IBM PowerHA SystemMirror for AIX

Standard Edition

V7.2.1

PowerHA SystemMirror 命令

IBM PowerHA SystemMirror for AIX

Standard Edition

V7.2.1

PowerHA SystemMirror 命令

注意

在使用本资料及其支持的产品之前，请阅读第 91 页的『声明』中的信息。

此版本适用于 IBM PowerHA SystemMirror 7.2.1 Standard Edition for AIX 以及所有后续发行版和修订版，除非在新版本中另有说明。

目录

关于本文档

可使用命令来管理和配置 PowerHA® SystemMirror® 集群。每个命令具有语法和示例。

突出显示

在本文档中使用下列突出显示约定：

粗体	标识命令、子例程、关键字、文件、结构、目录和名称由系统预先定义的其他项。粗体突出显示还标识图形对象，例如您选择的按钮、标签和图标。
斜体	标识您提供的实际名称或值的参数。
等宽字体	标识特定数据值的示例、类似于您可能看到显示出来的文本的示例、您可编写为参数的程序代码部分的示例、系统产生的消息或您必须输入的文本。

AIX 中的区分大小写

AIX® 操作系统中的所有内容都区分大小写，这表示它区分大写字母和小写字母。例如，您可以使用 **ls** 命令来列示文件。如果输入 LS，那么系统对该命令的响应是未找到。同样，**FILEA**、**FiLea** 和 **filea** 是三个不同的文件名，即使它们在同一个目录中也是如此。为了避免导致执行不期望的操作，请始终确保使用正确的大小写字母。

ISO 9000

在本产品的开发和制造过程中，使用了 ISO 9000 注册质量体系。

相关信息

- PowerHA SystemMirror V7.2.1 PDF 文档在以下主题提供：PowerHA SystemMirror 7.2.1 PDFs。
- PowerHA SystemMirror V7.2.1 发行说明在以下主题提供：PowerHA SystemMirror 7.2.1 release notes。

PowerHA SystemMirror 命令

以下命令通常用于获取有关集群环境的信息或者用于运行特定功能。以下每个命令具有语法和示例。

有关命令的功能和限制的完整信息，请参阅联机帮助页。PowerHA SystemMirror for AIX 命令的联机帮助页安装在 /usr/share/man/info/EN_US/a_doc_lib/cmds/powerha_cmds 目录中。

要查看命令的联机帮助页信息，请使用以下命令：

```
man command-name
```

command-name 是 PowerHA SystemMirror 命令或脚本的实际名称。例如，输入 man clpasswd 以获取有关 PowerHA SystemMirror **clpasswd** 命令的信息。

"PowerHA SystemMirror 命令"中的新增内容

阅读 PowerHA SystemMirror 命令主题集合的新增信息或已进行显著更改的信息。

如何查看新增内容或已更改的内容

在本 PDF 文件中，您可能会在左页边缘处看到用于标识新信息和已更改信息的修订线 (|)。

2016 年 12 月

以下信息是本主题集合的更新摘要：

- 已在用于仅启动 CAA 服务的 [START_CAA={no|yes|only}] 语法中添加 only 选项。有关更多信息，请参阅第 37 页的『clmgr 命令』主题。
- 已向以下主题中的以下项添加缺失标志：
  - 第 9 页的『cl_rc.cluster 命令』
  - 第 89 页的『rc.cluster 命令』

cl_clstop 命令

用途

使用系统资源控制器 (SRC) 工具来停止集群守护程序。

语法

```
cl_clstop [-cspoc "[-f] [-n NodeList | -g ResourceGroup]"] -f
cl_clstop [-cspoc "[-f] [-n NodeList | -g ResourceGroup]"] -g [-s] [-y] [-N | -R | -B]
cl_clstop [-cspoc "[-f] [-n NodeList | -g ResourceGroup]"] -gr [-s] [-y] [-N | -R | -B]
```

描述

cl_clstop 命令关闭集群节点间的集群服务。缺省情况下，**cl_clstop** 命令停止所有集群节点间的集群服务。但是，可指定要停止集群服务的节点列表。**cl_clstop** 使用系统资源控制器 (SRC) 停止集群守护程序。通过使用"正常关闭"选项或强制选项停止集群守护程序。此命令可选择通过 /etc/inittab 文件中的条目禁止重新引导时自动启动。如果使用"正常关闭并接管"选项来关闭集群守护程序，那么必须指定节点列表。缺省情况下，

cl_clstop 命令要求集群中的所有节点或节点列表中的所有节点可通过网络访问或联机访问；否则，**cl_clstop** 命令将失败。

标志

-cspoc
可对 C-SPOC 选项使用以下自变量：

-f 强制 C-SPOC 命令跳过缺省验证。如果设置了此标志并且集群节点不可访问，那么 **cl_clstop** 命令将报告一条警告并且继续在其他节点上执行。

-n NodeList
在节点列表中指定的节点上关闭集群服务。

-g ResourceGroup
生成节点列表，这些节点参与运行 **cl_clstop** 命令的资源组。

-f 强制关闭。集群守护程序终止，并且不运行任何本地过程。

-g 正常关闭并且不接管。

-gr
正常关闭，资源被此节点释放并由另一节点接管。守护程序正常终止，节点释放其资源，资源被另一节点接管。必须对"正常关闭并接管"选项指定节点列表。

-s 执行静默关闭。此标志不会通过 **wall** 命令广播关闭消息。缺省设置是广播。

-y
关闭集群节点前不要求操作员确认。此标志是缺省值。

-B 立即停止，并在后续系统重新启动时停止。

-N
立即关闭。

-R 在后续系统重新启动时停止并移除 /etc/inittab 文件中的该条目。

示例

1. 关闭集群节点时，如果要在 node1 上使用"正常关闭并接管"选项（释放资源），并且在停止集群进程和释放资源前不向用户发送警告消息，请输入：

   ```
   cl_clstop -cspoc "-n node1" -ysNgr
   ```

2. 要在所有集群节点上强制立即关闭该集群（不释放资源），并在停止集群进程前向用户广播警告消息，请输入：

   ```
   cl_clstop -yNf
   ```

3. 关闭集群节点时，如果要使用"在所有集群节点上正常关闭"选项，并在停止集群进程前向用户广播警告消息，请输入：

   ```
   cl_clstop -yg
   ```

   注：如果未指定 -g 或 -n 标志，那么将在所有集群节点上执行缺省操作。

相关参考:

第 37 页的『clmgr 命令』

cl_convert 命令

用途

将 PowerHA SystemMirror 软件升级到最新版本涉及到将配置数据库从先前发行版转换为该配置数据库的当前发行版。安装 PowerHA SystemMirror 时，自动运行 **cl_convert**。但是，如果安装失败，那么您必须从命令行中运行 **cl_convert**。需要 Root 用户特权以运行 **cl_convert**。

语法

[-F] -v < *release*> [-s< *simulationfile*>][-i]

描述

此命令将先前版本的 ODM 数据复制到新版本的 ODM 结构。如果新版本中删除了某些字段，那么数据将保存到 **/tmp/cl_convert_PowerHA SystemMirror_OLD**。然后，此命令确保数据对于新版本具有正确的格式。

在安装新版本时，安装脚本会将后缀 OLD 添加到 **/etc/objrepos** 目录中存储的 PowerHA SystemMirrorxxx 类中，并且为新版本创建新的 PowerHA SystemMirrorxxx 类。安装脚本发出 **cl_convert** 命令，此命令将 PowerHA SystemMirrorxxxOLD 中的数据转换为 PowerHA SystemMirrorxxx 中对应的新类。

您可以从命令行中运行 **cl_convert** 命令，但是此命令期望 PowerHA SystemMirrorxxx 和 PowerHA SystemMirrorxxxOLD ODM 已存在。

您可能希望运行带 **-F** 选项的 **cl_convert** 命令。如果不指定此选项，那么 **cl_convert** 命令将检查新 ODM 类 PowerHA SystemMirrorcluster 中的已配置数据。如果数据存在，那么此命令退出，并且不执行转换。如果指定了 **-F** 选项，那么此命令将继续，而不检查现有数据。

请注意，在将最终数据写入到 PowerHA SystemMirrorxxx ODM 之前，**cl_convert** 会将 PowerHA SystemMirrorxxx 和 PowerHA SystemMirrorxxxOLD ODM 复制到临时文件 (**/tmp/tmpodmdir**) 以进行处理。如果 **cl_convert** 遇到任何类型的错误，那么不会覆盖 PowerHA SystemMirrorxxx ODM。如果不发生任何错误，那么将覆盖 PowerHA SystemMirrorxxx ODM 并且安装脚本将移除 PowerHA SystemMirrorxxxOLD ODM

注意，您必须处于 conversion 目录中才能运行此命令：

```
/usr/es/sbin/cluster/conversion
```

此外，cl_convert 假定为 ODMDIR 设置了正确的值。可以在 **/tmp/clconvert.log** 中找到 cl_convert 的结果。

标志

-F 强制标志。使 cl_convert 覆盖现有 ODM 对象类，无论现有条目的数量如何。省略此标志将导致 cl_convert 检查 PowerHA SystemMirrorcluster 中的数据（其中始终来自先前配置）并且在遇到数据的情况下将会退出。

-v 发行版版本标志。表示旧版本的发行版号。

要点：除非您知道要作为转换源的版本，否则请勿使用 cl_convert 命令。

-s <simulation_file>
模拟标志。指示将生成的 ODM 数据以文本格式写入到指定文件，而不是写回到新的 PowerHA SystemMirrorxxx ODM。

-i 忽略复制标志。指定不将 PowerHA SystemMirrorxxxOLD 数据复制到新的 PowerHA SystemMirrorxxx ODM，而是直接在新的 PowerHA SystemMirrorxxx ODM 上操作。此标志主要由 clconvert_snapshot 使用。

注： AIX 环境变量 ODMDIR 必须设置为您要转换的目录。

示例

如果已经为前发行版配置了集群，那么在安装新版本的 PowerHA SystemMirror 期间，安装脚本将以如下方式来调用 cl_convert：

```
cl_convert -F -v <version of prior release>
```

---

cl_lsfs 命令

用途

显示共享文件系统的特征。

注： 与特定标志关联的参数必须在紧邻标志的后面指定。

语法

```
cl_lsfs [-cspoc"[-f] [-g ResourceGroup | -n Nodelist ]" [-q]  [-c | -l] FileSystem ]...
```

标志

-cspoc
用于指定以下其中一个 C-SPOC 选项的参数：

-f - 与 **cl_lsfs** 命令一起使用时，此选项没有任何效果。

-g *ResourceGroup* - 生成节点的列表，这些节点加入了将要在其中执行此命令的资源组。

-n *nodelist* - 在此节点列表上运行命令。如果有一个以上的节点，请用逗号分隔列出的节点。

-c 指定不同搜索模式以确定底层的 AIX **lsfs** 命令是否返回了数据。

-l 指定以列表格式输出。

-q 查询逻辑卷管理器 (LVM) 以获取逻辑卷大小（以 512 字节块为单位）并且查询 JFS 超块以获取文件系统大小、片段大小、压缩算法（如果有）以及每个索引节点的字节数量 (nbpi)。除了显示 **lsfs** 命令报告的其他文件系统特征以外，还将显示此信息。

示例

1. 要显示有关集群中所有共享文件系统的特征，请输入：

   ```
   cl_lsfs
   ```

2. 显示有关 *resource_grp1* 中在参与节点之间共享的文件系统的特征。

   ```
   cl_lsfs -cspoc "-g resource_grp1"
   ```

---

cl_lsgroup 命令

用途

显示 PowerHA SystemMirror 集群上存在的组的属性。

注： 与特定标志关联的参数必须在紧邻标志的后面指定。

语法

```
cl_lsgroup [-cspoc "[-f] -g ResourceGroup | -n Nodelist"]
[-c|-f] [-a | -a List ] {ALL | Group [ ,Group] ... }
```

标志

-cspoc
用于指定以下 C-SPOC 选项的参数：

-f - 与 **cl_lsgroup** 命令一起使用时，此选项没有任何效果。

-g *ResourceGroup* - 生成节点的列表，这些节点加入了将要在其中执行此命令的资源组。

-n *nodelist* - 在此节点列表上运行命令。如果有一个以上的节点，请用逗号分隔列出的节点。

-a List
指定要显示的属性。*List* 参数可以包括定义在 **chgroup** 命令中的所有属性，属性之间需要一个空格。如果您通过仅使用 -a 标志指定了一个空列表，那么将仅列出组名。

-c 以冒号隔开的记录形式显示每组属性。如下所示：

```
#name:attribute1:attribute2: ...

Group:  value1:      value2:      ...
```

-f 以节形式显示组属性。每节由组名定义。每个 Attribute = Value 对在单独的行上列出：

```
group:
      attribute1=value
      attribute2=value
      attribute3=value
```

ALL | group [group]...
要显示的所有资源组或者一个或多个特定组。

示例

1. 要显示所有集群节点中 finance 组的属性，请输入：

   ```
   cl_lsgroup finance
   ```

2. 要以节格式显示所有集群节点中 finance 组的标识、成员 (users) 和管理员 (adms)，请输入：

   ```
   cl_lsgroup -f -a id users adms finance
   ```

3. 要以冒号分隔的格式显示所有集群节点中所有组的属性，请输入：

   ```
   cl_lsgroup -c ALL
   ```

---

cl_lslv 命令

用途

显示共享逻辑卷属性。

注：与特定标志关联的参数必须在紧邻标志的后面指定。

语法

```
cl_lslv [-cspoc "[-f] [-g ResourceGroup | -n Nodelist "]
] [-l | -m] LogicalVolume
```

标志

-cspoc
用于指定以下其中一个 C-SPOC 选项的参数：

-f - 与 **cl_lsfs** 命令一起使用时，此选项没有任何效果。

-g *ResourceGroup* - 生成节点的列表，这些节点加入了将要在其中执行此命令的资源组。

-n *Nodelist* - 在此节点列表上运行命令。如果有一个以上的节点，请用逗号分隔列出的节点。

-l *Logical Volume*
列示共享逻辑卷中的每个物理卷的信息。有关显示的字段的信息，请参阅 **lslv** 命令。

-m *Logical Volume*
列出每个逻辑分区的信息。有关显示的字段的信息，请参阅 **lslv** 命令。如果未指定任何标志，那么将显示有关共享逻辑卷及其底层共享卷组的信息。请参阅 **lslv** 命令，以获取有关显示的字段的信息。

示例

1. 要显示有关共享逻辑卷 *lv03* 的信息，请输入：

   ```
   cl_lslv -cspoc -g resource_grp1 lv03
   ```

   会显示关于逻辑卷 *lv03*、它的逻辑和物理分区及其所属卷组的信息。
2. 要使用标识显示特定逻辑卷的有关信息，请输入：

   ```
   cl_lslv -g resource_grp1 00000256a81634bc.2
   ```

   会显示此逻辑卷的所有可用特征和状态。

---

cl_lsuser 命令

用途

显示 PowerHA SystemMirror 集群上存在的用户的用户帐户属性。

注：与特定标志关联的参数必须在紧邻标志的后面指定。

语法

```
cl_lsuser [-cspoc "[-f] [-g ResourceGroup | -n Nodelist]"]
 [-c | -f] [-a List ] {ALL | Name [ ,Name ]... }
```

标志

-cspoc
用于指定以下 C-SPOC 选项的参数：

-f - 与 **cl_lsuser** 命令一起使用时，此选项没有任何效果。

-g *ResourceGroup* - 生成节点的列表，这些节点加入了将要在其中执行此命令的资源组。

-n *Nodelist* - 在此节点列表上运行命令。如果有一个以上的节点，请用逗号分隔列出的节点。

-a *Lists*
指定要显示的属性。List 变量可以包含在 **chuser** 命令中定义的任何属性并要求在属性间有一个空格。如果指定空列表，那么将仅显示用户名。

-c 以冒号隔开记录显示用户属性，如下：

```
# name:  attribute1:  attribute2:  ...
  User:  value1:      value2:      ...
```

-f 以节格式显示输出，每一节以用户名标识。每个 Attribute = Value 对在单独的行上列出：

```
user:
      attribute1=value
      attribute2=value
      attribute3=value
```

ALL | Name [name]...
显示所有用户或指定的一个或多个用户的信息。

示例

1. 要以节格式显示有关所有集群节点中 *smith* 帐户的用户标识和组的相关信息，请输入：

   ```
   cl_lsuser -fa id pgrp groups admgroups smith
   ```

2. 要以缺省格式显示所有集群节点中用户 *smith* 的所有属性，请输入：

   ```
   cl_lsuser smith
   ```

3. 要显示集群上所有用户的所有属性，请输入：

   ```
   cl_lsuser ALL
   ```

---

cl_lsvg 命令

用途

显示有关共享卷组的信息。

注：与特定标志关联的参数必须在紧邻标志的后面指定。

语法

cl_lsvg [-cspoc "[-f] [-g *ResourceGroup* | *n- Nodelist*]" [-o] |[-l | -M | -p] *Volume Group...*INFO HERE

