始または停止できなかった場合、またはリソースモニターが正常ではないリソースを検出し、その結果再起動またはフェイルオーバーを要求する場合に RGM が取る回復アクションを変更します。

NONE、SOFT、または HARD (メソッドの失敗)

これらの設定が影響するのは、起動または停止メソッド (Prenet_start、Start、Monitor_stop、Stop、Postnet_stop) が失敗した場合のフェイルオーバー動作のみです。RESTART_ONLY および LOG_ONLY 設定も、リソースモニターが scha_control コマンドまたは scha_control() 関数の実行を開始できるかどうかに影響を与えます。

NONE は、前述の起動メソッドまたは停止メソッドが失敗する場合に RGM が何の復旧アクションも行わないことを示します。SOFT または HARD は、Start または Prenet_start メソッドが失敗した場合、RGM がリソースのグループを別のノードに再配置することを示します。Start または Prenet_start の失敗に関しては、SOFT と HARD は同じになります。

停止メソッド (Monitor_stop、Stop、または Postnet_stop) の失敗に関しては、SOFT は NONE と同じになります。これらの停止メソッドのいずれかが失敗したときに Failover_mode が HARD に設定されている場合、RGM はノードをリブートして、強制的にリソースグループをオフライン状態にします。RGM は、別のノードでグループの起動を試みることが可能になります。ただし、リソースグループが clresourcegroup quiesce サブコマンドによって休止されている場合は、Failover_mode が HARD であるときに停止メソッドが失敗しても、ノードはリブートされません。この場合、リソースは代わりに STOP_FAILED 状態に移動します。

RESTART_ONLY または LOG_ONLY

起動メソッドまたは停止メソッドが失敗するとフェイルオーバー動作に影響を与える NONE、SOFT、HARD とは異なり、RESTART_ONLY と LOG_ONLY はすべてのフェイルオーバー動作に影響を与えます。フェイルオーバー動作には、モニター起動 (scha_control) によるリソースおよびリソースグループの再起動や、リソースモニターによって開始されるギブオーバーなどがあります。

RESTART_ONLY は、モニターが scha_control を実行してリソースまたはリソースグループを再起動できることを意味します。RGM では、Retry_interval の間に Retry_count 回数だけ再起動を試行できます。試行回数が Retry_count を超えると、それ以上の再起動は許可されません。

注記 - Retry_count の負の値は、リソースタイプによっては全部適用できませんが、一部適用できます。リソースを無制限に再起動できることを指定します。より確実に無制限の再起動を指定するには、次の手順を実行します。

- Retry_interval に 1 や 0 などの小さい値を指定します。
- Retry_count に 1000 などの大きい値を指定します。

リソースタイプが Retry_count および Retry_interval プロパティーを宣言しない場合は、リソースは回数の制限なく再起動できます。

Failover_mode が LOG_ONLY に設定されている場合、リソースの再起動またはギブオーバーは許可されません。Failover_mode を LOG_ONLY に設定するの

は、Failover_mode を RESTART_ONLY に設定し、Retry_count をゼロに設定するのと同じことです。

RESTART_ONLY または LOG_ONLY (メソッドの失敗)

Prenet_start、Start、Monitor_stop、Stop、または Postnet_stop メソッドが失敗した場合、RESTART_ONLY と LOG_ONLY は NONE と同じことです。つまり、データソースのフェイルオーバーやリブートはどちらも行われません。

データサービスに対する Failover_mode 設定の影響

Failover_mode の各設定がデータサービスに及ぼす影響は、データサービスがモニターされているかどうか、およびデータサービスが Data Services Development Library (DSDL) に基づいているかどうかによって決まります。

- データサービスがモニタリングの対象となるのは、そのサービスが Monitor_start メソッドを実装しており、かつリソースのモニタリングが有効になっている場合です。RGM は、リソースそれ自体を起動したあとで Monitor_start メソッドを実行することにより、リソースモニターを起動します。リソースモニターはリソースが正常であるかどうかを検証します。検証が失敗した場合、リソースモニターは scha_control() 関数を呼び出すことにより、再起動またはフェイルオーバーを要求できます。DSDL ベースのリソースの場合、検証によりデータサービスの部分的な障害 (機能低下) または完全な障害が明らかになる場合があります。部分的な障害が繰り返し蓄積されると、完全な障害になります。
- データサービスが Monitor_start メソッドを提供していない場合やリソースのモニタリングが無効である場合、データサービスはモニタリングされません。
- DSDL ベースのデータサービスには、Agent Builder や GDS により開発されたデータサービス、または DSDL を直接使用して開発されたデータサービスが含まれます。HA Oracle など一部のデータサービスは、DSDL を使用せずに開発されています。

NONE、SOFT、または HARD (検証機能の失敗)

Failover_mode が NONE、SOFT、または HARD に設定されていて、データサービスがモニター対象の DSDL ベースのサービスであり、検証が完全に失敗した場合、モニターは scha_control() 関数を呼び出してリソースの再起動をリクエストします。検証が失敗し続ける場合、Retry_interval の間、Retry_count に指定されている最大の回数を限度にリソースの再起動が行われます。Retry_count の再起動数に到達後も検証がふたたび失敗した場合、モニターは別のノードに対してリソースのグループのフェイルオーバーをリクエストします。

Failover_mode が NONE、SOFT、または HARD に設定した場合、データサービスがモニター対象外の DSDL ベースのサービスであり、検出される障害はリソースのプロセスツリーの終了のみです。リソースのプロセスツリーが終了すると、リソースが再起動されます。

データサービスが DSDL ベースのサービスではない場合、再起動またはフェイルオーバー動作は、リソースモニターがどのようにコード化されているかによって決まります。たとえば Oracle リソースモニターは、リソースまたはリソースグループを再起動するか、リソースグループのフェイルオーバーを行うことで回復します。

`RESTART_ONLY` (検証機能の失敗)

`Failover_mode` が `RESTART_ONLY` に設定されていて、データサービスがモニター対象の DSDL ベースのサービスであり、検証が完全に失敗した場合、リソースは `Retry_interval` 内で `Retry_count` の回数再起動されます。ただし、`Retry_count` の回数を超えると、リソースモニターは終了し、リソースのステータスを `FAULTED` に設定して、ステータスメッセージ「`Application faulted, but not restarted. Probe quitting.`」を生成します。この時点でモニタリングはまだ有効ですが、リソースがクラスタ管理者により修復および再起動されるまで、リソースは事実上モニタリング対象外になります。

`Failover_mode` を `RESTART_ONLY` に設定した場合、データサービスがモニター対象外の DSDL ベースのサービスであり、プロセスツリーが停止した際は、リソースは再起動*されません*。

モニター対象データサービスが DSDL ベースのデータサービスではない場合、回復動作はリソースモニターがどのようにコード化されているかに依存します。`Failover_mode` が `RESTART_ONLY` に設定されている場合、リソースまたはリソースグループは、`Retry_interval` 内で `Retry_count` の回数 `scha_control()` 関数を呼び出すことで再起動できます。リソースグループが `Retry_count` を超過すると、再起動の試行が失敗します。モニターが `scha_control()` を呼び出してフェイルオーバーを要求する場合、その要求も同様に失敗します。

`LOG_ONLY` (検証機能の失敗)

`Failover_mode` がデータサービスに対して `LOG_ONLY` に設定されている場合、リソースまたはリソースグループの再起動またはグループのフェイルオーバーを行うためのすべての `scha_control()` リクエストは不可能になります。データサービスが DSDL ベースである場合、検証が完全に失敗した場合メッセージが記録されますが、リソースは再起動されません。`Retry_interval` 内で `Retry_count` の回数超えて検証が完全に失敗した場合、リソースモニターが終了し、リソースステータスが `FAULTED` になり、ステータスメッセージ「`Application faulted, but not restarted. Probe quitting.`」が生成されます。この時点でモニタリングはまだ有効ですが、リソースがクラスタ管理者により修復および再起動されるまで、リソースは事実上モニタリング対象外になります。

`Failover_mode` が `LOG_ONLY` に設定した場合、データサービスがモニター対象外の DSDL ベースのサービスであり、プロセスツリーが停止した際は、メッセージが記録されますが、リソースは再起動されません。

モニター対象データサービスが DSDL ベースのデータサービスではない場合、回復動作はリソースモニターがどのようにコード化されているかに依存します。`Failover_mode` が `LOG_ONLY` に設定されている場合、リソースまたはリソースグループの再起動またはグループのフェイルオーバーを行うためのすべての `scha_control()` リクエストは失敗します。

カテゴリ　　任意

デフォルト　　NONE

調整可能　　すべての時間

`Global_zone_override` (boolean)

このプロパティーは、RTR ファイルで `Global_zone=TRUE` プロパティーを設定しているリソースタイプにのみ許可されます。`Global_zone_override` プロパティーの設定は、特定のリソースのりリソースタイププロパティー `Global_zone` の値をオーバーライドします。詳細は、1335 ページの`rt_properties(5)` のマニュアルページを参照してください。

`Global_zone_override` プロパティーを `FALSE` に設定すると、リソースグループが構成されている非大域ゾーンで、リソースメソッドが強制的に実行されるようになり、これは、`Global_zone` プロパティーが `TRUE` に設定されている場合に、通常は常に大域ゾーンで実行されるのとは異なります。

RTR ファイル内でデフォルト値が指定されている場合、このプロパティーは任意です。

RTR ファイル内で `Tunable` 属性が指定されていない場合は、プロパティーの `Tunable` 値は、`At_creation` になります。RTR ファイル内の `Tunable` 属性は、`At_creation`、`When_disabled`、または `Anytime` に設定できます。

注記 - RTR ファイル内で `Tunable` 属性を `Anytime` に設定するときには、十分注意してください。`Global_zone_override` プロパティーへの変更は、リソースがオンラインであってもただちに有効になります。たとえば、調整可能性 `Global_zone_override` が `ANYTIME` に設定され、`Global_zone_override` プロパティーが現在、非大域ゾーン内に構成されているリソースに対して `FALSE` に設定されているとします。そのリソースがオンラインに切り換えられると、起動メソッドが非大域ゾーンで実行されます。その後、`Global_zone_override` プロパティーを `TRUE` に設定し、リソースがオフラインに切り替わると、停止メソッドが大域ゾーンで実行されます。メソッドのコードでこの可能性に対処する必要があります。メソッドのコードでこの可能性に対処しない場合は、代わりに `Tunable` 属性を `When_disabled` または `At_creation` に設定する必要があります。

カテゴリ　　条件付き/任意

デフォルト　　TRUE

調整可能　　作成時

`Load_balancing_policy` (string)

使用する負荷分散ポリシーを定義する文字列。このプロパティーは、スケーラブルサービス専用です。RTR ファイルに `Scalable` プロパティーが宣言されている場合、RGM は自動的にこのプロパティーを作成します。

`Load_balancing_policy` には次の値を設定できます。

- `Lb_weighted` (デフォルト)。`Load_balancing_weights` プロパティーに設定されている重みにより、さまざまなノードに負荷が分散されます。
- `Lb_sticky`. アプリケーションリソースの構成時に複数のポートが認識されます。スケーラブルサービスの指定のクライアント (クライアントの IP アドレスで識別される) は、常に同じクラスタノードに送信されます。
- `Lb_sticky_wild`. ポート番号をあらかじめ把握することはできず、ポート番号は動的に割り当てられます。指定のクライアント (クライアントの IP アドレスで識別される) は、ワイルドカードスティッキーサービスの IP アドレスに接続され、送信時に使用されるポート番号とは無関係に、常に同じクラスタノードに送信されます。

カテゴリ　条件付き/任意

デフォルト　`Lb_weighted`

調整可能　作成時

`Load_balancing_weights` (`string_array`)

このプロパティーは、スケーラブルサービス専用です。RTR ファイルに `Scalable` プロパティーが宣言されている場合、RGM は自動的にこのプロパティーを作成します。形式は、*weight*`@`*node*`,`*weight*`@`*node*`...` で、*weight* は、指定の *node* に対する負荷分散の割合を示す整数になります。ノードに分散される負荷の割合は、すべてのウエイトの合計でこのノードのウエイトを割った値になります。たとえば、`1@1,3@2` という指定では、ノード 1 が負荷の 1/4、ノード 2 が 3/4 を引き受けることになります。空の文字列 (“”) では、デフォルトの均一な配分に設定されます。明示的にウエイトを割り当てられていないノードのウエイトは、デフォルトで 1 になります。ノードには、負荷を割り当てないウェイト 0 を指定できます。

`Tunable` 属性がリソースタイプファイルに指定されていない場合は、プロパティーの `Tunable` 値は `Anytime` (任意の時点) になります。このプロパティーを変更すると、新しい接続時にのみ分散が変更されます。

カテゴリ　条件付き/任意

デフォルト　NULL

調整可能　すべての時間

method`_timeout` for each callback method (`integer`)

RGM がメソッドの呼び出しに失敗したと判断するまでの時間 (秒)。

注記 - `Max` 属性を使用してメソッドタイムアウトの最大値を指定できません。同様に、最小値ゼロ (`Min=0`) を指定することもできません。

カテゴリ　条件付き/任意

デフォルト　RTR ファイルにメソッド自体が宣言されている場合は 3,600 秒 (1 時間)

調整可能　すべての時間

`Monitored_switch` (`enum`)

このプロパティーを直接設定できません。より正確には、このプロパティーは、特定のノード上またはクラスタ全体で、RGM によって `Enabled` または `Disabled` に設定されます。RGM は、クラスタ管理者が特定のノード上またはクラスタ全体で管理ユーティリティーを使用してモニターを有効または無効にする場合に、このように実行します。無効の場合、モニターがふたたび有効にされるまで、そのリソースに対して `Monitor_start` メソッドが呼び出されることはありません。リソースがモニターコールバックメソッドを持たない場合、このプロパティーは `Disabled` と評価されます。

カテゴリ　照会のみ

デフォルト　リソースタイプがモニターメソッドを持っている場合は有効、それ以外の場合は無効です。

調整可能　説明を参照

`Network_resources_used` (`string_array`)

このリソースが依存関係を持っている論理ホスト名または共有アドレスリソースのリスト。このリストには、プロパティー `Resource_dependencies`、`Resource_dependencies_weak`、`Resource_dependencies_restart`、または `Resource_dependencies_offline_restart` にあるすべてのネットワークアドレスリソースが含まれます。

RTR ファイルに `Scalable` プロパティーが宣言されている場合、RGM は自動的にこのプロパティーを作成します。`Scalable` プロパティーが RTR ファイルで宣言されていない場合、`Network_resources_used` は RTR ファイルで明示的に宣言されていないかぎり使用できません。

`Network_resources_used` プロパティーに値を割り当てなかった場合、このプロパティーの値は、リソース依存関係プロパティーの設定に基づいて、RGM により自動的に更新されます。このプロパティーを直接設定する必要はありません。その代わりに、`Resource_dependencies`、`Resource_dependencies_offline_restart`、`Resource_dependencies_restart`、または `Resource_dependencies_weak` プロパティーを設定します。ノード単位の依存関係が指定されている場合、`Network_resources_used` プロパティーの派生値には、ローカルノードで有効な依存関係のみが含まれます。値はノードごとに異なる場合があります。

旧リリースの Oracle Solaris Cluster ソフトウェアとの互換性を維持するために、`Network_resources_used` プロパティーの値を直接設定することもできます。`Network_resources_used` プロパティーの値を直接設定した場合は、`Network_resources_used` プロパティーの値がリソース依存関係プロパティーの設定から派生されることはありません。`Network_resources_used` プロパティーにリソース名を追加する場合、そのリソース名は自動的に `Resource_dependencies` プロパティー

にも追加されます。依存関係を解除するには、Network_resources_used プロパティーから依存関係を削除する方法しかありません。ネットワークリソースの依存関係が元々 Resource_dependencies プロパティーに追加されたものなのか Network_resources_used プロパティーに追加されたものなのかがわからない場合は、両方のプロパティーから依存関係を削除します。たとえば、次のコマンドは、リソース r1 のネットワーク資源 r2 に対する依存関係を削除します。依存関係が Network_resources_used プロパティーと Resource_dependencies プロパティーのどちらに追加されているかは問いません。

```
# clresource set -p Network_resources_used-=r2 -p Resource_dependencies-=r2 r1
```

簡潔になるように、Network_resources_used プロパティーに値を設定しないようにしてください。リソース依存関係プロパティーのみを設定し、Network_resources_used プロパティーを読み取り専用のプロパティーとして扱います。

カテゴリ　条件付き/任意

デフォルト　空のリスト

調整可能　すべての時間

各クラスタノード上の Num_resource_restarts (integer)

過去 *n* 秒以内にこのリソースで発生した再起動リクエストの数 (*n* は、Retry_interval プロパティーの値)。

再起動要求は、次に示す呼び出しのいずれかです。

- RESOURCE_RESTART 引数を持つ scha_control コマンド
- SCHA_RESOURCE_RESTART 引数を持つ scha_control() 関数
- RESOURCE_IS_RESTARTED 引数を持つ scha_control コマンド
- scha_control() 引数を持つ SCHA_RESOURCE_IS_RESTARTED 関数

リソースが次のいずれかを実行した場合、RGM は、ある特定のノード上にある特定のリソースに対して再起動カウンタを必ず 0 にリセットします:

- GIVEOVER 引数を持つ scha_control コマンド
- SCHA_GIVEOVER 引数を持つ scha_control() 関数

カウンタは、ギブオーバーの試行が成功した場合でも失敗した場合でもリセットされます。

リソースタイプが Retry_interval プロパティーを宣言していない場合、このタイプのリソースに Num_resource_restarts プロパティーを使用できません。

カテゴリ　照会のみ

デフォルト　デフォルトなし

調整可能　　説明を参照

各クラスタノード上の Num_rg_restarts (integer)

過去 *n* 秒以内にこのリソースに対して発生したリソースグループ再起動リクエストの数 (*n* は、Retry_interval プロパティーの値)。

リソースグループ再起動要求は、次に示す呼び出しのいずれかです。

- RESTART 引数を持つ scha_control コマンド
- scha_control() 引数を持つ SCHA_RESTART 関数

リソースタイプが Retry_interval プロパティーを宣言していない場合、このタイプのリソースに Num_resource_restarts プロパティーを使用できません。

カテゴリ　　照会のみ

デフォルト　　デフォルトなし

調整可能　　説明を参照

On_off_switch (enum)

このプロパティーを直接設定できません。より正確には、このプロパティーは、特定のノード上またはクラスタ全体で、RGM によって Enabled または Disabled に設定されます。RGM は、クラスタ管理者が特定のノード上またはクラスタ全体のいずれかで管理ユーティリティーを使用してリソースを有効または無効にする場合に、このように実行します。無効に設定されると、ふたたび有効に設定されるまで、リソースはコールバックを呼び出しません。

カテゴリ　　照会のみ

デフォルト　　無効

調整可能　　説明を参照

Outgoing_Connection(boolean)

クラスタ外部のサーバーへの外部リクエストを起動するときに、スケーラブルなサービスが仮想ネットワークアドレスを使用するかどうかを指定します (Network_resources_used プロパティーを参照)。ロードバランサは、受信応答が起動ノードに転送されたことを確認します。

注記 - ある時点で、1 台のサーバーへのリクエストを起動できるノードは 1 つのみです。

このプロパティーは、Generic_Affinity が TRUE に、Load_balancing_policy が LB_STICKY_WILD に設定されているスケーラブルなサービスのみに適用されます。次の値がサポートされています:

- FALSE (デフォルト) – スケーラブルなサービスは、Network_resources_used プロパティーで指定された仮想ネットワークアドレスを使用して、外部サーバーに向かうリクエストを起動しません。
- TRUE – スケーラブルなサービスは、Network_resources_used プロパティーで指定された仮想ネットワークアドレスを使用して、外部サーバーに向かうリクエストを起動します。ロードバランサは、受信応答を起動ノードに転送します。

カテゴリ	条件付き/任意
デフォルト	False
調整可能	作成時

Port_list (string_array)

サーバーが待機するポート番号をコンマで区切ったリストです。各ポート番号には、スラッシュ (/) と、そのポートで使用されるプロトコルが付加されます (たとえば、Port_list=80/tcp や Port_list=80/tcp6,40/udp6 など)。

指定できる可能性があるプロトコルは次のとおりです。

- tcp、TCP IPv4 の場合のみ
- tcp6、TCP IPv4 と TCP IPv6 の両方の場合
- udp、UDP IPv4 の場合のみ
- udp6、UDP IPv4 と UDP IPv6 の両方の場合

Scalable プロパティーが RTR ファイルで宣言されている場合、RGM は自動的に Port_list を作成します。それ以外の場合、このプロパティーは RTR ファイルで明示的に宣言されていないかぎり使用できません。

このプロパティーを Oracle Solaris Cluster HA for Apache で使用するように設定する方法については、Unresolved link to " Oracle Solaris Cluster Data Service for Apache Guide "を参照してください。

カテゴリ	条件付き/必須
デフォルト	デフォルトなし
調整可能	すべての時間

Pre_evict (boolean)

より優先順位の低いリソースグループの削除は、切り替えられるグループに対して別のグループが強い否定的なアフィニティーを宣言している場合や、ノード上で強い負荷制限値を超えている場合は、リソースグループの切り替えに関連して実行される可能性があります。

このプロパティーによって、RGM が、このリソースを含むリソースグループのスイッチオーバーを開始する前にリソースグループの削除を実行しようとするかどうかが決まります。

次の値がサポートされています:

- FALSE (デフォルト) – リソースグループの削除は、切り替えているリソースグループが、ターゲットノード上でオンラインへの移行を開始したときに実行されます。
- TRUE – リソースグループの削除は、切り替えているリソースグループがその現在のマスターからオフラインへの移行を開始する前に、スイッチオーバーターゲットノード上で実行されます。この設定は、Maximum_primaries プロパティーが 1 に設定されたリソースグループである単一マスターリソースグループに対してのみ有効です。

RTR ファイル内で Tunable 属性が指定されていない場合、このプロパティーの Tunable 値は When disabled です。

カテゴリ	任意
デフォルト	False
調整可能	無効時

R_description (string)

リソースの簡単な説明。

カテゴリ	任意
デフォルト	空の文字列
調整可能	すべての時間

Resource_dependencies (string_array)

このリソースが強い依存関係を持つ、同じリソースグループまたは異なるリソースグループ内のリソースのリスト。リスト内の任意のリソースの起動に失敗した場合、このリソースは起動されません。このリソースと、リストのリソースの 1 つが同時に起動されると、RGM は、リストのリソースが始動してからこのリソースを起動します。このリソースの Resource_dependencies リスト内のリソースが起動しない場合 (たとえば、リスト内のリソースのリソースグループがオフラインのままであったり、リスト内のリソースが Start_failed 状態にあったりする場合)、このリソースはオフラインのままです。このリソースが依存する別のリソースグループのリソースが起動しないために、このリソースがオフラインのままの状態である場合、このリソースのグループは Pending_online_blocked 状態に入ります。

このリソースが、リストのリソースと同時にオフラインにされる場合は、このリソースが停止されてから、リストのほかのリソースが停止されます。しかし、このリソースがオンラインのままであるか、停止に失敗すると、リストのリソースは停止されます。

デフォルトでは、リソースグループ内では、アプリケーションリソースがネットワークアドレスリソースに暗黙的に強いリソース依存関係を持ちます。詳細は、Implicit_network_dependencies in 1319 ページのrg_properties(5) のマニュアルページを参照してください。

同じリソースグループ内では、依存性の順序に従って Prenet_start メソッドが Start メソッドより先に実行されます。Postnet_stop メソッドは依存性の順序に従って Stop メソッドよりあとに実行されます。異なるリソースグループ内では、依存先のリソースが Prenet_start と Start を実行してから、依存しているリソースが Prenet_start を実行します。同様に、依存しているリソースが Stop と Postnet_stop を終了してから、依存先のリソースが Stop を実行します。

依存関係の範囲を指定するには、このプロパティーを指定するときに、次の修飾子を、中括弧 ({ }) やアットマーク (@) を含めてリソース名に付加します。

{ANY_NODE}

指定された依存関係を任意のノードに拡張します。依存関係の動作は、どのノードでも依存先のリソースに影響されます。依存しているリソースは、自分が起動する前に依存先のリソースがプライマリノードで起動するまで待機します。停止と再起動で同じような状況になります。

{FROM_RG_AFFINITIES}

リソースの依存関係の有効範囲が、そのリソースが属するリソースグループの RG_affinities 関係から派生するように指定します。依存しているリソースのリソースグループが、依存先のリソースのリソースグループに対して肯定的なアフィニティーを持っていて、同じノード上で起動または停止する場合、依存関係は {LOCAL_NODE} になります。このような肯定的なアフィニティーが存在しない場合、またはグループが別のノード上で起動する場合、依存関係は {ANY_NODE} になります。

{LOCAL_NODE}

指定される依存関係をノード単位に限定します。依存関係の動作は、同じノード上でのみ依存先のリソースに影響されます。依存しているリソースは、依存先のリソースが同じノードで起動されるまで待機します。停止と再起動で同じような状況になります。

{LOCAL_NODE} 依存リソースがフェイルオーバー (すなわち単一マスター) リソースグループに存在する場合、また、あるノードとの {LOCAL_NODE} 依存関係が満たされていない場合、リソースグループは、依存関係が満たされないノードで Pending_online_blocked 状態のままいるのではなく、{LOCAL_NODE} 依存関係が満たされている別のノードにフェイルオーバーする可能性があります。

@nodename

ほかのノードに影響しない特定のノードに制限される {LOCAL_NODE} 依存関係を指定します。これにより、リソースがクラスタのノードごとに異なる依存関係を持つことができます。*nodename* は、ノード名またはノード ID です。

たとえば、次のリストは、ノード node1 上のリソース res1 の依存関係、およびノード node2 上のリソース res2 の依存関係を示しています:

```
res1@node1,res2@node2
```

複数のノードに同じ依存関係を適用可能な場合は、各ノード名でリソース名を繰り返します。例:

```
myres@node1,myres@node2,myres@node3,...
```

同じリソースグループ内の 2 つのリソース間の依存関係は、常に {LOCAL_NODE} です。

修飾子を指定しない場合は、FROM_RG_AFFINITIES がデフォルトで使用されます。

691 ページのscha_resource_get(1HA)、1127 ページのscha_resource_get(3HA)、および 1017 ページのscds_property_functions(3HA) のマニュアルページのドキュメントには、修飾子を含む依存関係リスト、または修飾子を含まない依存関係リストを取得するための代わりのクエリーフォームが記載されています。

カテゴリ	任意
デフォルト	空のリスト
調整可能	すべての時間

Resource_dependencies_offline_restart (string_array)

このリソースがオフライン再起動の依存関係を持つ、同じリソースグループまたは異なるリソースグループ内のリソースのリスト。

このプロパティーは、Resource_dependencies とまったく同じように動作します。ただし、このリソースは、オフライン再起動依存関係リスト内のいずれかのリソースが停止すると、停止します。オフライン再起動依存関係リスト内のリソースがあとで再起動する場合は、このリソースも再起動します。

リスト内の任意のリソースの起動に失敗した場合、このリソースは起動されません。このリソースと、リストのリソースの 1 つが同時に起動されると、RGM は、リストのリソースが始動してからこのリソースを起動します。このリソースの Resource_dependencies リスト内のリソースが起動しない場合 (たとえば、リスト内のリソースのリソースグループがオフラインのままであったり、リスト内のリソースが Start_failed 状態にあったりする場合)、このリソースはオフラインのままです。このリソースが依存する別のリソースグループのリソースが起動しないために、このリソースがオフラインのままの状態である場合、このリソースのグループは Pending_online_blocked 状態に入ります。

このリソースが、リストのリソースと同時にオフラインにされる場合は、このリソースが停止されてから、リストのほかのリソースが停止されます。しかし、このリソースがオンラインのままであるか、停止に失敗すると、リストのリソースは停止されます。

ノード上にある「依存先の」リソースで障害が発生して、リソースを回復できない場合、RGM はそのノード上のそのリソースをオフラインにします。また、RGM は、すべての依存先のリソースのオフライン再起動依存リソースで再起動をトリガーすることによって、これらをオフラインします。クラスタ管理者が障害を解決し、依存先のリソースを再度有効にすると、RGM は、リソースのオフライン再起動依存リソースも再度オンラインにします。

依存関係の範囲を指定するには、このプロパティーを指定するときに、次の修飾子を、中括弧 ({ }) やアットマーク (@) を含めてリソース名に付加します。

{ANY_NODE}

指定された依存関係を任意のノードに拡張します。依存関係の動作は、どのノードでも依存先のリソースに影響されます。依存しているリソースは、自分が起動する前に依存

先のリソースがプライマリノードで起動するまで待機します。停止と再起動で同じような状況になります。

`{FROM_RG_AFFINITIES}`

リソースの依存関係の有効範囲が、そのリソースが属するリソースグループの `RG_affinities` 関係から派生するように指定します。依存しているリソースのリソースグループが、依存先のリソースのリソースグループに対して肯定的なアフィニティーを持っていて、同じノード上で起動または停止する場合、依存関係は `{LOCAL_NODE}` になります。このような肯定的なアフィニティーが存在しない場合、またはグループが別のノード上で起動する場合、依存関係は `{ANY_NODE}` になります。

`{LOCAL_NODE}`

指定される依存関係をノード単位に限定します。依存関係の動作は、同じノード上でのみ依存先のリソースに影響されます。依存しているリソースは、依存先のリソースが同じノードで起動されるまで待機します。停止と再起動で同じような状況になります。

`{LOCAL_NODE}` 依存リソースがフェイルオーバー (すなわち単一マスター) リソースグループに存在する場合、また、あるノードとの `{LOCAL_NODE}` 依存関係が満たされていない場合、リソースグループは、依存関係が満たされないノードで `Pending_online_blocked` 状態のままいるのではなく、`{LOCAL_NODE}` 依存関係が満たされている別のノードにフェイルオーバーする可能性があります。

@nodename

ほかのノードに影響しない特定のノードに制限される `{LOCAL_NODE}` 依存関係を指定します。これにより、リソースがクラスタのノードごとに異なる依存関係を持つことができます。*nodename* は、ノード名またはノード ID です。

たとえば、次のリストは、ノード `node1` 上のリソース `res1` の依存関係、およびノード `node2` 上のリソース `res2` の依存関係を示しています:

```
res1@node1,res2@node2
```

複数のノードに同じ依存関係を適用可能な場合は、各ノード名でリソース名を繰り返します。例:

```
myres@node1,myres@node2,myres@node3,...
```

同じリソースグループ内の 2 つのリソース間の依存関係は、常に `{LOCAL_NODE}` です。

修飾子を指定しない場合は、`FROM_RG_AFFINITIES` がデフォルトで使用されます。

691 ページの`scha_resource_get`(1HA)、1127 ページの`scha_resource_get`(3HA)、および 1017 ページの`scds_property_functions`(3HA) のマニュアルページのドキュメントには、修飾子を含む依存関係リスト、または修飾子を含まない依存関係リストを取得するための代わりのクエリーフォームが記載されています。

カテゴリ　　任意

デフォルト　　空のリスト

調整可能　　すべての時間

Resource_dependencies_restart (string_array)

このリソースが再起動の依存関係を持つ、同じリソースグループまたは異なるリソースグループ内のリソースのリスト。

このプロパティーは、Resource_dependencies とまったく同じように動作します。ただし、このリソースは、再起動依存関係リスト内のいずれかのリソースが再起動されると、再起動されます。このリソースの再起動は、リストのリソースがオンラインに復帰したあとで行われます。

リスト内の任意のリソースの起動に失敗した場合、このリソースは起動されません。このリソースと、リストのリソースの 1 つが同時に起動されると、RGM は、リストのリソースが始動してからこのリソースを起動します。

このリソースの Resource_dependencies_restart リスト内のリソースが起動しない場合 (たとえば、リスト内のリソースのリソースグループがオフラインのままであったり、リスト内のリソースが Start_failed 状態にあったりする場合)、このリソースはオフラインのままです。このリソースが依存する別のリソースグループのリソースが起動しないために、このリソースがオフラインのままの状態である場合、このリソースのグループは Pending_online_blocked 状態に入ります。

このリソースが、リストのリソースと同時にオフラインにされる場合は、このリソースが停止されてから、リストのほかのリソースが停止されます。しかし、このリソースがオンラインのままであるか、停止に失敗すると、リストのリソースは停止されます。

同じリソースグループ内では、依存性の順序に従って Prenet_start メソッドが Start メソッドより先に実行されます。Postnet_stop メソッドは依存性の順序に従って Stop メソッドよりあとに実行されます。異なるリソースグループ内では、依存先のリソースが Prenet_start と Start を実行してから、依存しているリソースが Prenet_start を実行します。同様に、依存しているリソースが Stop と Postnet_stop を終了してから、依存先のリソースが Stop を実行します。

依存関係の範囲を指定するには、このプロパティーを指定するときに、次の修飾子を、中括弧 ({ }) やアットマーク (@) を含めてリソース名に付加します。

{LOCAL_NODE}

指定される依存関係をノード単位に限定します。依存関係の動作は、同じノード上でのみ依存先のリソースに影響されます。依存しているリソースは、依存先のリソースが同じノードで起動されるまで待機します。停止と再起動で同じような状況になります。

{LOCAL_NODE} 依存リソースがフェイルオーバー (すなわち単一マスター) リソースグループに存在する場合、また、あるノードとの {LOCAL_NODE} 依存関係が満たされていない場合、リソースグループは、依存関係が満たされないノードで Pending_online_blocked 状態のままいるのではなく、{LOCAL_NODE} 依存関係が満たされている別のノードにフェイルオーバーする可能性があります。