标志

-cspoc
用于指定其中一个选项的参数

-f - 与 **cl_lsvg** 命令一起使用时，此选项没有任何效果。

-g *ResourceGroup* - 指定资源组（其参与节点共享卷组）的名称。命令在这些节点上执行。

-n *Nodelist* - 在此节点列表上运行命令。如果有一个以上的节点，请用逗号分隔列出的节点。

-p 列出通过 **VolumeGroup** 参数指定的组内每个物理卷的以下信息：

- Physical volume：组中的物理卷。

- PVstate：卷组的状态。

- **Total PPs**：物理卷上物理分区的总数。

- Free PPs：物理卷上空闲物理分区的数量。

- **Distribution**：分配在物理卷每个区域内的物理分区的数目，这些区域包括：物理卷的外部边缘、外部中间、中间、中心和内部边缘。

-l 列出由 *VolumeGroup* 参数指定的组内每个逻辑卷的以下信息：

- LV：卷组中的逻辑卷。

- **Type**：逻辑卷类型。
- **LPs**：逻辑卷中逻辑分区的数量。
- **PPs**：逻辑卷使用的物理分区的数量。
- **PVs**：逻辑卷使用的物理卷的数量。

-M 列出物理卷中的每个逻辑卷的以下字段：

- **PVname:** *PPnum* **[** *LVname* **:** *LPnum* **[:** *Copynum* **] [** *PPstate* **]]**
- **PVname**：系统所指定的物理卷的名称。
- **PPnum**：物理分区编号。物理分区号范围为 1 到 1016。

-o 仅列出活动的卷组（那些变化的卷组）。一个活动的卷组是可以使用的卷组。请参阅 **lsvg** 命令，以获取在未指定任何标志的情况下显示的信息。

示例

1. 要显示集群中所有共享卷组的名称，请输入：

```
cl_lsvg
nodeA: testvg
nodeB: testvg
```

2. 要显示集群中所有活动共享卷组的名称，请输入：

```
cl_lsvg -o
nodeA: testvg
```

3. 要显示有关共享卷组 *vg02* 的信息，请输入：

```
cl_lsvg -cspoc testvg
```

---

cl_nodecmd 命令

用途

在给定的一组节点上以并行方式运行给定命令。

语法

```
cl_nodecmd [-q] [-cspoc "[-f] [-n nodelist | -g resourcegroup ]"
] command args
```

标志

-q 指定静默方式。将禁止所有标准输出。

-cspoc
用于指定以下其中一个 C-SPOC 选项的参数：

-f - 强制 **cl_nodecmd** 跳过 PowerHA SystemMirror 版本兼容性检查和节点可访问性验证。

-g *resource group* - 生成节点的列表，这些节点加入了将要在其中执行此命令的资源组。

-n *nodelist* - 在此节点列表上运行命令。如果有一个以上的节点，请用逗号分隔列出的节点。

command
指定要在节点列表中的所有节点上运行的命令。

args
指定要传递至 **cl_nodecmd** 命令的自变量。

示例

1. 在所有集群节点上运行 **lspv** 命令。

   ```
   cl_nodecmd lspv
   ```

2. 在节点 *beaver* 和 *dam* 上运行 **lsvg rootvg** 命令，禁止标准输出。

   ```
   cl_nodecmd -cspoc "-n beaver,dam" lsvg rootvg
   ```

---

cl_rc.cluster 命令

用途

设置操作系统环境并跨集群节点启动集群守护程序。

语法

```
cl_rc.cluster [-cspoc "[-f] [-g ResourceGroup | -nNodeList ]"] [-boot]
[b] [-i | I] [-N | -R | -B] [-M | -A] [-x] [-r] [-v] [-C interactive|yes]
```

注：与特定标志关联的参数必须在紧邻标志的后面指定。

标志

-cspoc
用于指定以下 C-SPOC 选项的参数：

-f - 强制 **cl_rc.cluster** 跳过 PowerHA SystemMirror 版本兼容性检查和节点可访问性验证。

-g *ResourceGroup* - 指定资源组（其参与节点共享卷组）的名称。命令在这些节点上执行。

-n *Nodelist* - 跨节点列表中的节点来执行底层 AIX 命令。

-boot
已启用 IPAT 时，将服务网络接口配置为在其引导地址上。

-i 使用缺省选项来启动集群信息 (**clinfoES**) 守护程序。

-I 在启用陷阱的情况下启动集群信息 (**clinfoES**) 守护程序。

-b 广播启动。

-N
立即启动守护程序（无 **inittab** 更改）。

-R 仅在系统重新启动时启动 PowerHA SystemMirror 守护程序（PowerHA SystemMirror 启动命令将添加到 **inittab** 文件）。

-B 立即启动守护程序并将 PowerHA SystemMirror 条目添加到 **inittab** 文件。

-C 指定要用于在发生问题时执行更正操作的方式。指定 **yes** 以自动更正问题。指定 **interactive** 以在运行每个更正操作前显示提示。

-M 以手动资源获取方式来启动集群服务。如果您希望手动使资源组联机，请使用此选项。

-A 以自动资源获取方式来启动集群服务。如果您希望在集群启动时自动将资源组联机，请使用此选项。这是缺省选项。

-f 强制启动。集群守护程序应该对正在运行的本地过程进行初始化。

-r 在强制关闭之后重新获取集群资源。如果您在集群强制关闭期间更改了任何集群资源（IP 标签、磁盘和应用程序）的状态，请使用此选项。

-v 在启动期间忽略验证错误 (auto ver sync)

| **-x** 激活 NFS 交叉装配。

示例

1. 要启动集群并且在所有集群节点上运行 **clinfo**，请运行以下命令：

   ```
   cl_rc.cluster -boot -i
   ```

2. 要启动集群并且在其启用了陷阱的所有集群节点上运行 **clinfo**，请运行以下命令：

   ```
   cl_rc.cluster -boot -I
   ```

---

clconvert_snapshot 命令

用途

此命令将 snapshot_file 中先前版本 ODM 数据复制为新版本 ODM 结构的格式。

语法

```
clconvert_snapshot -v release -s < snapshotfile >
```

描述

您可以运行 **clconvert_snapshot** 以将集群快照从先前版本的 PowerHA SystemMirror 升级到最新版本的 PowerHA SystemMirror。缺省情况下，此命令假定您要转换为最新版本的软件。

如果新版本中删除了某些字段，那么数据将保存到 */tmp/cl_convert_PowerHA SystemMirror_OLD*。然后，此命令确保数据对于新版本具有正确的格式。

一旦快照文件已升级，将为其分配一个与先前版本中相同的名称，并且无法恢复为先前版本。将为您保存旧版本快照的副本，名称为原始名称加上扩展名 *.old*。

您必须处于生成快照的同一节点上的 **/usr/es/sbin/cluster/conversion** 目录中才能运行 clconvert_snapshot 命令。

一旦集群快照已升级，并且集群中的所有节点都安装了当前级别，那么便可以应用升级的快照，并且随后可以将集群联机。

脚本 clconvert_snapshot 创建旧版本的 ODM，并使用由用户提供的快照文件中的值来填充这些 ODM。然后，它调用 *cl_convert* 所使用的同一命令来将这些 ODM 转换为当前版本。将从升级的 ODM 中获取新快照，并且将新快照复制到用户提供的快照文件。

在安装期间，**clconvert_snapshot** 不会自动运行，并且必须始终从命令行中运行。

表 *1. clconvert_snapshot* 标志

标志	描述
-v	发行版版本标志。指定要在其中执行转换的发行版号。 **要点：**除非您知道要作为转换源的版本，否则请勿使用 clconvert_snapshot 命令。
-s	快照文件标志。指定要转换的快照文件。如果不为快照文件指定路径，那么命令将使用 **$SNAPSHOTPATH** 变量中的指定的路径。缺省值为 **/usr/es/sbin/cluster/snapshots**。

示例

运行以下命令，以将 PowerHA SystemMirror 5.3 快照转换为名为"mysnapshot"的当前 PowerHA SystemMirror 快照。

```
clconvert_snapshot -v 5.3 -s mysnapshot
```

随后将文件"mysnapshot"放入 **$SNAPSHOTPATH** 环境变量所指定的目录中。如果未指定 **$SNAPSHOTPATH** 变量，那么会将此文件放入 **/usr/es/sbin/cluster/snapshots** 中。

---

clcheck_server 命令

用途

返回 PowerHA SystemMirror 集群中的守护程序的状态。

语法

```
clcheck_server daemon
```

描述

clcheck_server 命令返回所指定守护程序的状态。此命令适合在需要可靠确定守护程序状态的 shell 脚本中使用。此命令完成 **lssrc** 命令（由系统资源控制器 (SRC) 提供）不会进行的额外检查。

使用 **clcheck_server** 命令之前，必须了解要检查的守护程序的用途。

标志

daemon
指定要检查的守护程序的名称。

示例

要检查 clinfo 守护程序的状态，请输入：

```
    if ! clcheck_server clinfoES
    then
 echo "clinfo is active"
    else
 echo "clinfo is inactive"
    fi
```

---

clfindres 命令

用途

在集群配置中查找给定的一个或多个资源组。

语法

```
clfindres [-s] [resgroup1] [resgroup2]...
```

描述

在您运行 **clfindres** 时，它将调用 **clRGinfo**，并且 **clfindres** 的命令输出与 **clRGinfo** 命令的命令输出相同。因此，请使用 **clRGinfo** 命令来查找资源组的状态和位置。**clfindres** 命令的 **-s** 标志请求简略（仅包含位置）的输出。请参阅 **clRGinfo** 命令以获取更多信息。

clgetactivenodes 命令

用途

检索所有集群节点的名称。

语法

clgetactivenodes [-n *nodename*] [-o *odmdir*] [-t*timeout*] [-v *verbose*]

表 2. *clgetactivenodes* 标志

标志	描述
-n nodename	确定指定节点是否处于活动状态。
-o odmdir	将 *odmdir*（而非缺省的 **/etc/objrepos**）指定为 ODM 对象存储库目录。
-t timeout	指定在接收有关活动节点的信息时的最大时间间隔。
-v verbose	指定有关要显示为详细输出的活动节点的信息。

示例

运行以下命令以验证节点 *java* 是否处于活动状态。

```
clgetactivenodes -n java
```

clgetaddr 命令

用途

返回指定节点名的可成功进行 ping 操作的地址。

语法

clgetaddr [-o *odmdir*] *nodename*

-o 指定备用 ODM 目录。

示例

要获取节点 *seaweed* 的可 ping 通地址，请输入：

```
clgetaddr seaweed
```

将返回以下地址：2361059035

cli_assign_pvids 命令

用途

向每个磁盘分配 PVID（作为参数传递），然后使用这些 PVID 更新所有其他集群节点。

语法

cli_assign_pvids PhysicalVolume ...

描述

逻辑卷管理器 (LVM) 向列表中的每个物理卷分配 PVID（如果它们还没有 PVID），然后使这些 PVID 在所有集群节点上被识别。

示例

要向磁盘列表分配 PVID 并让这些 PVID 在集群中被识别，请输入：

```
cli_assign_pvids hdisk101 hdisk102 hdisk103
```

---

cli_chfs 命令

用途

更改集群中所有节点上的文件系统的属性。

语法

```
cli_chfs [ -m NewMountPoint ] [ -u MountGroup ] [ -p { ro | rw  } ]
         [ -t { yes | no } ] [ -a Attribute=Value ] [ -d Attribute ]
   FileSystem
```

描述

使用 C-SPOC 运行带有指定参数的 **chfs** 命令，并在所有集群节点上更新文件系统定义。

标志

-d Attribute
从 /etc/filesystems 文件为指定的文件系统删除指定的属性。

-m NewMountPoint
为指定的文件系统指定新的安装点。以下值有效：

-p 设置文件系统的许可权。以下值有效：

ro 指定只读许可权。

rw
指定读写许可权。

-t 为指定的文件系统设置记帐属性。以下值有效：

是 文件系统记帐由记帐子系统处理。

no 文件系统记帐并非由记帐子系统处理。这是缺省值。

-u MountGroup
指定安装组。安装组用于组合相关安装以便它们可作为一个组进行安装，而不是分别安装。例如，执行某些测试时，如果需要将一些临时文件系统安装在一起，那么可将它们都放在测试安装组中。可使用单个命令（例如，**mount -t** 命令）安装此安装组。

-a Attribute=Value
指定依赖于虚拟文件系统类型的 Attribute=Value 对。要指定多个 Attribute=Value 对，请提供多个 -a Attribute=Value 参数。

示例

要更改名为 */test_fs* 的共享文件系统的大小，请输入：

```
cli_chfs -a size=32768 /test_fs
```

相关信息:

cli_chlv 命令

用途

更改集群中所有节点上的逻辑卷的属性。

语法

```
cli_chlv  [-a Position] [-b BadBlocks] [-d Schedule] [-e Range]
          [-L label] [-p Permission] [-r Relocate] [-s Strict]
          [-t Type] [-u Upperbound] [-v Verify] [-w MirrorWriteConsistency]
          [-x Maximum] [-U userid] [-G groupid] [-P modes] LogicalVolume
```

描述

使用 C-SPOC 运行带有指定参数的 **chlv** 命令，并在所有集群节点上更新逻辑卷定义。

标志

-a Position
设置物理卷分配策略（物理卷上逻辑分区的位置）。以下 *Position* 变量有效：

- **m** 在每个物理卷的外部中间扇区内分配逻辑分区。此变量为缺省设置。
- **c** 在每个物理卷的中间扇区内分配逻辑分区。
- **e** 在每个物理卷的外部边缘部分来分配逻辑分区。
- **ie** 在每个物理卷的内部边缘段内分配逻辑分区。
- **im** 在每个物理卷的内部中间部分来分配逻辑分区。

-b BadBlocks
设置坏区重定位策略。以下 *BadBlocks* 变量有效：

- **y** 导致坏区重定位发生。
- **n** 防止坏区重定位发生。

-d Schedule
写入多个逻辑分区时，设置调度策略。必须使用并行处理或顺序处理来镜像带区逻辑卷。以下 *Schedule* 变量有效：

- **p** 建立并行调度策略。
- **ps** 使用顺序读策略进行并行写入。所有镜像并行写入，但始终读取第一个可用镜像。
- **pr** 针对所有镜像完成并行写入和读取。此策略与并行策略类似，只是尝试更均衡地将针对逻辑卷的读取分布到所有镜像间。
- **s** 建立顺序调度策略。如果指定严格并行或顺序（超级严格）策略，请使用此变量。

-e Range
设置物理卷分配策略。分配策略是使用提供最佳分配的卷以便在其中进行扩展的物理卷数。*Range* 变量的值受使用 **-u** 标志设置的 *Upperbound* 变量限制。以下 *Range* 变量有效：

- **x** 覆盖最大数目的物理卷分配逻辑分区。
- **m** 根据最小物理卷数量分配逻辑分区。

-G Groupid
为逻辑卷特别文件指定组标识。

-L Label
设置逻辑卷标号。此变量的最大大小为 127 个字符。

-n NewLogicalVolume
更改 *NewLogicalVolume* 变量指定的逻辑卷的名称。逻辑卷名称在系统范围内必须是唯一的，并且最多可有 15 个字符。

-p Permission
将访问许可权设置为读/写或只读。以下 *Permission* 变量有效：

w 将访问许可权设置为读/写。

r 将访问许可权设置为只读。不支持在只读逻辑卷上安装 JFS 文件系统。

-P Modes
为逻辑卷特殊文件指定许可权（文件方式）。

-r Relocate
指定您是想允许还是阻止在重组期间重定位逻辑卷。以下 *Relocate* 变量有效：

y 允许在重组织过程中重定位逻辑卷。如果逻辑卷被分割，那么不能使用 **chlv** 命令将重定位标志更改为 y。

n 防止在重组织过程中重定位逻辑卷。

-s Strict
确定严格的分配策略。对于同一物理卷，可分配逻辑分区副本以共享或不共享。以下 *Strict* 变量有效：

y 设置严格的分配策略，导致逻辑分区副本不能共享同一物理卷。

n 不设置严格的分配策略，导致逻辑分区副本可共享同一物理卷。

s 设置一个超级严格的分配策略，导致为一个镜像分配的分区不能与另一个镜像中的分区共享同一物理卷。将非超级严格逻辑卷更改为超级严格逻辑卷时，必须使用 **-u** 标志。

-t Type
设置逻辑卷类型。最大大小为 31 个字符。如果逻辑卷被分割，那么不能将 *Type* 变量更改为 boot。

-U Userid
指定逻辑卷特殊文件的用户标识。

-u Upperbound
设置新分配的最大物理卷数。*Upperbound* 变量的值介于 1 到物理卷总数之间。如果使用超级严格策略，那么上限指示允许对每个镜像副本使用的最大物理卷数。如果使用带区逻辑卷，那么上限必须为 *Stripe_width* 变量的倍数。

-v Verify
设置逻辑卷的写验证状态。导致所有对逻辑卷的写入进行后续读取验证或不进行后续读取验证。以下 *Verify* 变量有效：

y 对逻辑卷的所有写入使用后续读取进行验证。

n 对逻辑卷的所有写入不使用后续读取进行验证。

-w MirrorWriteConsistency
以下 *MirrorWriteConsistency* 变量有效：

y 启用主动镜像写一致性。此变量验证常规 I/O 处理期间逻辑卷的镜像副本上的数据一致性。

p 启用被动镜像写一致性。系统中断后，此变量验证卷组同步期间镜像副本上的数据一致性。此功能仅在大型卷组上可用。

n 没有镜像写一致性。

-x Maximum
设置可分配给逻辑卷的最大逻辑分区数。每个逻辑卷的最大逻辑分区数为 32,512。

示例

要更改名为 *lv01* 的逻辑卷的物理卷分配，请输入：

```
cli_chlv -e m lv01
```

相关信息:

chlv 命令

---

cli_chvg 命令

用途

更改集群中所有节点上的卷组的属性。

语法

```
cli_chvg  [ -s Sync { y | n }] [ -L LTGSize ] [ -Q { n | y } ] [ -u ]
    [ -t [factor ] ] [ -B ] [ -C  ]   VolumeGroup
```