{ANY_NODE}

指定された依存関係を任意のノードに拡張します。依存関係の動作は、どのノードでも依存先のリソースに影響されます。依存しているリソースは、自分が起動する前に依存

先のリソースがプライマリノードで起動するまで待機します。停止と再起動で同じような状況になります。

@nodename

ほかのノードに影響しない特定のノードに制限される `{LOCAL_NODE}` 依存関係を指定します。これにより、リソースがクラスタのノードごとに異なる依存関係を持つことができます。*nodename* は、ノード名またはノード ID です。

たとえば、次のリストは、ノード `node1` 上のリソース `res1` の依存関係、およびノード `node2` 上のリソース `res2` の依存関係を示しています:

```
res1@node1,res2@node2
```

複数のノードに同じ依存関係を適用可能な場合は、各ノード名でリソース名を繰り返します。例:

```
myres@node1,myres@node2,myres@node3,...
```

`{FROM_RG_AFFINITIES}`

リソースの依存関係の有効範囲が、そのリソースが属するリソースグループの `RG_affinities` 関係から派生するように指定します。依存しているリソースのリソースグループが、依存先のリソースのリソースグループに対して肯定的なアフィニティーを持っていて、同じノード上で起動または停止する場合、依存関係は `{LOCAL_NODE}` になります。このような肯定的なアフィニティーが存在しない場合、またはグループが別のノード上で起動する場合、依存関係は `{ANY_NODE}` になります。

同じリソースグループ内の 2 つのリソース間の依存関係は、常に `{LOCAL_NODE}` です。

修飾子を指定しない場合は、`FROM_RG_AFFINITIES` がデフォルトで使用されます。

691 ページの`scha_resource_get`(1HA)、1127 ページの`scha_resource_get`(3HA)、および 1017 ページの`scds_property_functions`(3HA) のマニュアルページのドキュメントには、修飾子を含む依存関係リスト、または修飾子を含まない依存関係リストを取得するための代わりのクエリーフォームが記載されています。

カテゴリ	任意
デフォルト	空のリスト
調整可能	すべての時間

`Resource_dependencies_weak` (`string_array`)

このリソースが弱い依存関係を持つ、同じリソースグループまたは異なるリソースグループ内のリソースのリスト。グループ内でのメソッド呼び出しの順序は、弱い依存性によって決まります。RGM は、このリスト内のリソースの `Start` メソッドを呼び出してから、このリソースの `Start` メソッドを呼び出します。さらに、RGM は、このリソースの `Stop` メソッドを呼び出してからリストのリソースの `Stop` メソッドを呼び出します。リスト内のリソースが始動に失敗したりオフラインのままであっても、リソースは起動されます。

このリソースと、その Resource_dependencies_weak リスト内のリソースが同時に起動された場合、RGM は、リストのリソースが起動してからこのリソースを起動します。リスト内のリソースが起動しない場合でも (たとえば、リスト内のリソースのリソースグループがオフラインのままであったり、リスト内のリソースが Start_failed 状態であったりする場合)、このリソースは起動します。このリソースの Resource_dependencies_weak リストのリソースが起動する際に、このリソースのリソースグループが一時的に Pending_online_blocked 状態に入ることがあります。リスト内のすべてのリソースが起動するか、起動に失敗すると、このリソースが起動し、そのグループが再度 Pending_online 状態になります。

このリソースが、リストのリソースと同時にオフラインにされる場合は、このリソースが停止されてから、リストのほかのリソースが停止されます。しかし、このリソースがオンラインのままであるか、停止に失敗すると、リストのリソースは停止されます。

同じリソースグループ内では、依存性の順序に従って Prenet_start メソッドが Start メソッドより先に実行されます。Postnet_stop メソッドは依存性の順序に従って Stop メソッドよりあとに実行されます。異なるリソースグループ内では、依存先のリソースが Prenet_start と Start を実行してから、依存しているリソースが Prenet_start を実行します。同様に、依存しているリソースが Stop と Postnet_stop を終了してから、依存先のリソースが Stop を実行します。

依存関係の範囲を指定するには、このプロパティーを指定するときに、次の修飾子を、中括弧 ({ }) やアットマーク (@) を含めてリソース名に付加します。

{LOCAL_NODE}

指定される依存関係をノード単位に限定します。依存関係の動作は、同じノード上でのみ依存先のリソースに影響されます。依存しているリソースは、依存先のリソースが同じノードで起動されるまで待機します。停止と再起動で同じような状況になります。

{LOCAL_NODE} 依存リソースがフェイルオーバー (すなわち単一マスター) リソースグループに存在する場合、また、あるノードとの {LOCAL_NODE} 依存関係が満たされていない場合、リソースグループは、依存関係が満たされないノードで Pending_online_blocked 状態のままいるのではなく、{LOCAL_NODE} 依存関係が満たされている別のノードにフェイルオーバーする可能性があります。

{ANY_NODE}

指定された依存関係を任意のノードに拡張します。依存関係の動作は、どのノードでも依存先のリソースに影響されます。依存しているリソースは、自分が起動する前に依存先のリソースがプライマリノードで起動するまで待機します。停止と再起動で同じような状況になります。

@*nodename*

ほかのノードに影響しない特定のノードに制限される {LOCAL_NODE} 依存関係を指定します。これにより、リソースがクラスタのノードごとに異なる依存関係を持つことができます。*nodename* は、ノード名またはノード ID です。

たとえば、次のリストは、ノード node1 上のリソース res1 の依存関係、およびノード node2 上のリソース res2 の依存関係を示しています:

```
res1@node1,res2@node2
```

複数のノードに同じ依存関係を適用可能な場合は、各ノード名でリソース名を繰り返します。例:

```
myres@node1,myres@node2,myres@node3,...
```

`{FROM_RG_AFFINITIES}`

リソースの依存関係の有効範囲が、そのリソースが属するリソースグループの `RG_affinities` 関係から派生するように指定します。依存しているリソースのリソースグループが、依存先のリソースのリソースグループに対して肯定的なアフィニティーを持っていて、同じノード上で起動または停止する場合、依存関係は `{LOCAL_NODE}` になります。このような肯定的なアフィニティーが存在しない場合、またはグループが別のノード上で起動する場合、依存関係は `{ANY_NODE}` になります。

同じリソースグループ内の 2 つのリソース間の依存関係は、常に `{LOCAL_NODE}` です。

修飾子を指定しない場合は、`FROM_RG_AFFINITIES` がデフォルトで使用されます。

691 ページの`scha_resource_get`(1HA)、1127 ページの`scha_resource_get`(3HA)、および 1017 ページの`scds_property_functions`(3HA) のマニュアルページのドキュメントには、修飾子を含む依存関係リスト、または修飾子を含まない依存関係リストを取得するための代わりのクエリーフォームが記載されています。

カテゴリ	任意
デフォルト	空のリスト
調整可能	すべての時間

`Resource_name` (`string`)

リソースインスタンスの名前です。クラスタ構成内で一意にする必要があります。リソースが作成されたあとで変更はできません。

カテゴリ	必須
デフォルト	デフォルトなし
調整可能	しない

`Resource_project_name` (`string`)

リソースに関連付けられた Oracle Solaris プロジェクト名 (Unresolved link to " projects1" を参照)。このプロパティーは、CPU の共有、クラスタデータサービスのリソースプールといった Solaris のリソース管理機能に適用できます。RGM は、リソースをオンラインにすると、このプロジェクト名を持つ関連プロセスを起動します。このプロパティーが指定されていない場合、プロジェクト名は、リソースを含むリソースグループの `RG_project_name` プロパティーから取得されます (1319 ページの`rg_properties`(5) のマニュアルページを参照)。どちらのプロパティーも指定されなかった場合、RGM は事前定義済みのプロジェクト名 `default` を使用します。指定されたプロジェクト名は、プロジェクトデータベース内に存在する必要があります (Unresolved link to " projects1" のマニュアルページおよびUnresolved link to " Oracle Solaris ゾーンの紹介 "を参照)。

注記 - このプロパティーへの変更は、リソースが次に起動されるときに有効になります。

カテゴリ	任意
デフォルト	NULL
調整可能	すべての時間
有効な値	任意の有効な Oracle Solaris プロジェクト名、または null

各クラスタノード上の `Resource_state` (enum)

RGM が判断した各クラスタノード上のリソースの状態。この状態には、`Online`、`Offline`、`Start_failed`、`Stop_failed`、`Monitor_failed`、`Online_not_monitored`、`Starting`、`Sto` があります。

`Online`

起動メソッド (`Prenet_start`、`Start`、`Monitor_start`) が、このノードのリソースに対して正常に実行されました。

`Offline`

そのリソースはこのノードで一度も起動されていないか、停止メソッド (`Monitor_stop`、`Stop`、`Postnet_stop`。どのメソッドであるかはリソースによって異なる) が、このノードのリソースに対して正常に実行されています。

`Start_failed`

`Prenet_start` または `Start` メソッドが、このノードのリソースで失敗しました。`Start_failed` は、メソッドがゼロ以外の終了ステータスで終了したか、タイムアウトになったことを意味します。このリソースの状態で表されるサービスは、このノードで実際に起動していることもあり、また起動していないこともあります。

`Stop_failed`

`Monitor_stop`、`Stop`、または `Postnet_stop` メソッドが、このノードのリソースで失敗しました。`Stop_failed` は、メソッドがゼロ以外の終了ステータスで終了したか、タイムアウトになったことを意味します。このリソースの状態で表されるサービスは、このノードで実際に停止されていることもあり、また停止されていないこともあります。

リソースがこの状態に入ると、リソースグループの状態は `Error_stop_failed` になり、ユーザーの介入が必要です。`Error_stop_failed` は、1319 ページの`rg_properties`(5) のマニュアルページで詳細に説明されています。

`Monitor_failed`

リソースがその `Prenet_start` または `Start` メソッドを正常に実行しました (どちらのメソッドであるかはリソースタイプによって異なる)。しかし、リソースの `Monitor_start` メ

ソッドが、ゼロ以外の終了ステータスで終了したか、タイムアウトになりました。リソースモニターは、このノードで実際に起動されていることもあり、また起動されていないこともあります。

`Online_not_monitored`

リソースがその `Prenet_start` または `Start` メソッドを正常に実行しました (どちらのメソッドであるかはリソースタイプによって異なる)。`Monitor_start` メソッドは、このリソースに対してまだ実行されていません。モニタリングされていないリソース (そのリソースに対する `Monitor_start` メソッドが存在しない場合や、そのリソースに対するモニタリングが無効にされている場合) は、そのリソースグループが `Online` 状態になっても、この状態のままです。

`Starting`

リソースが `Prenet_start` か `Start` メソッドを実行してオンラインになろうとしています。

`Stopping`

リソースが `Start` か `Postnet_stop` メソッドを実行してオフラインになろうとしています。

ユーザーはこのプロパティーを構成できません。

カテゴリ	照会のみ
デフォルト	デフォルトなし
調整可能	しない

`Retry_count` (`integer`)

起動に失敗したリソースをモニターが再起動する回数です。`Retry_count` を超過した場合、特定のデータサービス、および `Failover_mode` プロパティーの設定に応じて、モニターは次のいずれかの動作を行います:

- リソースが障害状態であっても、リソースグループが現在のプライマリノード上に残ることを許可します。
- 別のノードへのリソースグループのフェイルオーバーをリクエストします。

このプロパティーは RGM によって作成されます。RTR ファイルに宣言されている場合、クラスタ管理者のみ使用を許可されます。RTR ファイル内でデフォルト値が指定されている場合、このプロパティーは任意です。

リソースタイプファイル内で `Tunable` 属性が指定されていない場合は、プロパティーの `Tunable` 値は、`When_disabled` (無効にするとき) になります。

このプロパティーに負の値を指定すると、モニターは無限回リソースを再起動を試みます。

注記 - 一部のリソースタイプでは、Retry_count に負の値を設定できません。より確実に無制限の再起動を指定するには、次の手順を実行します。

- Retry_interval に 1 や 0 などの小さい値を指定します。
- Retry_count に 1000 などの大きい値を指定します。

カテゴリ	条件付き
デフォルト	上記を参照
調整可能	無効時

Retry_interval (integer)

失敗したリソースの再起動の間、この秒数だけ待機します。リソースモニターは、このプロパティーと Retry_count を組み合わせて使用します。このプロパティーは RGM によって作成されます。RTR ファイルに宣言されている場合、クラスタ管理者のみ使用を許可されます。RTR ファイル内でデフォルト値が指定されている場合、このプロパティーは任意です。

リソースタイプファイル内で Tunable 属性が指定されていない場合は、プロパティーの Tunable 値は、When_disabled (無効にするとき) になります。

注記 - Retry_interval プロパティーが宣言されていない場合、scha_resource_get (num_*_restarts) は終了コード 13 (SCHA_ERR_RT) で終了します。

カテゴリ	条件付き
デフォルト	上記を参照
調整可能	無効時

Round_robin (boolean)

各ノードに割り当てられた関連の load_balancing_weight 値を考慮して、受信リクエストをラウンドロビン方式で特定のサーバーノードに割り当てます。リクエストは、スティッキ以外の load_balancing_policy 設定のリソースで接続ごとに割り当てられ、それ以外の場合は、クライアント単位の IP アドレスごとに割り当てられます。

Round_Robin は、予想される接続またはクライアント数が少ない受信リクエストの決定的負荷分散が必要なリソースで有効になるようにしてください。

リソースプロパティー Conn_threshold、およびクラスタプロパティー udp_session_timeout はラウンドロビンスキームをサポートし、Round_robin リソースプロパティーがサービスに設定されている場合はオプションで構成できます。

Round_robin プロパティーを使用するのにアップグレードが必要な既存のリソースタイプ登録 (RTR) ファイルはありません。

カテゴリ　　任意

デフォルト　　FALSE

調整可能　　無効時

`Scalable` (`boolean`)

リソースがスケーラブルであるかどうか、つまり、リソースが Data Service for Apache Guide Cluster ソフトウェアのネットワーキング負荷分散機能を使用するかどうかを示します。

このプロパティーが RTR ファイルで宣言されて
いる場合は、そのタイプのリソースに対して、RGM
は、`Affinity_timeout`、`Load_balancing_policy`、`Load_balancing_weights`、`Network_resources_used`、`Port_`
および `Weak_affinity` のスケーラブルサービスプロパティーを自動的に作成します。これらのプロパティーは、RTR ファイル内で明示的に宣言されないかぎり、デフォルト値を持ちます。RTR ファイルに `Scalable` が宣言されている場合、このプロパティーのデフォルトは `True` です。

RTR ファイルにこのプロパティーが宣言されている場合、`At_creation` 以外の `Tunable` 属性の割り当ては許可されません。

RTR ファイルにこのプロパティーが宣言されていない場合、このリソースはスケーラブルではないため、このプロパティーを調整できません。RGM は、スケーラビブルサービスプロパティーをいっさい設定しません。ただし、必要に応じて、RTR ファイルに明示的に `Network_resources_used` および `Port_list` プロパティーを宣言できますが、これは、これらのプロパティーが、スケーラブルサービスだけでなく、非スケーラブルサービスでも有用なためです。

`Scalable` リソースプロパティと `Failover` リソースタイププロパティーの組み合わせと、その説明は次の通りです。

`Failover/Scalable`	**説明**
`True/True`	この非論理的な組み合わせは指定しないでください。
`True/False`	この組み合わせは、フェイルオーバーサービスに対して指定します。
`False/True`	この組み合わせは、ネットワーク負荷分散に `SharedAddress` リソースを使用するスケーラブルサービスに指定します。Unresolved link to " Oracle Solaris Cluster Concepts Guide "Oracle Solaris Cluster Concepts Guide を参照してください。
`False/False`	この組み合わせを使用して、ネットワーク負荷分散を使用しないマルチマスターサービスを構成します

詳細は、1335 ページの`rt_properties(5)` のマニュアルページにある `Failover` リソースタイププロパティーの説明を参照してください。

カテゴリ　　任意

デフォルト　　上記を参照

調整可能　　作成時

各クラスタノード上の `Status` (`enum`)

リソースモニターによって設定されます。可能な値は、`Online`、`Degraded`、`Faulted`、`Unknown`、および `Offline` です。`Start` (または `Prenet_start`) メソッドによって値がまだ設定されていない場合、値は、リソースの起動時に RGM により `Online` に設定されます。`Stop` (または `Postnet_stop`) メソッドによって値がまだ設定されていない場合、値は、リソースの停止時に RGM により `Offline` に設定されます。

カテゴリ　　照会のみ

デフォルト　　デフォルトなし

調整可能　　`scha_resource_setstatus` コマンドの使用によりのみ

各クラスタノード上の `Status_msg` (`string`)

リソースモニターによって、`Status` プロパティーと同時に設定されます。リソースが `Offline` になると、RGM はこの値に空文字列を設定します (`Stop` または `Postnet_stop` メソッドでまだ設定されていない場合)。

カテゴリ　　照会のみ

デフォルト　　デフォルトなし

調整可能　　`scha_resource_setstatus` の使用によりのみ

`Thorough_probe_interval` (`integer`)

高オーバーヘッドのリソース障害検証の呼び出し間隔 (秒)。このプロパティーは RGM によって作成されます。RTR ファイルに宣言されている場合、クラスタ管理者のみ使用を許可されます。RTR ファイル内でデフォルト値が指定されている場合、このプロパティーは任意です。

リソースタイプファイル内で `Tunable` 属性が指定されていない場合は、プロパティーの `Tunable` 値は、`When_disabled` (無効にするとき) になります。

カテゴリ　　条件付き

デフォルト　　デフォルトなし

調整可能　　無効時

`Timeout_delay` (`boolean`)

`hatimerun` コマンドに `-d` (遅延) オプションを渡すかどうかを決定します。`Timeout_delay` リソースプロパティーを宣言するリソースタイプは、`hatimerun` コマンドを使用して強制され

た時間制限の下でコマンドを実行します。-d オプションは、コマンドの実行が開始されるまでタイムアウトクロックの開始を遅延させます。これにより、割り当てられた期間に対して、実行前のスケジューリング遅延がカウントされなくなります。

このプロパティーを宣言する各リソースタイプは、そのリソースタイプに固有の方法で使用します。たとえば、ORCL.gds は hatimerun(1HA) を使用して、プローブコマンドを起動します。詳細は、各データサービスのドキュメントを参照してください。

Timeout_delay プロパティーのデフォルト値は FALSE、デフォルトの調整可能性は Any time です。これらの属性は、RTR ファイルでオーバーライドできます。

カテゴリ	条件付き/任意
デフォルト	False
調整可能	すべての時間

Type (string)

インスタンスのリソースタイプ。

カテゴリ	必須
デフォルト	デフォルトなし
調整可能	しない

Type_version (string)

現在このリソースに関連付けられているリソースタイプのバージョンを指定します。このプロパティーは RTR ファイル内に宣言できません。したがって、RGM によって自動的に作成されます。このプロパティーの値は、リソースタイプの RT_version プロパティーと等しくなります。リソースの作成時、Type_version プロパティーは明示的には指定しませんが、リソースタイプ名の接尾辞として表示されます。リソースを編集すると、Type_version の値を新しい値に変更する場合があります。

カテゴリ	上記を参照
デフォルト	なし
調整可能	調整は次のものから派生されます。 ■ 現在のリソースタイプのバージョン ■ リソースタイプ登録ファイル内の #$upgrade_from 指令 (1251 ページのrt_reg(4) のマニュアルページを参照)。

UDP_affinity (boolean)

true の場合、指定のクライアントからの UDP トラフィックはすべて現在クライアントの TCP トラフィックを処理しているサーバーノードに送信されます。

このプロパティーは、Load_balancing_policy が Lb_sticky または Lb_sticky_wild の場合にかぎり有効です。さらに、Weak_affinity を FALSE (デフォルト値) に設定する必要があります。

このプロパティーは、スケーラブルサービス専用です。

カテゴリ	条件付き/任意
デフォルト	False
調整可能	無効時

Weak_affinity (boolean)

true の場合、弱い形式のクライアントアフィニティーが有効になります。これによって、あるクライアントからの接続を同じサーバーノードに送信することが可能になりますが、サーバーリスナが起動される場合 (障害モニターの再起動があったり、リソースのフェイルオーバーやスイッチオーバーがあったり、ノードが障害後にクラスタに再結合されたりするため) や、管理アクションによってスケーラブルリソースの load_balancing_weights が変更される場合を除きます。

弱いアフィニティーはメモリーの消費とプロセッササイクルの点で、デフォルトの形式よりもオーバーヘッドを低く抑えられます。

このプロパティーは、Load_balancing_policy が Lb_sticky または Lb_sticky_wild の場合にかぎり有効です。

このプロパティーは、スケーラブルサービス専用です。

カテゴリ	条件付き/任意
デフォルト	False
調整可能	無効時

Unresolved link to " projects1", 273 ページのclresource(1CL),
305 ページのclresourcegroup(1CL), 333 ページのclresourcetype(1CL),
683 ページのscha_control(1HA), 691 ページのscha_resource_get(1HA),
699 ページのscha_resource_setstatus(1HA), 1101 ページのscha_control(3HA),
1127 ページのscha_resource_get(3HA), 1017 ページのscds_property_functions(3HA),
1251 ページのrt_reg(4), 1271 ページのproperty_attributes(5),
1319 ページのrg_properties(5), 1335 ページのrt_properties(5)

Unresolved link to " Oracle Solaris Cluster Concepts Guide "、Unresolved link to " Oracle Solaris ゾーンの紹介 "。

名前

SUNW.rac_framework, rac_framework — Oracle Solaris Cluster Support for Oracle Real Application Clusters (Oracle RAC) を有効にするフレームワークのためのリソースタイプ実装

SUNW.rac_framework リソースタイプは、Oracle Solaris Cluster Support for Oracle RAC を有効にするフレームワークを表します。このリソースタイプでは、このフレームワークのステータスをモニターできます。

SUNW.rac_framework リソースタイプは、シングルインスタンスリソースタイプです。したがって、クラスタに作成可能なリソースは 1 個だけです。

このリソースタイプを登録し、リソースタイプのインスタンスを作成するには、次のいずれかの方法を使用します。

- Oracle Solaris Cluster Manager
- Oracle Solaris Cluster Support for Oracle Real Application Clusters を構成するためのオプションを指定する 487 ページのclsetup(1CL) ユーティリティー
- 次の一連の Oracle Solaris Cluster 保守コマンド。

  1. このリソースタイプを登録するには、333 ページのclresourcetype(1CL) コマンドを使用します。
  2. このリソースタイプのインスタンスを作成するには、273 ページのclresource(1CL) コマンドを使用します。

この種類のリソースをノード上でオフラインにする場合、オンライン状態からオフライン状態へ切り替えるには、一定の時間がかかります。オフライン状態への切り替え中も、リソースは引き続き再構成処理に使用されています。ただし、リソースをノード上でオフラインにした場合、再度オンラインに戻すまで、リソースのプロパティーの変更は適用されません。Oracle Solaris Cluster は、この種類のリソースが無効になっている場合、このことを知らせる警告メッセージを表示します。

この種類のリソースを含むリソースグループを非管理状態に切り替えるには、一定の時間がかかります。非管理状態への切り替え中も、Oracle RAC フレームワークは引き続きフレームワーク再構成処理に使用されています。ただし、リソースグループを非管理状態に切り替えた場合、ノード上のリソースプロパティーの変更は有効になりません。Oracle RAC フレームワークを停止するには、ノードをリブートします。

SUNW.rac_framework リソースタイプの拡張プロパティーは次のとおりです。

`reservation_timeout`

整数型で、最小値は 100、最大値は 99999、デフォルト値は 325 です。このプロパティーは、Oracle Solaris Cluster Support for Oracle RAC の再構成の予約ステップに対するタイムアウト値を秒単位で指定します。このプロパティーはいつでも変更できます。

例 398 `rac_framework` リソースの作成

この例では、`SUNW.rac_framework` リソースタイプを登録し、`rac_framework-rs` という名前の `SUNW.rac_framework` リソースタイプのインスタンスを作成します。この例では、`rac-framework-rg` という名前のリソースグループがすでに作成されているものとします。

```
phys-host-sc1# clresourcetype register SUNW.rac_framework

phys-host-sc1# clresource create -g rac-framework-rg \
-t SUNW.rac_framework rac_framework-rs
```

例 399 `rac_framework` リソースのプロパティーの変更

この例では、Oracle Solaris Cluster Support for Oracle RAC の再構成の予約ステップのタイムアウトを 350 秒に設定します。この例では、`rac_framework-rs` という名前のリソースタイプ `SUNW.rac_framework` がすでに作成されているものとします。

```
phys-host-sc1# clresource set
\-p reservation_timeout=350 rac_framework-rs
```

注記 - ボリュームマネージャーリソースで使用される `SUNW.vucmm_framework` リソースを作成する例については、1433 ページの`SUNW.vucmm_framework`(5) を参照してください。

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/library/ucmm

273 ページの`clresource`(1CL), 333 ページの`clresourcetype`(1CL), 487 ページの`clsetup`(1CL), Unresolved link to " attributes5", 1433 ページの`SUNW.vucmm_framework`(5)

Unresolved link to " Oracle Solaris Cluster Data Service for Oracle Real Application Clusters ガイド "

名前

rg_properties — リソースグループプロパティー

下のリストで、Oracle Solaris Cluster が定義するリソースグループプロパティーについて説明します。

リソースグループプロパティーと説明

注記 - リソースグループプロパティー名 (`Auto_start_on_new_cluster` や `Desired_primaries` など) には、大文字と小文字の区別は*ありません*。リソースグループプロパティー名を指定する際には、大文字と小文字を任意に組み合わせることができます。

`Auto_start_on_new_cluster` (`boolean`)

このプロパティーは、新しいクラスタの形成時にリソースグループマネージャー (RGM) が自動的にリソースグループを起動するかどうかを制御します。デフォルトは、`TRUE` です。

`TRUE` に設定した場合、クラスタのすべてのノードが同時にリブートすると、RGM はリソースグループを自動的に起動して `Desired_primaries` を取得しようとします。

`FALSE` に設定した場合、クラスタのリブート時にリソースグループが自動的にリブートすることはありません。リソースグループは、305 ページの`clresourcegroup`(1CL) コマンドまたは同等のグラフィカルユーザーインタフェースコマンドを使用して、はじめてリソースグループが手動でオンラインに切り替えられるまで、オフラインのままです。その後、このリソースグループは通常のフェイルオーバー動作を再開します。

デフォルト	`TRUE`
調整可能	すべての時間

`Desired_primaries` (`integer`)

リソースグループが同時に動作できるノードの必要数です。

デフォルトは 1 です。`Desired_primaries` プロパティーの値は、`Maximum_primaries` プロパティーの値以下にする必要があります。

デフォルト	1 (上記を参照)
調整可能	すべての時間

`Failback` (`boolean`)

あるノードがクラスタメンバーシップに参加するか、またはあるノード上でリソースグループの強い肯定的なアフィニティーが有効になったときに、リソースグループがオンラインである一連のノードを再計算するかどうかを示すブール値。再計算により、RGM は優先度の低いノードでグループをオフラインにし、優先度の高いノードをオンラインにすることができます。詳細は、`RG_affinities` プロパティーを参照してください。

デフォルト　　FALSE

調整可能　　すべての時間

`Global_resources_used` (`string_array`)

クラスタファイルシステムがこのリソースグループ内のリソースによって使用されるかどうかを指定します。管理者はアスタリスク (*) か空文字列 ("") を指定できます。すべてのグローバルリソースを指定するときはアスタリスク、グローバルリソースを一切指定しない場合は空文字列を指定します。

デフォルト　　すべてのグローバルリソース

調整可能　　すべての時間

`Implicit_network_dependencies` (`boolean`)

ブール値で、`TRUE` の場合、RGM は、非ネットワークアドレスリソースの暗黙的で強い依存関係をグループ内のネットワークアドレスリソースに強制します。このとき、RGM は、すべてのネットワークアドレスリソースを起動してからその他のリソースを起動します。また、グループ内のその他のすべてのリソースを停止してからネットワークアドレスリソースを停止します。ネットワークアドレスリソースには、論理ホスト名と共有アドレスリソースタイプがあります。

スケーラブルリソースグループの場合、ネットワークアドレスリソースを含んでいないため、このプロパティーは効果がありません。

デフォルト　　TRUE

調整可能　　すべての時間

`Load_factors`

リソースグループが消費する負荷制限の量を決定します。

負荷制限はノードごとに構成でき、リソースグループにはそのノードに定義されている負荷制限に対応する負荷係数セットが割り当てられます。RGM がリソースグループをオンラインにすると、各ノードでのリソースグループの負荷係数が加算され、ノードの負荷制限に対して合計負荷が比較されます。リソースグループの負荷分散ポリシーは、`Priority` および `Preemption_mode` プロパティーの設定によっても影響されます。詳細は、`Preemption_mode` および `Priority` プロパティーを参照してください。

`clresourcegroup set -p` オプションを使用して、`load_factors` プロパティーの値を設定できます。`load_factors` プロパティーには、`limitname@value` という形式の 0 個以上の要素のコンマ区切りリストで構成される値があります。ここで、`limitname` は識別子文字列で、`value` は負にならない整数です。各負荷係数のデフォルト値は *0* で、最大許容値は *1000* です。リソースグループのノードリストに含まれるノードで `limitname` が `loadlimit` として定義されていない場合、そのノードでは無制限と見なされます。

一連のリソースグループが共通の負荷係数を使用している場合、ノードで対応する負荷制限が指定されていなくても (すなわち無制限)、それらのリソースグループはそれらのノード全体に分散されます。ゼロ以外の負荷係数が存在する場合、RGM は負荷を分散します。

負荷ベースのリソースグループ分散を行なわないようにするには、負荷係数を削除するか、ゼロに設定します。

注記 - 負荷係数または負荷制限が変更されると、現在オフラインになっているリソースグループが自動的にオンラインになることがあります。自動的にオンラインにならないようにするには、リソースグループで `clresourcegroup suspend` コマンドを実行します。

詳細は、305 ページの`clresourcegroup`(1CL) および 185 ページの`clnode`(1CL) のマニュアルページを参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、`solaris.cluster.admin` RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

`Maximum_primaries` (`integer`)

リソースグループが同時にオンラインになる可能性があるノードの最大数です。

`RG_mode` プロパティーが `Failover` の場合、プロパティーの値は、1 以下でなければなりません。`RG_mode` プロパティーが `Scalable` の場合は、1 より大きくてもかまいません。

デフォルト　1 (上記を参照)

調整可能　すべての時間

`Nodelist` (`string_array`)

優先順位に従ってグループをオンラインにできるノードのリスト。これらのノードは、リソースグループの潜在的なプライマリまたはマスターになります。

デフォルト　すべてのクラスタノードの順不同のリスト

調整可能　すべての時間

`Pathprefix` (`string`)

リソースグループ内のリソースが重要な管理ファイルを書き込むことができるクラスタファイルシステム内のディレクトリ。一部のリソースの必須プロパティーです。`Pathprefix` の値はリソースグループごとに固有の値を指定します。

デフォルト　空の文字列

調整可能　すべての時間

`Pingpong_interval` (`integer`)

再構成が生じた場合、あるいは `scha_control` ギブオーバーコマンドまたは関数の実行結果によって、どのノードでリソースグループをオンラインにするかを判断するときに RGM が使用する負以外の整数値 (秒)。

再構成の場合は、リソースグループが (リソースの Start または Prenet_start メソッドがゼロ以外で終了するか、またはタイムアウトしたために) 特定のノード上で Pingpong_interval 秒以内にオンラインにならなかったことが複数回発生すると、そのノードはリソースグループをホストする資格がないと見なされ、RGM は別のマスターを探します。

683 ページのscha_control(1HA) コマンドまたは 1101 ページのscha_control(3HA) ギブオーバーがあるリソースによって特定のノード上で実行され, それによってそのリソースグループが別のノードにフェイルオーバーされた場合, scha_control が呼び出された最初のノードは, Pingpong_interval 秒が経過するまで, 同じリソースによる別の scha_control ギブオーバーの宛先になることができません。

デフォルト　3600 (1 時間)

調整可能　すべての時間

Preemption_mode

ノードの過負荷のため、優先順位の高いリソースグループによってリソースグループがノードから横取りされる可能性を決定します。

clresourcegroup set -p オプションを使用して、preemption_mode プロパティーの列挙値を設定できます。preemption_mode プロパティーのデフォルト設定は HAS_COST です。

リソースグループの preemption_mode プロパティーは、次のいずれかの値を取ることができます。

- HAS_COST – 負荷制限を満たすため、優先順位の高いリソースグループはこのリソースグループを現在のマスターから奪うことができます。このリソースグループを横取りすることでコストが発生するため、可能な場合、RGM は優先順位の高いグループをマスターにするために別のノードを選択することでこれを回避しようとします。
- NO_COST – 負荷制限を満たすため、優先順位の高いリソースグループはこのリソースグループを現在のマスターから奪うことができます。このリソースグループを横取りしても、コストは発生しません。
- NEVER – 負荷制限を満たすため、このリソースグループを現在のマスターから奪うことはできません。

詳細は、305 ページのclresourcegroup(1CL) および 185 ページのclnode(1CL) のマニュアルページを参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

Priority

リソースグループがマスターノードに割り当てられる順序を決定します。優先順位が高いほど、サービスの重要度も高くなります。

clresourcegroup set -p オプションを使用して、priority プロパティーの符号のない整数値を設定できます。優先順位の値が高いリソースグループはほかのリソースグループよりも優先され、優先ノードによってマスターにされる可能性が高く、そのノードから奪われる可能性は低くなります。priority プロパティーのデフォルト値は *500* です。

2 つのリソースグループの優先順位が同等で、RG_dependencies または強い RG_affinities による関係がある場合、依存するリソースグループの前に、依存関係またはアフィニティーが指定されていないリソースグループにノードが割り当てられます。2 つのリソースグループの優先順位が同等で、依存関係や強いアフィニティーによる関係がない場合、任意の順序で一次ノードが割り当てられます。

詳細は、305 ページのclresourcegroup(1CL) および 185 ページのclnode(1CL) のマニュアルページを参照してください。

スーパーユーザー以外のユーザーがこのサブコマンドを使用するには、solaris.cluster.admin RBAC の承認が必要です。Unresolved link to " rbac5" のマニュアルページを参照してください。

Resource_list (string_array)

グループに含まれるリソースのリスト。管理者はこのプロパティーを直接設定しません。このプロパティーは、管理者がリソースグループにリソースを追加したり、リソースを削除したときに、RGM によって更新されます。

デフォルト	デフォルトなし
調整可能	しない

RG_affinities (string)

RGM は、(1) 現在別の指定リソースグループのマスターになっているマシン上のリソースグループ (肯定的なアフィニティー) を探すか、または、(2) 現在別の指定リソースグループのマスターになっていないマシン上のリソースグループ (否定的なアフィニティー) を探します。

RG_affinities には次の文字列を設定できます。

- + (弱い肯定的なアフィニティー)
- ++ (強い肯定的なアフィニティー)
- +++ (フェイルオーバー委託付きの強い肯定的なアフィニティー)
- - (弱い否定的なアフィニティー)
- -- (強い否定的なアフィニティー)

たとえば、RG_affinities=+RG2,--RG3 は、このリソースグループが RG2 に対して弱い肯定的なアフィニティーを、RG3 に対して強い否定的なアフィニティーを持つことを示しています。

RG_affinities の使用については、Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド の第 2 章データサービスリソースの管理"で説明されています。

デフォルト　　空の文字列

調整可能　　すべての時間

プロトタイプとして単一マシンクラスタを構成することがあります。そのようなクラスタ上の複数ノードでリソースグループを実行するように構成する場合、RG_affinities は、マシンレベルではなくてノードレベルとして解釈されます。たとえば、強い肯定的なアフィニティーでは、両方のリソースグループが単に同じマシン上で動作するだけでなく、同じノード上で動作する必要があります。単一マシンクラスタ上のすべてのノードは、同じマシン上のゾーンであることに注意してください。

RG_dependencies (string_array)

リソースグループの任意指定のリストで、同じノードでほかのグループをどのような順序でオンラインまたはオフラインにするかを示しています。すべての強い RG_affinities (肯定的および否定的) と RG_dependencies の関係図式の中にサイクルが含まれることが許可されません。

たとえば、リソースグループ RG2 がリソースグループ RG1 の RG_dependencies リストに含まれる場合です。言い換えると、RG1 が RG2 にリソースグループ依存関係を持っている場合です。次のリストに、リソースグループ依存関係の影響を要約します。

- あるノードがクラスタに参加すると、RG1 内のリソースに対するそのノード上の Boot メソッドは、RG2 内のリソースに対するそのノード上のすべての Boot メソッドが完了するまで実行されません。
- RG1 と RG2 が両方とも同じノード上で同時に Pending_online 状態である場合、起動メソッド (Prenet_start または Start) は、RG2 内のすべてのリソースがその起動メソッドを完了するまで RG1 内のどのリソース上でも実行されません。
- RG1 と RG2 が両方とも同じノード上で同時に Pending_offline 状態である場合、停止メソッド (Stop または Postnet_stop) は、RG1 内のすべてのリソースがその停止メソッドを完了するまで RG2 内のどのリソース上でも実行されません。
- RG1 または RG2 のプライマリをスイッチする試みは、スイッチによって RG1 がいずれかのノードでオンラインに、RG2 がすべてのノードでオフラインになる場合は、失敗します。
- RG2 に対する Desired_primaries がゼロに設定されている場合は、RG1 に対する Desired_primaries プロパティーをゼロより大きい値に設定することはできません。
- RG2 に対する Auto_start_on_new_cluster が FALSE に設定されている場合は、RG1 に対する Auto_start_on_new_cluster プロパティーを TRUE に設定することはできません。

デフォルト　　空のリスト

調整可能　　すべての時間

RG_description (string)

リソースグループの簡単な説明。

デフォルト　　空の文字列

調整可能　　すべての時間

RG_is_frozen (boolean)

あるリソースグループが依存しているグローバルデバイスをスイッチオーバーするかどうかを表します。このプロパティーが TRUE に設定されている場合、グローバルデバイスはスイッチオーバーされます。このプロパティーが FALSE に設定されている場合、グローバルデバイスはスイッチオーバーされません。リソースグループがグローバルデバイスに依存するかどうかは、Global_resources_used プロパティーの設定によります。

RG_is_frozen プロパティーをユーザーが直接設定することはありません。RG_is_frozen プロパティーは、グローバルデバイスのステータスが変わったときに、RGM によって更新されます。

デフォルト　　デフォルトなし

調整可能　　しない

RG_mode (enum)

リソースグループがフェイルオーバーグループなのか、スケーラブルグループなのかを指定します。値が Failover の場合、RGM はグループの Maximum_primaries プロパティーを 1 に設定し、リソースグループのマスターを単一のノードに制限します。

このプロパティーの値が Scalable の場合、RGM は Maximum_primaries プロパティーの値を 1 より大きい値に設定できるようにします。その結果、グループは、複数のノードで同時にマスターできます。RGM は、Failover プロパティーが TRUE であるリソースが、RG_mode が Scalable であるリソースグループに追加されることを許可しません。

Maximum_primaries が 1 の場合、デフォルトは Failover です。Maximum_primaries が 1 より大きい場合、デフォルトは Scalable です。

デフォルト　　Maximum_primaries の値によります。

調整可能　　作成時

RG_project_name(string)

リソースグループに関連付けられている Solaris プロジェクト名です (Unresolved link to " projects1" を参照)。このプロパティーは、CPU の共有やリソースプールなどの Solaris リソース管理機能をクラスタデータサービスに適用する場合に使用します。RGM は、リソースグループをオンラインにすると、Resource_project_name プロパティーが設定されていないリソースの関連プロセスをこのプロジェクト名で起動します (Unresolved link to " r_properties(5)" を参照)。指定されたプロジェクト名は、プロジェクトデータベース内に存在する必要があります (Unresolved link to " projects1" およびUnresolved link to " Oracle Solaris ゾーンの紹介 "を参照)。

注記 - このプロパティーへの変更は、リソースの次回起動時に有効になります。

デフォルト　テキスト文字列は、“default” です。

調整可能　すべての時間

有効な値　有効な任意の Solaris プロジェクト名

RG_SLM_CPU_SHARES(integer)

リソースグループに関連する CPU シェア数。

注記 - RG_SLM_CPU_SHARES プロパティーを設定できるのは、RG_SLM_TYPE に automated が設定されている場合のみです。詳細は、「RG_SLM_TYPE プロパティー」を参照してください。

RG_SLM_CPU_SHARES の最大値は 65535 です。シェア値にゼロを設定すると、CPU の負荷が高くなったときに、プロセスがスケジュールされなくなることがあるため、RG_SLM_CPU_SHARES の値にはゼロを指定できません。リソースグループがオンラインの間に RG_SLM_CPU_SHARES に行われた変更は、動的に考慮されます。

RG_SLM_TYPE が automated に設定されている場合、Oracle Solaris Cluster は、Unresolved link to " project4" を名前 SCSLM_*resourcegroup-name* で作成します。ここで、*resourcegroup-name* は、使用するリソースグループの名前になります。リソースグループに属するリソースの各メソッドは、このプロジェクトで実行されます。これらのプロジェクトは、そのリソースグループのゾーン (大域ゾーン) 内に作成されます。

プロジェクト SCSLM_*resourcegroup-name* の project.cpu-shares の値には、RG_SLM_CPU_SHARES の値が設定されます。RG_SLM_CPU_SHARES プロパティーが設定されていない場合、このプロジェクトの作成時、project.cpu-shares の値には 1 が設定されます。

RG_SLM_PSET_TYPE プロパティーが strong または weak に設定されている場合は、作成される pset のサイズを計算するために RG_SLM_CPU_SHARES プロパティーの値も使用されます (慣例上、100 の共有は 1 個の CPU と同等)。詳細は、RG_SLM_PSET_TYPE プロパティーを参照してください。

プロセッサセットについては、Unresolved link to " Oracle Solaris ゾーンの紹介 "を参照してください。

デフォルト　1

調整可能　すべての時間

RG_SLM_PSET_MIN (integer)

リソースグループが実行するプロセッサセット中のプロセッサの最小値。このプロパティーは、次の条件が真の場合だけ使用できます。

- 使用されているオペレーティングシステムが Solaris 11 である。

- RG_SLM_TYPE は automated に設定されます。
- RG_SLM_PSET_TYPE は strong または weak に設定されます。(RG_SLM_PSET_TYPE プロパティーを参照)
- RG_SLM_PSET_MIN の値は、RG_SLM_CPU_SHARES の値を 100 で割った値以下でなければなりません。

RG_SLM_PSET_MIN の最大数は 655 です。RG_SLM_PSET_MIN プロパティーの値は、Oracle Solaris Cluster がプロセッサセットの最小サイズを計算するのに使用されます。

リソースグループがオンラインの間に RG_SLM_CPU_SHARES と RG_SLM_PSET_MIN に行われた変更は、動的に考慮されます。ただし、RG_SLM_PSET_TYPE に strong が設定されている場合、および、変更を受け入れるための CPU が十分に存在しない場合、RG_SLM_PSET_MIN にリクエストされた変更は適用されません。この場合は、警告メッセージが表示されます。次のスイッチオーバー時、利用できる CPU が十分になく、RG_SLM_PSET_MIN に構成した値を遵守できない場合、CPU の不足によるエラーが発生する可能性があります。

プロセッサセットについては、Unresolved link to " Oracle Solaris ゾーンの紹介 "を参照してください。

デフォルト　0

調整可能　すべての時間

RG_SLM_PSET_TYPE (string)

専用のプロセッサセットの作成を可能にします。

RG_SLM_PSET_TYPE に指定可能な値は、default、strong、および weak です。

RG_SLM_PSET_TYPE に strong または weak を設定できるのは、次のすべての条件が真の場合だけです。

- 使用されているオペレーティングシステムが Solaris 11 である。
- 非大域ゾーンだけで実行するようにリソースグループが構成されている。
- RG_SLM_TYPE は automated に設定されます。

RG_SLM_PSET_TYPE に指定可能な値は、default、strong、および weak です。

リソースグループが strong または weak として実行される場合は、そのリソースグループのノードリストには非大域ゾーンだけが存在するように、リソースグループを構成してください。

非大域ゾーンは、デフォルトプールである pool_default 以外のプールに対して構成しないでください。ゾーン構成についての詳細は、Unresolved link to " zonecfg1M" を参照してください。非大域ゾーンは、デフォルトプール以外のプールに動的にバインドしないでください。プールバインドについての詳細は、Unresolved link to " poolbind1M" を参照してください。バインドされた 2 つのプールの状態は、リソースグループ内のリソースのメソッドが起動されている場合だけ、確認されます。

strong と weak の値は、ノードリストに同じゾーンを持つリソースグループと相互に排他的です。一部のリソースグループの RG_SLM_PSET_TYPE に strong を設定し、それ以外に weak を設定するように、同じゾーン内のリソースグループを構成できません。

RG_SLM_PSET_TYPE に strong または weak が設定されており、RG_SLM_TYPE に示されているアクションに automated が設定されている場合、リソースグループがオンラインになるとき、Oracle Solaris Cluster は次のことを行ないます。

- プールを作成して、このプールを動的にリソースグループが起動する非大域ゾーンにバインドします。
- プロセッサセットを作成して、そのサイズを最小値から最大値までの間にします。
  - 最小値は、このリソースグループが起動するゾーンでオンラインであるすべてのリソースグループの RG_SLM_PSET_MIN 値の合計になります。この合計がゼロの場合、最小値は 1 になります。
  - 最大値は、このリソースグループが起動するゾーンでオンラインであるすべてのリソースグループの RG_SLM_SPU_SHARES 値の合計を 100 で割り、もっとも近い整数に切り上げた値になります。この計算の結果がゼロの場合、最大値は 1 になります。
- プロセッサセットをプールに関連付けます。
- zone.cpu-shares に、ゾーンで実行しているすべてのリソースグループの RG_SLM_CPU_SHARES の合計を設定します。

RG_SLM_PSET_TYPE に strong または weak が設定されている場合、そのリソースグループはオフラインに設定されます (より正確には、リソースグループの最初のリソースの STOP メソッドまたは POSTNET_STOP メソッドが実行される)。ゾーンにオンラインのリソースグループが 1 つも存在しなくなった場合、Oracle Solaris Cluster はプロセッサセットとプールを破棄し、そのゾーンをデフォルトのプール (pool_default) にバインドします。

RG_SLM_PSET_TYPE に strong が設定されている場合、リソースグループの動作は、RG_SLM_PSET_TYPE に strong が設定されていた場合と同じです。しかし、プロセッサセットの作成に十分なプロセッサを使用できない場合、プールはデフォルトプロセッサセットに関連付けられます。

RG_SLM_PSET_TYPE が strong に設定されており、プロセッサセットの作成に十分なプロセッサを使用できない場合は、リソースグループモニター (Resource Group Monitor、RGM) にエラーが返され、そのノードまたはゾーンではリソースグループは起動されません。

CPU 割り当ての優先順位は、defaultpsetmin の最小サイズが strong よりも優先され、weak よりも優先されます。(defaultpsetmin プロパティーについては、185 ページのclnode(1CL) を参照。)しかし、clnode コマンドを使用してデフォルトプロセッサセットのサイズを大きくしようとしたときに、十分なプロセッサが使用できない場合、この優先順位は維持されません。

clnode コマンドを使用して CPU の最小数をデフォルトプロセッサセットに割り当てる場合、操作は動的に実行されます。選択された数の CPU を使用できない場合、Oracle Solaris Cluster は定期的に、選択された CPU の最小値が割り当てられるまで、選択された数の CPU をデフォルトプロセッサセットに割り当てようとし、次にそれより少ない数の CPU を割り当てようとします。このアクションは、いくつかの weak なプロセッサセットを破壊することがありますが、strong なプロセッサセットは破壊しません。

RG_SLM_PSET_TYPE が strong として構成されているリソースグループが起動すると、両方のプロセッサセットに対してノード上で十分な CPU を使用できないときは、このアクションが weak プロセッサセットに関連付けられているプロセッサセットを破棄する場合があります。その場合、weak プロセッサセットで動作しているリソースグループのプロセスは、デフォルトプロセッサセットに関連付けられます。

プロセッサセットを weak から strong に、または、strong から weak に変更するには、まずプロセッサセットの RG_SLM_PSET_TYPE を default に設定してください。

RG_SLM_PSET_TYPE にデフォルトを設定すると、Oracle Solaris Cluster はプール SCSLM_pool_*zone-name* を作成しますが、プロセッサセットは作成しません。この場合、SCSLM_pool_*zone-name* はデフォルトプロセッサセットに関連付けられます。ゾーンに割り当てられるシェアは、そのゾーンで実行しているすべてのリソースグループの RG_SLM_CPU_SHARES に設定されている値の合計で決定されます。

非大域ゾーン内の CPU コントロールに対して構成されているオンラインリソースグループがもはや 1 つもない場合、非大域ゾーンに対する CPU のシェアの値は、ゾーン構成内にある zone.cpu-shares の値となります。このパラメータの値はデフォルトで 1 です。ゾーン構成についての詳細は、Unresolved link to " zonecfg1M" を参照してください。

リソースプールとプロセッサセットについては、Unresolved link to " Oracle Solaris ゾーンの紹介 "を参照してください。

デフォルト	テキスト文字列は、“default”です。
調整可能	すべての時間

RG_SLM_TYPE (string)

システムリソース使用率を制御できるようにし、システムリソース管理のための Oracle Solaris OS の構成手順の一部を自動化します。RG_SLM_TYPE に指定可能な値は、automated と manual です。

RG_SLM_TYPE に automated が設定されている場合、リソースがオンラインになるとき、Oracle Solaris Cluster は次のことを行ないます。

- SCSLM_*resourcegroup-name* という名前のプロジェクトを作成します。このリソースグループ内のリソースのすべてのメソッドは、このプロジェクト内で実行されます。このプロジェクトは、このリソースグループ内のリソースのメソッドがノードまたはゾーンで初めて実行されるときに作成されます。

- このプロジェクトに関連付けられている project.cpu_shares の値に RG_SLM_CPU_SHARES の値を設定します。project.cpu_shares の値はデフォルトでは 1 です。
- zone.cpu-shares を、そのゾーンの RG_SLM_TYPE が automated に設定されているすべてのリソースグループの RG_SLM_CPU_SHARES の合計に設定します。ゾーンはグローバルにできます。専用のプロセッサセットについての詳細は、「RG_SLM_PSET_TYPE プロパティー」を参照してください。

RG_SLM_TYPE が automated に設定されている場合は、どのアクションを実行した場合でもメッセージがログに記録されます。

RG_SLM_TYPE が manual に設定されている場合、リソースグループは RG_project_name プロパティーで指定されたプロジェクトで実行されます。

リソースプールとプロセッサセットについては、Unresolved link to " Oracle Solaris ゾーンの紹介 "を参照してください。

- 58 文字を超えるリソースグループ名は指定しないでください。リソースグループ名が 58 文字を超える場合、CPU 制御を構成できなくなる、つまり、RG_SLM_TYPE プロパティーに automated を設定できなくなります。
- リソースグループ名にはダッシュ (-) を含めないでください。Oracle Solaris Cluster ソフトウェアは、プロジェクトの作成時に、リソースグループ名にあるすべてのダッシュを下線 (_) に置き換えます。たとえば、Oracle Solaris Cluster が rg-dev というリソースグループに対して SCSLM_rg_dev というプロジェクトを作成する場合です。Oracle Solaris Cluster がリソースグループ rg-dev に対してプロジェクトを作成しようとするとき、rg_dev という名前のリソースグループがすでに存在する場合、競合が発生します。

デフォルト	manual
調整可能	すべての時間

各クラスタノード上の RG_state (enum)

RGM によって Unmanaged、Online、Offline、Pending_online、Pending_offline、Error_stop_failed、Online_faulted、または Pending_online_blocked に設定され、各クラスタノード上のリソースグループの状態を示します。

ユーザーはこのプロパティーを構成できません。ただし、305 ページのclresourcegroup(1CL) を使用するか、または同等の Oracle Solaris Cluster グラフィカルユーザーインタフェースコマンドを使用して、このプロパティーを間接的に設定することは可能です。RGM の制御下にないときは、リソースグループは Unmanaged 状態で存在可能です。

各状態の説明は次のとおりです。

注記 - 状態は、すべてのノードにわたって適用される Unmanaged 状態を除き、個々のノードに対してのみ適用されます。たとえば、リソースグループがノード A 上では Offline であるが、ノード B 上では Pending_online である可能性があります。

Error_stop_failed

リソースグループ内の 1 つ以上のリソースが正常に停止できず、Stop_failed リソース状態にあります。グループ内のほかのリソースがオンラインまたはオフラインである可能性があります。このリソースグループは、Error_stop_failed 状態がクリアされるまで、どのノード上でも起動することを許可されません。

clresourcegroup clear などの管理コマンドを使用して、Stop_failed リソースを手動で停止させ、その状態を Offline にリセットする必要があります。

Offline

リソースグループはノードですでに停止しています。つまり、有効になっているそのグループのすべてのリソースに対して、停止メソッド (Monitor_stop、Stop、Postnet_stop) がすでに正常に実行されています (どのメソッドであるかはリソースによって異なります)。リソースグループがノードで最初に起動される前も、この状態が適用されます。

Online

リソースグループはノード上ですでに起動しています。つまり、有効になっているそのグループのすべてのリソースに対して、起動メソッド (Prenet_start、Start、Monitor_start) がすでに正常に実行されています (どのメソッドであるかはリソースによって異なります)。

Online_faulted

リソースグループは Pending_online の状態でしたが、このノードで起動が終わりました。しかし、1 つ以上のリソースが Start_failed リソース状態または Faulted ステータスになりました。

Pending_offline

リソースグループはノード上で停止中です。有効になっているそのグループのリソースに対して、停止メソッド (Monitor_stop、Stop、Postnet_stop) が実行されようとしています (どのメソッドであるかはリソースによって異なります)。

Pending_online

リソースグループはノード上で起動中です。有効になっているそのグループのリソースに対して、起動メソッド (Prenet_start、Start、Monitor_start) が実行されようとしています (どのメソッドであるかはリソースによって異なります)。

`Pending_online_blocked`

リソースグループは、完全な起動を行うことに失敗しました。これは、リソースグループの 1 つまたは複数のリソースが、ほかのリソースグループのリソースに対して強いリソース依存性があり、それが満たされていないためです。このようなリソースは `Offline` のままになります。リソース依存性が満たされると、リソースグループは自動的に `Pending_online` 状態に戻ります。

`Unmanaged`

新しく作成されたリソースグループの最初の状態や、過去には管理されていたリソースグループの状態。そのグループのリソースに対して `Init` メソッドがまだ実行されていないか、そのグループのリソースに対して `Fini` メソッドがすでに実行されています。

このグループは RGM によって管理されていません。

デフォルト　　デフォルトなし

調整可能　　しない

`RG_system` (`boolean`)

リソースグループの `RG_system` プロパティーの値が `TRUE` の場合、そのリソースグループとそのリソースグループ内のリソースに関する特定の操作が制限されます。この制限は、重要なリソースグループやリソースを間違って変更または削除してしまうことを防ぐためにあります。このプロパティーによって影響を受けるのは、273 ページの`clresource`(1CL) コマンドと 305 ページの`clresourcegroup`(1CL) コマンドだけです。683 ページの`scha_control`(1HA) と 1101 ページの`scha_control`(3HA) の操作は影響を受けません。

リソースグループ (または、リソースグループ内のリソース) の制限された操作を実行する前に、最初にリソースグループの `RG_system` プロパティーを `FALSE` に設定する必要があります。クラスタサービスをサポートするリソースグループ (または、リソースグループ内のリソース) を変更または削除するときには注意してください。

次の表に、リソースグループの `RG_system` プロパティーが `TRUE` に設定されているときに制限される操作を示します。

操作	例
リソースグループを削除する	**`clresourcegroup delete RG1`**
リソースグループプロパティーを編集する (`RG_system` を除く)	**`clresourcegroup set -p RG_description=... +`**
リソースグループへソースを追加する	**`clresource create -g RG1 –t SUNW.nfs R1`** リソースは作成後に有効な状態になり、リソースモニタリングも有効になります。
リソースグループからリソースを削除する	**`clresource delete R1`**
リソースグループに属するリソースのプロパティーを編集する	**`clresource set -g RG1 –t SUNW.nfs -p r_description="HA-NFS res" R1`**

操作	例
リソースグループをオフラインに切り替える	**clresourcegroup offline RG1**
リソースグループを管理する	**clresourcegroup manage RG1**
リソースグループを管理しない	**clresourcegroup unmanage RG1**
リソースを使用可能にする	**clresource enable R1**
リソースのモニタリングを有効にする	**clresource monitor R1**
リソースを使用不可にする	**clresource disable R1**
リソースのモニタリングを無効にする	**clresource unmonitor R1**

リソースグループの RG_system プロパティーの値が TRUE の場合、そのリソースグループで編集可能な唯一のプロパティーは RG_system プロパティー自身です。つまり、RG_system プロパティーの編集が制限されることはありません。

デフォルト　　FALSE

調整可能　　すべての時間

Suspend_automatic_recovery (boolean)

リソースグループの自動復旧が中断されるかどうかを指定するブール値です。クラスタ管理者が自動復旧を再開するコマンドを明示的に実行するまで、中断されたリソースグループが自動的に再開またはフェイルオーバーされることは*ありません*。オンラインかオフラインかにかかわらず、中断されたデータサービスは現在の状態のままです。

リソースグループが一時停止されている間、clresourcegroup(1CL) または clresource(1CL) コマンドを switch、online、offline、disable、enable などのサブコマンドとともに使用すると、リソースグループまたはそのリソースを特定のノード上の別の状態に手動で切り替えることができます。アプリケーションプロセスの強制終了やアプリケーション固有のコマンドの実行など、リソースに対して直接に操作するのではなく、clresourcegroup(1CL) または clresource(1CL) コマンドを使用してください。これにより、クラスタのフレームワークがリソースおよびリソースグループの現在のステータスを正確に把握することができ、resume サブコマンドが実行されたときに可用性が正しく復元されます。

Suspend_automatic_recovery プロパティーに TRUE が設定されると、リソースグループの自動復旧は中断されます。このプロパティーに FALSE が設定されると、リソースグループの自動復旧が再開され、アクティブになります。

クラスタ管理者はこのプロパティーを直接設定しません。RGM は、クラスタ管理者がリソースグループの自動復旧を中断または再開したときに Suspend_automatic_recovery プロパティーの値を変更します。クラスタ管理者は、clresourcegroup suspend コマンドで自動復旧を中断します。クラスタ管理者は、clresourcegroup resume コマンドで自動復旧を再開します。RG_system プロパティーの設定にかかわらず、リソースグループは中断または再開できます。

デフォルト　　FALSE

調整可能　　しない

`Target_nodes` (`string_array`)

`clrg switch`、`clrg remaster`、`scha_control` ギブオーバーまたはフェイルバックアクションによって切り替えられている、リソースグループの現在の切り替え先ノードのリスト。リソースグループが現在、これらのアクションのいずれかによって新しいノードに切り替えられていない場合、このクエリーは空のリストを返します。

このクエリーは通常、グループの切り替え先である指定された (1 つまたは複数の) ターゲットノードが存在するかどうかを検出するために、`Stop` または `Postnet_stop` 停止メソッドによって実行されます。

デフォルト　　なし

調整可能　　なし (クエリーのみ)

Unresolved link to " projects1", 185 ページの`clnode`(1CL),
273 ページの`clresource`(1CL), 305 ページの`clresourcegroup`(1CL),
683 ページの`scha_control`(1HA), Unresolved link to " poolbind1M",
1101 ページの`scha_control`(3HA), Unresolved link to " project4",
1271 ページの`property_attributes`(5), 1287 ページの`r_properties`(5),
1335 ページの`rt_properties`(5), 703 ページの`scha_resourcegroup_get`(1HA), および
1185 ページの`scha_resourcegroup_get`(3HA)。

Unresolved link to " Oracle Solaris Cluster Concepts Guide "、Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド "、Unresolved link to " Oracle Solaris ゾーンの紹介 "。

名前

rt_properties — resource-type プロパティー

下のリストで、Oracle Solaris Cluster ソフトウェアが定義するリソースタイププロパティーについて説明します。この説明は、データサービスの開発者を対象としたものです。特定のデータサービスについての詳細は、そのデータサービスのマニュアルページを参照してください。

リソースタイププロパティーの値

必須	プロパティーはリソースタイプ登録 (RTR) ファイルに明示的な値を必要とします。そうでない場合、プロパティーが属するオブジェクトは作成できません。空白文字または空の文字列を値として指定できません。
条件付き	RTR ファイル内に宣言を必要とするプロパティーです。宣言がない場合、RGM はこのプロパティーを作成しません。したがって、このプロパティーを管理ユーティリティーから利用できません。空白文字または空の文字列を値として指定できます。プロパティーが RTR ファイル内で宣言されており、値が指定されていない場合には、RGM はデフォルト値を使用します。
条件付き / 明示	RTR ファイル内に宣言と明示的な値を必要とするプロパティーです。宣言がない場合、RGM はこのプロパティーを作成しません。したがって、このプロパティーを管理ユーティリティーから利用できません。空白文字または空の文字列を値として指定できません。
任意	RTR ファイル内に宣言できるプロパティーです。プロパティーが RTR ファイル内で宣言されていない場合は、RGM がこれを作成し、デフォルト値を与えます。プロパティーが RTR ファイル内で宣言されており、値が指定されていない場合は、RGM は、プロパティーが RTR ファイル内で宣言されないときのデフォルト値と同じ値を使用します。
照会のみ	管理ユーティリティーでは直接設定できないプロパティーです。このプロパティーは RTR ファイル内には設定されません。このプロパティーの値は情報のためだけに提供されます。

注記 - `Installed_nodes` と `RT_system` を除くリソースタイププロパティーは、管理ユーティリティーでは更新できません。`Installed_nodes` は RTR ファイル内に宣言できないため、クラスタ管理者のみが設定できます。`RT_system` には RTR ファイル内で初期値を割り当てることができ、またクラスタ管理者が設定することもできます。

リソースタイププロパティーと説明

リソースタイプはリソースタイプ登録ファイルで定義されます。リソースタイプ登録ファイルには、そのリソースタイプの標準プロパティーと拡張プロパティーの値が指定されています。

注記 - resource-type プロパティー名 (`API_version` や `Boot` など) には、大小文字の区別は*ありません*。プロパティー名を指定する際には、大文字と小文字を任意に組み合わせることができます。

`API_version` (`integer`)

このリソースタイプの実装が使用するリソース管理 API のバージョン。

Oracle Solaris Cluster ソフトウェアの各リリースでサポートされる最新の `API_version` は次のとおりです。

3.1 以前	2
3.1 10/03	3
3.1 4/04	4
3.1 9/04	5
3.1 8/05	6
3.2	7
3.2 2/08	8
3.2 1/09	9
3.2 11/09	10
3.3	11
3.3 5/11	12
3.3 12/12	13
4.0	20
4.1	21
4.2	22

RTR ファイルで `API_version` に 2 より大きい値を宣言すると、そのリソースタイプは、その値より古いバージョンをサポートする Oracle Solaris Cluster ソフトウェアにはインストールされません。たとえば、あるリソースタイプに `API_version=7` を宣言した場合、そのリソースタイプは、Sun Cluster 3.2 リリースより前にリリースされたクラスタソフトウェアのどのバージョンにもインストールできません。

カテゴリ	任意
デフォルト	2
調整可能	しない

`Boot` (`string`)

オプションのコールバックメソッド: 次に示す条件が発生したときに、RGM がノード上で呼び出すプログラムのパスです:

- ノードがクラスタに結合または再結合する場合。
- このタイプのリソースを含むリソースグループが管理されている場合。

このメソッドは、`Init` メソッドと同様に、このタイプのリソースの初期化を行う必要があります。

カテゴリ	条件付き / 明示
デフォルト	なし
調整可能	しない

`Failover` (`boolean`)

このプロパティーに `TRUE` を設定した場合、このタイプのリソースは、複数のノードで同時にオンラインになる可能性があるどのグループ内でも構成できません。

このリソースタイプのプロパティーは、次のように `Scalable` リソースプロパティーと一緒に使用します:

Failover/Scalable	説明
TRUE/TRUE	この非論理的な組み合わせは指定しないでください。
TRUE/FALSE	この組み合わせは、フェイルオーバーサービスに対して指定します。
FALSE/TRUE	この組み合わせは、ネットワーク負荷分散に `SharedAddress` リソースを使用するスケーラブルサービスに指定します。 Unresolved link to " Oracle Solaris Cluster Concepts Guide "Oracle Solaris Cluster Concepts Guide を参照してください。
FALSE/FALSE	この組み合わせを使用して、ネットワーク負荷分散を使用しないマルチマスターサービスを選択します。

詳細は、Scalable in 1287 ページのr_properties(5) の説明およびUnresolved link to " Oracle Solaris Cluster Concepts Guide の第 3 章Key Concepts for System Administrators and Application Developers"を参照してください。

カテゴリ	任意
デフォルト	FALSE
調整可能	しない

Fini (string)

オプションのコールバックメソッド: このタイプのリソースを RGM 管理の対象外にするとき、RGM によって呼び出されるプログラムのパスです。

カテゴリ	条件付き / 明示
デフォルト	デフォルトなし
調整可能	しない

Global_zone (boolean)

あるリソースタイプのこのプロパティーに TRUE を設定した場合、そのメソッドはすべての状況で、大域ゾーン内で実行されます。このプロパティーに TRUE を設定した場合、リソースグループがゾーンクラスタで構成されている場合でも、メソッドは大域ゾーンで実行されます。このプロパティーに TRUE を設定するのは、ネットワークアドレスやファイルシステムなど、大域ゾーンから管理できるサービスに対してだけです。

注意 - 信頼できる既知のソースであるリソースタイプを除いて、Global_zone プロパティーに TRUE が設定されているリソースタイプは登録しないでください。このプロパティーに TRUE を設定したリソースタイプは、ゾーン分離をすり抜け、脅威をもたらします。

ゾーンクラスタ内にある RTR ファイルで Global_zone プロパティーを TRUE に設定しないでください。このプロパティーに TRUE を設定するすべてのタイプのリソースは、グローバルクラスタの大域ゾーンに配置する必要があります。

非大域ゾーンで起動するように構成され、Global_zone プロパティーに TRUE が設定されているリソースのメソッドは、常に大域ゾーンで実行されます。このようなリソースは、非大域ゾーンで構成されても、CPU 共有および専用プロセッサセット構成の利点を得ません。このリソースは、RG_slm_type プロパティーに AUTOMATED を設定した場合でも利点は得られません。Oracle Solaris Cluster ソフトウェアは、RG_slm_type プロパティーに MANUAL が設定されているリソースグループ内にあるかのように、このようなリソースを扱います。