描述

使用 C-SPOC 运行带有指定参数的 **chvg** 命令，并使已更新的卷组定义在所有集群节点上可用。

标志

-B 将卷组更改为大的 VG 格式。此标志最多可累积 128 个物理卷和 512 个逻辑卷。如果集群节点中有任何旧物理分区，那么不能使用此标志。要使用此标志，每个物理卷上必须有足够可用分区进行 VGDA 扩展。

由于 VGDA 驻留在磁盘的边缘，并且它需要邻接空间用于扩展，所以在磁盘的边缘上需要具有可用分区。如果这些分区已被分配以供应用程序数据使用，那么它们将迁移至同一磁盘上的其他可用分区。对其余的物理分区重新编号，以反映由于 VGDA 使用而造成的分区丢失。此过程会更改卷组中所有物理卷上的逻辑分区至物理分区映射。

如果保存逻辑卷映射以用于将来的潜在恢复操作，那么您可在完成转换操作后再次生成映射。如果进行卷组备份时带有映射选项，并且您计划使用这些映射进行复原，那么复原操作可能失败，因为分区编号可能不存在（因为减少）。建议在启动转换过程之前备份逻辑卷，如果您使用映射选项，那么转换过程完成后也应立即进行备份。

因为 VGDA 空间大幅增长，所以每个 VGDA 更新操作（创建逻辑卷、更改逻辑卷、添加物理卷等等）可能需要更长时间运行。

-C 将卷组更改为增强的支持并行的卷组。将卷组从非并行方式（联机）更改为增强并行方式。此过程要求激活增强并行方式之前在所有其他节点上重新导入该卷组。将卷组从非并行方式（联机）更改为增强并行方式。只能在 PowerHA SystemMirror 集群中使用此标志，并且必须在激活增强并行卷组前配置该集群。

-L LTGSize
将逻辑磁道组大小设置为磁盘的通用最大传输大小（如果卷组已联机）。*LTGSize* 变量的值必须为

0、128、256、512 或 1024。*LTGSize* 变量必须小于或等于卷组中所有磁盘的最大传输大小。*LTGSize* 变量的缺省值为 128。如果您指定 *LTGSize* 为 0，那么 **varyonvg** 命令会将逻辑磁道组大小设置为磁盘的通用最大传输大小。

-Q 确定丢失物理卷的限额后卷组是否自动脱机。缺省值为 yes。此更改在卷组下次激活时生效。以下值有效：

y 卷组在丢失其物理卷限额后自动脱机。

n 卷组保持活动状态直到丢失其所有物理卷。

-s Sync

设置 *VolumeGroup* 变量指定的卷组的同步特征。此标志不影响非镜像逻辑卷。支持并行的卷组不支持此标志。

自动同步是一种恢复机制，仅应在逻辑卷管理器 (LVM) 设备驱动程序在 AIX 操作系统错误日志中记录 LVM_SA_STALEPP 之后尝试。通过任何其他途径（例如，**mklvcopy** 命令）变旧的分区不会自动重新同步。以下值有效：

y 尝试自动同步旧分区。

n 禁止旧文件分区的自动同步。此值为卷组的缺省设置。

-t factor

更改每个物理卷的物理分区数限制（由因子指定）。对于包含 32 个物理卷的组，此因子必须为 1 到 16。对于包含 128 个物理卷的组，此因子必须为 1 到 64。

如果未指定此因子，那么它设置为最低值，以便卷组中的最大磁盘的物理分区数小于因子值乘以 1016。

如果指定了因子，那么卷组的每个物理卷的最大物理分区数更改为该因子值乘以 1016。

确定因子值时查看以下信息：

- 对于可伸缩类型的卷组，此标志被忽略。
- 如果卷组以并行方式联机，那么不能使用此标志。
- 如果卷组中有任何旧物理分区，那么不能更改因子值。
- 此卷组中允许的最大物理卷数将减至您使用 MAXPVS 除以因子值 (MAXPVS/factor) 时获取的值。
- 将现有卷组更改为可伸缩卷组格式会将所有关联逻辑卷的设备子类型（由 IOCINFO ioctl() 调用进行报告）更改为 *DS_LVZ*，不管之前子类型如何都是如此。此更改不会改变所报告子类型之类的逻辑卷的任何行为。

-u 解锁卷组。如果逻辑卷组因为另一 LVM 操作的异常终止（例如，命令核心转储或系统崩溃）而保持锁定状态，那么此标志可用。使用此标志之前，必须验证该卷组是否未被另一 LVM 命令使用。

示例

要关闭名为 *vg01* 的卷组的定额，请输入：

```
cli_chvg -Q n vg01
```

相关信息:

chvg 命令

---

cli_crfs 命令

用途

创建新文件系统并使其在集群中的所有节点上可用。

语法

```
cli_crfs  -v VfsType { -g VolumeGroup | -d Device } [ -l LogPartitions ]
    -m MountPoint [ -u MountGroup ] [ -A { yes | no } ]
    [ -p {ro | rw } ] [ -a Attribute=Value ... ] [ -t { yes | no } ]
```

描述

使用 C-SPOC 运行带有指定参数的 **crfs** 命令，并使已更新的文件系统定义在所有集群节点上可用。

标志

-a Attribute=Value
指定依赖于虚拟文件系统的属性和值对。要指定多个属性/值对，请提供多个 -a Attribute=Value 参数。

-d Device
指定在其上创建文件系统的设备或逻辑卷的设备名。此标志用于在现有逻辑卷上创建文件系统。

-g VolumeGroup
指定在其上创建文件系统的现有卷组。卷组是一个或多个物理卷的集合。

-l LogPartitions
指定日志逻辑卷大小（表示为逻辑分区数）。此标志仅适用于没有日志设备的 JFS 和 JFS2 文件系统。

-m MountPoint
指定安装点，即在其中提供文件系统的目录。如果指定相对路径名，那么它会先转换为绝对路径名，然后再插入至 /etc/filesystems 文件。

-p 设置文件系统的许可权。

ro 只读许可权

rw
读/写许可权

-t 指定文件系统是否由记帐子系统处理。以下值有效：

是 已在文件系统上启用记帐。

no 未在文件系统上启用记帐。此值为缺省设置。

-u MountGroup
指定安装组。

-v VfsType
指定虚拟文件系统类型。*agblksize* 属性是在创建文件系统时设置的，创建文件系统后不能更改。*size* 属性用于定义最小文件系统大小。创建文件系统后，不能使用 *size* 属性降低文件系统大小。

示例

要在名为 *lv01* 的现有逻辑卷上创建 JFS 文件系统，请输入：

```
cli_crfs -v jfs -d lv01 -m /tstvg -a 'size=32768'
```

相关信息:

crfs 命令

cli_crlvfs 命令

用途

创建新逻辑卷和文件系统并使其在集群中的所有节点上可用。

语法

```
cli_crlvfs -v VfsType -g VolumeGroup  [ -l LogPartitions ] -m MountPoint
        [ -u MountGroup ] [ -A { yes | no } ] [ -p {ro | rw } ]
  [ -a Attribute=Value ... ] [ -t { yes | no } ]
```

描述

使用 C-SPOC 运行带有指定参数的 **crfs** 命令，并使已更新的文件系统定义在所有集群节点上可用。

标志

-a Attribute=Value
指定依赖于虚拟文件系统的属性和值对。要指定多个属性/值对，请提供多个 -a Attribute=Value 参数。

-g VolumeGroup
指定在其上创建文件系统的现有卷组。卷组是一个或多个物理卷的集合。

-l LogPartitions
指定日志逻辑卷大小（表示为逻辑分区数）。此标志仅适用于没有日志设备的 JFS 和 JFS2 文件系统。

-m MountPoint
指定安装点，即在其中提供文件系统的目录。如果指定相对路径名，那么它会先转换为绝对路径名，然后再插入至 /etc/filesystems 文件。

-p 设置文件系统的许可权。

ro 只读许可权

rw
读/写许可权

-t 指定文件系统是否由记帐子系统处理。以下值有效：

是 已在文件系统上启用记帐。

no 未在文件系统上启用记帐。此值为缺省设置。

-u MountGroup
指定安装组。

-v VfsType
指定虚拟文件系统类型。*agblksize* 属性是在创建文件系统时设置的，创建文件系统后不能更改。*size* 属性用于定义最小文件系统大小。创建文件系统后，不能使用 *size* 属性降低文件系统大小。

示例

要在名为 *vg01* 的卷组上创建 JFS 文件系统，请输入：

```
cli_crlvfs -v jfs -g vg01 -m /tstvg -a 'size=32768'
```

cli_extendlv 命令

用途

通过添加卷组内未分配的物理分区来增加所有节点上的逻辑卷大小。

语法

```
cli_extendlv [ -a Position ] [ -e Range ] [ -u Upperbound ] [ -s Strict ]
             LogicalVolume Partitions [ PhysicalVolume ... ]
```

描述

使用 C-SPOC 运行带有指定参数的 **extendlv** 命令，并使已更新的逻辑卷定义在所有集群节点上可用。

标志

-a Position
设置物理卷分配策略（物理卷上逻辑分区的位置）。以下 *Position* 变量有效：

- **m** 在每个物理卷的外部中间扇区内分配逻辑分区。此变量为缺省设置。
- **c** 在每个物理卷的中间扇区内分配逻辑分区。
- **e** 在每个物理卷的外部边缘部分来分配逻辑分区。
- **ie** 在每个物理卷的内部边缘段内分配逻辑分区。
- **im** 在每个物理卷的内部中间部分来分配逻辑分区。

-e Range
设置物理卷分配策略。分配策略是使用提供最佳分配的卷以便在其中进行扩展的物理卷数。*Range* 变量的值受使用 **-u** 标志设置的 *Upperbound* 变量限制。以下 *Range* 变量有效：

- **x** 覆盖最大数目的物理卷分配逻辑分区。
- **m** 根据最小物理卷数量分配逻辑分区。

-s Strict
确定严格的分配策略。对于同一物理卷，可分配逻辑分区副本以共享或不共享。以下 *Strict* 变量有效：

- **y** 设置严格的分配策略，导致逻辑分区副本不能共享同一物理卷。
- **n** 不设置严格的分配策略，导致逻辑分区副本可共享同一物理卷。
- **s** 设置一个超级严格的分配策略，导致为一个镜像分配的分区不能与另一个镜像中的分区共享同一物理卷。将非超级严格逻辑卷更改为超级严格逻辑卷时，必须使用 **-u** 标志。

-u Upperbound
设置新分配的最大物理卷数。*Upperbound* 变量的值介于 1 到物理卷总数之间。如果使用超级严格策略，那么上限指示允许对每个镜像副本使用的最大物理卷数。如果使用带区逻辑卷，那么上限必须为 *Stripe_width* 变量的倍数。

示例

要将名为 *lv01* 的逻辑卷的大小增大 3 个逻辑分区，请输入：

```
cli_extendlv lv01 3
```

相关信息:

extendlv 命令

cli_extendvg 命令

用途

在集群中的所有节点上向卷组添加物理卷。

语法

```
cli_extendvg VolumeGroup PhysicalVolume ...
```

描述

使用 C-SPOC 运行带有指定参数的 **extendvg** 命令，并使已更新的卷组定义在所有集群节点上可用。

必须验证要包含的物理卷 (hdisk) 是否在所有集群节点可用并且在运行此命令前是否对其分配了 PVID。

示例

要将名为 *hdisk101* 和 *hdisk111* 的磁盘添加至名为 *vg01* 的卷组，请输入：

```
cli_extendvg vg01 hdisk101 hdisk111
```

相关信息:

extendvg 命令

cli_importvg 命令

用途

从集群中所有节点上的一组物理卷导入新的卷组定义

语法

```
cli_importvg [ -y VolumeGroup ] [ -V MajorNumber ]  PhysicalVolume
```

描述

使用 C-SPOC 运行带有指定参数的 **importvg** 命令。此命令导致每个集群节点上的逻辑卷管理器 (LVM) 读取卷组中的磁盘上的 LVM 信息，然后更新逻辑卷组定义。

标志

-V MajorNumber
指定所导入卷组的主编号。

-y VolumeGroup
指定要用于新卷组的名称。如果未使用此标志，那么系统会自动生成新名称。卷组名称只能包含以下字符：

- A - Z
- a - z
- 0 - 9
- _（下划线字符）
- -（减号字符）
- . （句点字符）

示例

为使名为 *hdisk07* 的物理卷中名为 *bkvg* 的卷组在所有集群节点上可用，请输入：

```
cli_importvg -y bkvg hdisk07
```

相关信息:

importvg 命令

---

cli_mirrorvg 命令

用途

语法

```
cli_mirrorvg  [-S | -s] [-Q] [-c Copies] [-m] VolumeGroup [PhysicalVolume...]
```

描述

使用 C-SPOC 运行带有指定参数的 **mirrorvg** 命令，并使已更新的卷组定义在所有集群节点上可用。

标志

-c Copies
指定在运行 **mirrorvg** 命令后每个逻辑卷必须具有的最小副本数。运行 **mirrorvg** 命令后，通过独立使用 **mklvcopy** 命令，某些逻辑卷具有的副本数可能超过所指定的最小副本数。您可指定的最小值和最大值分别为 2 和 3。值 1 被忽略。

-m exact map
允许按原始副本中的确切物理分区顺序镜像逻辑卷。必须指定确切映射副本所以的物理卷。如果确切映射的空间不足，那么该命令失败。必须添加新的驱动器，或选择满足整个卷组的确切逻辑卷映射的另一组驱动器。所指定磁盘必须等于或大于要镜像的驱动器的大小（不管是否使用整个磁盘）。如果任何逻辑卷已镜像，那么该命令失败。

-Q Quorum Keep
缺省情况下，镜像卷组内容时，卷组的定额被禁用。如果要在镜像完成后保持卷组定额需求，那么可使用此标志。有关后续定额更改，请参阅 **chvg** 命令。

-S Background Sync
立即返回 **mirrorvg** 命令并在后台对卷组启动 **syncvg** 命令。如果使用此标志，那么镜像完成同步的时间不明显。但是，镜像的部分开始同步时，逻辑卷管理器 (LVM) 立即将它们用于镜像。

-s Disable Sync
立即返回 **mirrorvg** 命令而不执行任何类型的镜像同步。如果使用此标志，那么逻辑卷的镜像可能存在，但在使用 **syncvg** 命令同步逻辑卷前，操作系统不会使用该逻辑卷。

示例

要在名为 *vg01* 的共享卷组中对每个逻辑卷指定两个副本，请输入：

```
cli_mirrorvg -c 2 vg01
```

相关信息:

mirrorvg 命令

cli_mklv 命令

用途

在集群中所有节点上创建新逻辑卷。

语法

```
cli_mklv [ -a Position ] [ -b BadBlocks ] [ -c Copies ]  [ -d Schedule ]
         [ -e Range ] [ -i ] [ -L Label ] [ -o y / n ] [ -r Relocate ]
         [ -s Strict ] [ -t Type ] [ -u UpperBound ] [ -v Verify ]
         [ -w MirrorWriteConsistency ] [ -x Maximum ] [ -y NewLogicalVolume |
         -Y Prefix ] [ -S StripSize ] [ -U Userid ] [ -G Groupid ] [ -P Modes ]
   VolumeGroup NumberOfLPs [ PhysicalVolume ... ]
```

描述

使用 C-SPOC 运行带有参数的 **mklv** 命令，并使新的逻辑卷定义在所有集群节点上可用。

标志

-a Position
设置物理卷分配策略（物理卷上逻辑分区的位置）。以下 *Position* 变量有效：

m 在每个物理卷的外部中间扇区内分配逻辑分区。此变量为缺省设置。

c 在每个物理卷的中间扇区内分配逻辑分区。

e 在每个物理卷的外部边缘部分来分配逻辑分区。

ie 在每个物理卷的内部边缘段内分配逻辑分区。

im 在每个物理卷的内部中间部分来分配逻辑分区。

-b BadBlocks
设置坏区重定位策略。以下 *BadBlocks* 变量有效：

y 导致坏区重定位发生。

n 防止坏区重定位发生。

-c Copies
指定在运行 **mirrorvg** 命令后每个逻辑卷必须具有的最小副本数。运行 **mirrorvg** 命令后，通过独立使用 **mklvcopy** 命令，某些逻辑卷具有的副本数可能超过所指定的最小副本数。您可指定的最小值和最大值分别为 2 和 3。值 1 被忽略。

-d Schedule
写入多个逻辑分区时，设置调度策略。必须使用并行处理或顺序处理来镜像带区逻辑卷。以下 *Schedule* 变量有效：

p 建立并行调度策略。

ps 使用顺序读策略进行并行写入。所有镜像并行写入，但始终读取第一个可用镜像。

pr 针对所有镜像完成并行写入和读取。此策略与并行策略类似，只是尝试更均衡地将针对逻辑卷的读取分布到所有镜像间。

s 建立顺序调度策略。如果指定严格并行或顺序（超级严格）策略，请使用此变量。

-e Range
设置物理卷分配策略。分配策略是使用提供最佳分配的卷以便在其中进行扩展的物理卷数。*Range* 变量的值受使用 **-u** 标志设置的 *Upperbound* 变量限制。以下 *Range* 变量有效：

x 覆盖最大数目的物理卷分配逻辑分区。

m 根据最小物理卷数量分配逻辑分区。

-G Groupid
为逻辑卷特别文件指定组标识。

-L Label
设置逻辑卷标号。此变量的最大大小为 127 个字符。

-n NewLogicalVolume
更改 *NewLogicalVolume* 变量指定的逻辑卷的名称。逻辑卷名称在系统范围内必须是唯一的，并且最多可有 15 个字符。

-p Permission
将访问许可权设置为读/写或只读。以下 *Permission* 变量有效：

w 将访问许可权设置为读/写。

r 将访问许可权设置为只读。不支持在只读逻辑卷上安装 JFS 文件系统。

-P Modes
为逻辑卷特殊文件指定许可权（文件方式）。

-r Relocate
指定您是想允许还是阻止在重组期间重定位逻辑卷。以下 *Relocate* 变量有效：

y 允许在重组织过程中重定位逻辑卷。如果逻辑卷被分割，那么不能使用 **chlv** 命令将重定位标志更改为 y。

n 防止在重组织过程中重定位逻辑卷。

-s Strict
确定严格的分配策略。对于同一物理卷，可分配逻辑分区副本以共享或不共享。以下 *Strict* 变量有效：

y 设置严格的分配策略，导致逻辑分区副本不能共享同一物理卷。

n 不设置严格的分配策略，导致逻辑分区副本可共享同一物理卷。

s 设置一个超级严格的分配策略，导致为一个镜像分配的分区不能与另一个镜像中的分区共享同一物理卷。将非超级严格逻辑卷更改为超级严格逻辑卷时，必须使用 -u 标志。

-S StripSize
指定每个条带单元的字节数（阵列中的磁盘数乘以条带单元大小等于条带大小）。有效值包括 4K、8K、16K、32K、64K、128K、256K、512K、1M、2M、4M、8M、16M、32M、64M 以及 128M。使用此标志创建带区逻辑卷时，不能使用 -d、-e 和 -s 标志。

-t Type
设置逻辑卷类型。以下是标准类型：

- jfs（日志文件系统）
- jfslog（日志文件系统日志）
- jfs2（增强日志文件系统）
- jfs2log（增强日志文件系统日志）
- paging（调页空间）

您可以使用此标志定义其他逻辑卷类型。无法创建类型为 boot 的带区逻辑卷。缺省值为 jfs。如果逻辑卷在创建时指定为 jfslog 或 jfs2log 类型，那么 C-SPOC 会自动运行 **logform** 命令以使可使用该逻辑卷。

-U Userid
指定逻辑卷特殊文件的用户标识。

-u Upperbound
设置新分配的最大物理卷数。*Upperbound* 变量的值介于 1 到物理卷总数之间。如果使用超级严格策略，那么上限指示允许对每个镜像副本使用的最大物理卷数。如果使用带区逻辑卷，那么上限必须为 *Stripe_width* 变量的倍数。

-v Verify
设置逻辑卷的写验证状态。导致所有对逻辑卷的写入进行后续读取验证或不进行后续读取验证。以下 *Verify* 变量有效：

y 对逻辑卷的所有写入使用后续读取进行验证。

n 对逻辑卷的所有写入不使用后续读取进行验证。

-w MirrorWriteConsistency
以下 *MirrorWriteConsistency* 变量有效：

y 启用主动镜像写一致性。此变量验证常规 I/O 处理期间逻辑卷的镜像副本上的数据一致性。

p 启用被动镜像写一致性。系统中断后，此变量验证卷组同步期间镜像副本上的数据一致性。此功能仅在大型卷组上可用。

n 没有镜像写一致性。

-x Maximum
设置可分配给逻辑卷的最大逻辑分区数。每个逻辑卷的最大逻辑分区数为 32,512。

-y NewLogicalVolume
指定要代替系统生成的名称使用的逻辑卷名称。逻辑卷名称在系统范围内必须唯一，可包含 1 到 15 个字符。新名称必须在定义卷组的所有节点间必须唯一。该名称不能以设备配置数据库中的预定义设备数据库 (PdDv) 类中对其他设备定义的前缀开头。

-Y Prefix
指定前缀值以替代新逻辑卷的系统生成名中的前缀。此前缀值不能超过 13 个字符。该名称不能以设备配置数据库中的预定义设备数据库 (PdDv) 类中对其他设备定义的前缀开头，并且不能是另一设备已在使用的名称。

示例

要在名为 *vg02* 的卷组中创建带有一个逻辑分区和总共两个数据副本的逻辑卷，请输入：

```
cli_mklv -c 2 vg01 1
```

相关信息:

mklv 命令

---

cli_mklvcopy 命令

用途

增加集群中所有节点的逻辑卷的每个逻辑分区中的副本数。

语法

```
cli_mklvcopy  [ -a Position ] [ -e Range ] [ -k ] [ -s Strict ]
  [ -u UpperBound ] LogicalVolume Copies [ PhysicalVolume... ]
```

描述

使用 C-SPOC 运行带有参数的 **mklvcopy** 命令，并使已更新的逻辑卷定义在所有集群节点上可用。

标志

-a Position
设置物理卷分配策略（物理卷上逻辑分区的位置）。以下 *Position* 变量有效：

- **m** 在每个物理卷的外部中间扇区内分配逻辑分区。此变量为缺省设置。
- **c** 在每个物理卷的中间扇区内分配逻辑分区。
- **e** 在每个物理卷的外部边缘部分来分配逻辑分区。
- **ie** 在每个物理卷的内部边缘段内分配逻辑分区。
- **im** 在每个物理卷的内部中间部分来分配逻辑分区。

-e Range
设置物理卷分配策略。分配策略是使用提供最佳分配的卷以便在其中进行扩展的物理卷数。*Range* 变量的值受使用 **-u** 标志设置的 *Upperbound* 变量限制。以下 *Range* 变量有效：

- **x** 覆盖最大数目的物理卷分配逻辑分区。
- **m** 根据最小物理卷数量分配逻辑分区。

-k 新分区中的同步数据。

-s Strict
确定严格的分配策略。对于同一物理卷，可分配逻辑分区副本以共享或不共享。以下 *Strict* 变量有效：

- **y** 设置严格的分配策略，导致逻辑分区副本不能共享同一物理卷。
- **n** 不设置严格的分配策略，导致逻辑分区副本可共享同一物理卷。
- **s** 设置一个超级严格的分配策略，导致为一个镜像分配的分区不能与另一个镜像中的分区共享同一物理卷。将非超级严格逻辑卷更改为超级严格逻辑卷时，必须使用 **-u** 标志。

-u Upperbound
设置新分配的最大物理卷数。*Upperbound* 变量的值介于 1 到物理卷总数之间。如果使用超级严格策略，那么上限指示允许对每个镜像副本使用的最大物理卷数。如果使用带区逻辑卷，那么上限必须为 *Stripe_width* 变量的倍数。

示例

要将物理分区添加至名为 *lv01* 的逻辑卷的逻辑分区以便每个逻辑分区存在总共三个副本，请输入：

```
cli_mklvcopy lv01 3
```

相关信息:

mklvcopy 命令

cli_mkvg 命令

用途

在集群中所有节点上创建卷组。

语法

```
cli_mkvg   [ -B ] [ -t factor ] [ -C ] [ -G ] [ -x ] [ -s Size ]
     [ -V MajorNumber ] [ -v LogicalVolumes ] [ -y VolumeGroup ]
     PhysicalVolume ...
```

描述

可使用 C-SPOC 运行带有参数的 **mkvg** 命令，并使新的逻辑卷定义在所有集群节点上可用。

标志

-B 创建大类型卷组。此类型的卷组最多可累积 128 个物理卷和 512 个逻辑卷。因为 vgda 空间大幅增长，所以每个 vgda 更新操作（创建逻辑卷、更改逻辑卷、添加物理卷等等）可能需要更长时间运行。

-C 创建增强的支持并行的卷组。只能在已配置 PowerHA SystemMirror 集群中使用此标志。可使用此标志创建增强的支持并行的卷组。

增强的并行卷组使用组服务。组服务是随 PowerHA SystemMirror 提供的，必须在以此方式激活卷组之前配置。

只有增强的支持并行的卷组才受 64 位内核支持。支持并行的卷组不受 64 位内核支持。

-p partitions

指定卷组中的分区总数。*partitions* 变量表示为 1024 分区单元数。以下值对此标志有效：

- 32
- 64
- 128
- 256
- 512
- 768
- 1024
- 2048

缺省值为 32k（32768 个分区）。可使用 **chvg** 命令将分区数增加至最多 2048 k（2097152 个分区）。此标志仅对 **-s** 标志有效。

-s size

设置每个物理分区中的兆字节 (MB) 数。size 变量表示为兆字节单元数，范围在 1 (1 MB) 到 131072 (128 GB) 之间。大小变量必须等于 2 的幂（例如，1、2、4 和 8）。32 个物理卷组和 128 个物理卷组的缺省值是在每个物理卷 1016 个物理分区的限制内的最低值。可伸缩卷组的缺省值为对每个物理卷累积 2040 个物理分区的最低值。

-t factor

更改每个物理卷的物理分区数限制（由因子指定）。对于包含 32 个物理卷的组，此因子必须为 1 到 16。对于包含 128 个物理卷的组，此因子必须为 1 到 64。此卷组的每个物理卷的最大物理分区数更改为该因子值乘以 1016。缺省值是要保留在 factor x 1016 个物理分区限制内的最低值。可包含在卷组中的最大物理卷数为 maxpvs/factor。如果使用 **-s** 标志，那么此标志被忽略。

-V majornumber
指定要创建卷组的主号码。

-v 指定可创建的逻辑卷数。以下值对此标志有效：

- 256
- 512
- 1024
- 2048
- 4096

缺省值为 256。可使用 **chvg** 命令将逻辑卷数增加至最多 4096。此标志仅对 **-s** 标志有效。保留最后一个逻辑卷供元数据使用。

-y volumegroup
指定卷组名而不是让名称自动地生成。卷组名称在系统范围内必须是唯一的，可以有 1 到 15 个字符。该名称不能以设备配置数据库中的预定义设备数据库 (PdDv) 类中对其他设备定义的前缀开头。创建的卷组名将发送至标准输出。卷组名称只能包含以下字符：

- a - z
- 0 - 9
- _（下划线字符）
- -（减号字符）
- .（句点字符）

示例

要创建包含名为 *hdisk3*、*hdisk5* 和 *hdisk6* 的磁盘的卷组并将物理分区大小设置为 1，请输入：

```
cli_mkvg -s 1 hdisk3 hdisk5 hdisk6
```

相关信息:

mkvg 命令

---

cli_on_cluster 命令

用途

在集群中的所有节点上运行命令。

语法

```
cli_on_cluster  [ -S | -P ]  'command string'
```

描述

在所有集群节点上以 root 用户身份串行或并行运行命令。该命令的输出（stdout 和 stderr）显示在命令行上。每行输出前应加上节点名（后跟冒号）。

标志

-S 在集群中的每个节点上运行命令（一次一个）。该命令完成后，运行下一个命令。

-P 在集群中的所有节点上同步运行命令。

示例

要在集群中重新引导每个节点，请输入：

```
cli_on_cluster -S 'shutdown -Fr'
```

cli_on_node 命令

用途

在集群中的特定节点上运行任意命令。

语法

```
cli_on_node  [ -V <volume group> | -R <resource group | -N <node> ] 'command string'
```

描述

在显式指定的节点或拥有指定卷组或资源组的集群节点上以 root 用户身份运行命令。该命令的任何输出（stdout 和 stderr）显示在命令行上。

标志

-V volume group
在指定卷组处于联机状态的节点上运行该命令。如果卷组以并行方式在多个节点上处于联机状态，那么该命令在所有节点上运行。

-R resource group
在当前拥有指定资源组的节点上运行该命令。

-N node
在指定节点上运行该命令。此标志用于标识 PowerHA SystemMirror 节点名。

示例

要在名为 awesome 的节点上运行 **ps -efk** 命令，请输入：

```
cli_on_node -N awesome 'ps -efk'
```

cli_reducevg 命令

用途

从卷组中移除物理卷并使已更新的更改在所有集群节点上可用。如果从卷组移除所有物理卷，那么将在所有集群节点上删除该卷组。

语法

```
cli_reducevg  VolumeGroup PhysicalVolume ...
```

描述

使用 C-SPOC 运行带有参数的 **reducevg** 命令，并使已更新的卷组定义在所有集群节点上可用。

示例

要从名为 *vg01* 的卷组移除名为 *hdisk10* 的物理磁盘，请输入：

```
cli_reducevg vg01 hdisk101
```

相关信息:

reducevg 命令

---

cli_replacepv 命令

用途

将卷组中的物理卷替换为另一物理卷并使更改在所有集群节点上可用。

语法

```
cli_replacepv  SourcePhysicalVolume  DestinationPhysicalVolume
```

描述

使用 C-SPOC 运行带有参数的 **replacepv** 命令，并使已更新的卷组定义在所有集群节点上可用。

示例

要将名为 *hdisk10* 的磁盘替换为拥有 *hdisk10* 磁盘的卷组中名为 *hdisk20* 的磁盘，请输入：

```
cli_replacepv hdisk10 hdisk20
```

相关信息:

replacepv 命令

---

cli_rmfs 命令

用途

从集群中的所有节点移除文件系统。

语法

```
cli_rmfs [ -r ] FileSystem
```

描述

使用 C-SPOC 运行带有参数的 **rmfs** 命令，并从所有集群节点移除文件系统定义。

标志

-r 移除文件系统的安装点。

示例

要移除名为 */test_fs* 的共享文件系统，请输入：

```
cli_rmfs -r /test_fs
```

相关信息:

rmfs 命令

cli_rmlv 命令

用途

从集群中的所有节点移除逻辑卷。

语法

```
cli_rmlv  LogicalVolume ...
```

描述

使用 C-SPOC 运行带有参数的 **rmlv** 命令，并使已更新的逻辑卷定义在所有集群节点上可用。

示例

要更改名为 *lv01* 的共享逻辑卷，请输入：

```
cli_rmlv lv01
```

相关信息:

rmlv 命令

cli_rmlvcopy 命令

用途

从集群中的所有节点上的逻辑卷中移除副本。

语法

```
cli_rmlvcopy LogicalVolume Copies [ PhysicalVolume... ]
```

描述

在所有集群节点上使用 C-SPOC 运行带有参数的 **rmlvcopy** 命令。

示例

要将每个逻辑分区的属于名为 *lv01* 的逻辑卷的副本数减少至 1，请输入：

```
cli_rmlvcopy lv01 1
```

相关信息:

rmlvcopy 命令

cli_syncvg 命令

用途

运行带有参数的 **syncvg** 命令，并使已更新的卷组定义在所有集群节点上可用。

语法

```
cli_syncvg  [-f] [-H] [-P NumParallelLps] {-l|-v} Name
```

描述

使用 C-SPOC 运行 **syncvg** 命令，这会导致每个集群节点的逻辑卷管理器 (LVM) 读取卷组中的磁盘上的 LVM 信息。此命令还会更新逻辑卷组定义。

标志

-f 指定选择正常的物理副本并将其传播至逻辑分区的所有其他副本，即使这些副本不是旧文件。

-H 对于并行卷组处于活动状态的任何其他集群节点上的所选卷组，延迟写入操作直到同步操作完成。如果使用此标志，那么集群中的所有节点不必支持 **cli_syncvg** 命令的 **-P** 标志。如果卷组未以并行方式联机，那么会忽略该标志。

-l 指定 *Name* 变量表示逻辑卷设备名。

-P NumParallelLps
指定可并行同步的逻辑分区数。*NumParallelLps* 变量的有效范围为 1 - 32。*NumParallelLps* 变量必须特定于系统、卷组中的磁盘、系统资源及卷组方式。

-v 指定 *Name* 变量表示卷组设备名。

示例

要使名为 *v01* 的卷组上的副本同步，请输入：

```
cli_syncvg -v vg01
```

相关信息:

syncvg 命令

---

cli_unmirrorvg 命令

用途

在集群中所有节点上取消镜像卷组。

语法

```
cli_unmirrorvg  [ -c Copies ] VolumeGroup [ PhysicalVolume ... ]
```

描述

使用 C-SPOC 运行带有参数的 **unmirrorvg** 命令，并使已更新的卷组定义在所有集群节点上可用。

标志

-c Copies
指定在运行 **unmirrorvg** 命令后每个逻辑卷必须具有的最小副本数。如果不希望所有逻辑卷具有相同数目的副本，请使用 **rmlvcopy** 命令手动减少镜像。如果未使用此标志，那么系统会使用缺省值 1。