Global_zone プロパティーに TRUE が設定されているリソースタイプのメソッドは大域ゾーンで実行されるため、RGM は、非大域ゾーンが停止されても、これらのリソースタイプがオフラインであると即座にみなすわけではありません。実際、RGM は、Monitor_stop、Stop、

および Postnet_stop などのメソッドを LogicalHostname、SharedAddress、および HAStoragePlus をはじめとするこれらのリソースタイプ上で実行します。しかし、非大域ゾーンが終了したとき、RGM は、Global_zone プロパティーに FALSE が設定されているリソースはオフラインであると考えます。停止メソッドは非大域ゾーンで実行する必要があるため、RGM は、このようなリソースでは停止メソッドを実行できません。

Global_zone=TRUE を宣言するリソースタイプは、Global_zone_override リソースプロパティーも宣言する場合があります。その場合、Global_zone_override プロパティーの値が、そのリソースの Global_zone プロパティーの値より優先されます。Global_zone_override プロパティーの詳細は、1287 ページのr_properties(5) のマニュアルページを参照してください。

カテゴリ	任意
デフォルト	FALSE
調整可能	しない

Init (string)

オプションのコールバックメソッド: このタイプのリソースを RGM 管理対象にするとき RGM によって呼び出されるプログラムのパスです。

カテゴリ	条件付き / 明示
デフォルト	デフォルトなし
調整可能	しない

Init_nodes (enum)

RGM が Init、Fini、Boot、Validate メソッドを呼び出すノードを示します。このプロパティーには、RG_primaries (リソースをマスターできるノードのみ) と RT_installed_nodes (このリソースタイプがインストールされるすべてのノード) のいずれかを指定できます。

カテゴリ	任意
デフォルト	RG_primaries
調整可能	しない

Installed_nodes (string_array)

リソースタイプの実行が許可されるノードの名前のリスト。このプロパティーは RGM によって自動的に作成されます。クラスタ管理者は値を設定できます。このプロパティーは、RTR ファイル内には宣言できません。

カテゴリ	クラスタ管理者による構成が可能です。

デフォルト　すべてのクラスタノード

調整可能　すべての時間

`Is_logical_hostname` (`boolean`)

`TRUE` は、このリソースタイプが、フェイルオーバー IP アドレスを管理する `LogicalHostname` リソースタイプのいずれかのバージョンであることを示します。

カテゴリ　照会のみ

デフォルト　デフォルトなし

調整可能　しない

`Is_shared_address` (`boolean`)

`TRUE` は、このリソースタイプが、共有 IP (インターネットプロトコル) アドレスを管理する `SharedAddress` リソースタイプのいずれかのバージョンであることを示します。

カテゴリ　照会のみ

デフォルト　デフォルトなし

調整可能　しない

`Monitor_check` (`string`)

オプションのコールバックメソッド: 障害モニターの要求によってこのリソースタイプのフェイルオーバーを実行する前に、RGM によって呼び出されるプログラムのパスです。あるノードでモニター検査プログラムがゼロ以外の値で終了した場合、そのノードへのフェイルオーバーへの試みは妨げられます。

カテゴリ　条件付き / 明示

デフォルト　デフォルトなし

調整可能　しない

`Monitor_start` (`string`)

オプションのコールバックメソッド: このタイプのリソースの障害モニターを起動するために RGM によって呼び出されるプログラムのパスです。

カテゴリ　条件付き / 明示

デフォルト　デフォルトなし

調整可能　しない

`Monitor_stop` (`string`)

`Monitor_start` が設定されている場合、必須のコールバックメソッドになります: このタイプのリソースの障害モニターを停止するために RGM によって呼び出されるプログラムのパスです。

カテゴリ　条件付き / 明示

デフォルト　デフォルトなし

調整可能　しない

`Pkglist` (`string_array`)

リソースタイプのインストールに含まれている任意のパッケージリスト。

カテゴリ　条件付き / 明示

デフォルト　デフォルトなし

調整可能　しない

`Postnet_stop` (`string`)

オプションのコールバックメソッド: このタイプのリソースが、あるネットワークアドレスリソースに依存している場合、そのネットワークアドレスリソースの `Stop` メソッドの呼び出し後に RGM によって呼び出されるプログラムのパスです。このメソッドは、ネットワークインタフェースを停止したあとに、`Stop` アクションを行う必要があります。

カテゴリ　条件付き / 明示

デフォルト　デフォルトなし

調整可能　しない

`Prenet_start` (`string`)

オプションのコールバックメソッド: このタイプのリソースが、あるネットワークアドレスリソースに依存している場合、そのネットワークアドレスリソースの `Start` メソッドの呼び出し前に RGM によって呼び出されるプログラムのパスです。このメソッドは、ネットワークインタフェースが構成される前に実行する必要がある `Start` アクションを行う必要があります。

カテゴリ　条件付き / 明示

デフォルト　デフォルトなし

調整可能　しない

`Proxy` (`boolean`)

このタイプのリソースがプロキシリソースであるかどうかを示します。

*プロキシリソース*は、リソースの状態を Oracle Clusterware などの別のクラスタフレームワークからインポートする Oracle Solaris Cluster リソースです。Oracle Clusterware は、クラスタ環境のための、プラットフォームに依存しない一連のシステムサービスです。

プロキシリソースタイプは、`Prenet_start` メソッドを使用して、外部 (プロキシ) リソースの状態をモニターするデーモンを起動します。`Postnet_stop` メソッドは、このモニタリングデーモンを停止します。このモニタリングデーモンは、`CHANGE_STATE_ONLINE` または `CHANGE_STATE_OFFLINE` タグとともに `scha_control` コマンドを実行し、プロキシリソースの状態をそれぞれ `Online` または `Offline` に設定します。`scha_control()` 関数は、`SCHA_CHANGE_STATE_ONLINE` タグと `SCHA_CHANGE_STATE_OFFLINE` タグを同様に使用します。

このプロパティーに `TRUE` を設定した場合、このリソースはプロキシリソースです。

カテゴリ	任意
デフォルト	`FALSE`
調整可能	しない

`Resource_list` (`string_array`)

リソースタイプの全リソースのリストです。管理者はこのプロパティーを直接設定しません。ただし、管理者がこのタイプのリソースをリソースグループに追加したり、リソースグループから削除したりした場合、RGM はこのプロパティーを更新します。

カテゴリ	照会のみ
デフォルト	空のリスト
調整可能	しない

`Resource_type` (`string`)

リソースタイプの名前。現在登録されているリソースタイプ名を表示するには、次のように入力します。

`clresourcetype list`

リソースタイプ名には、バージョン (必須) が含まれます。

`vendor_id.resource_type:version`

リソースタイプ名の 3 つのコンポーネントは、RTR ファイル内に指定された 3 つのプロパティー *vendor-id*、*resource-type* および *RT-version* になります。`clresourcetype` コマンドはピリオド (.) およびコロン (:) 区切り文字を挿入します。リソースタイプ名の接尾辞 `RT_version` には、`RT_version` プロパティーと同じ値が入ります。*vendor-id* が必ず一意になるように、リソースタイプを作成した会社の株式の略号を使用することをお勧めします。

カテゴリ	必須
デフォルト	空の文字列

調整可能　　しない

`RT_basedir` (`string`)

コールバックメソッドの相対パスを補完するディレクトリパスです。このパスは、リソースタイプパッケージのインストール場所に設定します。このパスには、スラッシュ (`/`) で開始する完全なパスを指定する必要があります。すべてのメソッドパス名が絶対パスの場合は、このプロパティーを指定しなくてもかまいません。

カテゴリ　　必須 (すべてのメソッドパスが絶対パスである場合を除く)

デフォルト　　デフォルトなし

調整可能　　しない

`RT_description` (`string`)

リソースタイプの簡単な説明です。

カテゴリ　　条件付き

デフォルト　　空の文字列

調整可能　　しない

`RT_system` (`boolean`)

あるリソースタイプのこのプロパティーに `TRUE` を設定した場合、そのリソースタイプは削除できません (`clresourcetype unregister` *resource-type-name*)。このプロパティーは、`LogicalHostname` など、クラスタのインフラストラクチャのサポートに使用されるリソースタイプを間違って削除してしまうことを防ぐためにあります。しかし、`RT_system` プロパティーはどのリソースタイプにも適用できます。

`RT_system` プロパティーが `TRUE` に設定されたリソースタイプを削除するには、まず、このプロパティーに `FALSE` を設定する必要があります。クラスタサービスをサポートするリソースを持つリソースタイプを削除するときには注意してください。

カテゴリ　　任意

デフォルト　　`FALSE`

調整可能　　すべての時間

`RT_version` (`string`)

このリソースタイプ実装を識別する必須のバージョン文字列。`RT_version` は、完全なリソースタイプ名の末尾の部分です。

カテゴリ　　条件付き/明示または必須

デフォルト　　デフォルトなし

調整可能　　しない

`Single_instance` (`boolean`)

このプロパティーに `TRUE` を設定した場合、RGM は、このタイプのリソースがクラスタ内に 1 つだけ存在することを許可します。

カテゴリ　　任意

デフォルト　　`FALSE`

調整可能　　しない

`Start` (`string`)

コールバックメソッド。このタイプのリソースを起動するために RGM によって呼び出されるプログラムのパスです。

カテゴリ　　RTR ファイルで `Prenet_start` メソッドが宣言されていないかぎり必須

デフォルト　　デフォルトなし

調整可能　　しない

`Stop` (`string`)

コールバックメソッド。このタイプのリソースを停止させるために RGM によって呼び出されるプログラムのパスです。

カテゴリ　　RTR ファイルで `Postnet_stop` メソッドが宣言されていないかぎり必須

デフォルト　　デフォルトなし

調整可能　　しない

`Update` (`string`)

オプションのコールバックメソッド: このタイプの実行中のリソースのプロパティーが変更されたとき RGM によって呼び出されるプログラムのパスです。

カテゴリ　　条件付き / 明示

デフォルト　　デフォルトなし

調整可能　　しない

`Validate` (`string`)

オプションのコールバックメソッド: このタイプのリソースのプロパティー値を検査するために、RGM によって呼び出されるプログラムのパスです。

カテゴリ	条件付き / 明示
デフォルト	デフォルトなし
調整可能	しない

`Vendor_ID` (`string`)

「`Resource_type` プロパティー」を参照してください。

カテゴリ	条件付き
デフォルト	デフォルトなし
調整可能	しない

273 ページの`clresource`(1CL), 305 ページの`clresourcegroup`(1CL), 333 ページの`clresourcetype`(1CL), 1251 ページの`rt_reg`(4), 1389 ページの`SUNW.HAStoragePlus`(5), 1271 ページの`property_attributes`(5), 1287 ページの`r_properties`(5), 1319 ページの`rg_properties`(5), 683 ページの`scha_control`(1HA), 1101 ページの`scha_control`(3HA)

Unresolved link to " Oracle Solaris Cluster Concepts Guide ", Unresolved link to " Oracle Solaris Cluster Data Services Developer's Guide ", Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド "

名前

scalable_service — スケーラブルリソースタイプ

スケーラブルデータサービスとは、Oracle Solaris Cluster ネットワーキング機能を使用するデータサービスのことです。このようなサービスは、Resource Group Manager (RGM) で管理されるリソースタイプとして実装されます。

標準的なリソースプロパティー

すべてのスケーラブルリソースタイプに共通に適用される標準的なリソースプロパティーには、`Scalable`、`Network_resources_used`、`Port_list`、`Load_balancing_policy`、および `Load_balancing_weights` があります。これらのプロパティーの構文および説明については、1287 ページの`r_properties(5)` のマニュアルページを参照してください。

一部のデータサービスは、スケーラブルモードか非スケーラブルモードのいずれかでしか動作しません。これらのデータサービスでは、リソースの作成時に `True` プロパティーの値として `False` か `Scalable` を指定できます。このプロパティーがリソース上で `True` に設定されている場合、そのリソースは「スケーラブルモード」になっているとみなされます。次に、そのリソースをスケーラブルモードのリソースグループ (つまり、`Maximum_primaries` プロパティーを 1 よりも大きい値に設定できるグループ) に含める必要があります。

スケーラブルモードでのみ動作できるデータサービスの場合、`Scalable` プロパティーはこのタイプのリソースに対して暗黙に True であり、管理者が変更することはできません。

`Load_balancing_weights` および `Port_list` プロパティーは、リソースがオンラインである場合を含め、いつでも変更できます。`Network_resources_used` および `Load_balancing_policy` はリソースの作成時に設定され、このあとにこれらのプロパティーを編集できません。そのリソースタイプがどのように実装されているかによって、これらのプロパティーにデフォルト値が設定されている場合もあれば、リソースの作成時に値を指定する必要がある場合もあります。

ネットワークモニタリング

特定のノードで動作するスケーラブルサービスインスタンスは、パブリックネットワークを介してクライアントに応答できる必要があります。RGM は、スケーラブルサービスが動作するノードでのパブリックネットワークの状態を自動的にモニターし、特定のノードでパブリックネットワークがアクセス不能になると、そのノードで動作するスケーラブルサービスインスタンスを停止します。`clresource unmonitor` コマンドを使用してスケーラブルリソースのモニタリングを無効にすると、これらのネットワークチェックは無効になります。

リソースの検証

Scalable に設定された True リソースプロパティーが作成または更新されると、RGM はさまざまなプロパティーを検査し、これらのプロパティーの構成が正しくなければ更新を実行しません。この検査の一部には次のようなものがあります。

- Network_resources_used プロパティーは未設定のままにしないでください。このプロパティーには既存の SharedAddress リソースの名前を含めます。スケーラブルリソースを含むリソースグループの Nodelist プロパティーに対して指定する各ノードは、NetIfList プロパティー、またはいずれかの SharedAddress リソースの AuxNodeList プロパティーに含まれる必要があります。
- スケーラブルリソースを含むリソースグループの RG_dependencies プロパティーは、そのスケーラブルリソースの Network_resources_used プロパティーにリストされているすべての SharedAddress リソースのリソースグループを含むように設定されている必要があります。
- Port_list プロパティーは未設定のままにしないでください。このプロパティーには、ポートとプロトコルの組み合わせが列挙されている必要があります。その場合、プロトコルは、tcp、tcp6、udp、または udp6 のいずれかになります。指定可能なプロトコルは、TCP IPv4 のみの場合は tcp、TCP IPv4 と TCP IPv6 の両方を使用する場合は tcp6、UDP IPv4 のみの場合はudp、UDP IPv4 と UDP IPv6 の両方を使用する場合は udp6 になります。

  たとえば、Port_list=80/tcp,40/udp と指定します。

アフィニティー

IP アフィニティーにより、特定のクライアント IP アドレスからの接続はすべて同じクラスタノードに転送されるようになります。Affinity_timeout、UDP_affinity、Weak_affinity は、Load_balancing_policy が Lb_sticky または Lb_sticky_wild に設定されているときのみ有効です。詳細は、1287 ページのr_properties(5) を参照してください。

273 ページのclresource(1CL), 305 ページのclresourcegroup(1CL),
333 ページのclresourcetype(1CL), 645 ページのrt_callbacks(1HA),
1251 ページのrt_reg(4), 1287 ページのr_properties(5)

Unresolved link to " Oracle Solaris Cluster ソフトウェアのインストール "、Unresolved link to " Oracle Solaris Cluster Data Services Developer's Guide "

名前

SUNW.ScalDeviceGroup, ScalDeviceGroup — スケーラブルデバイスグループのリソースタイプ実装

SUNW.ScalDeviceGroup リソースタイプは、スケーラブルデバイスグループを表します。このリソースタイプのインスタンスは、次に示す種類のデバイスグループの 1 つを表します:

- Solaris Volume Manager for Sun Cluster の複数所有者ディスクセット

SUNW.ScalDeviceGroup リソースタイプは、スケーラブルリソースタイプです。このリソースタイプのインスタンスは、リソースを含むリソースグループのノードリスト内にある各ノードでオンラインです。

このリソースタイプを登録し、リソースタイプのインスタンスを作成するには、次のいずれかの方法を使用します。

- Oracle Solaris Cluster Manager
- Oracle Real Application Clusters 用 Oracle Solaris Cluster サポートを構成するためのオプションを指定する clsetup ユーティリティー
- 次の一連の Oracle Solaris Cluster 保守コマンド。

  1. このリソースタイプを登録するには、clresourcetype コマンドを使用します。
  2. このリソースタイプのインスタンスを作成するには、clresource コマンドを使用します。

SUNW.ScalDeviceGroup リソースタイプに対して定義される標準プロパティーと拡張プロパティーについて、これ以降のサブセクションで説明します。

標準プロパティー

すべての標準リソースプロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

標準のリソースタイププロパティーは、次のようにこのリソースタイプに対して無効にされます。

Monitor_start_timeout

最小
: 10

デフォルト
: 300

Monitor_stop_timeout

最小
10

デフォルト
300

Postnet_stop_timeout

最小
60

デフォルト
300

Prenet_start_timeout

最小
60

デフォルト
300

Start_timeout

最小
60

デフォルト
300

Stop_timeout

最小
60

デフォルト
300

Thorough_probe_interval

デフォルト
300

Update_timeout

最小

60

デフォルト

300

Validate_timeout

最小

60

デフォルト

300

拡張プロパティー

このリソースタイプには、次のような拡張プロパティーが設定されます。

Debug_level

このプロパティーは、このタイプのリソースからのデバッグメッセージが記録されるレベルを指定します。デバッグレベルを高くすると、より多くのデバッグメッセージがログファイルに書き込まれます。

データ型

整数

デフォルト

0

範囲

0–10

調整可能

すべての時間

DiskGroupName

このプロパティーは、リソースが表すデバイスグループの名前を指定します。このプロパティーには次に示す項目の 1 つを設定してください。

- Solaris ボリュームマネージャーの複数所有者ディスクセットの名前。この名前は、ディスクセットの作成に使用した Unresolved link to " metaset1M" コマンドで指定したものです。

指定するデバイスグループの要件は次のとおりです。

- デバイスグループは、既存の有効な複数所有者ディスクセットまたは共有ディスクグループとします。
- デバイスグループは、リソースをマスターできるすべてのノードでホストされるようにします。
- デバイスグループは、スケーラブルデバイスグループリソースをマスターできるすべてのノードからアクセス可能にします。
- デバイスグループには、1 つ以上のボリュームを含めます。

データ型
: 文字列

デフォルト
: デフォルトは定義されていません。

範囲
: 適用不可

調整可能
: 無効時

IOTimeout

このプロパティーは、I/O 検証のタイムアウト値 (秒) を指定します。

デフォルト
: 30

範囲
: 1–1800

調整可能
: すべての時間

LogicalDeviceList

このプロパティーは、リソースの障害モニターでモニターする論理ボリュームのコンマ区切りのリストを指定します。このプロパティーは省略可能です。このプロパティーの値を指定しない場合、デバイスグループ内のすべての論理ボリュームがモニターされます。

デバイスグループのステータスは、モニターされる個々の論理ボリュームのステータスから導き出されます。モニターされているすべての論理ボリュームが健全な場合、そのデバイスグループは健全です。モニターされている論理ボリュームが 1 つでも障害状態の場合、そのデバイスグループは障害状態となります。

個々の論理ボリュームのステータスを取得するには、そのボリュームのボリュームマネージャーに照会します。Solaris ボリュームマネージャーのボリュームのステータスを照会から判

断できない場合、障害モニターはファイルの入出力 (I/O) 操作を実行して、そのステータスを判断します。

デバイスグループが障害状態であることがわかると、グループを表すリソースのモニタリングが停止され、リソースが無効な状態に置かれます。

注記 - ミラー化ディスクの場合、1 つのサブミラーが障害状態でも、デバイスグループは健全であるとみなされます。

注記 - Solaris Volume Manager for Sun Cluster を使用している場合、ソフトパーティションのモニターを指定するには、対応する上または下のレベルのメタデバイスも指定する必要があります。

指定する各論理ボリュームの要件は次のとおりです。

- 論理ボリュームが存在する。
- 論理ボリュームが、`diskgroupname` プロパティーが指定するデバイスグループに含まれている。
- 論理ボリュームが、スケーラブルデバイスグループリソースをマスターできるすべてのノードからアクセス可能である。

データ型
: 文字列配列

デフォルト
: ""

範囲
: 適用不可

調整可能
: すべての時間

Monitor_retry_count

このプロパティーは、障害モニターに許可される、プロセスモニター機能 (PMF) による最大再起動回数を指定します。

データ型
: 整数

デフォルト
: 4

範囲
: 範囲は定義されていません。

調整可能

すべての時間

Monitor_retry_interval

このプロパティーは、PMF が障害モニターの再起動をカウントする期間 (分単位) を指定します。

データ型

整数

デフォルト

2

範囲

範囲は定義されていません。

調整可能

すべての時間

RebootOnFailure

このプロパティーは、検証によって障害が検出された場合に、ローカルシステムをリブートするかどうかを指定します。このプロパティーが `TRUE` に設定されている場合、リソースによって使用されるすべてのデバイスを、ディスクパスのモニタリングで直接または間接的にモニターする必要があります。

`RebootOnFailure` が `TRUE` に設定されており、`GlobalDevicePaths`、`FileSystemMountPoints`、または `Zpools` プロパティーで指定した各エントリで、使用可能なデバイスが少なくとも 1 つ検出された場合、ローカルシステムはリブートされます。ローカルシステムは、リソースがオンライン状態のグローバルクラスタノードまたはゾーンクラスタノードを参照します。

デフォルト

`FALSE`

調整可能

すべての時間

例 400 `SUNW.vucmm_svm` リソースタイプを使用して、`ScalDeviceGroup` リソースを作成

この例では、`SUNW.vucmm_svm` リソースタイプを使用した `datadg` という名前の Solaris Volume Manager for Sun Cluster 複数所有者ディスクセットを表す、`ScalDeviceGroup` リソースを作成する方法を示します。リソースの名前は `scaldatadg-rs` です。この例は、次に示す Oracle Solaris Cluster オブジェクトが存在することを前提とします。

- scaldatadg-rg という名前のスケーラブルリソースグループ。
- vucmm-svm-rs という名前の SUNW.vucmm_svm リソースタイプのインスタンス。

```
# clresourcetype register SUNW.ScalDeviceGroup
# clresource create -t SUNW.ScalDeviceGroup \
-g scaldatadg-rg \
-p Resource_dependencies=vucmm-svm-rs \
-p DiskGroupName=datadg \
scaldatadg-rs
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core

273 ページのclresource(1CL), 333 ページのclresourcetype(1CL), 487 ページのclsetup(1CL), Unresolved link to " metaset1M", Unresolved link to " attributes5", 1287 ページのr_properties(5), 1435 ページのSUNW.vucmm_svm(5)

Unresolved link to " Oracle Solaris Cluster Data Service for Oracle Real Application Clusters ガイド "

名前

SUNW.ScalMountPoint, ScalMountPoint — スケーラブルファイルシステムのマウントポイントのリソースタイプ

SUNW.ScalMountPoint リソースタイプは、スケーラブルファイルシステムマウントポイントを表します。このリソースタイプのインスタンスは、次に示す種類のファイルシステムの 1 つのマウントポイントを表します。

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

  NAS デバイスおよびファイルシステムは、Oracle Solaris Cluster で使用できるように構成済みである必要があります。詳細は、Unresolved link to " Oracle Solaris Cluster With Network-Attached Storage Device Manual "を参照してください。

SUNW.ScalMountPoint リソースタイプは、スケーラブルリソースタイプです。このリソースタイプのインスタンスは、リソースを含むリソースグループのノードリスト内にある各ノードでオンラインです。

このリソースタイプを登録し、リソースタイプのインスタンスを作成するには、次のいずれかの方法を使用します。

- Oracle Solaris Cluster Manager
- Oracle Real Application Clusters 用 Oracle Solaris Cluster サポートを構成するためのオプションを指定する clsetup ユーティリティー
- 次の一連の Oracle Solaris Cluster 保守コマンド。

  1. このリソースタイプを登録するには、clresourcetype コマンドを使用します。
  2. このリソースタイプのインスタンスを作成するには、clresource コマンドを使用します。

SUNW.ScalMountPoint リソースタイプに対して定義される標準プロパティーと拡張プロパティーについて、これ以降のサブセクションで説明します。

標準プロパティー

すべての標準リソースプロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

標準のリソースタイププロパティーは、次のようにこのリソースタイプに対して無効にされます。

Monitor_start_timeout

最小

10

デフォルト

300

Monitor_stop_timeout

最小

10

デフォルト

300

Postnet_stop_timeout

最小

60

デフォルト

300

Prenet_start_timeout

最小

60

デフォルト

300

Start_timeout

最小

60

デフォルト

300

Stop_timeout

最小

60

デフォルト

300

Thorough_probe_interval

デフォルト

300

Update_timeout

最小

60

デフォルト

300

Validate_timeout

最小

60

デフォルト

300

拡張プロパティー

このリソースタイプには、次のような拡張プロパティーが設定されます。

Debug_level

このプロパティーは、ファイルシステムマウントポイントのリソースからのデバッグメッセージが記録されるレベルを指定します。デバッグレベルを高くすると、より多くのデバッグメッセージがログファイルに書き込まれます。

データ型

整数

デフォルト

0

範囲

0–10

調整可能

すべての時間

FileSystemType

このプロパティーは、リソースが表すマウントポイントを持つファイルシステムの種類を指定します。このプロパティーを指定してください。このプロパティーには次に示す値の 1 つを設定します。

nas

ファイルシステムが NAS デバイス上のファイルシステムであることを指定します。

s-qfs

ファイルシステムが Sun QFS 共有ファイルシステムであることを指定します。

データ型

文字列

デフォルト

デフォルトは定義されていません。

範囲

適用不可

調整可能

無効時

IOTimeout

このプロパティーは、障害モニターが入出力 (I/O) 検証に使用するタイムアウト値を秒単位で指定します。障害モニターは、マウントされたファイルシステムが使用可能かどうかを判断するため、ファイルシステム上でテストファイルを開いたり、読み込んだり、書き込んだりするなどの I/O 操作を実行します。I/O 操作がタイムアウト期間内に完了しない場合、障害モニターはエラーを報告します。

データ型

整数

デフォルト

300

範囲

5–1800

調整可能

すべての時間

Monitor_retry_count

このプロパティーは、障害モニターに許可される、プロセスモニター機能 (PMF) による最大再起動回数を指定します。

データ型

整数

デフォルト

4

範囲

範囲は定義されていません。

調整可能

すべての時間

Monitor_retry_interval

このプロパティーは、PMF が障害モニターの再起動をカウントする期間 (分単位) を指定します。

データ型

整数

デフォルト

2

範囲

範囲は定義されていません。

調整可能

すべての時間

MountOptions

このプロパティーは、リソースが表すファイルシステムがマウントされるときに使用されるマウントオプションのコンマ区切りリストを指定します。このプロパティーは省略可能です。このプロパティーの値を指定しないと、マウントオプションは、ファイルシステムのデフォルトの表から取得されます。

- Sun QFS 共有ファイルシステムの場合、これらのオプションは `/etc/opt/SUNWsamfs/samfs.cmd` ファイルから取得されます。
- NAS デバイス上のファイルシステムの場合、これらのオプションは `/etc/vfstab` ファイルから取得されます。

このプロパティーを通して指定するマウントオプションは、ファイルシステムのデフォルトの表内のマウントオプションをオーバーライドします。

データ型

文字列

デフォルト

""

範囲

適用不可

調整可能

無効時

MountPointDir

このプロパティーは、リソースが表すファイルシステムのマウントポイントを指定します。マウントポイントは、ファイルシステムのマウント時にファイルシステムがファイルシステム階層に接続されるディレクトリへのフルパスです。このプロパティーを指定してください。

指定するディレクトリはすでに存在している必要があります。

データ型

文字列

デフォルト

デフォルトは定義されていません。

範囲

適用不可

調整可能

無効時

RebootOnFailure

このプロパティーは、検証によって障害が検出された場合に、ローカルシステムをリブートするかどうかを指定します。このプロパティーが `TRUE` に設定されている場合、リソースによって使用されるすべてのデバイスを、ディスクパスのモニタリングで直接または間接的にモニターする必要があります。

`RebootOnFailure` が `TRUE` に設定されており、`GlobalDevicePaths`、`FileSystemMountPoints`、または `Zpools` プロパティーで指定した各エントリで、使用可能なデバイスが少なくとも 1 つ検出された場合、ローカルシステムはリブートされます。ローカルシステムは、リソースがオンライン状態のグローバルクラスタノードまたはゾーンクラスタノードを参照します。

デフォルト

`FALSE`

調整可能

すべての時間

TargetFileSystem

このプロパティーは、`MountPointDir` 拡張プロパティーが指定するマウントポイントでマウントされるファイルシステムを指定します。このプロパティーを指定してください。ファイルシステムの種類は、`FileSystemType` プロパティーが指定する種類に一致させる必要があります。このプロパティーの書式は、ファイルシステムの種類によって次のように異なります。

- Sun QFS 共有ファイルシステムの場合、このプロパティーにはファイルシステムの作成時にファイルシステムに割り当てられた名前を設定します。ファイルシステムは、正しく構成してください。詳細は、使用している Sun QFS 共有ファイルシステムのドキュメントを参照してください。
- NAS デバイスのファイルシステムの場合は、このプロパティーに *nas-device*:*path* を設定します。上記書式の各項目の意味は次のとおりです。

  nas-device

  ファイルシステムをエクスポートしている NAS デバイスの名前を指定します。この名前は、状況に応じてドメインで修飾できます。

  path

  NAS デバイスがエクスポートしているファイルシステムへのフルパスを指定します。

  NAS デバイスおよびファイルシステムは、Oracle Solaris Cluster で使用できるように構成済みである必要があります。詳細は、Unresolved link to " Oracle Solaris Cluster With Network-Attached Storage Device Manual "を参照してください。

データ型
: 文字列

デフォルト
: デフォルトは定義されていません。

範囲
: 適用不可

調整可能
: 無効時

例 401 `ScalMountPoint` リソースの作成

この例では、Solaris Volume Manager for Sun Cluster と一緒に使用される Sun QFS 共有ファイルシステムのマウントポイントを表す `ScalMountPoint` リソースを作成する方法を示します。リソースの名前は `scal-db_qfs-Data-rs` です。ファイルシステムの特徴は次のとおりです。

- ファイルシステムのマウントポイントは `/db_qfs/Data` です。
- マウントされるファイルシステムは `Data` です。

- マウントオプションは、ファイルシステムのデフォルトの表、`/etc/opt/SUNWsamfs/samfs.cmd` ファイルから取得されます。

この例は、次に示す Oracle Solaris Cluster オブジェクトが存在することを前提とします。

- `scaldatadg-rg` という名前のスケーラブルリソースグループ。
- `qfs-db_qfs-Data-rs` という名前の `SUNW.qfs` リソースタイプのインスタンス。
- `scaldatadg-rs` という名前の `SUNW.ScalDeviceGroup` リソースタイプのインスタンス。

```
# clresourcetype register SUNW.ScalMountPoint
# clresource create -t SUNW.ScalMountPoint \
-g scaldatadg-rg \
-p Resource_dependencies=qfs-db_qfs-Data-rs,scaldatadg-rs \
-p MountPointDir=/db_qfs/Data \
-p FileSystemType=s-qfs \
-p TargetFileSystem=Data \
scal-db_qfs-Data-rs
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core

273 ページの`clresource`(1CL), 333 ページの`clresourcetype`(1CL), 487 ページの`clsetup`(1CL), Unresolved link to " vfstab4", Unresolved link to " attributes5", 1287 ページの`r_properties`(5), 1415 ページの`SUNWct.ScalDeviceGroup`(5), 1433 ページの`SUNW.vucmm_framework`(5)

Unresolved link to " Oracle Solaris Cluster Data Service for Oracle Real Application Clusters ガイド "、Unresolved link to " Oracle Solaris Cluster With Network-Attached Storage Device Manual "

名前

SUNW.SCTelemetry, SCTelemetry — システム資源使用状況に関するデータを収集するためのリソースタイプ

SUNW.SCTelemetry は、システム資源の使用状況に関するデータの収集を可能にするリソースタイプです。SUNW.SCTelemetry は、7 日間にわたり、Java DB データベース内のシステム資源使用状況データを格納します。タイプが SUNW.SCTelemetry のリソースは、タイプが SUNW.derby のリソースに依存します。詳細は、1371 ページのSUNW.derby(5) のマニュアルページを参照してください。

SUNW.SCTelemetry リソースタイプに関連付けられている拡張プロパティーは、次のとおりです。

Extended_accounting_cleanup(boolean)

拡張されているアカウンティングログファイルをクリーンアップするかどうか、つまり履歴データを削除するかどうかを指定します。Extended_accounting_cleanup の取り得る値は、TRUE と FALSE です。

カテゴリ	任意
デフォルト	TRUE
調整可能	すべての時間

Monitor_retry_count(integer)

障害モニターの再起動を制御します。このプロパティーには、障害モニターがプロセスモニター機能により再起動される回数を示しています。このプロパティーは、735 ページのpmfadm(1M) コマンドに渡される -n オプションに相当します。Resource Group Manager (RGM) は、指定された時間窓内の再起動回数をカウントします。詳細については、Monitor_retry_interval プロパティーを参照してください。なお、Monitor_retry_count は、タイプが SUNW.SCTelemetry のリソースではなく、障害モニター自体の再起動回数を指します。