示例

要对名为 *vg01* 的共享卷组指定单个副本，请输入：

```
cli_unmirrorvg -c 1 vg01
```

相关信息:

unmirrorvg 命令

cli_updatevg 命令

用途

在所有集群节点上更新卷组定义以与该卷组的当前实际状态匹配。

语法

```
cli_updatevg VolumeGroup
```

描述

使用 C-SPOC 运行 **updatevg** 命令，这会导致每个集群节点的逻辑卷管理器 (LVM) 读取卷组中的磁盘上的 LVM 信息并更新本地卷组定义。

示例

要在所有集群节点上更新名为 *vg11* 的卷组的卷组定义，请输入：

```
cli_updatevg vg11
```

cllscf 命令

用途

列示集群拓扑信息。

语法

```
cllscf
```

描述

cllscf 命令列示集群、网络和适配器配置 ODM 对象类中定义的集群拓扑信息。**cllscf** 命令概述集群配置信息。

示例

要显示缺省或活动集群配置中定义的集群信息，请输入：

```
cllscf
```

此命令显示类似以下示例的输出信息：

```
# /usr/es/sbin/cluster/utilities/cllscf
Cluster Name:    hadev11_cluster
Cluster Type:    Standard
Heartbeat Type:  Unicast
Repository Disk: hdisk10 (00c0f592e54367f2)

There were 2 networks defined: net_ether_01, net_ether_02
There are 2 nodes in this cluster

NODE hadev11:
        This node has 0 service IP label(s):

NODE hadev12:
        This node has 0 service IP label(s):

Breakdown of network connections:

Connections to network net_ether_01
```

```
        Node hadev11 is connected to network net_ether_01 by these interfaces:
                hadev11

        Node hadev12 is connected to network net_ether_01 by these interfaces:
                hadev12

Connections to network net_ether_02
        Node hadev12 is connected to network net_ether_02 by these interfaces:
                hadev12_en1_boot
                hadev12_en2_boot
```

相关参考:

第 37 页的『clmgr 命令』

---

cllsdisk 命令

用途

列出指定资源链中可访问磁盘的 PVID。

语法

```
cllsdisk {-g Resource Group }
```

示例

运行以下命令以列出可由资源组 *grp3* 中的所有参与节点访问的磁盘的 PVID。

```
cllsdisk -g grp3
```

---

cllsfs 命令

用途

列出可由资源组中所有参与节点访问的共享文件系统。

语法

```
cllsfs {-g resource group } [-n]
```

表 *3. cllsfs* 标志

标志	描述
-g resource group	指定要为其列出文件系统的资源组的名称。
-n	列出在资源组中共享文件系统的节点。

注：请勿在命令行中运行 **cllsfs** 命令。使用 SMIT 界面可检索文件系统信息，如"管理共享 LVM 组件"中所述（请参阅"管理共享 LVM 组件"）。

---

cllsgrp 命令

用途

列示为集群配置的所有资源组。

语法

```
cllsgrp
```

描述

显示集群中所有资源组的名称。

示例

要显示集群的资源组信息，请输入：

```
cllsgrp
```

此命令显示以下输出：

```
grp1
grp2
grp3
grp4
```

---

cllsparam 命令

用途

列出运行时参数。

语法

```
cllsparam {-n nodename } [-c] [-s] [-d odmdir ]
```

标志

-n nodename
指定要为其列出信息的节点。

-c 指定冒号输出格式。

-s 与 **-c** 标志一起使用，指定本机语言以代替英语。

-d odmdir
指定备用 ODM 目录。

示例

运行以下示例以显示节点 abalone 的运行时参数：

```
cllsparam -n abalone
```

---

cllsres 命令

用途

按名称和参数对 PowerHA SystemMirror for AIX 配置数据库资源数据进行排序。

语法

```
cllsres [-g group ] [-e] [-c] [-s] [-d odmdir ] [-qquery ]
```

标志

-g group
指定要列出的资源组的名称。

-c 指定冒号输出格式。

-e 以"resourcetype=resourcenames"格式展开用户定义的资源列表。

-s 与 **-c** 标志一起使用，指定母语以代替英语。

-d odmdir
指定备用 ODM 目录。

-q query
指定 ODM 检索的搜索条件。请参阅 **odmget** 联机帮助页，以获取有关搜索条件的信息。

示例

1. 运行以下命令以列出所有资源组的资源数据。
   ```
   cllsres
   ```
2. 运行以下命令以列出 grp1 资源组的资源数据。
   ```
   cllsres -g grp1
   ```
3. 运行以下命令以列出 grp1 资源组的文件系统资源数据。
   ```
   cllsres -g grp1 -q"name = FILESYSTEM"
   ```

---

cllsserv 命令

用途

按名称来列出应用程序控制器。

语法

```
cllsserv [-c] [-h] [-n name ] [-d odmdir ]
```

标志

-c 指定冒号输出格式。

-h 指定打印标题。

-n name
指定要为其检查信息的应用程序控制器。

-d odmdir
指定备用 ODM 目录。

示例

1. 运行以下命令以列出所有应用程序控制器。
   ```
   cllsserv
   ```
2. 运行以下命令以使用冒号格式列出 test1 应用程序控制器的信息。
   ```
   cllsres -c -n test1
   ```

---

cllsvg 命令

用途

列出由集群中的节点共享的卷组。如果某个卷组可以由某个已配置资源组中所有参与节点来访问，那么便认为此卷组为共享。请注意，列出的卷组不一定已配置为任何资源组中的资源。如果既没有选择 **-s** 也没有选择 **-c**，那么将同时列出共享卷组和并发卷组。

语法

```
cllsvg {-g resource group } [-n] [-v] [-s | -c]]
```

标志

-g resource group
指定要为其列出卷组（这些卷组在参与该资源组的节点中共享）的资源组的名称。

-n nodes
指定参与每个资源组的所有节点。

-v 仅列出已联机并且与其他命令行条件匹配的卷组。

-s 仅列出还匹配其他条件的共享卷组。

-c 仅列出还匹配其他条件的并发卷组。

示例

运行以下命令以列出 grp1 资源组中的所有共享卷组。

```
cllsvg -g grp1
```

---

clmgr 命令

用途

clmgr 命令提供了一致且可靠的接口，可用于通过终端或脚本来执行 PowerHA SystemMirror 集群操作。

语法

以下是 **clmgr** 命令的完整语法：

```
clmgr {[-c|-d <DELIMITER>] [-S] | [-x]}
[-v][-f] [-D] [-T <#####>]
[-l {error|standard|low|med|high|max}][-a {<ATTR#1>,<ATTR#2>,...}]
<ACTION> <CLASS> [<NAME>]
[-h | <ATTR#1>=<VALUE#1> <ATTR#2>=<VALUE#2> <ATTR#n>=<VALUE#n>]

clmgr {[-c|-d <DELIMITER>] [-S] | [-x]}
[-v][-f] [-D] [-T <#####>]
[-l {error|standard|low|med|high|max}][-a {<ATTR#1>,<ATTR#2>,...}]
[-M]- "
<ACTION> <CLASS> [<NAME>] <ATTR#1>=<VALUE#1> <ATTR#n>=<VALUE#n>]
.
.
."
         ACTION={add|modify|delete|query|online|offline|...}
          CLASS={cluster|site|node|network|resource_group|...}

    clmgr {-h|-?} [-v]
    clmgr [-v] help
```

以下是使用 **clmgr** 命令的基本格式：

```
clmgr <ACTION> <CLASS> [<NAME>] [<ATTRIBUTES...>]
```

可以从命令行中获取 **clmgr** 命令的帮助。例如，当您不带有任何标志或参数运行 **clmgr** 命令时，将显示可用 ACTION 的列表。如果在命令行中输入 `clmgr ACTION` 但不提供 CLASS，那么将生成指定的 ACTION 的所有可用 CLASS 的列表。如果输入 `clmgr ACTION CLASS` 但不提供 NAME 或 ATTRIBUTES，那么将略有不同，因为某些 ACTION+CLASS 组合不需要任何其他参数。要在此情况中显示帮助，您必须通过将 `-h` 标志追

加到 clmgr ACTION CLASS 命令来显式请求帮助。您无法从命令行中显示每个 **clmgr** 命令的各个 ATTRIBUTES 的帮助。

描述

clmgr 命令所使用的高度一致性有助于使该命令更易于学习和使用。除执行的一致性以外，**clmgr** 还提供一致的返回码，从而使脚本编制更为容易。该命令还为数据查询提供了多种输出格式，从而使收集集群信息尽可能的简单。

所有 **clmgr** 命令操作都记录在 clutils.log 文件中，包括已执行的命令的名称、命令启动和停止时间以及启动命令的用户名。

注：如果资源组具有多个依赖项，那么不能使用 **clmgr** 命令移动多个资源组。

标志

ACTION

描述要执行的操作。

注：ACTION 不区分大小写。所有 ACTION 标志都提供一个更简短的别名。例如，rm 是 delete 的别名。为方便起见，在命令行中提供了别名，并且不得在脚本中使用别名。

几乎所有受支持的 CLASS 对象上都可以使用以下四个 ACTION 标志：

- add （别名：a）
- query （别名：q、ls、get）
- modify （别名：mod、ch、set）
- delete （别名：de、rm、er）

其他 ACTION 通常仅在一小部分支持的 CLASS 对象上受支持：

- 集群、节点和资源组：
  - start （别名：online、on）
  - stop （别名：offline、off）
- 资源组、服务 IP 和持久性 IP：
  - move （别名：mv）
- 集群、接口、日志、节点、快照、网络、应用程序监视器：
  - manage （别名：mg）
- 集群和文件集合：
  - sync （别名：sy）
- 集群、方法：
  - verify （别名：ve）
- 日志、报告、快照：
  - view （别名：vi）
- 存储库：
  - replace（别名：rep、switch、swap）

类

对其执行 ACTION 的对象的类型。

注：CLASS 不区分大小写。所有 CLASS 对象都提供一个更简短的别名。例如，fc 是 file_collection 的别名。为方便起见，在命令行中提供了别名，并且不得在脚本中使用别名。

以下是受支持 CLASS 对象的完整列表：

- cluster （别名：cl）
- repository （别名：rp）
- site （别名：st）
- node （别名：no）
- interface （别名：in、if）
- network （别名：ne、nw）
- resource_group （别名：rg）
- service_ip （别名：si）
- persistent_ip （别名：pi）
- application_controller （别名：ac、app）
- application_monitor （别名：am、mon）
- tape （别名：tp）
- dependency （别名：de）
- file_collection （别名：fi、fc）
- snapshot （别名：sn、ss）
- method （别名：me）
- volume_group （别名：vg）
- logical_volume （别名：lv）
- file_system （别名：fs）
- physical_volume （别名：pv、disk）
- mirror_pool （别名：mp）
- user （别名：ur）
- group （别名：gp）
- ldap_server （别名：ls）
- ldap_client （别名：lc）
- event
- hmc
- cod （别名：cuod、dlpar）

名称

将对其执行 ACTION 的 CLASS 类型的特定对象。

ATTR=VALUE

一个可选标志，具有特定于 ACTION+CLASS 组合的属性对和值对。使用这些对标志可指定配置设置或调整特定操作。

与查询操作结合使用时，ATTR=VALUE 指定内容可以用于执行基于属性的搜索和过滤。用于此目的时，您可以使用简单通配符。例如，"*"匹配零个或更多个任意字符，"?"匹配零个或一个任意字符。

注：可能并非始终需要完全输入 ATTR。仅必需的前导字符（用来唯一标识可用于指定操作的属性集中的属性）数目是必须提供的。要进行添加集群操作，不必输入 FC_SYNC_INTERVAL，而是可以输入 FC 来获得相同结果。

-a 仅显示指定属性，并且仅对以下 ACTION 有效：query、add 和 modify。属性名称不区分大小写，并且可以与标准 UNIX 通配符"*"和"?"结合使用.

-c 以冒号分隔格式显示所有数据，并且仅对以下 ACTION 有效：query、add 和 modify。

-d 仅对 *query*、*add* 和 *modify* ACTION 标志有效，用于请求以指定定界符分隔的格式显示所有数据。

-D 在 **clmgr** 命令中禁用依赖性机制，依赖性机制会尝试使用缺省值来创建任何必要资源（如果集群中尚未定义这些资源）。

-f 覆盖任何交互式提示，从而强制尝试当前操作（如果能够强制操作）。

-h 显示帮助信息。

-l 激活以下跟踪日志记录值以实现可维护性：

- Error：仅当检测到错误时才更新日志文件。
- Standard：记录有关每个 clmgr 操作的基本信息。
- Low：每个函数的基本入口和出口跟踪。
- Med：执行 *low* 跟踪，同时添加函数入口参数和函数返回值。
- High：执行 *med* 跟踪，同时添加每个执行行的跟踪并省略例程函数和实用程序函数。
- Max - 执行 *high* 跟踪，同时添加例程函数和实用程序函数。向函数入口消息和出口消息中添加时间和日期戳记。

注：所有跟踪数据都将写入到 clutils.log 文件。此标志非常适合于对问题进行故障诊断。

-M
允许通过 **clmgr** 的一次调用指定和运行多个操作，每行指定一个操作。所有操作将共享一个公共事务标识。

-S 显示禁止了列标题的数据，并且仅对 query ACTION 和 -c 标志有效。

-T 事务标识将应用于所有已记录的输出，以帮助将一个或多个活动分组为可从日志中抽取的单个输出主体，以供分析。此标志非常适合于对问题进行故障诊断。

-v 在输出中显示最大详细程度。

注：如果与 query ACTION 结合使用并且不指定任何特定对象名，那么将显示指定类的所有实例。例如，输入 clmgr -v query mode 将显示所有节点及其属性。如果此标志与 add 或 modify ACTION 结合使用，那么将在操作完成后显示生成的属性（仅当操作成功时）。

-x 以简单 XML 格式显示所有数据，并且仅对以下 ACTION 有效：query、add 和 modify。

语法

以下部分描述所有可能的 **clmgr** 操作的语法。

- 应用程序控制器
- 应用程序监视器
- Cluster
- CoD
- 依赖性
- EFS

- 事件
- 回退计时器
- 文件集合
- 文件系统
- 组
- HMC
- Interface
- LDAP 服务器
- LDAP 客户机
- Log
- 方法
- 镜像组
- 镜像对
- 镜像池
- 网络
- Node
- 持久性 IP/标签
- 物理卷
- Report
- 存储库
- 资源组
- 服务 IP/标签
- 站点
- 快照
- 存储代理程序
- 存储系统
- 磁带
- User
- 卷组

Cluster

```
clmgr add cluster \
      [ <cluster_label> ] \
      [ NODES=<host>[,<host#2>,...] ] \
      [ TYPE={NSC|SC} ] \
      [ HEARTBEAT_TYPE={unicast|multicast} ] \
          [ CLUSTER_IP=<IP_Address> ] \
      [ REPOSITORIES=<disk>[,<backup_disk>,...] ] \
      [ FC_SYNC_INTERVAL=## ] \
      [ RG_SETTLING_TIME=## ] \
      [ MAX_EVENT_TIME=### ] \
      [ MAX_RG_PROCESSING_TIME=### ] \
      [ DAILY_VERIFICATION={Enabled|Disabled} ] \
      [ VERIFICATION_NODE={Default|<node>} ] \
      [ VERIFICATION_HOUR=<00..23> ] \
      [ VERIFICATION_DEBUGGING={Enabled|Disabled} ] \
```

```
      [ HEARTBEAT_FREQUENCY=<10..600> ] \
      [ GRACE_PERIOD=<5..30> ] \
      [ SITE_POLICY_FAILURE_ACTION={fallover|notify} ] \
      [ SITE_POLICY_NOTIFY_METHOD="<FULL_PATH_TO_FILE>" ] \
      [ SITE_HEARTBEAT_CYCLE=<1..10> ] \
      [ SITE_GRACE_PERIOD=<10..30> ] \
      [ TEMP_HOSTNAME={disallow|allow} ] \
      [ MONITOR_INTERFACES={enable|disable} ] \
      [ NETWORK_FAILURE_DETECTION_TIME=<0..590> ]


clmgr add cluster \
      [ <cluster_label> ] \
      [ NODES=<host>[,<host#2>,...] ] \
      [ TYPE="LC" ] \
      [ HEARTBEAT_TYPE={unicast|multicast} ] \
      [ FC_SYNC_INTERVAL=## ] \
      [ RG_SETTLING_TIME=## ] \
      [ MAX_EVENT_TIME=### ] \
      [ MAX_RG_PROCESSING_TIME=### ] \
      [ DAILY_VERIFICATION={Enabled|Disabled} ] \
      [ VERIFICATION_NODE={Default|<node>} ] \
      [ VERIFICATION_HOUR=<00..23> ] \
      [ VERIFICATION_DEBUGGING={Enabled|Disabled} ] \
      [ HEARTBEAT_FREQUENCY=<10..600> ] \
      [ GRACE_PERIOD=<5..30> ] \
      [ SITE_POLICY_FAILURE_ACTION={fallover|notify} ] \
      [ SITE_POLICY_NOTIFY_METHOD="<FULL_PATH_TO_FILE>" ]
      [ SITE_HEARTBEAT_CYCLE=<1..10> ] \
      [ SITE_GRACE_PERIOD=<10..30> ] \
      [ TEMP_HOSTNAME={disallow|allow} ] \
      [ MONITOR_INTERFACES={enable|disable} ] \
      [ NETWORK_FAILURE_DETECTION_TIME=<0..590> ]
```

表 *4.* 首字母缩略词及其含义

缩写	含义
NSC	非站点集群（将不定义任何站点）
SC	延伸集群（简化的基础结构，非常适合有限距离数据复制；必须定义站点）
LC	链接集群（全功能的基础结构，非常适合长距离数据复制；必须定义站点）。