カテゴリ	任意
デフォルト	4
調整可能	すべての時間

Monitor_retry_interval(integer)

RGM が障害モニターで問題が発見された回数をカウントする時間を分単位で指定します。このプロパティーは、735 ページのpmfadm(1M) コマンドに渡される -t オプションに相

当します。障害モニターで問題が発見された回数が Monitor_retry_count プロパティーの値を超過すると、プロセスモニター機能は障害モニターを再起動しません。

カテゴリ	任意
デフォルト	2 分
調整可能	すべての時間

Probe_timeout(integer)

検証機能のタイムアウト値を秒単位で指定します。

カテゴリ	任意
デフォルト	60 秒
最小	2 秒
調整可能	すべての時間

Sampling_interval(integer)

モニタリングデータの収集頻度を指定します。Telemetry_sampling_interval プロパティーには 30 と 3600 の間の値を指定する必要があります。

カテゴリ	必須
デフォルト	60
最小	30 秒
最大	3600 秒
調整可能	すべての時間

735 ページのpmfadm(1M), 1371 ページのSUNW.derby(5)

名前

SUNW.crs_framework, crs_framework — Oracle Clusterware のシャットダウンを調整するリソースタイプ実装

SUNW.crs_framework リソースタイプは、Oracle Solaris Cluster Support for Oracle Real Application Clusters (Oracle RAC) 構成で、Oracle Clusterware リソースと Oracle Solaris Cluster リソースのシャットダウンを調整します。このリソースタイプは、Oracle Solaris Cluster が Oracle Clusterware を停止できるようにすることで、Oracle Solaris Cluster と Oracle Clusterware の相互運用を可能にします。

注記 - このリソースタイプは、Oracle Solaris Cluster 管理コマンドを使用して Oracle Clusterware を起動できるようにするものでは*ありません*。Oracle Clusterware を起動するには、Oracle コマンドを使用するか、ノードをブートするかの方法しかありません。

Oracle Clusterware 投票ディスクと Oracle クラスタレジストリ (OCR) ファイルは、SUNW.ScalDeviceGroup または SUNW.ScalMountPoint タイプのリソースが表すストレージに存在することがあります。この状況では、SUNW.ScalDeviceGroup または SUNW.ScalMountPoint タイプのリソースをオフラインにする前に、Oracle Clusterware を停止する必要があります。この要件に適合するため、SUNW.crs_framework タイプのリソースは、次の状況になると、ノードにある Oracle Clusterware プロセスを停止します:

- そのノードでタイプ SUNW.ScalDeviceGroup または SUNW.ScalMountPoint のリソースがオフラインになるとき。Oracle Clusterware プロセスを次の理由のため停止する必要があります。
  - SUNW.ScalDeviceGroup または SUNW.ScalMountPointタイプのリソースが正しく停止するのを確認するため。
  - Oracle Clusterware プロセスまたは Oracle RAC プロセスがストレージにアクセスしているときに、SUNW.ScalDeviceGroup または SUNW.ScalMountPoint タイプのリソースがオフラインになった場合に、データベースまたはノードに障害が発生するのを防ぐため。
- ノードが停止するとき。Oracle Clusterware プロセスが停止しない場合、ノードはシャットダウンできません。

SUNW.crs_framework リソースタイプは、シングルインスタンスリソースタイプです。したがって、クラスタに作成可能なリソースは 1 個だけです。

Oracle Solaris Cluster が正しい順序でリソースを停止するようにするには、SUNW.crs_framework タイプのリソースを次のように構成します。

- SUNW.crs_framework リソースを含むリソースグループに対して、SUNW.ScalDeviceGroup または SUNW.ScalMountPoint タイプのリソースを含むすべてのリソースグループが、強いポジティブなアフィニティーを宣言することを確認します。
- Oracle Clusterware 投票ディスクと OCR ファイルのストレージを表すすべてのリソースに対して、SUNW.crs_framework リソースによるオフライン再起動の依存関係を設定します。これらのリソースのタイプは、SUNW.ScalDeviceGroup または SUNW.ScalMountPoint です。各依存関係のスコープを、SUNW.ScalDeviceGroup リソースまたは SUNW.ScalMountPoint リソースが動作しているノードだけに限定します。
- SUNW.rac_framework タイプのリソースに対して、SUNW.crs_framework タイプのリソースによる強い依存関係を設定します。

これらの依存関係およびアフィニティーは、Oracle RAC 用 Oracle Solaris Cluster サポートデータサービスのデータベースリソースを構成するときに作成します。詳細は、Unresolved link to " Oracle Solaris Cluster Data Service for Oracle Real Application Clusters ガイド のOracle RAC のサポート データベースインスタンスのリソースの構成"を参照してください。

このリソースタイプを登録し、リソースタイプのインスタンスを作成するには、次のいずれかの方法を使用します。

- Oracle Solaris Cluster Manager
- Oracle Solaris Cluster Support for Oracle Real Application Clusters を構成するためのオプションを指定する 487 ページのclsetup(1CL) ユーティリティー
- 次の一連の Oracle Solaris Cluster 保守コマンド。

  1. このリソースタイプを登録するには、333 ページのclresourcetype(1CL) コマンドを使用します。
  2. このリソースタイプのインスタンスを作成するには、273 ページのclresource(1CL) コマンドを使用します。

標準プロパティー

すべての標準リソースプロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

標準のリソースタイププロパティーは、次のようにこのリソースタイプに対して無効にされます。

Monitor_start_timeout

最小 60

デフォルト 300

`Monitor_stop_timeout`

最小 60

デフォルト 300

`Start_timeout`

最小 60

デフォルト 300

`Stop_timeout`

最小 60

デフォルト 1200

`Update_timeout`

最小 60

デフォルト 300

`Validate_timeout`

最小 60

デフォルト 300

拡張プロパティー

`SUNW.crs_framework` リソースタイプには拡張プロパティーはありません。

例 402 `SUNW.crs_framework` リソースの作成

この例では、`SUNW.crs_framework` リソースタイプを登録し、`crs_framework-rs` という名前の `SUNW.crs_framework` リソースタイプのインスタンスを作成する方法を示します。この例では、次のように仮定します。

- C シェルを使用します。
- `crs-framework-rg` という名前のリソースグループが存在します。
- 次のリソースが存在します。

- `rac_framework-rs` という名前のタイプ `SUNW.rac_framework` のリソースで、Oracle RAC フレームワークを表します。
- `db-storage-rs` という名前のタイプ `SUNW.ScalDeviceGroup` のリソースで、Oracle Clusterware 投票ディスクと OCR ファイルを格納するスケーラブルデバイスグループを表します。

```
phys-schost-1# clresourcetype register SUNW.crs_framework

phys-schost-1# clresource create -g crs-framework-rg \
-t SUNW.crs_framework \
-p resource_dependencies=rac_framework-rs \
-p resource_dependencies_offline_restart=db-storage-rs\{local_node\} \
crs_framework-rs
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	`ha-cluster/library/ucmm`

273 ページの`clresource`(1CL), 333 ページの`clresourcetype`(1CL), 487 ページの`clsetup`(1CL), Unresolved link to " attributes5", 1411 ページの`SUNW.rac_framework`(5), 1415 ページの`SUNWct.ScalDeviceGroup`(5), 1423 ページの`SUNW.ScalMountPoint`(5)

Unresolved link to " Oracle Solaris Cluster Data Service for Oracle Real Application Clusters ガイド "

名前

SUNW.derby, derby — Java DBデータベースのリソースタイプの実装

SUNW.derby は、Oracle Solaris Cluster と一緒に Java DB データベースを使用できるようにするフェイルオーバーリソースタイプです。Java DB データベースは Derby データベースに基づきます。このデータベースについては、http://db.apache.org/derby/ を参照してください。

SUNW.derby リソースタイプに関連付けられている拡張プロパティーは、次のとおりです。

DB_path(string)

Java DB データベースのデータファイルの位置を指定します。

DB_path の値は PATH を指定する文字列です。指定されたパスは選択されたストレージ、たとえば、HAStoragePlus によって制御されたパスにする必要があります。

カテゴリ	必須
調整可能	無効時

DB_port(integer)

Java DB データベースのポートを指定します。

カテゴリ	必須
デフォルト	1527
調整可能	無効時

DB_probe_port(integer)

Oracle Solaris Cluster が Java DB データベースのサーバーの健全性をテストするために使用するポートを指定します。

カテゴリ	必須
デフォルト	1528
調整可能	無効時

Monitor_retry_count(integer)

障害モニターの再起動を制御します。このプロパティーには、障害モニターがプロセスモニター機能により再起動される回数を示しています。このプロパティーは、735 ページのpmfadm(1M) コマンドに渡される -n オプションに相当します。Resource

Group Manager(RGM) は、指定された時間枠の再起動回数をカウントします (`Monitor_retry_interval` を参照)。なお、`Monitor_retry_count` は、タイプが `SUNW.HADerby` のリソースではなく、障害モニター自体の再起動を指します。

カテゴリ	任意
デフォルト	4
調整可能	すべての時間

`Monitor_retry_interval`(integer)

RGM が障害モニターで問題が発見された回数をカウントする時間を分単位で指定します。このプロパティーは、735 ページの`pmfadm`(1M) コマンドに渡される `-t` オプションに相当します。障害モニターで問題が発見された回数が拡張プロパティー `Monitor_retry_count` の値を超過すると、プロセスモニター機能は障害モニターを再起動しません。

カテゴリ	任意
デフォルト	2 分
調整可能	すべての時間

`Probe_timeout`(integer)

プローブコマンドのタイムアウト値を秒単位で指定します。

カテゴリ	任意
デフォルト	120 秒
デフォルト	2 秒
調整可能	すべての時間

735 ページの`pmfadm`(1M)

名前

SUNW.Event — Cluster Reconfiguration Notification Protocol (CRNP) のリソースタイプの実装

SUNW.Event リソースタイプ実装は、高可用性を備えた Oracle Solaris Cluster の CRNP サービスを提供します。この実装は、Oracle Solaris Cluster Resource Group Manager (RGM) の下でリソースを管理することにより、通知デーモン (/usr/cluster/lib/sc/cl_apid) の可用性を高めます。SUNW.Event リソースを含むリソースグループは、同じリソースグループ内に構成されたネットワークグループを持つ必要があります。クラスタ上には、SUNW.Event 型の単一リソースしか存在できません。

このオプションは大域ゾーンだけで実行できます。

標準プロパティー

このセクションでは、この実装の動作を制御する主な標準プロパティーについて説明します。SUNW.Event リソース上でこれらのプロパティーを設定するには、clresource コマンドを使用します。これらのリソースプロパティーの詳細は、1287 ページのr_properties(5) のマニュアルページを参照してください。

Network_resources_used (string_array)

このリソースが依存関係を持っている論理ホスト名または共有アドレスネットワーク資源のリスト。このリストには、プロパティー Resource_dependencies、Resource_dependencies_weak、Resource_dependencies_restart、または Resource_dependencies_offline_restart にあるすべてのネットワークアドレスリソースが含まれます。

このプロパティーは、リソース依存関係プロパティーの設定に基づいて、RGM により自動的に更新されます。このプロパティーを直接設定することはありません。その代わりに、Resource_dependencies プロパティーを使用します。

カテゴリ	条件付き/任意
デフォルト	空のリスト
調整可能	無効時

Port_list (string_array)

サーバーが待機するポート番号をコンマで区切ったリストです。Port_list の詳細は、1287 ページのr_properties(5) のマニュアルページを参照してください。

カテゴリ	条件付き/必須

デフォルト　　9444/tcp

調整可能　　無効時

`Resource_dependencies` (`string array`)

リソースが依存するリソースのリスト。このリストには、リソースが使用するすべての論理ホスト名または共有アドレスネットワーク資源が含まれます。このプロパティーのデフォルト値は null です。アプリケーションを 1 つ以上の特定のアドレスにバインドする必要がある場合は、このプロパティーを指定してください。ネットワーク資源の依存関係を指定しない場合、アプリケーションはすべてのアドレスで待機します。

イベントリソースを作成する前に、`LogicalHostname` または `SharedAddress` リソースの構成を済ませてください。

リソース名は 1 つでも複数でも指定できます。各ネットワーク資源には 1 つまたは複数の論理ホスト名を指定できます。詳細は、251 ページの`clreslogicalhostname`(1CL) および 347 ページの`clressharedaddress`(1CL) のマニュアルページを参照してください。

`Resource_dependencies` プロパティーの代わりに、`Resource_dependencies_weak`、`Resource_dependencies_restart`、または `Resource_dependencies_offline_restart` プロパティーを使用すると、ほかの種類の依存関係も指定できます。詳細は、1287 ページの`r_properties`(5) のマニュアルページを参照してください。

カテゴリ　　任意

デフォルト　　空のリスト

調整可能　　すべての時間

`Retry_count` (`integer`)

起動に失敗したリソースをモニターが再起動する回数です。`Retry_count` の詳細は、1287 ページの`r_properties`(5) のマニュアルページを参照してください。

注記 - このプロパティーに負の値を指定すると、モニターは無限回リソースを再起動を試みます。

カテゴリ　　条件付き

デフォルト　　2

最大　　10

調整可能　　すべての時間

`Retry_interval` (`integer`)

失敗したリソースの再起動の間、この秒数だけ待機します。`Retry_interval` の詳細は、1287 ページの`r_properties`(5) のマニュアルページを参照してください。

カテゴリ 条件付き

デフォルト 300

最大 3600

調整可能 すべての時間

`Thorough_probe_interval` (整数)

高オーバーヘッドのリソース障害検証の呼び出し間隔 (秒)。`Thorough_probe_interval` の詳細は、1287 ページの`r_properties(5)` のマニュアルページを参照してください。

カテゴリ 条件付き

デフォルト 60

最大 3600

調整可能 すべての時間

拡張プロパティー

このセクションでは、この実装の動作を制御する主な拡張プロパティーについて説明します。

`Allow_hosts` (`string_array`)

このプロパティーは、この実装で登録可能な一連のクライアントを制御して、クラスタ再構成イベントを受信します。このプロパティーは、通常 `ipaddress/masklength` の形式で登録可能なクライアントのサブネットを定義します。たとえば、`129.99.77.0/24` の設定では、サブネット`129.99.77` 上のクライアントのイベント登録が許可されます。また `192.9.84.231/32` の設定では、クライアント `192.9.84.231` だけのイベント登録が許可されます。

さらに、次の特殊キーワードが認識されます。

- `LOCAL` は、クラスタの直接接続されたサブネット上に存在する全クライアントです。
- `ALL` では、すべてのクライアントが登録可能です。

注記 - `Allow_hosts` と `Deny_hosts` プロパティーの両方で 1 つのエントリと一致するクライアントの場合、この実装で登録できません。

カテゴリ 任意

デフォルト `LOCAL`

調整可能　すべての時間

`Client_retry_count` (整数)

このプロパティーは、外部クライアントとの通信中に実装が試行する回数を制御します。`Client_retry_count` の試行回数内に応答しなかったクライアントはタイムアウトになり、クラスタ再構成イベントの受信資格を持つクライアントの登録リストから削除されます。続いてクライアントは、クラスタ再構成イベントを受け取ることができる登録済みクライアントのリストから削除されます。再びイベントを受信し始めるには、このクライアントは再登録する必要があります。`Client_retry_interval` プロパティーのセクションでは、この実装が再試行する頻度について説明します。

カテゴリ　任意

デフォルト　3

最小　1

調整可能　すべての時間

`Client_retry_interval` (整数)

このプロパティーは、応答しない外部クライアントとの通信中に使用される時間範囲 (秒) を定義します。この時間内に、最大 `Client_retry_count` 回のクライアント接続が試行されます。

ここで定義した値は、随時変更できます。

カテゴリ　任意

デフォルト　1800

最小　30

調整可能　すべての時間

`Client_timeout` (integer)

このプロパティーでは、外部クライアントとの通信中に使用されるタイムアウト値 (秒) を指定します。ただしこの実装は、調整可能な回数だけクライアントとの通信を試みます。`Client_retry_count` プロパティーと `Client_retry_interval` プロパティーのセクションに、このプロパティーの調整方法についての説明があります。

カテゴリ　任意

デフォルト　60

最小　30

調整可能　すべての時間

Deny_hosts (string_array)

このプロパティーは、クラスタ再構成イベントの受信候補として登録できないクライアントの設定を制御します。アクセスを確定するために、このプロパティーの設定は Allow_hosts リストの設定に優先します。このプロパティーの形式は、Allow_hosts で定義された形式と同じです。

カテゴリ	任意
デフォルト	NULL
調整可能	すべての時間

Max_clients (整数)

このプロパティーは、クラスタイベント通知の受信候補として登録可能なクライアントの最大数を制御します。新規クライアントによるイベント登録の試行は拒否されます。個々のクライアント登録にはクラスタ上のリソースが使用されるため、このプロパティーの値を調整することで、外部クライアントによるクラスタ上のリソース使用を制御することができます。

カテゴリ	任意
デフォルト	1000
最小	1
調整可能	無効時

例 403 デフォルトのプロパティーを使用した SUNW.Event リソースの作成

この例では、CRNP という名前のフェイルオーバー SUNW.Event リソースを、events-rg という既存グループ内に作成する方法を示します。events-rg リソースグループには、対応するフェイルオーバーホスト名を示す LogicalHostname または SharedAddress のリソースが含まれます。

```
# clresourcetype register SUNW.Event
# clresource create -g events-rg -t SUNW.Event CRNP
```

この例で作成される SUNW.Event リソースの名前は CRNP です。このリソースは、ポート 9444 上で通信を行い、直接接続されたサブネット上の全クライアントのイベント登録を可能にします。

例 404 デフォルトではないプロパティーを使用した SUNW.Event リソースの作成

この例では、CRNP という名前の SUNW.Event リソースを、events-rg というリソースグループ内に作成する方法を示します。CRNP リソースは、ポート 7000 上と、固有のネットワーク資源 foo-1 (events-rg 内に構成済み) 上で通信するように構成されます。この CRNP リソースは、サブ

ネット 192.9.77.0 上のクライアントと、直接接続されたサブネット上のクライアントの登録を可能にしますが、クライアント 192.9.77.98 に対して実装の使用を拒否します。

```
# clresource create -g events-rg -t SUNW.Event \
-p Port_list=7000/tcp -p Network_resources_used=foo-1 \
-p Allow_hosts=LOCAL,192.9.77.0/24 \
-p Deny_hosts=192.9.77.98/32 CRNP
```

`/usr/cluster/lib/sc/cl_apid`

CRNP デーモン。

`/usr/cluster/lib/sc/events/dtds`

CRNP プロトコル用のデータタイプ定義を含むディレクトリ。

次の属性の詳細は、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core

273 ページの`clresource`(1CL), 305 ページの`clresourcegroup`(1CL), 333 ページの`clresourcetype`(1CL), 691 ページの`scha_resource_get`(1HA), Unresolved link to " attributes5", 1287 ページの`r_properties`(5)

名前

SUNW.gds — 単純なネットワーク認識型およびネットワーク非認識型アプリケーションの可用性またはスケーラビリティーを向上させるためのリソースタイプ

汎用データサービス (GDS: Generic Data Service) は、単純なネットワーク認識型およびネットワーク非認識型アプリケーションを Oracle Solaris Cluster Resource Group Manager (RGM) フレームワークにプラグインすることにより、これらのアプリケーションの可用性またはスケーラビリティーを向上できるようにするためのメカニズムです。

GDS には、コールバックメソッド (645 ページの rt_callbacks(1HA)) やリソースタイプ登録 (RTR) ファイル (1251 ページの rt_reg(4)) を備えた、フル機能の Oracle Solaris Cluster リソースタイプが含まれています。

標準プロパティー

`Failover_mode` (`enum`)

リソースが正常に開始または停止できなかった場合、またはリソースモニターが正常ではないリソースを検出し、その結果再起動またはフェイルオーバーを要求する場合に RGM が取る回復アクションを変更します。

`Failover_mode` プロパティーの詳細は、1287 ページの r_properties(5) のマニュアルページを参照してください。

カテゴリ	任意
デフォルト	SOFT
調整可能	すべての時間

`Network_resources_used` (`string array`)

このリソースが依存関係を持っている論理ホスト名または共有アドレスネットワーク資源のリスト。このリストには、プロパティー `Resource_dependencies`、`Resource_dependencies_weak`、`Resource_dependencies_restart`、または `Resource_dependencies_offline_restart` にあるすべてのネットワークアドレスリソースが含まれます。

このプロパティーは、リソース依存関係プロパティーの設定に基づいて、RGM により自動的に更新されます。このプロパティーを直接設定することはありません。その代わりに、`Resource_dependencies` プロパティーを使用します。

カテゴリ	条件付き/任意
デフォルト	空のリスト

調整可能　　無効時

`Port_list` (`string array`)

アプリケーションが待機するポート番号をコンマで区切ったリストで指定します。1287 ページの`r_properties`(5) のマニュアルページを参照してください。

`Port_list` プロパティーは、Agent Builder が生成する起動スクリプトか、`clresource` コマンド (Oracle Solaris Cluster 管理コマンドを使用する場合) に指定されていなければなりません。

注記 - `Network_aware` プロパティーが `False` に設定されている場合、`Port_list` は不要です。

各ポート番号には、スラッシュ (`/`) と、`Port_list=80/tcp` や `Port_list=80/tcp6,40/udp6` などのそのポートで使用されるプロトコルが付加されます。

プロトコルには、次の値を指定できます。

- `tcp` (TCP IPv4)
- `tcp6` (TCP IPv6)
- `udp` (UDP IPv4)
- `udp6` (UDP IPv6)

カテゴリ　　条件付き/必須

デフォルト　　デフォルトなし

調整可能　　すべての時間

`Resource_dependencies` (`string array`)

リソースが依存するリソースのリストを指定します。このリストには、リソースが使用するすべての論理ホスト名または共有アドレスネットワーク資源が含まれます。このプロパティーのデフォルト値は null です。`Network_aware` プロパティーが true に設定されている場合は、このプロパティーの値として、アプリケーションが待機している論理ホスト名または共有アドレスネットワーク資源を設定します。

GDS リソースを作成する前には、`LogicalHostname` または `SharedAddress` リソースがすでに構成されている必要があります。

リソース名は 1 つでも複数でも指定できます。各ネットワーク資源には 1 つまたは複数の論理ホスト名を指定できます。詳細は、251 ページの`clreslogicalhostname`(1CL) および 347 ページの`clressharedaddress`(1CL) のマニュアルページを参照してください。

`Resource_dependencies` プロパティーの代わりに、`Resource_dependencies_weak` 、`Resource_dependencies_restart`、または `Resource_dependencies_offline_restart` プロパティーを使用すると、ほかの種類の依存関係も指定できます。詳細は、1287 ページの`r_properties`(5) のマニュアルページを参照してください。

カテゴリ　任意

デフォルト　空のリスト

調整可能　すべての時間

`Retry_count` (`integer`)

起動に失敗したリソースをモニターが再起動する回数です。

`Retry_count` プロパティーの詳細は、1287 ページの`r_properties(5)` のマニュアルページを参照してください。

カテゴリ　条件付き

デフォルト　2

調整可能　すべての時間

`Retry_interval` (`integer`)

失敗したリソースの再起動の間、この秒数だけ待機します。

`Retry_interval` プロパティーの詳細は、1287 ページの`r_properties(5)` のマニュアルページを参照してください。

カテゴリ　条件付き

デフォルト　370 秒

調整可能　すべての時間

`Scalable` (`boolean`)

リソースがスケーラブルであるかどうか、つまり、リソースが Data Service for Apache Guide Cluster ソフトウェアのネットワーキング負荷分散機能を使用するかどうかを示します。

`Scalable` プロパティーが `TRUE` に設定されている場合は、`Load_balancing_policy` や `Load_balancing_weights` などの追加プロパティーを使用して負荷分散動作を構成します。

`Scalable`、`Load_balancing_policy`、`Load_balancing_weights` の各プロパティーの詳細は、1287 ページの`r_properties(5)` のマニュアルページを参照してください。

カテゴリ　任意

デフォルト　FALSE

調整可能　作成時

`Start_timeout` (`integer`)

起動コマンドのタイムアウト値を秒単位で指定します。

カテゴリ	任意
最小	60 秒
デフォルト	300 秒
調整可能	すべての時間

`Stop_timeout` (`integer`)

停止コマンドのタイムアウト値を秒単位で指定します。

カテゴリ	任意
最小	60 秒
デフォルト	300 秒
調整可能	すべての時間

`Validate_timeout` (`integer`)

検証コマンドのタイムアウト値を秒単位で指定します。

カテゴリ	任意
最小	60 秒
デフォルト	300 秒
調整可能	すべての時間

拡張プロパティー

`Child_mon_level` (`integer`)

プロセスモニター機能 (PMF) によってモニターされるプロセスを制御します。このプロパティーには、フォークされた子プロセスのモニターレベルを指定します。このプロパティーを省略したり、このプロパティーをデフォルト値に設定したりすると、735 ページの`pmfadm`(1M) の `-C` オプションを省略した場合と同じになり、すべての子 (およびその子孫) がモニターされます。

カテゴリ	任意
デフォルト	`-1`
調整可能	作成時

`Failover_enabled` (`boolean`)

リソースのフェイルオーバーを許可します。プロパティーの値が `False` の場合、リソースのフェイルオーバーは無効です。このプロパティーを使用して、アプリケーションリソースによるリソースグループのフェイルオーバーを防ぐことができます。

カテゴリ　任意

デフォルト　`True`

調整可能　無効時

注記 - `Failover_mode=RESTART_ONLY` 設定は、`Failover_enabled=False` 設定の動作と一致します。`Failover_mode=LOG_ONLY` 設定はさらに、リソースが再起動されないようにします。フェイルオーバー動作をより適切に制御するには、`Failover_enabled` 拡張プロパティーの代わりに `Failover_mode` プロパティーを使用します。詳細は、1287 ページの`r_properties(5)` にある `Failover_mode` の `LOG_ONLY` および `RESTART_ONLY` 値の説明を参照してください。

`Log_level` (`enum`)

GDS によってログに記録される診断メッセージのレベル (つまり、種類) を指定します。`None`、 `Info`、または `Err` のいずれかを指定できます。`None` を指定した場合、GDS によって診断メッセージは記録されません。`Info` を指定した場合、情報メッセージとエラーメッセージが記録されます。`Err` を指定した場合、エラーメッセージだけが記録されます。

カテゴリ　任意

デフォルト　`Info`

調整可能　すべての時間

`Monitor_retry_count`

`Monitor_retry_interval` プロパティーで指定する時間枠内でプロセスモニター機能 (PMF) が障害モニターを再起動する回数。このプロパティーは、リソースではなく障害モニター自身の再起動を参照します。システム定義のプロパティー `Retry_interval` および `Retry_count` は、リソースの再起動を制御します。

カテゴリ　任意

データ型　整数

デフォルト　`4`

範囲　`0` - `2147483647`

`-1` は、無限の再試行回数を示します。

調整可能　　常時

`Monitor_retry_interval`

障害モニターの障害が発生している時間 (分)。障害モニターで障害が発生した回数が、この期間内で拡張プロパティー `Monitor_retry_count` で指定した値を超過すると、PMF は障害モニターを再起動しません。

カテゴリ　　任意

データ型　　整数

デフォルト　　`2`

範囲　　`0` – `2147483647`

`-1`は、無限の再試行間隔を示します。

調整可能　　常時

`Network_aware` (`boolean`)

このプロパティーは、アプリケーションがネットワークを使用するかどうかを指定します。

カテゴリ　　任意

デフォルト　　`True`

調整可能　　作成時

`Probe_command` (`string`)

ネットワーク認識型アプリケーションまたはネットワーク非認識型アプリケーションの健全性を定期的にチェックするコマンドを指定します。アプリケーションを検証するには、シェルに直接渡すことができる完全なコマンドを指定します。アプリケーションが正常に実行されていれば、検証コマンドは終了ステータスとして `0` を返します。

プローブコマンドの終了ステータスは、アプリケーションの障害の重大度を判断するために使用されます。終了ステータス (プローブステータス) は、`0` (正常) から `100` (全面的な障害) までの整数になります。検証ステータスの値が `201` で、`Failover_enabled` の値が `False` に設定されていなければ、アプリケーションのフェイルオーバーが行われます。

検証ステータスは、GDS 検証アルゴリズム内で使用され、アプリケーションをローカルで再起動するか別のノードにフェイルオーバーするかを決定します。`probe` コマンドを省略した場合は、GDS 自身が、ネットワーク資源上のアプリケーションに接続する単純な検証手段を提供します。GDS は、接続に成功すると、接続をただちに切り離します。接続と接続の切断が両方とも正常なら、アプリケーションは正常に動作しているものとみなされます。

GDS では、ネットワーク非認識型アプリケーションに対する「デフォルトの」プローブ動作は提供されません。ただし、ネットワーク非認識型アプリケーションは、アプリケーションをモニ

ターし、障害が発生した場合はアプリケーションを再起動する PMF の下で起動されます。詳細は、735 ページのpmfadm(1M) のマニュアルページを参照してください。

カテゴリ　任意

デフォルト　NULL

調整可能　無効時

`Probe_timeout` (`integer`)

プローブコマンドのタイムアウト値を秒単位で指定します。

カテゴリ　任意

最小　2 秒

デフォルト　30 秒

調整可能　すべての時間

`Start_command` (`string`)

アプリケーションを起動するコマンドを指定します。直接シェルに渡すことができる完全なコマンドを指定してください。

起動コマンド、またはそれにフォークされる子プロセスは、デーモンのように長く実行され、クライアントにサービスを実際に提供するプログラムである必要があります。起動コマンドのプロセスツリーはプロセスモニター機能 (PMF) によってモニターされます (`Child_mon_level` 拡張プロパティーを参照)。このようにモニターされているプロセスが終了すると、これらのプロセスは、リソースプロパティー `Retry_count` と `Retry_interval` の設定に従って再起動されます。再試行の回数を超えると、そのリソースグループを異なるノードに再配置することが試みられます。

起動コマンド (または、その子プロセス) が返す終了ステータスは無視されます。

カテゴリ　必須

最小　1

デフォルト　デフォルトなし

調整可能　無効時

`Stop_command` (`string`)

アプリケーションを停止するコマンドを指定します。直接シェルに渡すことができる完全なコマンドを指定してください。このプロパティーを指定しない場合、または停止コマンドがゼロ以外で終了した場合は、GDS はシグナルを使用して、このアプリケーションを停止します。

カテゴリ　任意

デフォルト　NULL

調整可能　無効時

`Stop_signal` (`integer`)

アプリケーションを停止するシグナルを指定します。このプロパティーの値は、Unresolved link to " signal3HEAD" に定義されているものと同じです。

カテゴリ　任意

最小　1

最大　37

デフォルト　15

調整可能　無効時

`Validate_command` (`string`)

アプリケーションを検証するコマンドへの絶対パスを指定します。絶対パスを指定しない場合、アプリケーションは検証されません。

検証コマンドの終了スタータスは、GDS リソースの作成または更新を許可するべきかどうかを決定するのに使用されます。指定された検証コマンドは、リソースを作成または更新する前に、そのリソースを含むリソースグループのノードリストにある各ノードで実行されます。検証コマンドがゼロ以外で終了した場合、要求されたリソースの作成または更新は許可されません。検証コマンドが `stdout` または `stderr` に書き込んだ出力は、リソースを作成または更新するための管理コマンドを発行したユーザーに戻されます。このような出力を使用すると、リソースの検証が失敗した理由を説明できます。

検証コマンドは、`gds` リソースがオンラインになるときに、`Start_command` 拡張プロパティーを実行する前にも実行されます。検証コマンドがゼロ以外で終了した場合、起動の失敗として扱われます。

検証コマンドはまた、リソースグループを新しいノードに再配置するために `scha_control` コマンドの `GIVEOVER` オプションを実行する前にも実行されます。このコマンドがゼロ以外で終了した場合、ギブオーバーはブロックされ、リソースグループは引き続き現在のノードをマスターとします。

カテゴリ　任意

デフォルト　NULL

調整可能　無効時

次では、GDS を使って`app` という名前のアプリケーションを高可用性アプリケーションにする例を紹介します。また、Oracle Solaris Cluster Agent Builder

(653 ページのscdsbuilder(1HA)) を使用して、これらのコマンドを含むスクリプトを作成することもできます。

基本的な例

この例は、SUNW.gds リソースタイプの登録、アプリケーションのリソースグループの作成、論理ホスト hhead のリソース LogicalHostname の作成、ネットワーク認識型アプリケーションリソースの作成、リソースグループの管理、すべてのリソースの有効化、リソースのオンライン化を行う方法を示しています。

この時点で、このアプリケーションは起動し、高可用性になり、GDS により指定された単純な検証プログラムによってモニターされます。これでアプリケーションのステータスをチェックできるようになります。

```
# clresourcetype register SUNW.gds
# clresourcegroup create rg1
# clreslogicalhostname create -g rg1 -h hhead
# clresource create -g rg1 -t SUNW.gds \
-p Start_command="/usr/local/app/bin/start" \
-p Port_list="1234/tcp" -p Network_aware=True \
-p Resource_dependencies=hhead app-rs
# clresourcegroup online -M rg1
# clresourcegroup status +
```

複雑な例

この例は、SUNW.gds リソースタイプの登録、アプリケーションのリソースグループの作成、論理ホスト hhead のリソース LogicalHostname の作成、ネットワーク認識型アプリケーションリソースの作成、エラーメッセージのみの記録、リソースグループの管理、すべてのリソースの有効化、リソースのオンライン化を行う方法を示しています。

この時点で、このアプリケーションは起動し、高可用性になり、Probe_command により指定された障害モニターによってモニターされます。これでアプリケーションのステータスをチェックできるようになります。

```
# clresourcetype register SUNW.gds
# clresourcegroup create rg1
# clreslogicalhostname create -g rg1 -h hhead
# clresource create -g rg1 -t SUNW.gds \
-p Start_command="/usr/local/app/bin/start" \
-p Stop_command="/usr/local/app/bin/stop" \
-p Validate_command="/usr/local/app/bin/config" \
-p Probe_command="/usr/local/app/bin/probe" \
-p stop_signal=9 -p failover_enabled=FALSE \
-p Start_timeout=120 -p Stop_timeout=180 \
-p Port_list="1234/tcp" -p Probe_timeout=60 \
-p Network_aware=True \
```

```
-p Validate_timeout=120 -p Log_level=Err \
-p Resource_dependencies=hhead app-rs
# clresourcegroup online -M rg1
# clresourcegroup status +
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/ha-service/gds

251 ページの clreslogicalhostname(1CL), 273 ページの clresource(1CL), 305 ページの clresourcegroup(1CL), 333 ページの clresourcetype(1CL), 347 ページの clressharedaddress(1CL), 645 ページの rt_callbacks(1HA), 653 ページの scdsbuilder(1HA), 683 ページの scha_control(1HA), 691 ページの scha_resource_get(1HA), 733 ページの hatimerun(1M), 735 ページの pmfadm(1M), Unresolved link to " signal3HEAD", 1251 ページの rt_reg(4), Unresolved link to " attributes5", 1287 ページの r_properties(5), 1347 ページの scalable_service(5)

名前

SUNW.HAStoragePlus — Oracle Solaris Cluster デバイスサービス、ファイルシステム、およびデータサービス間の依存関係を強制し、これらのエンティティーをモニターするリソースタイプ

SUNW.HAStoragePlus は、データサービスリソースと、デバイスグループ、クラスタファイルシステム、およびローカルファイルシステム間の依存関係を指定するためのリソースタイプです。

注記 - ローカルファイルシステムには、UFS、Oracle の Sun QFS、および Oracle Solaris ZFS が含まれます。

このリソースタイプを使用すると、依存するデバイスグループおよびファイルシステムが使用可能なことが保証されてはじめてデータサービスをオンラインにできます。SUNW.HAStoragePlus リソースタイプは、ファイルシステムのマウント、マウント解除、およびチェックをサポートします。

リソースグループだけでは、ディスクデバイスグループ、クラスタファイルシステム、またはローカルファイルシステムとの直接同期は提供されません。このため、クラスタのリブートまたはフェイルオーバー時に、依存する広域デバイスやクラスタファイルシステムは使用できないにも関わらず、データサービスの起動が試行されます。その結果、データサービスの START メソッドがタイムアウトになり、データサービスで障害が発生する可能性があります。

SUNW.HAStoragePlus リソースタイプは、1 つ以上のデータサービスリソースによって使用されるデバイスグループ、クラスタ、およびローカルファイルシステムを表します。これらは、リソースグループに SUNW.HAStoragePlus タイプのリソースを追加し、その他のリソースと SUNW.HAStoragePlus リソース間の依存関係を設定します。

アプリケーションリソースが HAStoragePlus リソース上に構成されている場合、アプリケーションリソースはその下にある HAStoragePlus リソースとのオフライン再起動の依存関係を定義しなければなりません。これによって、依存する HAStoragePlus リソースがオンラインになったあとでアプリケーションリソースがオンラインになり、HAStoragePlus resource がオフラインになる前にアプリケーションリソースがオフラインになります。例:

```
# clresource set \
-p Resource_dependencies_offline_restart=hasp_rs \
applicaton_rs
```

このような依存関係により、データサービスリソースは次の状況が発生したあとにオンラインになることが保証されます。

1. 指定のすべてのサービスが使用可能な状態になっており、必要に応じて連結されている。

2. 指定のすべてのファイルシステムがチェックされ、マウントされている

SUNW.HAStoragePlus リソースタイプも、グローバルデバイス、ファイルシステム、ZFS ストレージプールなどの HAStoragePlus リソースによって管理されるエンティティーの健全性をモニターするための障害モニターを提供します。障害モニターは定期的に障害プローブを実行します。いずれかのエンティティーが使用できない状態になると、リソースが再起動されるか、別のノードへのフェイルオーバーが実行されます。

複数のエンティティーがモニターされている場合、障害モニターはすべてのエンティティーを同時にプローブします。グローバルデバイスのモニター対象、raw デバイスグループ、Oracle Solaris Volume Manager デバイスグループ、ファイルシステム、および ZFS ストレージプールのリストを表示するには、Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド の第 2 章データサービスリソースの管理"を参照してください。

HAStoragePlus リソースの障害モニターは、ファイルシステムの読み取りと書き込みを行なうことで、管理対象のデバイスおよびファイルシステムを検証します。読み取り操作が I/O スタックのソフトウェアによってブロックされており、HAStoragePlus リソースがオンラインである必要がある場合、ユーザーは障害モニターを無効にしなければなりません。たとえば、Availability Suite は、ビットマップボリュームまたは NEED SYNC 状態のデータボリュームからの読み取りをブロックするため、Oracle Solaris の Availability Suite Remote Replication ボリューム機能を管理する HAStoragePlus リソースのモニターを解除する必要があります。Availability Suite ボリュームを管理する HAStoragePlus リソースは、常にオンラインである必要があります。

ZFS ファイルシステムの mountpoint プロパティーが none か legacy に設定されているか、または canmount プロパティーが off に設定されている場合、HAStoragePlus リソースはそのファイルシステムをモニターしません。その他のすべての ZFS ファイルシステムについて、HAStoragePlus リソースの障害モニターは、そのファイルシステムがマウントされているかどうかを確認します。ファイルシステムがマウントされている場合、HAStoragePlus リソースは、ReadOnly/ReadWrite という IOOption プロパティーの値に応じて、ファイルシステムの読み取りと書き込みを行うことによってそのアクセス可能性をプローブします。

ZFS ファイルシステムがマウントされていないか、またはファイルシステムのプローブが失敗した場合、リソースの障害モニターは失敗し、リソースは Faulted に設定されます。RGM はファイルシステムを再起動しようとします (リソースの retry_count および retry_interval プロパティーによって決定される)。先に説明した mountpoint プロパティーと canmount プロパティーの特定の設定が有効でない場合は、このアクションによってファイルシステムが再マウントされます。障害モニターが引き続き失敗し、retry_interval 内に retry_count を超えた場合、RGM はリソースを別のノードにフェイルオーバーします。

標準プロパティー

次の標準プロパティーは、SUNW.HAStoragePlus リソースタイプと関連付けられています。

Thorough_probe_interval

障害プローブの起動とリソースの起動の間の時間ウィンドウを秒単位で定義します。

カテゴリ

任意

最小

5

最大

3600

デフォルト

180

調整可能

すべての時間

拡張プロパティー

SUNW.HAStoragePlus リソースタイプには、次に示す拡張プロパティーが関連付けられています:

AffinityOn

SUNW.HAStoragePlus リソースが、GlobalDevicePaths および FilesyStemMountPoints 拡張プロパティーに定義されているすべてのグローバルデバイスのアフィニティースイッチオーバーを実行する必要があるかどうかを指定します。TRUE または FALSE を指定できます。

カテゴリ

任意

デフォルト

TRUE

調整可能

無効時

Zpools 拡張プロパティーは、AffinityOn 拡張プロパティーを無視します。AffinityOn 拡張プロパティーは、GlobalDevicePaths および FileSystemMountPoints 拡張プロパティーとのみ一緒に使用するよう意図されています。

AffinityOn 拡張プロパティーを FALSE に設定すると、SUNW.HAStoragePlus リソースは、指定されたグローバルサービスが使用可能な状態になるのを受動的に待機します。この場

合、各オンライングローバルデバイスサービスのプライマリノードがリソースグループのプライマリノードと一致しなくなる可能性があります。

アフィニティースイッチオーバーの目的は、特定のノードでのデバイスグループとリソースグループのコロケーションを確保し、パフォーマンスを向上することです。データの読み取りおよび書き込みは、常にデバイスのプライマリパスで行われます。アフィニティースイッチオーバーを行うには、リソースグループのプライマリノード候補のリストとデバイスグループのノードリストが等価でなければなりません。SUNW.HAStoragePlus リソースがオンラインになると、SUNW.HAStoragePlus リソースは 1 回だけ、各デバイスサービスにアフィニティースイッチオーバーを実行します。

スケーラブルサービスの場合、AffinityOn フラグの設定は無視されます。スケーラブルリソースグループでアフィニティースイッチオーバーを実行できません。

FileSystemCheckCommand

SUNW.HAStoragePlus がファイルシステムをマウントする前に、マウントされていないファイルシステムに対して行うチェックを変更します。この拡張プロパティーには、すべてのマウントされていないファイルシステムで呼び出される、代替のコマンド文字列または実行可能ファイルを指定できます。

カテゴリ

任意

デフォルト

NULL

調整可能

すべての時間

SUNW.HAStoragePlus リソースがスケーラブルリソースグループ内に構成されている場合、マウントされていない個々のクラスタファイルシステムに対するファイルシステムチェックは省略されます。この拡張プロパティーを NULL に設定すると、Oracle Solaris Cluster は /usr/sbin/fsck -o p コマンドを実行することで UFS をチェックします。Oracle Solaris Cluster は /usr/sbin/fsck コマンドを実行して、ほかのファイルシステムをチェックします。

FileSystemCheckCommand 拡張プロパティーに別のコマンド文字列を設定すると、SUNW.HAStoragePlus は、ファイルシステムのマウントポイントを引数として、このコマンド文字列を呼び出します。この方法であらゆる任意の実行可能ファイルを指定できます。ゼロ以外の戻り値は、ファイルシステムチェック処理中に発生したエラーとして扱われます。このエラーが発生すると、START メソッドが失敗します。

ファイルシステムチェック処理が必要ない場合は、 FileSystemCheckCommand 拡張プロパティーに /bin/true を設定してください。

FileSystemMountPoints

有効なファイルシステムのマウントポイントのリストを指定します。グローバルまたはローカルのどちらのファイルシステムも指定できます。グローバルファイルシステムには、クラスタ

のすべてのノードからアクセス可能です。ローカルファイルシステムには、1 つのクラスタノードからアクセス可能です。SUNW.HAStoragePlus リソースによって管理されているローカルファイルシステムは、1 つのクラスタノードにマウントされます。これらのローカルファイルシステムでは、配下のデバイスは Oracle Solaris Cluster グローバルデバイスであることが必要です。

カテゴリ

任意

デフォルト

空のリスト

調整可能

すべての時間

これらのファイルシステムマウントポイントは、paths[,…] という形式で定義されます。

すべてのクラスタノードおよびすべての大域ゾーンにある /etc/vfstab には、各ファイルシステムのマウントポイントに対応する等価なエントリが存在するはずです。SUNW.HAStoragePlus リソースタイプは、非大域ゾーンにある /etc/vfstab をチェックしません。

ローカルファイルシステムを指定する SUNW.HAStoragePlus リソースは、アフィニティースイッチオーバーが有効になっているフェイルオーバーリソースグループ以外には所属できません。このため、これらのローカルファイルシステムはフェイルオーバーファイルシステムと呼ばれます。ローカルおよびグローバルファイルシステムのマウントポイントは同時に指定できます。

ファイルシステムのマウントポイントが FileSystemMountPoints 拡張プロパティーに指定され、/etc/vfstab エントリが次の条件を両方とも満たしている場合、このファイルシステムはローカルとみなされます。

1. グローバルではないマウントオプションが指定されている
2. エントリの「mount at boot」フィールドが「no」に設定されている。

Oracle Solaris ZFS ファイルシステムは、常にローカルファイルシステムです。/etc/vfstab に ZFS ファイルシステムを列挙しないでください。また、ZFS マウントポイントを FileSystemMountPoints プロパティーに含めないでください。

GlobalDevicePaths

有効な広域デバイスグループ名または広域デバイスパスのリストを指定します。これらのパスは、paths[,…] という書式で定義されます。

カテゴリ

任意

デフォルト

空のリスト

調整可能

無効時

IOOption

ファイルシステムをプローブするために実行する I/O のタイプを定義します。サポートされている値は `ReadOnly` と `ReadWrite` のみです。`ReadOnly` 値は、障害モニターが管理対象ファイルシステム (`FileSystemMountPoints` プロパティーで指定されているファイルシステムと、`Zpools` プロパティーで指定されている ZFS ストレージプールに属する ZFS ファイルシステムを含む) で読み取り専用 I/O を実行できることを示します。`ReadWrite` 値は、障害モニターが管理対象ファイルシステムで読み取りおよび書き込み I/O を実行できることを示します。

カテゴリ

任意

デフォルト

ReadOnly

調整可能

すべての時間

IOTimeout

I/O プローブのタイムアウト値を秒単位で定義します。

カテゴリ

任意

最小

10

最大

3600

デフォルト

300

調整可能

すべての時間

Monitor_retry_count

障害モニターで可能なプロセスモニター機能 (PMF) の再起動回数を制御します。

カテゴリ

任意

最小

1

デフォルト

4

調整可能

すべての時間

Monitor_retry_interval

障害モニターが再起動される時間間隔を分単位で定義します。

カテゴリ

任意

最小

2

デフォルト

2

調整可能

すべての時間

RebootOnFailure

障害が検証によって検出された場合、ローカルシステムをリブートするかどうかを指定します。TRUE に設定されている場合、リソースによって使用されるすべてのデバイスを、ディスクパスのモニタリングで直接または間接的にモニターする必要があります。

RebootOnFailure が TRUE に設定されており、GlobalDevicePaths、FileSystemMountPoints、または Zpools プロパティーで指定した各エントリで、使用可能なデバイスが少なくとも 1 つ検出された場合、ローカルシステムはリブートされます。ローカルシステムは、リソースがオンライン状態のグローバルクラスタノードまたはゾーンクラスタノードを参照します。

カテゴリ

任意

デフォルト

FALSE

調整可能

すべての時間

Zpools

少なくとも 1 つの ZFS が含まれている有効な ZFS ストレージプールのリストを指定します。これらの ZFS ストレージプールは、`paths[,…]` という書式で定義されます。ZFS ストレージプール内のすべてのファイルシステムは、一緒にマウントおよびマウント解除されます。

カテゴリ

任意

デフォルト

空のリスト

調整可能

すべての時間

`Zpools` 拡張プロパティーでは、ZFS ストレージプールを指定できます。ZFS ストレージプールを構成するデバイスは、`SUNW.HAStoragePlus` リソースが属するリソースグループのノードリスト内に構成されているすべてのノードからアクセスできる必要があります。ZFS ストレージプールを管理する `SUNW.HAStoragePlus` リソースは、フェイルオーバーリソースグループだけに属することができます。

ZFS ストレージプールを管理する `SUNW.HAStoragePlus` リソースがオンラインになると、ZFS ストレージプールがインポートされ、ZFS ストレージプールに含まれるすべてのファイルシステムがマウントされます。

リソースがノード上でオフラインになると、管理されている ZFS ストレージプールごとに、すべてのファイルシステムがマウント解除され、ZFS ストレージプールがエクスポートされます。

注記 - `SUNW.HAStoragePlus` は ZFS ボリューム上に作成されたファイルシステムをサポートしません。

ZpoolsSearchDir

`Zpools` のデバイスを検索する場所を指定します。`ZpoolsSearchDir` 拡張プロパティーは、`zpool` コマンドの `-d` オプションと類似しています。

カテゴリ

任意

デフォルト

`/dev/dsk`

調整可能

無効時

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core

1251 ページのrt_reg(4), Unresolved link to " attributes5"

指定のリソースグループが持つデータサービスリソースはSUNW.HAStoragePlus リソースに依存させてください。依存していない場合、データサービスと広域デバイスまたはファイルシステム間の同期を取ることはできません。オフライン再起動のリソース依存関係により、SUNW.HAStoragePlus リソースがほかのリソースの前に確実にオンラインになります。SUNW.HAStoragePlus リソースによって管理されているローカルファイルシステムは、リソースがオンラインにされた場合だけマウントされます。

UFS システムへのログオンを有効にしてください。

複数の SUNW.HAStoragePlus リソースを別々のリソースグループ内に構成し、同じデバイスグループを参照させ、AffinityOn フラグを TRUE に設定することは避けてください。冗長デバイススイッチオーバーが発生する場合があります。この結果、リソースとデバイスグループの位置が変更される場合もあります。

異なるリソースグループ内の複数の SUNW.HAStoragePlus リソースで ZFS ストレージプールを構成するのは避けてください。

障害モニターのエラー

障害モニターは、グローバルデバイス、ファイルシステム、ZFS ストレージプールなどの HAStoragePlus リソースによって管理されるエンティティーをモニターします。モニター対象のエンティティーのステータスは次のいずれかです。

- オンライン – 部分エラーや重大なエラーはありません。
- 縮退 – 部分エラーがあります。
- 障害発生 – 重大なエラーがあります。Resource Group Manager (RGM) はリソースの再起動と別のクラスタノードへのフェイルオーバーを試みます。

複数のエンティティーがモニター対象になっている場合、リソースのステータスはモニター対象のすべてのエンティティーの集約ステータスによって決定されます。

注記 - 障害モニターの実行中にモニター対象エンティティーの構成を変更すると、障害モニターが失敗して終了し、リソースが再起動されることがあります。モニター対象エンティティーの構成を変更する前に障害モニターを無効にし、あとで障害モニターを再度有効にしてください。構成の変更には、ZFS ストレージプールまたはプール内の ZFS ファイルシステム、または Solaris Volume Manager のディスクセットまたはボリュームの削除が含まれます。

`SUNW.HAStoragePlus` リソースは、マウント解除状態になっているどのクラスタファイルシステムでもマウントできます。

すべてのファイルシステムはオーバーレイモードでマウントされます。

ローカルファイルシステムは強制的にマウント解除されます。

すべてのデバイスサービスおよびファイルシステムが使用可能な状態になるまでの待ち時間は、`SUNW.HAStoragePlus` の `Prenet_start_timeout` プロパティーで指定されます。これは調節可能なプロパティーです。

名前

SUNW.Proxy_SMF_failover, Proxy_SMF_failover — フェイルオーバー SMF サービスのプロキシ作成のためのリソースタイプ

SUNW.Proxy_SMF_failover リソースタイプは、サービス管理機能 (SMF) サービスのフェイルオーバーのプロキシを表します。

SUNW.proxysmffailover リソースタイプに対して定義される標準プロパティーと拡張プロパティーについては、これ以降のサブセクションで説明します。これらのプロパティーを SUNW.Proxy_SMF_failover リソースタイプのインスタンスに設定するには、clresource コマンド (273 ページのclresource(1CL)) を使用します。

標準プロパティー

次のリソースプロパティーについては、1287 ページのr_properties(5) を参照してください。

Start_timeout
: デフォルト: 3600
: 最小: 60

Stop_timeout
: デフォルト: 3600
: 最小: 60

Init_timeout
: デフォルト: 3600
: 最小: 60

Boot_timeout
: デフォルト: 3600
: 最小: 60

Fini_timeout
: デフォルト: 3600
: 最小: 60

Validate_timeout
: デフォルト: 3600
: 最小: 60

`Failover_mode`

デフォルト: `SOFT`

調整可能: `Anytime`

`R_description`

デフォルト: `""`

調整可能: `Anytime`

`Retry_count`

デフォルト: `2`

最小: `0`

最大: `10`

調整可能: `Anytime`

`Retry_interval`

デフォルト: `300`

最大: `3600`

調整可能: `Anytime`

`Through_probe_interval`

デフォルト: `60`

調整可能: `Anytime`

拡張プロパティー

`Proxied_service_instances`

リソースによってプロキシされる SMF サービスに関する情報を含みます。値はプロキシされるすべての SMF サービスを含むファイルのパスです。ファイル内の各行は 1 つの SMF サービス専用で、`svc fmri` および対応するサービスマニフェストファイルのパスを指定します。たとえば、リソースが `restarter_svc_test_1:default` と `restarter_svc_test_2:default` の 2 つのサービスを管理する必要がある場合、このファイルには次の 2 行を含めるようにしてください。

```
<svc:/system/cluster/restarter_svc_test_1:default>,
  svc:/system/cluster/restarter_svc_test_1:default>,
  </var/svc/manifest/system/cluster/restarter_svc_test_1.xml>

<svc:/system/cluster/restarter_svc_test_2:default>,
  </var/svc/manifest/system/cluster/restarter_svc_test_2.xml>
```

注記 - 上記エントリはそれぞれ、本来 1 行になるべきものです。ここでは読みやすさを考慮して複数の行に分けています。

デフォルト: ""

調整可能: When disabled

例

この例では、SUNW.Proxy_SMF_failover リソースタイプの登録、アプリケーション用のリソースグループの作成、フェイルオーバーアプリケーションリソースの作成、リソースグループの管理、およびリソースを有効にする方法を示します。

リソースタイプを登録します。

```
# clresourcetype -f <path-to-rtrfile> SUNW.Proxy_SMF_failover
```

アプリケーション用の rg1 というリソースグループを作成します。

```
# clresourcegroup create rg1
```

myfailoverres というフェイルオーバーアプリケーションリソースを作成します。

```
# clresource create -t SUNW.Proxy_SMF_failover -g rg1 \
-x proxied_service_instances="/usr/local/app/svc myfailoverres"
```

/usr/local/app/svc はテキストファイルです。

リソースグループ rg1 の管理:

```
# clresourcegroup manage rg1
```

myfailoverres リソースを有効にします。

```
# clresource enable myfailoverres
```

次のコマンドを使用して、アプリケーションのステータスをチェックします。

```
# clresource status
```

735 ページのpmfadm(1M), 691 ページのscha_resource_get(1HA),
333 ページのclresourcetype(1CL), 273 ページのclresource(1CL),
305 ページのclresourcegroup(1CL), Unresolved link to " attributes5",
1287 ページのr_properties(5)

Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド "

名前

SUNW.Proxy_SMF_multimaster, Proxy_SMF_multimaster — マルチマスター SMF サービスのプロキシ作成のためのリソースタイプ

SUNW.Proxy_SMF_multimaster リソースタイプは、マルチマスターサービス管理機能 (SMF) サービスのプロキシを表します。

SUNW.proxysmfmultimaster リソースタイプに対して定義される標準プロパティーと拡張プロパティーについては、これ以降のサブセクションで説明します。これらのプロパティーを SUNW.Proxy_SMF_multimaster リソースタイプのインスタンスに設定するには、clresource コマンド (273 ページのclresource(1CL)) を使用します。

標準プロパティー

次のリソースプロパティーについては、1287 ページのr_properties(5) を参照してください。

Start_timeout
: デフォルト: 3600
: 最小: 60

Stop_timeout
: デフォルト: 3600
: 最小: 60

Init_timeout
: デフォルト: 3600
: 最小: 60

Boot_timeout
: デフォルト: 3600
: 最小: 60

Fini_timeout
: デフォルト: 3600
: 最小: 60

Validate_timeout
: デフォルト: 3600
: 最小: 60

`Failover_mode`

デフォルト: `SOFT`

調整可能: `Anytime`

`R_description`

デフォルト: `""`

調整可能: `Anytime`

`Retry_count`

デフォルト: `2`

最小: `0`

最大: `3`

調整可能: `Anytime`

`Retry_interval`

デフォルト: `300`

最大: `3600`

調整可能: `Anytime`

`Through_probe_interval`

デフォルト: `60`

調整可能: `Anytime`

拡張プロパティー

`Proxied_service_instances`

リソースによってプロキシされる SMF サービスに関する情報を含みます。値はプロキシされるすべての SMF サービスを含むファイルのパスです。ファイル内の各行は 1 つの SMF サービス専用で、`svc fmri` および対応するサービスマニフェストファイルのパスを指定します。たとえば、リソースが `restarter_svc_test_1:default` と `restarter_svc_test_2:default` の 2 つのサービスを管理する必要がある場合、このファイルには次の 2 行を含めるようにしてください。

```
<svc:/system/cluster/restarter_svc_test_1:default>,
  svc:/system/cluster/restarter_svc_test_1:default>,
  </var/svc/manifest/system/cluster/restarter_svc_test_1.xml>

<svc:/system/cluster/restarter_svc_test_2:default>,
  </var/svc/manifest/system/cluster/restarter_svc_test_2.xml>
```

注記 - 上記エントリはそれぞれ、本来 1 行になるべきものです。ここでは読みやすさを考慮して複数の行に分けています。

デフォルト: ""

調整可能: When disabled

例

この例では、SUNW.Proxy_SMF_multimaster リソースタイプを登録して、アプリケーション用のリソースグループを作成して、マルチマスターアプリケーションリソースを作成して、リソースグループを管理状態にして、リソースを有効にする方法を示します。

リソースタイプを登録します。

```
# clresourcetype -f <path-to-rtrfile> SUNW.Proxy_SMF_multimaster
```

アプリケーション用の rg1 というリソースグループを作成します。

```
# clresourcegroup create rg1
```

mymultimasterres というマルチマスターアプリケーションリソースを作成します。

```
# clresource create -t SUNW.Proxy_SMF_multimaster -g rg1 \
-x proxied_service_instances="/usr/local/app/svc" mymultimasterres
```

/usr/local/app/svc はテキストファイルです。

リソースグループ rg1 の管理:

```
# clresourcegroup manage rg1
```

mymultimasterres リソースを有効にします。

```
# clresource enable mymultimasterres
```

次のコマンドを使用して、アプリケーションのステータスをチェックします。

```
# clresource status
```

735 ページのpmfadm(1M), 691 ページのscha_resource_get(1HA),
333 ページのclresourcetype(1CL), 273 ページのclresource(1CL),
305 ページのclresourcegroup(1CL), Unresolved link to " attributes5",
1287 ページのr_properties(5)

Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド "

名前

SUNW.Proxy_SMF_scalable, Proxy_SMF_scalable — スケーラブル SMF サービスのプロキシ作成のためのリソースタイプ

SUNW.Proxy_SMF_scalable リソースタイプは、スケーラブルなサービス管理機能 (Service Management Facility、SMF) サービスのプロキシを表します。

SUNW.proxysmfscalable リソースタイプに対して定義される標準プロパティーと拡張プロパティーについて、これ以降のサブセクションで説明します。SUNW.Proxy_SMF_scalable リソースタイプのインスタンスに対してこれらのプロパティーを設定するには、clresource コマンドを使用します。

標準プロパティー

次のリソースプロパティーについては、1287 ページのr_properties(5) を参照してください。

Start_timeout
: デフォルト: 3600
: 最小: 60

Stop_timeout
: デフォルト: 3600
: 最小: 60

Init_timeout
: デフォルト: 3600
: 最小: 60

Boot_timeout
: デフォルト: 3600
: 最小: 60

Fini_timeout
: デフォルト: 3600
: 最小: 60

Validate_timeout
: デフォルト: 3600
: 最小: 60

`Failover_mode`

デフォルト: `SOFT`

調整可能: 常時

`R_description`

デフォルト: `""`

調整可能: 常時

`Retry_count`

デフォルト: `2`

最小: `0`

最大: `3`

調整可能: 常時

`Retry_interval`

デフォルト: `300`

最大: `3600`

調整可能: 常時

`Through_probe_interval`

デフォルト: `60`

調整可能: 常時

拡張プロパティー

`Proxied_service_instances`

リソースによってプロキシされる SMF サービスに関する情報を含みます。値はプロキシされるすべての SMF サービスを含むファイルのパスです。ファイル内の各行は 1 つの SMF サービス専用で、`svc fmri` および対応するサービスマニフェストファイルのパスを指定します。たとえば、リソースが `restarter_svc_test_1:default` と `restarter_svc_test_2:default` の 2 つのサービスを管理する必要がある場合、このファイルには次の 2 行を含めるようにしてください。

```
<svc:/system/cluster/restarter_svc_test_1:default>,
  svc:/system/cluster/restarter_svc_test_1:default>,
  </var/svc/manifest/system/cluster/restarter_svc_test_1.xml>

<svc:/system/cluster/restarter_svc_test_2:default>,
  </var/svc/manifest/system/cluster/restarter_svc_test_2.xml>
```

注記 - 上記エントリはそれぞれ、本来 1 行になるべきものです。ここでは読みやすさを考慮して複数の行に分けています。

デフォルト: ""

調整可能: When disabled

例

この例では、SUNW.Proxy_SMF_scalable リソースタイプを登録して、アプリケーション用のリソースグループを作成して、負荷分散アプリケーションリソースを作成して、リソースグループを管理状態にして、そのリソースをすべて有効にして、そのリソースをオンラインにする方法を示します。

リソースタイプを登録します。

```
# clresourcetype -f <path-to-rtrfile> SUNW.Proxy_SMF_scalable
```

アプリケーション用の rg1 というリソースグループを作成します。

```
# clresourcegroup create rg1
```

myloadbalancedres という負荷分散アプリケーションリソースを作成します。

```
# clresource create -t SUNW.Proxy_SMF_scalable -g rg1 \
-x proxied_service_instances="/usr/local/app/svc myloadbalancedres"
```

/usr/local/app/svc はテキストファイルです。

リソースグループ rg1 の管理:

```
# clresourcegroup manage rg1
```

myloadbalancedres リソースを有効にします。

```
# clresource enable myloadbalancedres
```

次のコマンドを使用して、アプリケーションのステータスをチェックします。

```
# clresource status
```

735 ページのpmfadm(1M), 691 ページのscha_resource_get(1HA),
333 ページのclresourcetype(1CL), 273 ページのclresource(1CL),
305 ページのclresourcegroup(1CL), Unresolved link to " attributes5",
1287 ページのr_properties(5)

Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド "

名前

SUNW.rac_framework, rac_framework — Oracle Solaris Cluster Support for Oracle Real Application Clusters (Oracle RAC) を有効にするフレームワークのためのリソースタイプ実装

`SUNW.rac_framework` リソースタイプは、Oracle Solaris Cluster Support for Oracle RAC を有効にするフレームワークを表します。このリソースタイプでは、このフレームワークのステータスをモニターできます。

`SUNW.rac_framework` リソースタイプは、シングルインスタンスリソースタイプです。したがって、クラスタに作成可能なリソースは 1 個だけです。

このリソースタイプを登録し、リソースタイプのインスタンスを作成するには、次のいずれかの方法を使用します。

- Oracle Solaris Cluster Manager
- Oracle Solaris Cluster Support for Oracle Real Application Clusters を構成するためのオプションを指定する 487 ページの`clsetup`(1CL) ユーティリティー
- 次の一連の Oracle Solaris Cluster 保守コマンド。

  1. このリソースタイプを登録するには、333 ページの`clresourcetype`(1CL) コマンドを使用します。
  2. このリソースタイプのインスタンスを作成するには、273 ページの`clresource`(1CL) コマンドを使用します。

この種類のリソースをノード上でオフラインにする場合、オンライン状態からオフライン状態へ切り替えるには、一定の時間がかかります。オフライン状態への切り替え中も、リソースは引き続き再構成処理に使用されています。ただし、リソースをノード上でオフラインにした場合、再度オンラインに戻すまで、リソースのプロパティーの変更は適用されません。Oracle Solaris Cluster は、この種類のリソースが無効になっている場合、このことを知らせる警告メッセージを表示します。

この種類のリソースを含むリソースグループを非管理状態に切り替えるには、一定の時間がかかります。非管理状態への切り替え中も、Oracle RAC フレームワークは引き続きフレームワーク再構成処理に使用されています。ただし、リソースグループを非管理状態に切り替えた場合、ノード上のリソースプロパティーの変更は有効になりません。Oracle RAC フレームワークを停止するには、ノードをリブートします。

SUNW.rac_framework リソースタイプの拡張プロパティーは次のとおりです。

`reservation_timeout`

整数型で、最小値は 100、最大値は 99999、デフォルト値は 325 です。このプロパティーは、Oracle Solaris Cluster Support for Oracle RAC の再構成の予約ステップに対するタイムアウト値を秒単位で指定します。このプロパティーはいつでも変更できます。

例 405 `rac_framework` リソースの作成

この例では、`SUNW.rac_framework` リソースタイプを登録し、`rac_framework-rs` という名前の `SUNW.rac_framework` リソースタイプのインスタンスを作成します。この例では、`rac-framework-rg` という名前のリソースグループがすでに作成されているものとします。

```
phys-host-sc1# clresourcetype register SUNW.rac_framework

phys-host-sc1# clresource create -g rac-framework-rg \
-t SUNW.rac_framework rac_framework-rs
```

例 406 `rac_framework` リソースのプロパティーの変更

この例では、Oracle Solaris Cluster Support for Oracle RAC の再構成の予約ステップのタイムアウトを 350 秒に設定します。この例では、`rac_framework-rs` という名前のリソースタイプ `SUNW.rac_framework` がすでに作成されているものとします。

```
phys-host-sc1# clresource set
\-p reservation_timeout=350 rac_framework-rs
```

注記 - ボリュームマネージャーリソースで使用される `SUNW.vucmm_framework` リソースを作成する例については、1433 ページの`SUNW.vucmm_framework`(5) を参照してください。

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/library/ucmm

273 ページの`clresource`(1CL), 333 ページの`clresourcetype`(1CL), 487 ページの`clsetup`(1CL), Unresolved link to " attributes5", 1433 ページの`SUNW.vucmm_framework`(5)

Unresolved link to " Oracle Solaris Cluster Data Service for Oracle Real Application Clusters ガイド "