注：*CLUSTER_IP* 只能与集群类型 *NSC* 或 *SC* 结合使用。对于 *LC* 集群，必须为每个站点设置多点广播地址。

注：*REPOSITORIES* 选项只能与集群类型 *NSC* 或 *SC* 配合使用。对于 *LC* 集群，将为每个站点确定 *REPOSITORIES* 选项。*REPOSITORIES* 选项可以使用七个磁盘。第一个磁盘是活动存储库磁盘，随后的磁盘是备份存储库磁盘。

```
clmgr modify cluster \
      [ NAME=<new_cluster_label> ] \
      [ NODES=<host>[,<host#2>,...] ] \
      [ TYPE={NSC|SC} ] \
      [ HEARTBEAT_TYPE={unicast|multicast} ] \
      [ CLUSTER_IP=<IP_Address> ] \
      [ REPOSITORIES=<backup_disk>[,<backup_disk>,...] ] \
      [ FC_SYNC_INTERVAL=## ] \
      [ RG_SETTLING_TIME=## ] \
      [ MAX_EVENT_TIME=### ] \
      [ MAX_RG_PROCESSING_TIME=### ] \
      [ DAILY_VERIFICATION={Enabled|Disabled} ] \
      [ VERIFICATION_NODE={Default|<node>} ] \
```

```
      [ VERIFICATION_HOUR=<00..23> ] \
      [ VERIFICATION_DEBUGGING={Enabled|Disabled} ] \
      [ HEARTBEAT_FREQUENCY=<10..600> ] \
      [ GRACE_PERIOD=<5..30> ] \
      [ SITE_POLICY_FAILURE_ACTION={fallover|notify} ] \
      [ SITE_POLICY_NOTIFY_METHOD="<FULL_PATH_TO_FILE>" ] \
      [ SITE_HEARTBEAT_CYCLE=<1..10> ] \
      [ SITE_GRACE_PERIOD=<10..30> ] \
      [ TEMP_HOSTNAME={disallow|allow} ] \
      [ MONITOR_INTERFACES={enable|disable} ] \
      [ LPM_POLICY={manage|unmanage} ] \
      [ HEARTBEAT_FREQUENCY_DURING_LPM=### ] \
      [ NETWORK_FAILURE_DETECTION_TIME=<0..590> ]
```

注： *REPOSITORIES* 选项只能与集群类型 *NSC* 或 *SC* 配合使用。对于 *LC* 集群，将为每个站点确定 *REPOSITORIES* 选项。*REPOSITORIES* 选项可以使用六个备份存储库磁盘。

```
clmgr modify cluster \
      [ NAME=<new_cluster_label> ] \
      [ NODES=<host>[,<host#2>,...] ] \
      [ TYPE="LC" ] \
      [ HEARTBEAT_TYPE={unicast|multicast} ] \
      [ FC_SYNC_INTERVAL=## ] \
      [ RG_SETTLING_TIME=## ] \
      [ MAX_EVENT_TIME=### ] \
      [ MAX_RG_PROCESSING_TIME=### ] \
      [ DAILY_VERIFICATION={Enabled|Disabled} ] \
      [ VERIFICATION_NODE={Default|<node>} ] \
      [ VERIFICATION_HOUR=<00..23> ] \
      [ VERIFICATION_DEBUGGING={Enabled|Disabled} ] \
      [ HEARTBEAT_FREQUENCY=<10..600> ] \
      [ GRACE_PERIOD=<5..30> ] \
      [ SITE_POLICY_FAILURE_ACTION={fallover|notify} ] \
      [ SITE_POLICY_NOTIFY_METHOD="<FULL_PATH_TO_FILE>" ]
      [ SITE_HEARTBEAT_CYCLE=<1..10> ] \
      [ SITE_GRACE_PERIOD=<10..30> ] \
      [ TEMP_HOSTNAME={disallow|allow} ] \
      [ MONITOR_INTERFACES={enable|disable} ] \
      [ LPM_POLICY={manage|unmanage} ] \
      [ HEARTBEAT_FREQUENCY_DURING_LPM=### ] \
      [ NETWORK_FAILURE_DETECTION_TIME=<0..590> ]
```

```
|     clmgr modify cluster \
|           [ SPLIT_POLICY={none|tiebreaker|manual|NFS} ] \
|           [ TIEBREAKER=<disk> ] \
|           [ MERGE_POLICY={none|majority|tiebreaker|manual|NFS} ] \
|           [ NFS_QUORUM_SERVER=<server> ] \
|           [ LOCAL_QUORUM_DIRECTORY=<local_mount>] \
|           [ REMOTE_QUORUM_DIRECTORY=<remote_mount>] \
|           [ QUARANTINE_POLICY=<disable|node_halt|fencing|halt_with_fencing>] \
|           [ CRITICAL_RG=<rgname> ] \
|           [ NOTIFY_METHOD=<method> ] \
|           [ NOTIFY_INTERVAL=### ] \
|           [ MAXIMUM_NOTIFICATIONS=### ] \
|           [ DEFAULT_SURVIVING_SITE=<site> ] \
|           [ APPLY_TO_PPRC_TAKEOVER={yes|no} ] \
|           [ ACTION_PLAN={reboot|disable_rgs_autostart|disable_cluster_services_autostart} ]
```

注： 在完全定义和同步站点之后，以及当站点仍在使用中时，无法修改集群类型。

```
clmgr query cluster [ ALL | {CORE,SECURITY,SPLIT-MERGE,HMC,ROHA} ]
     clmgr delete cluster [ NODES={ALL|<node>[,<node#2>,...}] ]
```

注： 缺省情况下，delete 操作将从所有可用节点完全删除集群。

```
clmgr discover cluster
clmgr recover clusterclmgr sync cluster \
      [ VERIFY={yes|no} ] \
      [ CHANGES_ONLY={no|yes} ] \
      [ DEFAULT_TESTS={yes|no} ] \
      [ METHODS=<method#1>[,<method#2>,...] ] \
      [ FIX={no|yes} ] \
      [ LOGGING={standard|verbose} ] \
      [ LOGFILE=<PATH_TO_LOG_FILE> ] \
      [ MAX_ERRORS=## ] \
      [ FORCE={no|yes} ]
```

注：所有选项均为验证参数，因此仅当 VERIFY 设置为 yes 时，这些选项才有效。

```
clmgr manage cluster {reset|unlock}

clmgr manage cluster security \
      [ LEVEL={Disable|Low|Med|High} ] \
      [ ALGORITHM={DES|3DES|AES} ]\
      [ GRACE_PERIOD=<SECONDS> ] \
      [ REFRESH=<SECONDS> ] ] \
      [ MECHANISM={OpenSSL|SSH} ] \
        CERTIFICATE=<PATH_TO_FILE> \
        PRIVATE_KEY=<PATH_TO_FILE>
```

注：如果对 MECHANISM 指定了 SSL 或 SSH，那么必须提供定制证书和专用密钥文件。

```
clmgr manage cluster security \
      [ LEVEL={Disable|Low|Med|High} ] \
      [ ALGORITHM={DES|3DES|AES} ]\
      [ GRACE_PERIOD=<SECONDS> ] \
      [ REFRESH=<SECONDS> ] ] \
      [ MECHANISM="SelfSigned" ] \
        [ CERTIFICATE=<PATH_TO_FILE> ] \
        [ PRIVATE_KEY=<PATH_TO_FILE> ]
```

注： 如果对 MECHANISM 指定了 Self-Signed，那么指定证书和专用密钥文件是可选的。如果均未提供，那么将自动生成缺省对。GRACE_PERIOD 的缺省值为 21600 秒（6 小时）。REFRESH 的缺省值为 86400 秒（24 小时）。

```
clmgr manage cluster hmc \
            [ DEFAULT_HMC_TIMEOUT=<MINUTES> ] \
            [ DEFAULT_HMC_RETRY_COUNT=<INTEGER> ] \
            [ DEFAULT_HMC_RETRY_DELAY=<SECONDS> ] \
            [ DEFAULT_HMCS_LIST=<HMCS> ]

clmgr manage cluster roha \
            [ ALWAYS_START_RG={YES|NO} ] \
            [ ADJUST_SPP_SIZE={YES|NO} ]\
            [ FORCE_SYNC_RELEASE={YES|NO} ] \
            [ AGREE_TO_COD_COSTS={YES|NO} ] ] \
            [ ONOFF_DAYS=<DAYS>} ]


clmgr verify cluster \
      [ CHANGES_ONLY={no|yes} ] \
      [ DEFAULT_TESTS={yes|no} ] \
      [ METHODS=<method#1>[,<method#2>,...] ] \
      [ FIX={no|yes} ] \
      [ LOGGING={standard|verbose} ] \
      [ LOGFILE=<PATH_TO_LOG_FILE> ] \
      [ MAX_ERRORS=## ]
      [ SYNC={no|yes} ] \
      [ FORCE={no|yes} ]
```

注：在 SYNC 设置为 yes 时，可以使用 FORCE 选项。

```
clmgr offline cluster \
      [ WHEN={now|restart|both} ] \
      [ MANAGE={offline|move|unmanage} ] \
      [ BROADCAST={true|false} ] \
      [ TIMEOUT=<seconds_to_wait_for_completion> ]
      [ STOP_CAA={no|yes} ]
clmgr online cluster \
      [ WHEN={now|restart|both} ] \
      [ MANAGE={auto|manual} ] \
      [ BROADCAST={false|true} ] \
      [ CLINFO={false|true|consistent} ] \
      [ FORCE={false|true} ] \
      [ FIX={no|yes|interactively} ] \
      [ TIMEOUT=<seconds_to_wait_for_completion> ] \
      [ START_CAA={no|yes|only} ]
```

注：RG_SETTLING_TIME 属性仅影响启动策略为"Online On First Available Node"的资源组。cluster 的别名是 cl。

注：STOP_CAA 和 START_CAA 选项将使 Cluster Aware AIX (CAA) 集群服务脱机或联机。请在对这些选项有特定的已知需要时或在 IBM® 支持机构的指导下使用这些选项。不要取消激活 CAA 集群服务，因为这将禁用在集群环境中检测问题的功能。only 选项仅启动 CAA 服务。

存储库

```
clmgr add repository <disk>[,<backup_disk#2>,...] \
     [ SITE=<site_label> ]\
   [ NODE=<reference_node> ]
```

注：如果尚未定义活动存储库，那么第一个磁盘将用作活动存储库。列表中的任何其他磁盘将定义为备份存储库磁盘。对于标准集群和延伸集群，您最多可以为每个集群标识六个备份存储库磁盘。对于链接集群，您最多可以为每个站点标识六个备份存储库磁盘。

```
clmgr replace repository [ <new_repository> ] \
      [ SITE=<site_label> ] \
      [ NODE=<reference_node>]
```

注：如果未指定任何磁盘，那么将使用备份列表中的第一个磁盘。

```
clmgr query repository [ <disk>[,<disk#2>,...] ]
clmgr delete repository {<backup_disk>[,<disk#2>,...] | ALL}\
   [ SITE=<site_label> ] \
   [ NODE=<reference_node> ]
```

注：无法删除活动存储库磁盘。只能移除备份存储库。

地点

```
clmgr add site <sitename> \
          NODES=<node>[,<node#2>,...] \
    [ SITE_IP=<multicast_address> ] \
    [ RECOVERY_PRIORITY={MANUAL|1|2} ] \
    [ REPOSITORIES=<disk>[,<backup_disk>,...] ]
```

注：*REPOSITORIES* 选项只能与集群类型 *LC* 配合使用。*REPOSITORIES* 选项可以使用七个磁盘。第一个磁盘是活动存储库磁盘，随后的磁盘是备份存储库磁盘。

```
clmgr modify site <sitename> \
      [ NAME=<new_site_label> ] \
      [ NODES=<node>[,<node#2>,...] ] \
      [ SITE_IP=<multicast_address> ] \
      [ RECOVERY_PRIORITY={MANUAL|1|2} ] \
      [ REPOSITORIES=<backup_disk>[,<backup_disk>,...] ] \
      [ HMCS=<hmc>[,<hmc#2>,...] ]
```

注：*SITE_IP* 属性只能与集群类型 *LC*（链接集群）和集群脉动信号类型 *multicast* 配合使用。

注：*REPOSITORIES* 选项只能与集群类型 *LC* 配合使用。*REPOSITORIES* 选项可以使用六个备份存储库磁盘。

```
   clmgr query site [ <sitename>[,<sitename#2>,...] ]
clmgr delete site {<sitename>[,<sitename#2>,...] | ALL}
   clmgr offline site <sitename> \
      [ WHEN={now|restart|both} ] \
      [ MANAGE={offline|move|unmanage} ] \
      [ BROADCAST={true|false} ] \
      [ TIMEOUT=<seconds_to_wait_for_completion> ] \
      [ STOP_CAA={no|yes} ]
   clmgr online site <sitename> \
      [ WHEN={now|restart|both} ] \
      [ MANAGE={auto|manual} ] \
      [ BROADCAST={false|true} ] \
      [ CLINFO={false|true|consistent} ] \
      [ FORCE={false|true} ] \
      [ FIX={no|yes|interactively} ] \
      [ TIMEOUT=<seconds_to_wait_for_completion> ] \
      [ START_CAA={no|yes|only} ]
clmgr manage site respond {continue|recover}
```

注：*site* 的别名是 st。

注：STOP_CAA 和 START_CAA 选项将使 Cluster Aware AIX (CAA) 集群服务脱机或联机。请在对这些选项有特定的已知需要时或在 IBM 支持机构的指导下使用这些选项。不要取消激活 CAA 集群服务，因为这将禁用在集群环境中检测问题的功能。only 选项仅启动 CAA 服务。

节点

```
clmgr add node <node> \
      [ COMMPATH=<ip_address_or_network-resolvable_name> ] \
      [ RUN_DISCOVERY={true|false} ] \
      [ PERSISTENT_IP=<IP> NETWORK=<network>
        {NETMASK=<255.255.255.0 | PREFIX=1..128} ] \
      [ START_ON_BOOT={false|true} ] \
      [ BROADCAST_ON_START={true|false} ] \
      [ CLINFO_ON_START={false|true|consistent} ] \
      [ VERIFY_ON_START={true|false} ] \
      [ SITE=<sitename> ]
clmgr modify node <node> \
      [ NAME=<new_node_label> ] \
      [ COMMPATH=<new_commpath> ] \
      [ PERSISTENT_IP=<IP> NETWORK=<network>
        {NETMASK=<255.255.255.0 | PREFIX=1..128} ] \
      [ START_ON_BOOT={false|true} ] \
      [ BROADCAST_ON_START={true|false} ] \
      [ CLINFO_ON_START={false|true|consistent} ] \
      [ VERIFY_ON_START={true|false} ] \
      [ HMCS=<hmc>[,<hmc#2>,...] ] \
      [ ENABLE_LIVE_UPDATE={true|false} ]
clmgr query node [ {<node>|LOCAL}[,<node#2>,...] ]
```

```
clmgr delete node {<node>[,<node#2>,...] | ALL}
clmgr manage node undo_changes
clmgr recover node <node>[,<node#2>,...]
clmgr online node <node>[,<node#2>,...] \
      [ WHEN={now|restart|both} ] \
      [ MANAGE={auto|manual} ] \
      [ BROADCAST={false|true} ] \
      [ CLINFO={false|true|consistent} ] \
      [ FORCE={false|true} ] \
      [ FIX={no|yes|interactively} ] \
      [ TIMEOUT=<seconds_to_wait_for_completion> ] \
      [ START_CAA={no|yes|only} ]
clmgr offline node <node>[,<node#2>,...] \
      [ WHEN={now|restart|both} ] \
      [ MANAGE={offline|move|unmanage} ] \
      [ BROADCAST={true|false} ] \
      [ TIMEOUT=<seconds_to_wait_for_completion> ] \
      [ STOP_CAA={no|yes} ]
```

注：TIMEOUT 属性的缺省值为 120 秒。*node* 的别名是 no。

注：STOP_CAA 和 START_CAA 选项将使 Cluster Aware AIX (CAA) 集群服务脱机或联机。请在对这些选项有特定的已知需要时或在 IBM 支持机构的指导下使用这些选项。不要取消激活 CAA 集群服务，因为这将禁用在集群环境中检测问题的功能。only 选项仅启动 CAA 服务。

网络

```
clmgr add network <network> \
      [ TYPE={ether|XD_data|XD_ip} ] \
      [ {NETMASK=<255.255.255.0 | PREFIX=1..128} ] \
      [ IPALIASING={true|false} ] \
      [ PUBLIC={true|false} ]
```

注：缺省情况下，将使用子网掩码 *255.255.255.0* 来构造 IPv4 网络。要创建 IPv6 网络，请指定一个有效前缀。

```
clmgr modify network <network> \
      [ NAME=<new_network_label> ] \
      [ TYPE={ether|XD_data|XD_ip} ] \
      [ {NETMASK=<255.255.255.0> | PREFIX=1..128} ] \
      [ PUBLIC={true|false} ] \
      [ RESOURCE_DIST_PREF={AC|ACS|C|CS|CPL|ACPL|ACPLS|NOALI} ] \
        [ SOURCE_IP=<service_or_persistent_ip> ]
```

注：RESOURCE_DIST_PREF 属性的可能值如下：

AC Anti-collocation

ACS
Anti-collocation with source

C Collocation

CS Collocation with source

CPL
Collocation with persistent label

ACPL
Anti-collocation with persistent label

ACPLS
Anti-collocation with persistent label and source

NOALI
禁用第一个别名

注：如果 RESOURCE_DIST_PREF 属性使用 CS 或 ACS 值，那么 SOURCE 属性必须是服务标签。

```
clmgr query network [ <network>[,<network#2>,...] ]
clmgr delete network {<network>[,<network#2>,...] | ALL}
```

注：*network* 的别名是 ne 和 nw。

Interface

```
clmgr add interface <interface> \
      NETWORK=<network> \
      [ NODE=<node> ] \
      [ TYPE={ether|XD_data|XD_ip} ] \
      [ INTERFACE=<network_interface> ]
clmgr modify interface <interface> \
      NETWORK=<network>
clmgr query interface [ <interface>[,<if#2>,...] ]
clmgr delete interface {<interface>[,<if#2>,...] | ALL}
clmgr discover interfaces
```

注：接口可以是 IP 地址或标签。 NODE 属性的缺省值为本地节点名称。TYPE 属性的缺省值为 ether。<network_interface> 它可能类似于 en1、en2、en3。*interface* 的别名是 in 和 if。

资源组

```
      clmgr add resource_group <resource_group>[,<rg#2>,...] \
      NODES=nodeA1,nodeA2,...                                 \
      [ SECONDARYNODES=nodeB2[,nodeB1,...]                  ] \
      [ SITE_POLICY={ignore|primary|either|both}            ] \
      [ STARTUP={OHN|OFAN|OAAN|OUDP}                        ] \
      [ FALLOVER={FNPN|FUDNP|BO}                            ] \
      [ FALLBACK={NFB|FBHPN}                                ] \
        [ FALLBACK AT=<FALLBACK TIMER>                      ] \
      [ NODE_PRIORITY_POLICY={default|mem|cpu|                \
                              disk|least|most}              ] \
      [ NODE_PRIORITY_POLICY_SCRIPT=</path/to/script>       ] \
      [ NODE_PRIORITY_POLICY_TIMEOUT=###                    ] \
      [ SERVICE_LABEL=service_ip#1[,service_ip#2,...]       ] \
      [ APPLICATIONS=appctlr#1[,appctlr#2,...]              ] \
      [ SHARED_TAPE_RESOURCES=<TAPE>[,<TAPE#2>,...]         ] \
      [ VOLUME_GROUP=<VG>[,<VG#2>,...]                      ] \
      [ FORCED_VARYON={true|false}                          ] \
      [ VG_AUTO_IMPORT={true|false}                         ] \
      [ FILESYSTEM=/file_system#1[,/file_system#2,...]      ] \
      [ DISK=<raw_disk>[,<raw_disk#2>,...]                  ] \
      [ FS_BEFORE_IPADDR={true|false}                       ] \
      [ WPAR_NAME="wpar_name"                               ] \
      [ EXPORT_FILESYSTEM=/expfs#1[,/expfs#2,...]           ] \
      [ EXPORT_FILESYSTEM_V4=/expfs#1[,/expfs#2,...]        ] \
      [ STABLE_STORAGE_PATH="/fs3"                          ] \
      [ NFS_NETWORK="nfs_network"                           ] \
      [ MOUNT_FILESYSTEM=/nfs_fs1;/expfs1,/nfs_fs2;,...     ] \
      [ MIRROR_GROUP=<replicated_resource>                  ] \
      [ FALLBACK_AT=<FALLBACK_TIMER>                        ]

STARTUP:
    OHN ----- Online Home Node (default value)
    OFAN ---- Online on First Available Node
    OAAN ---- Online on All Available Nodes (concurrent)
    OUDP ---- Online Using Node Distribution Policy
```

```
FALLOVER:
    FNPN ---- Fallover to Next Priority Node (default value)
    FUDNP --- Fallover Using Dynamic Node Priority
    BO ------ Bring Offline (On Error Node Only)

FALLBACK:
    NFB ----- Never Fallback
    FBHPN --- Fallback to Higher Priority Node (default value)

NODE_PRIORITY_POLICY:
    default - next node in the NODES list
    mem ----- node with most available memory
    disk ---- node with least disk activity
    cpu ----- node with most available CPU cycles
    least --- node where the dynamic node priority script
              returns the lowest value
    most ---- node where the dynamic node priority script
              returns the highest value
```

注： 仅当 FALLOVER 策略已设置为 FUDNP 时，才能建立 NODE_PRIORITY_POLICY 策略。

```
SITE_POLICY:
     ignore -- Ignore
     primary - Prefer Primary Site
     either -- Online On Either Site
     both ---- Online On Both Sites
```

```
clmgr modify resource_group <resource_group>                    \
      [ NAME=<new_resource_group_label>              ] \
      [ NODES=nodeA1[,nodeA2,...]                      ] \
      [ SECONDARYNODES=nodeB2[,nodeB1,...]             ] \
      [ SITE_POLICY={ignore|primary|either|both}       ] \
      [ STARTUP={OHN|OFAN|OAAN|OUDP}                   ] \
      [ FALLOVER={FNPN|FUDNP|BO}                       ] \
      [ FALLBACK={NFB|FBHPN}                           ] \
      [ FALLBACK_AT=<FALLBACK_TIMER>                   ] \
      [ NODE_PRIORITY_POLICY={default|mem|cpu|           \
                              disk|least|most}         ] \
      [ NODE_PRIORITY_POLICY_SCRIPT=</path/to/script>  ] \
      [ NODE_PRIORITY_POLICY_TIMEOUT=###               ] \
      [ SERVICE_LABEL=service_ip#1[,service_ip#2,...]  ] \
      [ APPLICATIONS=appctlr#1[,appctlr#2,...]         ] \
      [ VOLUME_GROUP=volume_group#1[,volume_group#2,...]] \
      [ FORCED_VARYON={true|false}                     ] \
      [ VG_AUTO_IMPORT={true|false}                    ] \
      [ FILESYSTEM=/file_system#1[,/file_system#2,...]  ] \
      [ DISK=<raw_disk>[,<raw_disk#2>,...]             ] \
      [ FS_BEFORE_IPADDR={true|false}                  ] \
      [ WPAR_NAME="wpar_name"                          ] \
      [ EXPORT_FILESYSTEM=/expfs#1[,/expfs#2,...]      ] \
      [ EXPORT_FILESYSTEM_V4=/expfs#1[,/expfs#2,...]   ] \
      [ STABLE_STORAGE_PATH="/fs3"                     ] \
      [ NFS_NETWORK="nfs_network"                      ] \
      [ MOUNT_FILESYSTEM=/nfs_fs1;/expfs1,/nfs_fs2;,... ] \
      [ MIRROR_GROUP=<replicated_resource>             ] \
      [ FALLBACK_AT=<FALLBACK_TIMER>                   ]
```

注：值 appctlr 是 application_controller 的缩写。

```
clmgr query resource_group [ <resource_group>[,<rg#2>,...] ]
clmgr delete resource_group {<resource_group>[,<rg#2>,...] |
                             ALL}
clmgr online { resource_group <resource_group>[,<rg#2>,...] | ALL} \
      [ NODES={<node>[,<node#2>,...] | ALL}]
clmgr offline resource_group {<resource_group>[,<rg#2>,...] | ALL} \
      [ NODES={<node>[,<node#2>,...] | ALL} ]
```

注：The special ALL target for the NODES attribute is only applicable to concurrent resource groups.

```
clmgr move resource_group <resource_group>[,<rg#2>,...] \
      {NODE|SITE}=<node_or_site_label> \
      [ SECONDARY={false|true} ] \
      [ STATE={online|offline} ] \
```

注：仅当集群中已配置站点时，*SITE* 和 *SECONDARY* 属性才适用。如果未显式指定 *STATE*，那么资源组 *STATE* 将保持不变。*resource_group* 的别名是 rg。

回退计时器

```
clmgr add fallback_timer <timer> \
      [ YEAR=<####> ] \
      [ MONTH=<{1..12 | Jan..Dec}> ] \
      [ DAY_OF_MONTH=<{1..31}> ] \
      [ DAY_OF_WEEK=<{0..6 | Sun..Sat}> ] \
        HOUR=<{0..23}> \
        MINUTE=<{0..59}>
clmgr modify fallback_timer <timer> \
      [ YEAR=<{####}> ] \
      [ MONTH=<{1..12 | Jan..Dec}> ] \
      [ DAY_OF_MONTH=<{1..31}> ] \
      [ DAY_OF_WEEK=<{0..6 | Sun..Sat}> ] \
      [ HOUR=<{0..23}> ] \
      [ MINUTE=<{0..59}> ] \
      [ REPEATS=<{0,1,2,3,4 |
        Never,Daily,Weekly,Monthly,Yearly}> ]
clmgr query fallback_timer [<timer>[,<timer#2>,...] ]
clmgr delete fallback_timer {<timer>[,<timer#2>,...] |\
                            ALL}
```

注：*fallback_timer* 的别名是 fa 和 timer。

持久性 IP/标签

```
clmgr add persistent_ip <persistent_IP> \
      NETWORK=<network> \
      [ {NETMASK=< 255.255.255.0 | PREFIX=1..128} ] \ ]
      [ NODE=<node>   ]
clmgr modify persistent_ip <persistent_label> \
      [ NAME=<new_persistent_label> ] \
      [ NETWORK=<new_network> ] \
      [ NETMASK=<node> 255.255.255.0 | PREFIX=1..128} ] \ ]
```

注：除非底层网络使用不同的协议（IPv4 相对于 IPv6），否则将忽略为 NETMASK 或 PREFIX 提供的任何值。使用不同协议时，将需要 NETMASK 或 PREFIX。

```
clmgr query persistent_ip [ <persistent_IP>[,<pIP#2>,...] ]
clmgr delete persistent_ip {<persistent_IP>[,<pIP#2>,...] |
                             ALL}
clmgr move persistent_ip <persistent_IP> \
                         INTERFACE=<new_interface>
```

注：*persistent_ip* 的别名是 pe。

服务 IP/标签

```
clmgr add service_ip <service_ip> \
      NETWORK=<network> \
      [ {NETMASK=<255.255.255.0 | PREFIX=1..128} ] \
      [ HWADDR=<new_hardware_address> ] \
      [ SITE=<new_site> ]
```

```
clmgr modify service_ip <service_ip> \
      [ NAME=<new_service_ip> ] \
      [ NETWORK=<new_network> ] \
      [ {NETMASK=<###.###.###.###> | PREFIX=1..128} ] \
      [ HWADDR=<new_hardware_address> ] \
      [ SITE=<new_site> ]
clmgr query service_ip [ <service_ip>[,<service_ip#2>,...] ]
clmgr delete service_ip {<service_ip>[,<service_ip#2>,,...] | ALL}
clmgr move service_ip <service_ip> \
      INTERFACE=<new_interface>
```

注：如果未指定 NETMASK/PREFIX 属性，那么将使用底层网络的子网掩码或前缀值。*service_ip* 的别名是 si。

应用程序控制器

```
clmgr add application_controller <application_controller> \
      STARTSCRIPT="/path/to/start/script" \
      STOPSCRIPT ="/path/to/stop/script" \
      [ MONITORS=<monitor>[,<monitor#2>,...] ] \
      [ STARTUP_MODE={background|foreground}
clmgr modify application_controller <application_controller> \
      [ NAME=<new_application_controller_label> ] \
      [ STARTSCRIPT="/path/to/start/script" ] \
      [ STOPSCRIPT ="/path/to/stop/script" ] \
      [ MONITORS=<monitor>[,<monitor#2>,...] ] \
      [ STARTUP_MODE={background|foreground}
clmgr query application_controller [ <appctlr>[,<appctlr#2>,...] ]
clmgr delete application_controller {<appctlr>[,<appctlr#2>,...] | \
                                     ALL}
clmgr manage application_controller {suspend|resume} \
      <application_controller> \
      RESOURCE_GROUP=<resource_group>
clmgr manage application_controller {suspend|resume} ALL
```

注：*appctlr* 值是 *application_controller* 的缩写。*application_controller* 的别名是 ac 和 app。

应用程序监视器

```
clmgr add application_monitor <monitor> \
      TYPE=Process \
      MODE={longrunning|startup|both} \
      PROCESSES="pmon1,dbmon,..."  \
      OWNER="<processes_owner_name>" \
      [ APPLICATIONS=<appctlr#1>[,<appctlr#2>,...] ] \
      [ STABILIZATION="1 .. 3600" ] \
      [ RESTARTCOUNT="0 .. 100" ] \
      [ FAILUREACTION={notify|fallover} ] \
      [ INSTANCECOUNT="1 .. 1024" ] \
      [ RESTARTINTERVAL="1 .. 3600" ] \
      [ NOTIFYMETHOD="</script/to/notify>" ] \
      [ CLEANUPMETHOD="</script/to/cleanup>" ] \
      [ RESTARTMETHOD="</script/to/restart>" ]

clmgr add application_monitor <monitor> \
      TYPE=Custom \
      MODE={longrunning|startup|both} \
      MONITORMETHOD="/script/to/monitor" \
      [ APPLICATIONS=<appctlr#1>[,<appctlr#2>,...] ] \
      [ STABILIZATION="1 .. 3600" ] \
      [ RESTARTCOUNT="0 .. 100" ] \
      [ FAILUREACTION={notify|fallover} ] \
      [ MONITORINTERVAL="1 .. 1024" ] \
      [ HUNGSIGNAL="1 .. 63" ] \
```

```
      [ RESTARTINTERVAL="1 .. 3600"  ] \
      [ NOTIFYMETHOD="</script/to/notify>" ] \
      [ CLEANUPMETHOD="</script/to/cleanup>" ] \
      [ RESTARTMETHOD="</script/to/restart>" ]
```

注：STABILIZATION 的缺省值为 180。RESTARTCOUNT 的缺省值为 3。

```
clmgr modify application_monitor <monitor> \
      [ See the "add" action, above, for a list
        of supported modification attributes. ]
clmgr query application_monitor [ <monitor>[,<monitor#2>,...] ]
clmgr delete application_monitor {<monitor>[,<monitor#2>,...] | ALL}
```

注：*appctlr* 值是 *application_controller* 的缩写。*application_monitor* 的别名是 am 和 mon。

依赖性

```
# Temporal Dependency (parent ==> child)
clmgr add dependency \
      PARENT=<rg#1> \
      CHILD="<rg#2>[,<rg#2>,...]"
clmgr modify dependency <parent_child_dependency> \
      [ TYPE=PARENT_CHILD ] \
      [ PARENT=<rg#1> ] \
      [ CHILD="<rg#2>[,<rg#2>,...]" ]

# Temporal Dependency (start/stop after)
clmgr add dependency \
      {STOP|START}="<rg#2>[,<rg#2>,...]" \
      AFTER=<rg#1>
clmgr modify dependency \
      [ TYPE={STOP_AFTER|START_AFTER} ] \
      [ {STOP|START}="<rg#2>[,<rg#2>,...]" ] \
      [ AFTER=<rg#1> ]

# Location Dependency (colocation)
clmgr add dependency \
      SAME={NODE|SITE } \
      GROUPS="<rg1>,<rg2>[,<rg#n>,...]"
clmgr modify dependency <colocation_dependency> \
      [ TYPE={SAME_NODE|SAME_SITE} ] \
      GROUPS="<rg1>,<rg2>[,<rg#n>,...]"

# Location Dependency (anti-colocation)
clmgr add dependency \
      HIGH="<rg1>,<rg2>,..." \
      INTERMEDIATE="<rg3>,<rg4>,..." \
      LOW="<rg5>,<rg6>,..."
clmgr modify dependency <anti-colocation_dependency> \
      [ TYPE=DIFFERENT_NODES ] \
      [ HIGH="<rg1>,<rg2>,..." ] \
      [ INTERMEDIATE="<rg3>,<rg4>,..." ] \
      [ LOW="<rg5>,<rg6>,..." ]

# Acquisition/Release Order
clmgr add dependency \
      TYPE={ACQUIRE|RELEASE} \
      { SERIAL="{<rg1>,<rg2>,...|ALL}" |
      PARALLEL="{<rg1>,<rg2>,...|ALL}" }
clmgr modify dependency \
      TYPE={ACQUIRE|RELEASE} \
      { SERIAL="{<rg1>,<rg2>,...|ALL}" |
      PARALLEL="{<rg1>,<rg2>,...|ALL}" }

clmgr query dependency [ <dependency> ]
```

```
clmgr delete dependency {<dependency> | ALL} \
      [ TYPE={PARENT_CHILD|STOP_AFTER|START_AFTER| \
      SAME_NODE|SAME_SITE|DIFFERENT_NODES} ]
clmgr delete dependency RESOURCE_GROUP=<RESOURCE_GROUP>
```