名前

SUNW.ScalDeviceGroup, ScalDeviceGroup — スケーラブルデバイスグループのリソースタイプ実装

SUNW.ScalDeviceGroup リソースタイプは、スケーラブルデバイスグループを表します。このリソースタイプのインスタンスは、次に示す種類のデバイスグループの 1 つを表します:

- Solaris Volume Manager for Sun Cluster の複数所有者ディスクセット

SUNW.ScalDeviceGroup リソースタイプは、スケーラブルリソースタイプです。このリソースタイプのインスタンスは、リソースを含むリソースグループのノードリスト内にある各ノードでオンラインです。

このリソースタイプを登録し、リソースタイプのインスタンスを作成するには、次のいずれかの方法を使用します。

- Oracle Solaris Cluster Manager
- Oracle Real Application Clusters 用 Oracle Solaris Cluster サポートを構成するためのオプションを指定する clsetup ユーティリティー
- 次の一連の Oracle Solaris Cluster 保守コマンド。

  1. このリソースタイプを登録するには、clresourcetype コマンドを使用します。
  2. このリソースタイプのインスタンスを作成するには、clresource コマンドを使用します。

SUNW.ScalDeviceGroup リソースタイプに対して定義される標準プロパティーと拡張プロパティーについて、これ以降のサブセクションで説明します。

標準プロパティー

すべての標準リソースプロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

標準のリソースタイププロパティーは、次のようにこのリソースタイプに対して無効にされます。

Monitor_start_timeout

最小
: 10

デフォルト
: 300

Monitor_stop_timeout

最小
10

デフォルト
300

Postnet_stop_timeout

最小
60

デフォルト
300

Prenet_start_timeout

最小
60

デフォルト
300

Start_timeout

最小
60

デフォルト
300

Stop_timeout

最小
60

デフォルト
300

Thorough_probe_interval

デフォルト
300

Update_timeout

最小
60

デフォルト
300

Validate_timeout

最小
60

デフォルト
300

拡張プロパティー

このリソースタイプには、次のような拡張プロパティーが設定されます。

Debug_level

このプロパティーは、このタイプのリソースからのデバッグメッセージが記録されるレベルを指定します。デバッグレベルを高くすると、より多くのデバッグメッセージがログファイルに書き込まれます。

データ型
整数

デフォルト
0

範囲
0–10

調整可能
すべての時間

DiskGroupName

このプロパティーは、リソースが表すデバイスグループの名前を指定します。このプロパティーには次に示す項目の 1 つを設定してください。

- Solaris ボリュームマネージャーの複数所有者ディスクセットの名前。この名前は、ディスクセットの作成に使用した Unresolved link to " metaset1M" コマンドで指定したものです。

指定するデバイスグループの要件は次のとおりです。

- デバイスグループは、既存の有効な複数所有者ディスクセットまたは共有ディスクグループとします。
- デバイスグループは、リソースをマスターできるすべてのノードでホストされるようにします。
- デバイスグループは、スケーラブルデバイスグループリソースをマスターできるすべてのノードからアクセス可能にします。
- デバイスグループには、1 つ以上のボリュームを含めます。

データ型
: 文字列

デフォルト
: デフォルトは定義されていません。

範囲
: 適用不可

調整可能
: 無効時

IOTimeout

このプロパティーは、I/O 検証のタイムアウト値 (秒) を指定します。

デフォルト
: 30

範囲
: 1–1800

調整可能
: すべての時間

LogicalDeviceList

このプロパティーは、リソースの障害モニターでモニターする論理ボリュームのコンマ区切りのリストを指定します。このプロパティーは省略可能です。このプロパティーの値を指定しない場合、デバイスグループ内のすべての論理ボリュームがモニターされます。

デバイスグループのステータスは、モニターされる個々の論理ボリュームのステータスから導き出されます。モニターされているすべての論理ボリュームが健全な場合、そのデバイスグループは健全です。モニターされている論理ボリュームが 1 つでも障害状態の場合、そのデバイスグループは障害状態となります。

個々の論理ボリュームのステータスを取得するには、そのボリュームのボリュームマネージャーに照会します。Solaris ボリュームマネージャーのボリュームのステータスを照会から判

断できない場合、障害モニターはファイルの入出力 (I/O) 操作を実行して、そのステータスを判断します。

デバイスグループが障害状態であることがわかると、グループを表すリソースのモニタリングが停止され、リソースが無効な状態に置かれます。

注記 - ミラー化ディスクの場合、1 つのサブミラーが障害状態でも、デバイスグループは健全であるとみなされます。

注記 - Solaris Volume Manager for Sun Cluster を使用している場合、ソフトパーティションのモニターを指定するには、対応する上または下のレベルのメタデバイスも指定する必要があります。

指定する各論理ボリュームの要件は次のとおりです。

- 論理ボリュームが存在する。
- 論理ボリュームが、`diskgroupname` プロパティーが指定するデバイスグループに含まれている。
- 論理ボリュームが、スケーラブルデバイスグループリソースをマスターできるすべてのノードからアクセス可能である。

データ型

文字列配列

デフォルト

""

範囲

適用不可

調整可能

すべての時間

Monitor_retry_count

このプロパティーは、障害モニターに許可される、プロセスモニター機能 (PMF) による最大再起動回数を指定します。

データ型

整数

デフォルト

4

範囲

範囲は定義されていません。

調整可能

すべての時間

Monitor_retry_interval

このプロパティーは、PMF が障害モニターの再起動をカウントする期間 (分単位) を指定します。

データ型

整数

デフォルト

2

範囲

範囲は定義されていません。

調整可能

すべての時間

RebootOnFailure

このプロパティーは、検証によって障害が検出された場合に、ローカルシステムをリブートするかどうかを指定します。このプロパティーが `TRUE` に設定されている場合、リソースによって使用されるすべてのデバイスを、ディスクパスのモニタリングで直接または間接的にモニターする必要があります。

`RebootOnFailure` が `TRUE` に設定されており、`GlobalDevicePaths`、`FileSystemMountPoints`、または `Zpools` プロパティーで指定した各エントリで、使用可能なデバイスが少なくとも 1 つ検出された場合、ローカルシステムはリブートされます。ローカルシステムは、リソースがオンライン状態のグローバルクラスタノードまたはゾーンクラスタノードを参照します。

デフォルト

`FALSE`

調整可能

すべての時間

例 407 `SUNW.vucmm_svm` リソースタイプを使用して、`ScalDeviceGroup` リソースを作成

この例では、`SUNW.vucmm_svm` リソースタイプを使用した `datadg` という名前の Solaris Volume Manager for Sun Cluster 複数所有者ディスクセットを表す、`ScalDeviceGroup` リソースを作成する方法を示します。リソースの名前は `scaldatadg-rs` です。この例は、次に示す Oracle Solaris Cluster オブジェクトが存在することを前提とします。

- scaldatadg-rg という名前のスケーラブルリソースグループ。
- vucmm-svm-rs という名前の SUNW.vucmm_svm リソースタイプのインスタンス。

```
# clresourcetype register SUNW.ScalDeviceGroup
# clresource create -t SUNW.ScalDeviceGroup \
-g scaldatadg-rg \
-p Resource_dependencies=vucmm-svm-rs \
-p DiskGroupName=datadg \
scaldatadg-rs
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core

273 ページのclresource(1CL), 333 ページのclresourcetype(1CL), 487 ページのclsetup(1CL), Unresolved link to " metaset1M", Unresolved link to " attributes5", 1287 ページのr_properties(5), 1435 ページのSUNW.vucmm_svm(5)

Unresolved link to " Oracle Solaris Cluster Data Service for Oracle Real Application Clusters ガイド "

名前

SUNW.ScalMountPoint, ScalMountPoint — スケーラブルファイルシステムのマウントポイントのリソースタイプ

SUNW.ScalMountPoint リソースタイプは、スケーラブルファイルシステムマウントポイントを表します。このリソースタイプのインスタンスは、次に示す種類のファイルシステムの 1 つのマウントポイントを表します。

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

  NAS デバイスおよびファイルシステムは、Oracle Solaris Cluster で使用できるように構成済みである必要があります。詳細は、Unresolved link to " Oracle Solaris Cluster With Network-Attached Storage Device Manual "を参照してください。

SUNW.ScalMountPoint リソースタイプは、スケーラブルリソースタイプです。このリソースタイプのインスタンスは、リソースを含むリソースグループのノードリスト内にある各ノードでオンラインです。

このリソースタイプを登録し、リソースタイプのインスタンスを作成するには、次のいずれかの方法を使用します。

- Oracle Solaris Cluster Manager
- Oracle Real Application Clusters 用 Oracle Solaris Cluster サポートを構成するためのオプションを指定する clsetup ユーティリティー
- 次の一連の Oracle Solaris Cluster 保守コマンド。

  1. このリソースタイプを登録するには、clresourcetype コマンドを使用します。
  2. このリソースタイプのインスタンスを作成するには、clresource コマンドを使用します。

SUNW.ScalMountPoint リソースタイプに対して定義される標準プロパティーと拡張プロパティーについて、これ以降のサブセクションで説明します。

標準プロパティー

すべての標準リソースプロパティーについては、1287 ページのr_properties(5) のマニュアルページを参照してください。

標準のリソースタイププロパティーは、次のようにこのリソースタイプに対して無効にされます。

Monitor_start_timeout

最小

10

デフォルト

300

Monitor_stop_timeout

最小

10

デフォルト

300

Postnet_stop_timeout

最小

60

デフォルト

300

Prenet_start_timeout

最小

60

デフォルト

300

Start_timeout

最小

60

デフォルト

300

Stop_timeout

最小

60

デフォルト

300

Thorough_probe_interval

デフォルト

300

Update_timeout

最小

60

デフォルト

300

Validate_timeout

最小

60

デフォルト

300

拡張プロパティー

このリソースタイプには、次のような拡張プロパティーが設定されます。

Debug_level

このプロパティーは、ファイルシステムマウントポイントのリソースからのデバッグメッセージが記録されるレベルを指定します。デバッグレベルを高くすると、より多くのデバッグメッセージがログファイルに書き込まれます。

データ型

整数

デフォルト

0

範囲

0–10

調整可能

すべての時間

FileSystemType

このプロパティーは、リソースが表すマウントポイントを持つファイルシステムの種類を指定します。このプロパティーを指定してください。このプロパティーには次に示す値の 1 つを設定します。

`nas`

ファイルシステムが NAS デバイス上のファイルシステムであることを指定します。

`s-qfs`

ファイルシステムが Sun QFS 共有ファイルシステムであることを指定します。

データ型

文字列

デフォルト

デフォルトは定義されていません。

範囲

適用不可

調整可能

無効時

IOTimeout

このプロパティーは、障害モニターが入出力 (I/O) 検証に使用するタイムアウト値を秒単位で指定します。障害モニターは、マウントされたファイルシステムが使用可能かどうかを判断するため、ファイルシステム上でテストファイルを開いたり、読み込んだり、書き込んだりするなどの I/O 操作を実行します。I/O 操作がタイムアウト期間内に完了しない場合、障害モニターはエラーを報告します。

データ型

整数

デフォルト

300

範囲

5–1800

調整可能

すべての時間

Monitor_retry_count

このプロパティーは、障害モニターに許可される、プロセスモニター機能 (PMF) による最大再起動回数を指定します。

データ型

整数

デフォルト

4

範囲

範囲は定義されていません。

調整可能

すべての時間

Monitor_retry_interval

このプロパティーは、PMF が障害モニターの再起動をカウントする期間 (分単位) を指定します。

データ型

整数

デフォルト

2

範囲

範囲は定義されていません。

調整可能

すべての時間

MountOptions

このプロパティーは、リソースが表すファイルシステムがマウントされるときに使用されるマウントオプションのコンマ区切りリストを指定します。このプロパティーは省略可能です。このプロパティーの値を指定しないと、マウントオプションは、ファイルシステムのデフォルトの表から取得されます。

- Sun QFS 共有ファイルシステムの場合、これらのオプションは `/etc/opt/SUNWsamfs/samfs.cmd` ファイルから取得されます。
- NAS デバイス上のファイルシステムの場合、これらのオプションは `/etc/vfstab` ファイルから取得されます。

このプロパティーを通して指定するマウントオプションは、ファイルシステムのデフォルトの表内のマウントオプションをオーバーライドします。

データ型

文字列

デフォルト

""

範囲

適用不可

調整可能

無効時

MountPointDir

このプロパティーは、リソースが表すファイルシステムのマウントポイントを指定します。マウントポイントは、ファイルシステムのマウント時にファイルシステムがファイルシステム階層に接続されるディレクトリへのフルパスです。このプロパティーを指定してください。

指定するディレクトリはすでに存在している必要があります。

データ型

文字列

デフォルト

デフォルトは定義されていません。

範囲

適用不可

調整可能

無効時

RebootOnFailure

このプロパティーは、検証によって障害が検出された場合に、ローカルシステムをリブートするかどうかを指定します。このプロパティーが `TRUE` に設定されている場合、リソースによって使用されるすべてのデバイスを、ディスクパスのモニタリングで直接または間接的にモニターする必要があります。

`RebootOnFailure` が `TRUE` に設定されており、`GlobalDevicePaths`、`FileSystemMountPoints`、または `Zpools` プロパティーで指定した各エントリで、使用可能なデバイスが少なくとも 1 つ検出された場合、ローカルシステムはリブートされます。ローカルシステムは、リソースがオンライン状態のグローバルクラスタノードまたはゾーンクラスタノードを参照します。

デフォルト

`FALSE`

調整可能

すべての時間

TargetFileSystem

このプロパティーは、`MountPointDir` 拡張プロパティーが指定するマウントポイントでマウントされるファイルシステムを指定します。このプロパティーを指定してください。ファイルシステムの種類は、`FileSystemType` プロパティーが指定する種類に一致させる必要があります。このプロパティーの書式は、ファイルシステムの種類によって次のように異なります。

- Sun QFS 共有ファイルシステムの場合、このプロパティーにはファイルシステムの作成時にファイルシステムに割り当てられた名前を設定します。ファイルシステムは、正しく構成してください。詳細は、使用している Sun QFS 共有ファイルシステムのドキュメントを参照してください。
- NAS デバイスのファイルシステムの場合は、このプロパティーに *nas-device*:*path* を設定します。上記書式の各項目の意味は次のとおりです。

  nas-device

  ファイルシステムをエクスポートしている NAS デバイスの名前を指定します。この名前は、状況に応じてドメインで修飾できます。

  path

  NAS デバイスがエクスポートしているファイルシステムへのフルパスを指定します。

  NAS デバイスおよびファイルシステムは、Oracle Solaris Cluster で使用できるように構成済みである必要があります。詳細は、Unresolved link to " Oracle Solaris Cluster With Network-Attached Storage Device Manual "を参照してください。

データ型
: 文字列

デフォルト
: デフォルトは定義されていません。

範囲
: 適用不可

調整可能
: 無効時

例 408 `ScalMountPoint` リソースの作成

この例では、Solaris Volume Manager for Sun Cluster と一緒に使用される Sun QFS 共有ファイルシステムのマウントポイントを表す `ScalMountPoint` リソースを作成する方法を示します。リソースの名前は `scal-db_qfs-Data-rs` です。ファイルシステムの特徴は次のとおりです。

- ファイルシステムのマウントポイントは `/db_qfs/Data` です。
- マウントされるファイルシステムは `Data` です。

- マウントオプションは、ファイルシステムのデフォルトの表、`/etc/opt/SUNWsamfs/samfs.cmd` ファイルから取得されます。

この例は、次に示す Oracle Solaris Cluster オブジェクトが存在することを前提とします。

- `scaldatadg-rg` という名前のスケーラブルリソースグループ。
- `qfs-db_qfs-Data-rs` という名前の `SUNW.qfs` リソースタイプのインスタンス。
- `scaldatadg-rs` という名前の `SUNW.ScalDeviceGroup` リソースタイプのインスタンス。

```
# clresourcetype register SUNW.ScalMountPoint
# clresource create -t SUNW.ScalMountPoint \
-g scaldatadg-rg \
-p Resource_dependencies=qfs-db_qfs-Data-rs,scaldatadg-rs \
-p MountPointDir=/db_qfs/Data \
-p FileSystemType=s-qfs \
-p TargetFileSystem=Data \
scal-db_qfs-Data-rs
```

次の属性の説明は、Unresolved link to " attributes5" を参照してください:

属性タイプ	属性値
使用条件	ha-cluster/system/core

273 ページの`clresource`(1CL), 333 ページの`clresourcetype`(1CL), 487 ページの`clsetup`(1CL), Unresolved link to " vfstab4", Unresolved link to " attributes5", 1287 ページの`r_properties`(5), 1415 ページの`SUNWct.ScalDeviceGroup`(5), 1433 ページの`SUNW.vucmm_framework`(5)

Unresolved link to " Oracle Solaris Cluster Data Service for Oracle Real Application Clusters ガイド "、Unresolved link to " Oracle Solaris Cluster With Network-Attached Storage Device Manual "

名前

SUNW.SCTelemetry, SCTelemetry — システム資源使用状況に関するデータを収集するためのリソースタイプ

SUNW.SCTelemetry は、システム資源の使用状況に関するデータの収集を可能にするリソースタイプです。SUNW.SCTelemetry は、7 日間にわたり、Java DB データベース内のシステム資源使用状況データを格納します。タイプが SUNW.SCTelemetry のリソースは、タイプが SUNW.derby のリソースに依存します。詳細は、1371 ページのSUNW.derby(5) のマニュアルページを参照してください。

SUNW.SCTelemetry リソースタイプに関連付けられている拡張プロパティーは、次のとおりです。

Extended_accounting_cleanup(boolean)

拡張されているアカウンティングログファイルをクリーンアップするかどうか、つまり履歴データを削除するかどうかを指定します。Extended_accounting_cleanup の取り得る値は、TRUE と FALSE です。

カテゴリ	任意
デフォルト	TRUE
調整可能	すべての時間

Monitor_retry_count(integer)

障害モニターの再起動を制御します。このプロパティーには、障害モニターがプロセスモニター機能により再起動される回数を示しています。このプロパティーは、735 ページのpmfadm(1M) コマンドに渡される -n オプションに相当します。Resource Group Manager (RGM) は、指定された時間窓内の再起動回数をカウントします。詳細については、Monitor_retry_interval プロパティーを参照してください。なお、Monitor_retry_count は、タイプが SUNW.SCTelemetry のリソースではなく、障害モニター自体の再起動回数を指します。

カテゴリ	任意
デフォルト	4
調整可能	すべての時間

Monitor_retry_interval(integer)

RGM が障害モニターで問題が発見された回数をカウントする時間を分単位で指定します。このプロパティーは、735 ページのpmfadm(1M) コマンドに渡される -t オプションに相

当します。障害モニターで問題が発見された回数が Monitor_retry_count プロパティーの値を超過すると、プロセスモニター機能は障害モニターを再起動しません。

カテゴリ	任意
デフォルト	2 分
調整可能	すべての時間

Probe_timeout(integer)

検証機能のタイムアウト値を秒単位で指定します。

カテゴリ	任意
デフォルト	60 秒
最小	2 秒
調整可能	すべての時間

Sampling_interval(integer)

モニタリングデータの収集頻度を指定します。Telemetry_sampling_interval プロパティーには 30 と 3600 の間の値を指定する必要があります。

カテゴリ	必須
デフォルト	60
最小	30 秒
最大	3600 秒
調整可能	すべての時間

735 ページのpmfadm(1M), 1371 ページのSUNW.derby(5)

名前

SUNW.vucmm_framework, vucmm_framework — Oracle Solaris Cluster ボリュームマネージャー再構成フレームワーク用のリソースタイプ実装

SUNW.vucmm_framework リソースタイプは、Oracle Solaris Cluster 構成において、異なるクラスタボリュームマネージャーを有効にするフレームワークを表します。このリソースタイプでは、このフレームワークのステータスをモニターできます。

SUNW.vucmm_framework リソースタイプは、シングルインスタンスリソースタイプです。クラスタ内にはこのタイプのリソースは 1 つだけ作成できます。

このリソースタイプを登録し、リソースタイプのインスタンスを作成するには、次のいずれかの方法を使用します。

- このリソースタイプを登録するには、333 ページのclresourcetype(1CL) コマンドを使用します。
- このリソースタイプのインスタンスを作成するには、273 ページのclresource(1CL) コマンドを使用します。

この種類のリソースをノード上でオフラインにする場合、オンライン状態からオフライン状態へ切り替えるには、一定の時間がかかります。オフライン状態への切り替え中も、リソースは引き続き再構成処理に使用されています。ただし、リソースをノード上でオフラインにした場合、再度オンラインに戻すまで、リソースのプロパティーの変更は適用されません。Oracle Solaris Cluster ソフトウェアは、この種類のリソースが無効になっている場合、このことを知らせる警告メッセージを表示します。

この種類のリソースを含むリソースグループを非管理状態に切り替えるには、一定の時間がかかります。非管理状態への切り替え中も、フレームワークは引き続きフレームワーク再構成処理に使用されています。ただし、リソースグループを非管理状態に切り替えた場合、ノード上のリソースプロパティーの変更は有効になりません。フレームワークを停止するには、ノードをリブートする必要があります。

SUNW.vucmm_framework リソースタイプの拡張プロパティーは次のとおりです。

reservation_timeout

型の整数は、最小 100、最大 99999、デフォルト値を 325 で指定します。プロパティーは枠組みの再構成の予約ステップに対して秒単位でタイムアウトを指定します。このプロパティーはいつでも変更できます。

例 409 SUNW.vucmm_framework リソースの作成

この例では、SUNW.vucmm_framework リソースタイプを登録し、vucmm_framework-rs という名前の SUNW.vucmm_framework リソースタイプのインスタンスを作成します。この例では、vucmm_framework-rg という名前のリソースグループがすでに作成されているものとします。

```
phys-host-sc1# clresourcetype register SUNW.vucmm_framework
phys-host-sc1# clresource create -g vucmm_framework-rg \
-t SUNW.vucmm_framework vucmm_framework-rs
```

例 410 vucmm_framework リソースのプロパティーの変更

この例では、枠組みの再構成の予約ステップタイムアウトを 350 秒に設定します。この例では、vucmm_framework-rs という名前のリソースタイプ SUNW.vucmm_framework がすでに作成されているものとします。

```
phys-host-sc1# clresource set -p reservation_timeout=350 vucmm_framework-rs
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core

273 ページのclresource(1CL), 333 ページのclresourcetype(1CL), 487 ページのclsetup(1CL), 1435 ページのSUNW.vucmm_svm(5), Unresolved link to " attributes5"

Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド "

名前

SUNW.vucmm_svm, vucmm_svm — ボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster コンポーネントのリソースタイプ実装を表します

SUNW.vucmm_svm リソースタイプは、Oracle Solaris Cluster ボリュームマネージャー再構成フレームワーク の Solaris Volume Manager for Sun Cluster コンポーネントを表します。

SUNW.vucmm_svm リソースタイプのインスタンスは、Solaris Volume Manager for Sun Cluster コンポーネントの構成パラメータを保持しています。このタイプのインスタンスは、Solaris ボリュームマネージャー for Sun Cluster コンポーネントの再構成のステータスも表示します。

SUNW.vucmm_svm リソースタイプはシングルインスタンスのリソースタイプです。したがって、クラスタに作成可能なリソースは 1 個だけです。

このリソースタイプを登録し、リソースタイプのインスタンスを作成するには、次のいずれかの方法を使用します。

- このリソースタイプを登録するには、333 ページのclresourcetype(1CL) コマンドを使用します。
- このリソースタイプのインスタンスを作成するには、273 ページのclresource(1CL) コマンドを使用します。

SUNW.vucmm_svm リソースは、SUNW.vucmm_framework リソースタイプを使用するリソースグループにおいてのみ有効です。SUNW.rac_framework リソースグループではこのリソースタイプを使用しないでください。

この種類のリソースをノード上でオフラインにする場合、オンライン状態からオフライン状態へ切り替えるには、一定の時間がかかります。オフライン状態への切り替え中も、リソースは引き続き再構成処理に使用されています。ただし、リソースをノード上でオフラインにした場合、再度オンラインに戻すまで、リソースのプロパティーの変更は適用されません。Oracle Solaris Cluster ソフトウェアは、この種類のリソースが無効になっている場合、このことを知らせる警告メッセージを表示します。

この種類のリソースを含むリソースグループを非管理状態に切り替えるには、一定の時間がかかります。非管理状態への切り替え中も、ボリュームマネージャ再構成枠組みは引き続き枠組み再構成処理に使用されています。ただし、リソースグループを非管理状態に切り替えた場合、

ノード上のリソースプロパティーの変更は有効になりません。ボリュームマネージャ再構成枠組みを停止するには、ノードをリブートします。

SUNW.vucmm_svm リソースタイプの拡張プロパティーは次のとおりです。

debug_level

整数型で、最小値は 0、最大値は 10、デフォルトは 1 です。このプロパティーはボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster モジュールのデバッグレベルを指定します。デバッグレベルが上がると、再構成中にログファイルに書き込まれるメッセージが増えます。このプロパティーはいつでも変更できます。

svm_abort_step_timeout

整数型で、最小値は 30、最大値は 99999、デフォルトは 120 です。このプロパティーでは、ボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster モジュールの再構成の中止ステップに対するタイムアウト値を秒単位で設定します。このプロパティーはいつでも変更できます。

svm_return_step_timeout

整数型で、最小値は 30、最大値は 99999、デフォルトは 120 です。このプロパティーでは、ボリュームマネージャ再構成フレームワークの Solaris Volume Manager for Sun Cluster モジュールの再構成の戻りステップに対するタイムアウト値を秒単位で設定します。このプロパティーはいつでも変更できます。

svm_start_step_timeout

整数型で、最小値は 30、最大値は 99999、デフォルトは 120 です。このプロパティーでは、ボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster モジュールの再構成の開始ステップに対するタイムアウト値を秒単位で設定します。このプロパティーはいつでも変更できます。

svm_step1_timeout

整数型で、最小値は 30、最大値は 99999、デフォルトは 120 です。このプロパティーでは、ボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster モジュールの再構成ステップ 1 に対するタイムアウト値を秒単位で設定します。このプロパティーはいつでも変更できます。

svm_step2_timeout

整数型で、最小値は 30、最大値は 99999、デフォルトは 120 です。このプロパティーでは、ボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster モジュールの再構成ステップ 2 に対するタイムアウト値を秒単位で設定します。このプロパティーはいつでも変更できます。

svm_step3_timeout

整数型で、最小値は 30、最大値は 99999、デフォルトは 120 です。このプロパティーでは、ボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster

モジュールの再構成ステップ 3 に対するタイムアウト値を秒単位で設定します。このプロパティーはいつでも変更できます。

`svm_step4_timeout`

整数型で、最小値は 100、最大値は 99999、デフォルトは 120 です。このプロパティーでは、ボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster モジュールの再構成ステップ 4 に対するタイムアウト値を秒単位で設定します。このプロパティーはいつでも変更できます。

`svm_stop_step_timeout`

整数型で、最小値は 30、最大値は 99999、デフォルトは 120 です。このプロパティーでは、ボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster モジュールの再構成の停止ステップに対するタイムアウト値を秒単位で設定します。このプロパティーはいつでも変更できます。

例 411 `vucmm_svm` リソースの作成

この例では、`SUNW.vucmm_svm` リソースタイプを登録し、`vucmm_svm-rs` という名前の `SUNW.vucmm_svm` リソースタイプのインスタンスを作成します。この例では、次に示す Oracle Solaris Cluster オブジェクトが作成済みであることを前提とします。

- 名前が `vucmm_framework-rg` のリソースグループ
- `vucmm_framework-rs` という名前のタイプ `SUNW.vucmm_framework` のリソース

```
phys-schost-1# clresourcetype register SUNW.vucmm_svm
phys-schost-1# clresource create -g vucmm_framework-rg \
-t SUNW.vucmm_svm \
-p resource_dependencies=vucmm_framework-rs vucmm_svm-rs
```

例 412 `vucmm_svm` リソースのプロパティーの変更

この例では、ボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster コンポーネントの再構成のステップ 4 に対するタイムアウト値を 300 秒に設定します。この例では、`vucmm_svm-rs` という名前の `SUNW.vucmm_svm` リソースタイプのインスタンスがすでに作成されているものとします。

```
phys-schost-1# clresource set \
-p svm_step4_timeout=300 vucmm_svm-rs
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core

273 ページの clresource(1CL), 333 ページの clresourcetype(1CL), 487 ページの clsetup(1CL), 1433 ページの SUNW.vucmm_framework(5), Unresolved link to " attributes5"

名前

SUNW.vucmm_framework, vucmm_framework — Oracle Solaris Cluster ボリュームマネージャー再構成フレームワーク用のリソースタイプ実装

SUNW.vucmm_framework リソースタイプは、Oracle Solaris Cluster 構成において、異なるクラスタボリュームマネージャーを有効にするフレームワークを表します。このリソースタイプでは、このフレームワークのステータスをモニターできます。

SUNW.vucmm_framework リソースタイプは、シングルインスタンスリソースタイプです。クラスタ内にはこのタイプのリソースは 1 つだけ作成できます。

このリソースタイプを登録し、リソースタイプのインスタンスを作成するには、次のいずれかの方法を使用します。

- このリソースタイプを登録するには、333 ページのclresourcetype(1CL) コマンドを使用します。
- このリソースタイプのインスタンスを作成するには、273 ページのclresource(1CL) コマンドを使用します。

この種類のリソースをノード上でオフラインにする場合、オンライン状態からオフライン状態へ切り替えるには、一定の時間がかかります。オフライン状態への切り替え中も、リソースは引き続き再構成処理に使用されています。ただし、リソースをノード上でオフラインにした場合、再度オンラインに戻すまで、リソースのプロパティーの変更は適用されません。Oracle Solaris Cluster ソフトウェアは、この種類のリソースが無効になっている場合、このことを知らせる警告メッセージを表示します。

この種類のリソースを含むリソースグループを非管理状態に切り替えるには、一定の時間がかかります。非管理状態への切り替え中も、フレームワークは引き続きフレームワーク再構成処理に使用されています。ただし、リソースグループを非管理状態に切り替えた場合、ノード上のリソースプロパティーの変更は有効になりません。フレームワークを停止するには、ノードをリブートする必要があります。

SUNW.vucmm_framework リソースタイプの拡張プロパティーは次のとおりです。

reservation_timeout

型の整数は、最小 100、最大 99999、デフォルト値を 325 で指定します。プロパティーは枠組みの再構成の予約ステップに対して秒単位でタイムアウトを指定します。このプロパティーはいつでも変更できます。

例 413 SUNW.vucmm_framework リソースの作成

この例では、SUNW.vucmm_framework リソースタイプを登録し、vucmm_framework-rs という名前の SUNW.vucmm_framework リソースタイプのインスタンスを作成します。この例では、vucmm_framework-rg という名前のリソースグループがすでに作成されているものとします。

```
phys-host-sc1# clresourcetype register SUNW.vucmm_framework
phys-host-sc1# clresource create -g vucmm_framework-rg \
-t SUNW.vucmm_framework vucmm_framework-rs
```

例 414 vucmm_framework リソースのプロパティーの変更

この例では、枠組みの再構成の予約ステップタイムアウトを 350 秒に設定します。この例では、vucmm_framework-rs という名前のリソースタイプ SUNW.vucmm_framework がすでに作成されているものとします。

```
phys-host-sc1# clresource set -p reservation_timeout=350 vucmm_framework-rs
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core

273 ページのclresource(1CL), 333 ページのclresourcetype(1CL), 487 ページのclsetup(1CL), 1435 ページのSUNW.vucmm_svm(5), Unresolved link to " attributes5"

Unresolved link to " Oracle Solaris Cluster データサービス計画および管理ガイド "

名前

SUNW.vucmm_svm, vucmm_svm — ボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster コンポーネントのリソースタイプ実装を表します

SUNW.vucmm_svm リソースタイプは、Oracle Solaris Cluster ボリュームマネージャー再構成フレームワーク の Solaris Volume Manager for Sun Cluster コンポーネントを表します。

SUNW.vucmm_svm リソースタイプのインスタンスは、Solaris Volume Manager for Sun Cluster コンポーネントの構成パラメータを保持しています。このタイプのインスタンスは、Solaris ボリュームマネージャー for Sun Cluster コンポーネントの再構成のステータスも表示します。

SUNW.vucmm_svm リソースタイプはシングルインスタンスのリソースタイプです。したがって、クラスタに作成可能なリソースは 1 個だけです。

このリソースタイプを登録し、リソースタイプのインスタンスを作成するには、次のいずれかの方法を使用します。

- このリソースタイプを登録するには、333 ページのclresourcetype(1CL) コマンドを使用します。
- このリソースタイプのインスタンスを作成するには、273 ページのclresource(1CL) コマンドを使用します。

SUNW.vucmm_svm リソースは、SUNW.vucmm_framework リソースタイプを使用するリソースグループにおいてのみ有効です。SUNW.rac_framework リソースグループではこのリソースタイプを使用しないでください。

この種類のリソースをノード上でオフラインにする場合、オンライン状態からオフライン状態へ切り替えるには、一定の時間がかかります。オフライン状態への切り替え中も、リソースは引き続き再構成処理に使用されています。ただし、リソースをノード上でオフラインにした場合、再度オンラインに戻すまで、リソースのプロパティーの変更は適用されません。Oracle Solaris Cluster ソフトウェアは、この種類のリソースが無効になっている場合、このことを知らせる警告メッセージを表示します。

この種類のリソースを含むリソースグループを非管理状態に切り替えるには、一定の時間がかかります。非管理状態への切り替え中も、ボリュームマネージャ再構成枠組みは引き続き枠組み再構成処理に使用されています。ただし、リソースグループを非管理状態に切り替えた場合、

ノード上のリソースプロパティーの変更は有効になりません。ボリュームマネージャ再構成枠組みを停止するには、ノードをリブートします。

SUNW.vucmm_svm リソースタイプの拡張プロパティーは次のとおりです。

debug_level

整数型で、最小値は 0、最大値は 10、デフォルトは 1 です。このプロパティーはボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster モジュールのデバッグレベルを指定します。デバッグレベルが上がると、再構成中にログファイルに書き込まれるメッセージが増えます。このプロパティーはいつでも変更できます。

svm_abort_step_timeout

整数型で、最小値は 30、最大値は 99999、デフォルトは 120 です。このプロパティーでは、ボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster モジュールの再構成の中止ステップに対するタイムアウト値を秒単位で設定します。このプロパティーはいつでも変更できます。

svm_return_step_timeout

整数型で、最小値は 30、最大値は 99999、デフォルトは 120 です。このプロパティーでは、ボリュームマネージャ再構成フレームワークの Solaris Volume Manager for Sun Cluster モジュールの再構成の戻りステップに対するタイムアウト値を秒単位で設定します。このプロパティーはいつでも変更できます。

svm_start_step_timeout

整数型で、最小値は 30、最大値は 99999、デフォルトは 120 です。このプロパティーでは、ボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster モジュールの再構成の開始ステップに対するタイムアウト値を秒単位で設定します。このプロパティーはいつでも変更できます。

svm_step1_timeout

整数型で、最小値は 30、最大値は 99999、デフォルトは 120 です。このプロパティーでは、ボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster モジュールの再構成ステップ 1 に対するタイムアウト値を秒単位で設定します。このプロパティーはいつでも変更できます。

svm_step2_timeout

整数型で、最小値は 30、最大値は 99999、デフォルトは 120 です。このプロパティーでは、ボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster モジュールの再構成ステップ 2 に対するタイムアウト値を秒単位で設定します。このプロパティーはいつでも変更できます。

svm_step3_timeout

整数型で、最小値は 30、最大値は 99999、デフォルトは 120 です。このプロパティーでは、ボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster

モジュールの再構成ステップ 3 に対するタイムアウト値を秒単位で設定します。このプロパティーはいつでも変更できます。

`svm_step4_timeout`

整数型で、最小値は 100、最大値は 99999、デフォルトは 120 です。このプロパティーでは、ボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster モジュールの再構成ステップ 4 に対するタイムアウト値を秒単位で設定します。このプロパティーはいつでも変更できます。

`svm_stop_step_timeout`

整数型で、最小値は 30、最大値は 99999、デフォルトは 120 です。このプロパティーでは、ボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster モジュールの再構成の停止ステップに対するタイムアウト値を秒単位で設定します。このプロパティーはいつでも変更できます。

例 415 `vucmm_svm` リソースの作成

この例では、`SUNW.vucmm_svm` リソースタイプを登録し、`vucmm_svm-rs` という名前の `SUNW.vucmm_svm` リソースタイプのインスタンスを作成します。この例では、次に示す Oracle Solaris Cluster オブジェクトが作成済みであることを前提とします。

- 名前が `vucmm_framework-rg` のリソースグループ
- `vucmm_framework-rs` という名前のタイプ `SUNW.vucmm_framework` のリソース

```
phys-schost-1# clresourcetype register SUNW.vucmm_svm
phys-schost-1# clresource create -g vucmm_framework-rg \
-t SUNW.vucmm_svm \
-p resource_dependencies=vucmm_framework-rs vucmm_svm-rs
```

例 416 `vucmm_svm` リソースのプロパティーの変更

この例では、ボリュームマネージャ再構成枠組みの Solaris Volume Manager for Sun Cluster コンポーネントの再構成のステップ 4 に対するタイムアウト値を 300 秒に設定します。この例では、`vucmm_svm-rs` という名前の `SUNW.vucmm_svm` リソースタイプのインスタンスがすでに作成されているものとします。

```
phys-schost-1# clresource set \
-p svm_step4_timeout=300 vucmm_svm-rs
```

次の属性については、Unresolved link to " attributes5" を参照してください。

属性タイプ	属性値
使用条件	ha-cluster/system/core

273 ページの clresource(1CL), 333 ページの clresourcetype(1CL), 487 ページの clsetup(1CL), 1433 ページの SUNW.vucmm_framework(5), Unresolved link to " attributes5"

OSC4 5cl

名前

clconfiguration — では、Oracle Solaris Cluster 構成用 XML ファイルのクラスタ DTD について記述します。

`clconfiguration` では、Oracle Solaris Cluster 構成用 XML (eXtensible Markup Language) ファイルの文書型定義 (Documentation Type Definition、DTD) を規定します。Oracle Solaris Cluster 構成ファイルには、XML 要素付きの Oracle Solaris Cluster 構成情報が含まれています。このファイルには、1 つまたは複数のクラスタ、あるいはクラスタの一部の構成情報が含まれます。この Oracle Solaris Cluster 構成情報は、クラスタ構成のバックアップやクラスタの複製など、多数のクラスタ機能で使用できます。

DTD は、各要素、それらの関係、およびそれらの属性を定義します。要素名は、要素の内容を反映したものになります。たとえば、`<devicegroup>` という要素では、クラスタデバイスグループを定義します。要素の属性が、プロパティーまたは特性の変更や詳細設定に使用される場合もあります。オブジェクト指向の多数の Oracle Solaris Cluster コマンドに含まれる `export` サブコマンドは、DTD で規定されているフォーマットでクラスタオブジェクト情報をエクスポートします。多くの Oracle Solaris Cluster コマンドでは、クラスタ構成用 XML データの使用により、Oracle Solaris Cluster オブジェクトの追加、作成、および変更が行えます。

要素の階層

DTD に必要な要素の階層を、次のリストに示します。このリストでは、子および属性プロパティーのデフォルト値が次のようになっています。

必須	特に断らないかぎり、1 つ以上必要です。
省略可能	特に断らないかぎり、ゼロまたは 1 つ存在できます。

```
<propertyList>
  <property>
<state>
<allNodes>

<-- Cluster -->

<cluster>

  <-- Cluster Nodes -->

  <nodelist>
    <node>

  <-- Cluster Transport -->
```

```
<clusterTransport>
  <transportNodeList>
    <transportNode>
      <transportAdapter>
        <transportType>
  <transportSwitchList>
    <transportSwitch>
  <transportCableList>
    <transportCable>
      <endpoint>

<-- Cluster Global Devices -->

<deviceList>
  <device>
    <devicePath>

<-- Cluster Quorum -->

<clusterQuorum>
  <quorumNodeList>
    <quorumNode>
  <quorumDeviceList>
    <quorumDevice>
      <quorumDevicePathList>
        <quorumDevicePath>

<-- Cluster Device Groups -->

<devicegroupList>
  <devicegroup>
    <memberDeviceList>
      <memberDevice>
    <devicegroupNodeList>
      <devicegroupNode>

<-- Cluster Resource Types -->

<resourcetypeList>
  <resourcetype>
     <resourcetypeRTRFile>
     <resourcetypeNodeList>
       <resourcetypeNode>
       <methodList>
       <method>
       <parameterList>
         <parameter>

<-- Cluster Resources -->

<resourceList>
  <resource>
    <resourceNodeList>
```

```
      <resourceNode)
    <monitoredState>

<-- Cluster Resource Groups -->

<resourcegroupList>
  <resourcegroup>
    <failoverMode>
    <managedState>
    <resourcegroupNodeList>
      <resourcegroupNode>
    <resourcegroupResourceList>
      <resourcegroupResource>

<-- Cluster NAS Devices -->

<nasdeviceList>
  <nasdevice>
     <nasdir>

<-- Cluster SNMP -->

<snmpmibList>
  <snmpmib>
<snmphostList>
  <snmphost>
<snmpuserList>
  <snmpuser>
<-- Cluster Telemetrics -->

<telemetrics>
  <telemetryObjectType>
    <telemetryAttribute>
```

要素

このセクションでは、クラスタ DTD で定義されているすべての要素の一覧を示し、それらについて解説します。要素に子または属性が必要な場合、必須のデフォルト値は 1 です。オプションの要素のデフォルト数は、ゼロまたは 1 です。

`<allNodes>`

クラスタ内の全メンバーノードのリストです。`<allNodes>` 要素は、汎用要素です。

`<allNodes>` 要素は、クラスタの全ノードを表すために使用されます。

親:	`<resourcetypeNodeList>`
子:	なし
属性:	なし

<cluster>

クラスタ構成用 XML ファイル全体のルート要素。どのクラスタ構成用 XML ファイルも、ルートとしてこの要素で始めます。DTD では、<cluster> 要素を 1 つだけ受け入れることができます。クラスタ構成用 XML ファイル内の後続の <cluster> 要素は無視されます。

親: なし

子: (省略可能)

- <propertyList>
- <nodeList>
- <clusterTransport>
- <deviceList>
- <clusterQuorum>
- <deviceGroupList>
- <resourcetypeList>
- <resourcegroupList>
- <resourceList>
- <nasdeviceList>
- <snmpmibList>
- <snmphostList>
- <snmpuserList>

属性: 必須:

- name

  クラスタの名前。

<clusterQuorum>

クラスタ定足数構成のルート要素。クラスタ定足数情報はすべて、<clusterQuorum> 要素の子要素内で定義されます。

親: <cluster>

子: (省略可能)

- <quorumDeviceList>
- <quorumNodeList>

属性: なし

<clusterTransport>

クラスタトランスポート構成のルート要素。クラスタトランスポート情報はすべて、<clusterTransport> 要素のサブレベル内で表示されます。

親: <cluster>.

子: (省略可能)

- <transportNodeList>
- <transportSwitchList>
- <transportCableList>

属性: なし

<device>

クラスタデバイス ID 擬似ドライバ (DID) デバイス。

親: <deviceList>

子: (省略可能)

- <devicePath> (ゼロ個以上)

属性: 必須:

- ctd

  UNIX ディスク名。

- name

  デバイスのインスタンス番号。

<devicegroup>

クラスタデバイスグループインスタンスのルート要素。各デバイスグループの特性はすべて、<devicegroup> 要素の子要素内で定義されます。

親: <devicegroupList>

子: (省略可能)

- <devicegroupNodeList>
- <memberDeviceList>
- <propertyList>

属性: 必須:

- name

  デバイスグループの名前。name 属性は、有効な任意の文字列とすることができます。

- type

  デバイスグループのタイプ。type 属性は、rawdisk、vxvm、svm、または sds という値を持つことができます。

<devicegroupList>

すべてのクラスタデバイスグループのリスト。

親: <cluster>

子: (省略可能)

- <devicegroup>

  クラスタ内の各デバイスグループでは、<devicegroup> 要素を 1 つ使用できます。

属性: なし

<devicegroupNode>

デバイスグループが存在するノード。

親: <devicegroupNodeList>

子: なし

属性: 必須:

- nodeRef

  クラスタノードの名前を指定します。

<devicegroupNodeList>

デバイスグループが存在するノードのリスト。

親: <devicegroup> (1 つ以上)

子: 必須:

- <devicegroupNode> (1 つ以上)

属性: なし

<deviceList>

クラスタ DID デバイスのリスト。

親: <cluster>

子: (省略可能)

- <device>

属性: 固定:

- readonly

  readonly 属性は、true の固定値を持ちます。

<deviceNode>

特定の <device> が存在するノードおよびディスクデバイス。

親: <device>

子: なし

属性: 必須:

- nodeRef

  インスタンスが存在するノードの名前。

<endpoint>

トランスポート終端の 1 つ。

親: <transportCable>

子: なし

属性: 必須:

- name

  アダプタまたはスイッチの名前。

- nodeRef

  指定したアダプタがホストされているノードの名前。nodeRef 属性が必要なのは、type 属性に adapter が設定されている場合だけです。

- type

  type 属性には、adapter または switch を設定できます。

  type 属性に adapter を設定する場合は、nodeRef 属性を指定します。

  type 属性を switch に設定した場合、port 属性を指定できます。ただし、port 属性は必要ではありません。

(省略可能)

- port

  スイッチ上のポートの番号。port 属性は、type 属性に switch を設定する場合だけ指定します。

<failoverMode>

リソースグループのフェイルオーバーモード。

親: <resourcegroup>

子: なし

属性: 必須:

- value

  value 属性には、failover または scalable を設定できます。

<managedState>

リソースグループが管理されているのかどうかを示します。

親: <resourcegroup>

子: なし

属性: 必須:

- value

  value 属性は、true または false とすることができます。

<memberDevice>

特定のデバイスグループのメンバー名。<devicegroup> が一連の rawdisk のタイプである場合は、それぞれに raw ディスクパスの名前を含む 1 つ以上の <member> 要素を指定する必要があります。

親: <memberDeviceList>

子: なし

属性: 必須:

- name

  メンバーの名前。

<memberDeviceList>

デバイスグループメンバーのリスト。

親: <devicegroup>(1 つ以上)

子: 必須:

- <memberDevice>

属性: なし

<method>

汎用のメソッドタイプと特定のリソースタイプの実際のメソッド名の間のマッピング。

親: <methodList>

子: なし

属性: 必須:

- name

  リソースタイプのメソッドの実際の名前。

- type

  リソースタイプのメソッドのタイプ。次のタイプを指定できます。

  - MONITOR_CHECK
  - MONITOR_START
  - MONITOR_STOP

- PRENET_START
- START
- STOP
- VALIDATE
- UPDATE

<methodList>

特定の <resourcetype> で使用可能なすべての <method> 要素のリスト。

親: <resourcetype>

子: (省略可能)

- <method>

属性: 固定:

- readonly

  readonly 属性は、true の固定値を持ちます。

<monitoredState>

クラスタ内の要素の状態の一部を示すブール値。たとえば、リソースの <monitoredState> はそのリソースがモニター対象であるかどうかを示しますが、そのリソースが使用可能かどうかは示しません。

親: <resource>

子: なし

属性: 必須:

- value

  value 属性には、true または false を設定できます。

<nasdevice>

クラスタ上の NAS デバイスの単一インスタンス。

親: <nasdeviceList>

子: (省略可能)

- <nasdir>

属性: 必須:

- name

  NAS デバイスのホスト名。

- type

  NAS デバイスのタイプ。Oracle ZFS Storage Appliance に対して sun_uss を指定する必要があります。

(省略可能)

- userid

  NAS デバイスへのアクセスに必要なユーザー名。

<nasdeviceList>

クラスタ上のすべての NAS デバイスのリスト。

親: <cluster>

子: (省略可能)

- <nasdevice>

属性: なし

<nasdir>

NAS デバイス上のディレクトリの 1 つ。各 NAS デバイスは、複数の NAS ディレクトリを持つことができます。

親: <nasdevice>

子: なし

属性: 必須:

- path

  NAS ディレクトリへのパス。

<node>

1 つのクラスタノード。クラスタ内のノードごとに、<node> 要素を 1 つ指定します。

親: <nodeList>

子: (省略可能)

- <propertyList>

属性: 必須:

- name

  ノードの名前に等しくします。

(省略可能)

- id

  クラスタノード ID。指定しなかった場合、クラスタノード ID 属性のデフォルト値は空の文字列になります。

<nodeList>

クラスタ内の全ノードのリスト。

親: <cluster>

子: (省略可能)

- <node>

  クラスタ上のノードごとに、node 属性を 1 つ以上指定します。

属性: なし

<parameter>

<method> 要素のタイムアウト値とクラスタリソースタイプのほかのパラメータを規定する一連の属性。

親: <parameterList>

子: なし

属性: 必須:

- extension

  extension 属性には、true または false を設定できます。

- name

  パラメータの名前

- tunability

パラメータの調整時期の値。tunability 属性には、次の値のいずれか 1 つを設定できます。atCreation、anyTime、または whenDisabled。

- type

  パラメータのタイプ。type 属性には、次の値のいずれか 1 つを設定できます。boolean、enum、int、string、または stringArray。

(省略可能)

- default

  値が明示的に指定されなかった場合のこのパラメータのデフォルト値。たとえば、method 要素のタイムアウトのデフォルト値は START です。

- description

  パラメータの説明。未定義の場合、この属性のデフォルト値は空の文字列です。

- enumList

  オブジェクトの列挙型リスト。たとえば、優先度に従ったフェイルオーバーモードのリストが属性になる場合もあります。

- maxLength

  string または stringArray 型のパラメータの最大長。

- minArrayLength

  stringArray 型パラメータの最小サイズ。

- minLength

  string または stringArray 型のパラメータの最小長。

<parameterList>

リソースタイプを記述する <parameter> 要素のリスト。

親: <resourcetype>

子: (省略可能)

- <parameter>

属性: 固定:

- readonly

  readonly 属性は、true の固定値を持ちます。

<property>

1 つのプロパティーを規定する汎用要素。このプロパティーは、クラスタ関連構成のサブセットに固有なものではありません。

親: <propertyList>

子: なし

属性: 必須:

- name

  プロパティーの名前です。

- value

  プロパティーの値です。

(省略可能)

- readonly

  readonly 属性には、true または false を設定できます。この値を指定しないと、この属性のデフォルト値は false になります。

- type

  プロパティータイプ。

<propertyList>

<property> 要素のリスト。<propertyList> 要素は、汎用要素です。

親: <cluster>、<deviceGroup>、<node>、<quorumDevice>、<quorumNode>、<res

子: 省略可能:

- <property>

属性: (省略可能)

- extension

  この属性は次のいずれかの値を取ることができます。true、false、mixed、または doesNotApply。値を指定しないと、extension 属性はデフォルト値 doesNotApply になります。

- readonly

  この属性は、true または false の値を持つことができます。値を指定しないと、readonly 属性はデフォルト値 false を持ちます。

<quorumDevice>

それぞれのクラスタ定足数デバイス。

親: <quorumDeviceList>

子: (省略可能)

- <propertyList>

  <quorumDevice> 要素は、<propertyList> の子を 1 つだけ持つことができます。

- <quorumDevicePathList>

  <quorumDevice> 要素は、<quorumDevicePathList> の子を 1 つだけ持つことができます。

属性: 必須:

- name

  定足数デバイスの名前。

- type

  この要素によって参照される定足数デバイスのタイプ。type 属性には、scsi または quorum_server を設定できます。

<quorumDeviceList>

クラスタ内の定足数デバイスのリスト。

親: <clusterQuorum>

子: (省略可能)

- <quorumDevice>

属性: なし

<quorumDevicePath>

クラスタ定足数デバイスのパス。

親: <quorumDevicePathList>

子: (省略可能)

- <state>

  <quorumDevicePath> 要素は、<state> の子を 1 つだけ持つことができます。

属性: 必須:

- nodeRef

  定足数デバイスが存在するノードの名前。

<quorumDevicePathList>

特定の <quorumDevice> のすべてのパスのリスト。

親: <quorumDevice>

子: 必須:

- <quorumDevicePath>

属性: 固定:

- readonly

  readonly 属性には、true が設定されます。

<quorumNode>

クラスタ定足数に参加するクラスタ内の 1 つのノード。

親: <quorumNodeList>

子: (省略可能)

- <propertyList>

属性: 必須:

- <nodeRef>

  ノードの名前。

<quorumNodeList>

クラスタ定足数に参加するすべてのノードのリスト。installmode にない有効なクラスタの場合、このリストには通常、クラスタ内のすべてのノードが含まれます。まだ installmode にあるクラスタの場合、このリストにはクラスタノードの 1 つだけが含まれる場合があります。

親: <clusterQuorum>

子: 必須:

- <quorumNode>

属性: 固定:

- readonly

  readonly 属性には、true が設定されます。

<resource>

1 つのクラスタリソース。

親: <resourceList>

子: (省略可能)

- <resourceNodeList>
- <propertyList>

属性: 必須:

- name

  リソースの名前

- resourcegroupRef

  リソースが属するリソースグループ。

- resourcetypeRef

  この要素によって規定されるリソースの型。

<resourceList>

構成内で定義されているクラスタリソースのルート ノードのリスト。

親: <cluster>

子: (省略可能)

- <resource>

属性: なし

<resourcegroup>

クラスタリソースグループ。

親: <resourcegroupList>

子: 必須:

- <failoverMode>
- <managedState>
- <resourcegroupNodeList>
- <resourcegroupResourceList>
- <propertyList>

属性: 必須:

- name

  リソースの名前

<resourcegroupList>

構成内で定義されているクラスタリソースグループのルートノード。

親: <cluster>

子: (省略可能)

- <resourcegroup>

属性: なし

<resourcegroupNode>

リソースグループが定義されているノード。

親: <resourcegroupNodeList>

子: なし

属性: 必須:

- nodeRef

クラスタノードの名前。

(省略可能)

■ ゾーン

ゾーンの名前。

<resourcegroupNodeList>

特定のリソースグループが動作するクラスタノード。

親: <resourcegroup>

子: 必須:

■ <resourcegroupNode>

属性: なし

<resourcegroupResource>

特定のリソースグループに属するクラスタリソース。

親: <resourcegroupResourceList>

子: なし

属性: 必須:

■ resourceRef

リソースの名前

<resourcegroupResourceList>

リソースグループで定義されているリソースのリスト。

親: <resourcegroup>

子: (省略可能)

■ <resourcegroupResource>

属性: なし

<resourceNode>

リソースが定義されているノード。

親: <resourceNodeList>

子: 必須:

- <state>
- <monitoredState>

(省略可能)

- <propertyList>

属性: 必須:

- nodeRef

  リソースタイプの名前。

(省略可能)

- zone

  ゾーンの名前。

<resourcetype>

クラスタ内で使用可能な 1 つのクラスタリソースタイプ。

親: <resourcetypeList>

子: (省略可能)

- <resourcetypeRTRFile>
- <resourcetypeNodeList>
- <methodList>
- <parameterList>
- <propertyList>

属性: 必須:

- name

  リソースタイプの名前。

<resourcetypeList>

構成内で定義されているクラスタリソースタイプのルートノード。

親: <cluster>

子: (省略可能)

- <resourcetype>

属性: なし

<resourcetypeNode>

リソースタイプが定義されている 1 つのノード。

親: <resourcetypeNodeList>

子: なし

属性: 必須:

- nodeRef

  クラスタノードの名前。

<resourcetypeNodeList>

特定のリソースタイプが存在するクラスタノードのリスト。

親: <resourcetype>

子: 必須: <resourcetypeNodeList> 要素には、1 つ以上の <resourcetypeNode> 要素か、または正確に 1 つの <allNodes> 要素のどちらかを含める必要があります。

- <resourcetypeNode>
- <allNodes>

属性: なし

<resourcetypeRTRFile>

特定のリソースタイプを規定するリソースタイプ登録 (RTR) ファイルの名前。

親: <resourcetype>

子: なし

属性: 必須:

- name

  RTR ファイルの名前。

<snmphost>

クラスタノード上で構成される SNMP ホストおよびコミュニティー。

親: <snmphostList>

子: なし

属性: 必須:

- `community`

  SNMP コミュニティー名。

- `name`

  インスタンスの名前。

- `nodeRef`

  SNMP ホストとコミュニティーが存在するノード。

`<snmphostList>`

クラスタノード上で構成される SNMP ホストおよびコミュニティーのリスト。

親: `cluster>`

子: (省略可能)

- `<snmphost>`

属性: なし

`<snmpmib>`

クラスタノード上の SNMP MIB。

親: `<snmpmibList>`

子: (省略可能)

- `state`

属性: 必須:

- `name`

  MIB の名前。

- `nodeRef`

  SNMP MIB が存在するノード。

(省略可能)

- `protocol`

  MIB が使用する SNMP プロトコル。この属性のデフォルト値は `SNMPv2` です。

- value

  SNMPv3 または SNMPv2

snmpmibList

クラスタノード上の SNMP MIB のリスト。

親: <cluster>

子: (省略可能)

- <snmpmib>

属性: なし

<snmpuser>

クラスタノード上で構成される SNMPv3 ユーザー。

親: <snmpuserList>

子: なし

属性: 必須:

- name

  ユーザーの名前。

- nodeRef

  SNMPv3 ユーザーが存在するノード。

- auth

  auth 属性は、MD5 または SHA に設定できます。

(省略可能)

- defaultUser

  defaultUser 属性は、yes または no に設定できます。値が指定されない場合、この属性のデフォルト値はノード構成に基づき、いずれか適切な値になります。

- defaultSecurityLevel

  ユーザーのセキュリティーレベル。security 属性は次のいずれかの値に設定できます:

- authPriv
- authNoPriv
- noAuthNoPriv

<snmpuserList>
クラスタノード上で構成される SNMPv3 ユーザーのリスト。

親: <cluster>

子: <snmpuser>

属性: なし

<state>
クラスタ構成内のさまざまなオブジェクトの状態。<state> 要素は、汎用要素です。

親: <quorumDevicePath>、<resourceNode>、snmpmib>、telemetryAttribute、<

子: なし

属性: 必須:

- value

  value 属性には、enabled または disabled を設定できます。

<telemetrics>
クラスタモニタリングしきい値

親: <cluster>

子: (省略可能)

- <telemetryObjectType>

属性: なし

<telemetryAttribute>
ユーザーがモニターできるシステムリソースの属性。

親: <telemetryObjectType>

子: 必須:

■ <state> (1 つ以上)

属性: 必須:

■ name

属性の名前

<telemetryObjectType>

ユーザーがモニターできるオブジェクトのタイプ。

親: <telemetrics>

子: 必須:

■ <telemetryAttibute>

属性: 必須:

■ name

属性の名前

<transportAdapter>

プライベートクラスタトランスポートで使用するネットワークアダプタ。

親: <transportNode>

子: (省略可能)

■ <state>

■ <transportType>

■ <propertyList>

属性: 必須:

■ name

ネットワークアダプタの名前。

<transportCable>

プライベートクラスタトランスポートで使用するネットワークケーブル。このケーブルは、物理ケーブルを指さない場合もあり、2 つの <endpoint> 要素間のパスとする方が適切です。

親: <transportCableList>

子: 必須:

■ <endpoint>

<transportCable> 要素は、2 つの <endpoint> 要素を持つ必要があります。それぞれの endpont 要素では、ケーブル終端の一方を規定します。

(省略可能)

■ <state>

<transportCable> 要素は、<state> 要素を 1 つ持つことができます。

属性: なし

<transportCableList>

2 つのクラスタ <endpoint> 要素を接続するのに使用するネットワークケーブルのリスト。

親: <clusterTransport>

子: **省略可能:**

■ <transportCable>

属性: なし

<transportNode>

プライベートクラスタトランスポートで使用されるクラスタノードの 1 つ。クラスタのノードごとに、<transportNode> 要素を 1 つ指定します。

親: <transportNodeList>

子: (省略可能)

■ <nodeRef>

属性: 必須:

■ transportAdapterList

クラスタノードの名前。

<transportNodeList>

プライベートクラスタトランスポートで使用するノードのリスト。このノードリストには常に、クラスタのメンバーと同じ一連のノードが含まれます。

親: <clusterTransport>

子: (省略可能)

- <transportNode>

属性: なし

<transportSwitch>

1 つのクラスタトランスポートスイッチ。

親: <transportSwitchList>

子: (省略可能)

- <state>

属性: 必須:

- name

  トランスポートスイッチの名前。

(省略可能)

- port

  スイッチ上のポートの番号。

<transportSwitchList>

プライベートクラスタトランスポートシステムによって使用されるネットワークスイッチのリスト。

親: <clusterTransport>

子: (省略可能)

- <transportSwitch>

属性: なし

<transportType>

<transportAdapter> 要素で使用されるネットワークトランスポートのタイプ。

親: <transportAdapter>

子: (省略可能)

- <propertyList>

属性: 必須:

- `value`

  `value` 属性には、`dlpi` または `rsm` を設定できます。

`/usr/cluster/lib/xml/cluster.dtd`

Oracle Solaris Cluster 構成用 XML ファイルの構造を定義する文書型定義 (DTD) ファイル。

19 ページの`Intro`(1CL), 551 ページの`cluster`(1CL)

http://www.w3.org/XML/

OSC4 7

名前

clprivnet — SUNW,clprivnet Oracle Solaris Cluster プライベートネットワークドライバ

`/dev/clprivnet`

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

`SUNW,clprivnet` Oracle Solaris Cluster プライベートネットワークドライバは、STREAMS 擬似ドライバです。このドライバは、Oracle Solaris Clustrer プライベートネットワーク経由で通信を行なう標準型 Solaris インタフェースを使用する Oracle Solaris Cluster の常駐アプリケーションをサポートします。データトラフィックを全リンクから外すことで、プライベートネットワークの帯域幅を自在に活用し、可用性と耐障害性に優れた通信を実現します。

アプリケーションプログラミングインタフェース

このドライバは、文字型特殊デバイス `/dev/clprivnet` でサポートされ、Oracle Solaris Cluster の内部処理と Solaris の標準ネットワークユーティリティーで使用されます。このデバイスインターフェースは、一般的なアプリケーション通信に直接使用しないでください。

管理

Oracle Solaris Cluster の内部インフラストラクチャーは、ドライバをインターネットインタフェースとして完全に管理、構成します。

`/dev/clprivnet`

`clprivnet` 文字型特殊デバイス

`/usr/kernel/drv/clprivnet.conf`

システム全体のデフォルトのデバイスドライバプロパティー

名前

did — user-configurable DID pseudo driver

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

DIDはユーザーによる構成が可能な擬似デバイスドライバであり、基本となるディスク、テープ、および CD-ROM デバイスへのアクセスを提供します。デバイスが一意のデバイス ID をサポートする場合、デバイス ID に従って 1 つのデバイスに複数のパスが決定されます。同じデバイス ID に複数のパスが使用可能な場合でも、実際のデバイスに与えられる DID 名は 1 つのみです。

クラスタ環境では、複数のホストまたはコントローラへの接続にかかわらず、特定の物理デバイスが同じ DID 名を持ちます。ただし、これは物理ディスクなど広域で単一のデバイス識別名をサポートするデバイスの場合にのみ該当します。

DID は、`/dev/did` 下で管理される各デバイスタイプ用に同じディレクトリを保持しています。これらのディレクトリにあるデバイスは、DID 以外のデバイスと同様の動作をします。たとえば、ディスクや CD-ROM デバイスのスライスの保持や、さまざまなテープデバイス操作に使用する名前の保持があります。`/dev/did/dsk` と `/dev/did/rdsk` により、ディスク用に raw デバイスアクセスとブロック型デバイスアクセスの両方もサポートされます。

どの時点でも、入出力はデバイスへの一方向のパスのみサポートされます。DID によるマルチパス機能は現在サポートされていません。

DID デバイスを使用するには、まず 793 ページの`scdidadm`(1M) コマンドを使用して初期化する必要があります。

DID ドライバは、管理ノードおよび各 DID デバイスマイナーのノードを保持します。

管理ノードでサポートされるユーザー ioctls はありません。

`DKIOCINFO` ioctl は、`/dev/did/rdsk/d0s2` などの DID デバイスノードに対して呼び出された場合にサポートされます。

その他の ioctls は、次のドライバに直接渡されます。

`/dev/did/dsk/d`*n*`s`*m*

ブロックディスクまたは CD-ROM デバイス (*n* はデバイス番号、*m* はスライス番号)

`/dev/did/rdsk/d`*n*`s`*m*

raw ディスクまたは CD-ROM デバイス (*n* はデバイス番号、*m* はスライス番号)

`/dev/did/rmt/`*n*

テープデバイス (*n* はデバイス番号)

`/dev/did/admin`

管理デバイス

`/kernel/drv/did`

ドライバモジュール

`/kernel/drv/did.conf`

ドライバ構成ファイル

`/etc/did.conf`

クラスタ化されていないシステム用の `cldevice` 構成ファイル

`Cluster Configuration Repository` (CCR)ファイル

61 ページの`cldevice`(1CL) は、クラスタ化されたシステムの CCR で構成を保守します。

19 ページの`Intro`(1CL), 61 ページの`cldevice`(1CL), Unresolved link to "devfsadm1M"

DID はデバイス用の名前を、デバイスホットプラグ時にオーバーヘッドを減らすためにまとめて作成します。ディスクのデバイス名は、`/dev/did/dsk` と`/dev/did/rdsk` に作成されます。一度に作成されるのは 100 個のディスクまでです。テープのデバイス名は `/dev/did/rmt` に作成されます。一度に作成されるのは 10 個のテープまでです。クラスタに追加されるデバイスが、現在の名前で処理できる数より多い場合は、別のグループが作成されます。

OSC4 7p

名前

sctransp_dlpi — dlpi クラスタインターコネクトの構成

注記 - Oracle Solaris Cluster ソフトウェアには、オブジェクト指向のコマンドセットが含まれます。Oracle Solaris Cluster ソフトウェアは従来のコマンドセットもサポートしますが、Oracle Solaris Cluster の手順に関するドキュメントではオブジェクト指向のコマンドセットのみを使用します。オブジェクト指向のコマンドセットの詳細は、19 ページの`Intro`(1CL) のマニュアルページを参照してください。

`dlpi` は、サポートされているクラスタトランスポートタイプです。

19 ページの`Intro`(1CL), 551 ページの`cluster`(1CL), 747 ページの`scconf`(1M), 819 ページの`scinstall`(1M)

索引

さ

た

な

は

C

D

P

Q

R

S

T

U

V