注： *dependency* 的别名是 de。

磁带

```
clmgr add tape <tape> \
      DEVICE=<tape_device_name> \
      [ DESCRIPTION=<tape_device_description> ] \
      [ STARTSCRIPT="</script/to/start/tape/device>" ] \
      [ START_SYNCHRONOUSLY={no|yes} ] \
      [ STOPSCRIPT="</script/to/stop/tape/device>" ] \
      [ STOP_SYNCHRONOUSLY={no|yes} ]
clmgr modify tape <tape> \
      [ NAME=<new_tape_label> ] \
      [ DEVICE=<tape_device_name> ] \
      [ DESCRIPTION=<tape_device_description> ] \
      [ STARTSCRIPT="</script/to/start/tape/device>" ] \
      [ START_SYNCHRONOUSLY={no|yes} ] \
      [ STOPSCRIPT="</script/to/stop/tape/device>" ] \
      [ STOP_SYNCHRONOUSLY={no|yes} ]
clmgr query tape [ <tape>[,<tape#2>,...] ]
clmgr delete tape {<tape> | ALL}
```

注： *tape* 的别名是 tp。

文件集合

```
clmgr add file_collection <file_collection> \
      FILES="/path/to/file1,/path/to/file2,..." \
      [ SYNC_WITH_CLUSTER={no|yes}  ] \
      [ SYNC_WHEN_CHANGED={no|yes} ] \
      [ DESCRIPTION="<file_collection_description>" ]
clmgr modify file_collection <file_collection> \
      [ NAME="<new_file_collection_label>" ] \
      [ ADD="/path/to/file1,/path/to/file2,..." ] \
      [ DELETE={"/path/to/file1,/path/to/file2,..."|ALL} ] \
      [ REPLACE={"/path/to/file1,/path/to/file2,..."|""} ] \
      [ SYNC_WITH_CLUSTER={no|yes}  ] \
      [ SYNC_WHEN_CHANGED={no|yes} ] \
      [ DESCRIPTION="<file_collection_description>" ]
clmgr query file_collection [ <file_collection>[,<fc#2>,...]]
clmgr delete file_collection {<file_collection>[,<fc#2>,...]|
                              ALL}
clmgr sync file_collection <file_collection>
```

注： REPLACE 属性将所有现有文件替换为指定集合。 *file_collection* 的别名是 fc 和 fi。

快照

```
clmgr add snapshot <snapshot> \
      DESCRIPTION="<snapshot_description>" \
      [ METHODS="method1,method2,..." ]
clmgr add snapshot <snapshot> TYPE="xml"
clmgr modify snapshot <snapshot> \
      [ NAME="<new_snapshot_label>" ] \
      [ DESCRIPTION="<snapshot_description>" ]
clmgr query snapshot [ <snapshot>[,<snapshot#2>,...] ]
clmgr view snapshot <snapshot> \
      [ TAIL=<number_of_trailing_lines> ] \
      [ HEAD=<number_of_leading_lines> ] \
      [ FILTER=<pattern>[,<pattern#2>,...] ] \
```

```
      [ DELIMITER=<alternate_pattern_delimiter> ] \
      [ CASE={insensitive|no|off|false} ]
clmgr delete snapshot {<snapshot>[,<snapshot#2>,...] |
                       ALL}
clmgr manage snapshot restore <snapshot> \
      [ CONFIGURE={yes|no} ] \
      [ FORCE={no|yes} ]
```

注：view 操作显示快照的 .info 文件的内容（如果该文件存在）。*snapshot* 的别名是 sn 和 ss。

```
clmgr manage snapshot restore <snapshot> \
      NODES=<HOST>,<HOST#2> \
      REPOSITORIES=<DISK>[,<BACKUP>][:<DISK>[,<BACKUP>]] \
      [ CLUSTER_NAME=<NEW_CLUSTER_LABEL> ] \
      [ CONFIGURE={yes|no} ] \
      [ FORCE={no|yes} ]
```

注：对于 REPOSITORIES 选项，在冒号后面指定的任何磁盘都应用于第二个站点。当您复原链接集群快照时，在 REPOSITORIES 选项中冒号后面指定的任何磁盘都适用于第二个站点。

方法

```
clmgr add method <method_label> \
      TYPE=snapshot \
      FILE=<executable_file> \
      [ DESCRIPTION=<description> ]
clmgr add method <method_label> \
      TYPE=verify \
      FILE=<executable_file> \
      [ SOURCE={script|library} ] \
      [ DESCRIPTION=<description> ]
clmgr modify method <method_label> \
      TYPE={snapshot|verify} \
      [ NAME=<new_method_label> ] \
      [ DESCRIPTION=<new_description> ] \
      [ FILE=<new_executable_file> ]
clmgr add method <method_label> \
      TYPE=notify \
      CONTACT=<number_to_dial_or_email_address> \
      EVENT=<event>[,<event#2>,...] \
      [ NODES=<node>[,<node#2>,...] ] \
      [ FILE=<message_file> ] \
      [ DESCRIPTION=<description> ] \
      [ RETRY=<retry_count> ] \
      [ TIMEOUT=<timeout> ]
```

注：NODES 的缺省值为本地节点。

```
clmgr modify method <method_label> \
      TYPE=notify \
      [ NAME=<new_method_label> ] \
      [ DESCRIPTION=<description> ] \
      [ FILE=<message_file> ] \
      [ CONTACT=<number_to_dial_or_email_address> ] \
      [ EVENT=<cluster_event_label> ] \
      [ NODES=<node>[,<node#2>,...] ] \
      [ RETRY=<retry_count> ] \
      [ TIMEOUT=<timeout> ]

clmgr query method [ <method>[,<method#2>,...] ] \
      [ TYPE={notify|snapshot|verify} ]
clmgr delete method {<method>[,<method#2>,...] | ALL} \
      [ TYPE={notify|snapshot|verify} ]
clmgr verify method <method>
```

注：verify 操作只能应用于 notify 方法。如果一个以上的方法利用同一事件，并且指定了该事件，那么将同时调用两个方法。*method* 的别名是 me。

Log

```
clmgr modify logs ALL DIRECTORY="<new_logs_directory>"
clmgr modify log {<log>|ALL} \
      [ DIRECTORY="{<new_log_directory>"|DEFAULT} ]
      [ FORMATTING={none|standard|low|high} ] \
      [ TRACE_LEVEL={low|high} ]
      [ REMOTE_FS={true|false} ]
clmgr query log [ <log>[,<log#2>,...] ]
clmgr view log [ {<log>|EVENTS} ] \
      [ TAIL=<number_of_trailing_lines> ] \
      [ HEAD=<number_of_leading_lines> ] \
      [ FILTER=<pattern>[,<pattern#2>,...] ] \
      [ DELIMITER=<alternate_pattern_delimiter> ] \
      [ CASE={insensitive|no|off|false} ]
clmgr manage logs collect \
      [ DIRECTORY="<directory_for_collection>" ] \
      [ NODES=<node>[,<node#2>,...] ] \
      [ RSCT_LOGS={yes|no} ] \
```

注：如果为 DIRECTORY 属性指定了 DEFAULT，那么将恢复原始的缺省 PowerHA SystemMirror 目录值

FORMATTING 属性仅适用于 hacmp.out 日志，对于所有其他日志，将忽略此属性。FORMATTING 和 TRACE_LEVEL 属性仅适用于 hacmp.out 和 clstrmgr.debug 日志，对于所有其他日志，将忽略这两个属性。

如果指定 ALL 来代替日志名称，那么会将提供的 DIRECTORY 和 REMOTE_FS 修改应用于所有日志

如果指定 EVENTS 来代替日志名称，那么将显示一个事件摘要报告。

卷组

```
clmgr add volume_group [ <vgname> ] \
      NODES="<node#1>,<node#2>[,...>]" \
      PHYSICAL_VOLUMES="<disk#1>[,<disk#2>,...]" \
      [ TYPE={original|big|scalable|legacy} ] \
      [ RESOURCE_GROUP=<RESOURCE_GROUP> ] \
      [ PPART_SIZE={1|2|4|8|16|32|64|128|256|512|1024} ] \
      [ MAJOR_NUMBER=## ] \
      [ CONCURRENT_ACCESS={false|true} ] \
      [ ACTIVATE_ON_RESTART={false|true} ] \
      [ QUORUM_NEEDED={true|false} ] \
      [ LTG_SIZE=### ] \
      [ MIGRATE_FAILED_DISKS={false|one|pool|remove} ] \
      [ MAX_PHYSICAL_PARTITIONS={32|64|128|256|512|768|1024} ] \
      [ MAX_LOGICAL_VOLUMES={256|512|1024|2048} ] \
      [ STRICT_MIRROR_POOLS={no|yes|super} ] \
      [ MIRROR_POOL_NAME="<mp_name>" ] \
      [ CRITICAL={false|true} ] \
      [ FAILUREACTION={halt|notify|fence|
                       stoprg|moverg} ] \
      [ NOTIFYMETHOD=</file/to/invoke> ]
```

注：设置卷组主数可能导致命令在当前不具有主数的节点上无法成功执行。在更改此设置之前，请检查在所有节点上通常可用的主数。

```
clmgr modify volume_group <vgname> \
      [ ADD=<disk#n> [ MIRROR_POOL_NAME="<mp_name>" ] ] \
```

```
      [ REMOVE=<disk#n> ] \
      [ TYPE={big|scalable} ] \
      [ ENHANCED_CONCURRENT_MODE={false|true} ] \
      [ ACTIVATE_ON_RESTART={false|true} ] \
      [ QUORUM NEEDED={true|false} ] \
      [ LTG_SIZE=### ] \
      [ MIGRATE_FAILED_DISKS={false|one|pool|remove} ] \
      [ MAX_PHYSICAL_PARTITIONS={32|64|128|256|512|768|1024} ] \
      [ MAX_LOGICAL_VOLUMES={256|512|1024|2048} ] \
      [ STRICT_MIRROR_POOLS={off|on|super} ] \
      [ CRITICAL={false|true} ] \
      [ FAILUREACTION={halt|notify|fence|
                       stoprg|moverg} ] \
      [ NOTIFYMETHOD="</file/to/invoke>" ] \
      [ SCSIPR_ACTION={clear} ]
```

注： 如果 ENHANCED_CONCURRENT_MODE 设置为 false，那么将自动建立快速磁盘接管。

MAX_PHYSICAL_PARTITIONS、MAX_LOGICAL_VOLUMES 和 MIRROR_POOL_NAME 仅适用于可伸缩卷组。

```
clmgr query volume_group [ <vg#1>[,<vg#2>,...] ] clmgr delete volume_group
      {<volume_group> [,<vg#2>,...] | ALL}
clmgr discover volume_groups
```

注： *volume_group* 的别名是 vg。

逻辑卷

```
clmgr add logical_volume [ <lvname> ] \
      VOLUME_GROUP=<vgname> \
      LOGICAL_PARTITIONS=## \
            [ DISKS="<disk#1>[,<disk#2>,...]" ] \
      [ TYPE={jfs|jfs2|sysdump|paging|
           jfslog|jfs2log|aio_cache|boot} ] \
      [ POSITION={outer_middle|outer_edge|center|
                inner_middle|inner_edge  } ] \
      [ PV_RANGE={minimum|maximum} ] \
      [ MAX_PVS_FOR_NEW_ALLOC=## ] \
      [ LPART_COPIES={1|2|3} ] \
      [ WRITE_CONSISTENCY={active|passive|off} ] \
      [ LPARTS_ON_SEPARATE_PVS={yes|no|superstrict} ] \
      [ RELOCATE={yes|no} ] \
            [ LABEL="<label>" ] \
      [ MAX_LPARTS=#### ] \
      [ BAD_BLOCK_RELOCATION={yes|no} ] \
      [ SCHEDULING_POLICY={parallel|sequential
        |parallel_sequential
        |parallel_round_robin} ] \
      [ VERIFY_WRITES={false|true} ] \
            [ ALLOCATION_MAP=<file> ] \
      [ STRIPE_SIZE={4K|8K|16K|32K|64K|128K|256K|512K|
          1M|2M|4M|8M|16M|32M|64M|128M} ] \
      [ SERIALIZE_IO={false|true} ] \
      [ FIRST_BLOCK_AVAILABLE={false|true} ] \
            [ FIRST_COPY_MIRROR_POOL=<mirror_pool> ] \
            [ SECOND_COPY_MIRROR_POOL=<mirror_pool> ] \
            [ THIRD_COPY_MIRROR_POOL=<mirror_pool> ] \
            [ GROUP=<group> ] \
            [ PERMISSIONS=<####> ] \
            [ NODE=<reference_node_in_vg> ]
```

注：STRIPE_SIZE 可能无法与 LPARTS_ON_SEPARATE_PVS、PV_RANGE 或 SCHEDULING_POLICY 结合使用。

```
clmgr query logical_volume [ <lvname>[,<LV#2>,...] ]
clmgr delete logical_volume {[ <lv#1>[,<LV#2>,...]  ] | ALL}
```

注： *logical_volume* 的别名是 lv。

文件系统

```
clmgr add file_system <fsname> \
            VOLUME_GROUP=<group> \
      TYPE=enhanced \
      UNITS=### \
        [ SIZE_PER_UNIT={megabytes|gigabytes|512bytes} ] \
      [ PERMISSIONS={rw|ro} ]
      [ OPTIONS={nodev,nosuid,all} ] \
      [ BLOCK_SIZE={4096|512|1024|2048} ] \
      [ LV_FOR_LOG={ <lvname> | "INLINE" } ] \
        [ INLINE_LOG_SIZE=#### ] \
      [ EXT_ATTR_FORMAT={v1|v2} ] \
      [ ENABLE_QUOTA_MGMT={no|all|user|group} ] \
      [ ENABLE_EFS={false|true} ]
```

注：

1. *BLOCK_SIZE* 以字节为单位。*LOG_SIZE* 以兆字节为单位。
2. 仅当 *INLINE_LOG* 设置为 true 时，才能使用 *LOG_SIZE* 和 *LV_FOR_LOG*。
3. 增强型文件系统的大小为 16 MB。

```
clmgr add file_system <fsname> \
      TYPE=enhanced \
      LOGICAL_VOLUME=<logical_volume> \
      [ PERMISSIONS={rw|ro} ] \
      [ OPTIONS={nodev,nosuid,all} ] \
      [ BLOCK_SIZE={4096|512|1024|2048} ] \
      [ LV_FOR_LOG={ <lvname> | "INLINE" } ] \
      [ INLINE_LOG_SIZE=#### ] \
      [ EXT_ATTR_FORMAT={v1|v2} ] \
      [ ENABLE_QUOTA_MGMT={no|all|user|group} ] \
      [ ENABLE_EFS={false|true} ]
clmgr add file_system <fsname> \
            VOLUME_GROUP=<group> \
      TYPE={standard|compressed|large} \
      UNITS=### \
         [ SIZE_PER_UNIT={megabytes|gigabytes|512bytes} ] \
      [ PERMISSIONS={rw|ro} ] \
      [ OPTIONS={nodev|nosuid|all} ] \
      [ DISK_ACCOUNTING={false|true} ] \
      [ FRAGMENT_SIZE={4096|512|1024|2048} ] \
      [ BYTES_PER_INODE={4096|512|1024|2048|8192|
                         16384|32768|65536|131072} ] \
      [ ALLOC_GROUP_SIZE={8|16|32|64} ] \
      [ LV_FOR_LOG=<lvname> ]
```

注：FRAGMENT_SIZE 仅对于标准文件系统和压缩文件系统有效。

```
clmgr add file_system <fsname> \
      TYPE={standard|compressed|large} \
      LOGICAL_VOLUME=<logical_volume> \
      [ PERMISSIONS={rw|ro} ] \
      [ OPTIONS={nodev|nosuid|all} ] \
      [ DISK_ACCOUNTING={false|true} ] \
      [ FRAGMENT_SIZE={4096|512|1024|2048} ] \
      [ BYTES_PER_INODE={4096|512|1024|2048|8192|
                         16384|32768|65536|131072} ] \
```

```
      [ ALLOC_GROUP_SIZE={8|16|32|64} ] \
      [ LV_FOR_LOG=<lvname> ]

clmgr query file_system [ <fs#1>[,<fs#2>,...] ]
clmgr delete file_system { <fsname>[,<FS#2>,...] | ALL } \
      [ REMOVE_MOUNT_POINT={false|true} ]
```

注： *file_system* 的别名是 fs。

物理卷

```
clmgr query physical_volume \
      [ <disk>[,<disk#2>,...] ] \
      [ NODES=<node>,<node#2>[,<node#3>,...] ] \
      [ TYPE={available|all|tiebreaker} ]
```

注：Node 可以是节点名称或网络可解析的名称，例如主机名或 IP 地址。

磁盘可以是设备名 (hdisk0) 或 PVID (00c3a28ed9aa3512)。

```
clmgr modify physical_volume <disk_name_or_PVID> \
      NAME=<new_disk_name> \
      [ NODE=<reference_node> ] \
      [ ALL_NODES={false|true} ] \
      [ SCSIPR_ACTION={clear} ]
```

注：如果指定的磁盘是使用设备名（如 hdisk#）来提供的，那么 NODE 属性是必需的。如果磁盘是使用 PVID 来指定的，那么无需引用 NODE 属性。

physical_volume 的别名是 pv。

镜像池

```
clmgr add mirror_pool <pool_name> \
      VOLUME_GROUP=<vgname> \
      [ PHYSICAL_VOLUMES="<disk#1>[,<disk#2>,...]" ] \
      [ MODE={sync|async} ] \
      [ ASYNC_CACHE_LV=<lvname> ] \
        [ ASYNC_CACHE_HW_MARK=## ]

clmgr add mirror_pool <pool_name> \
      [ VOLUME_GROUP=<vgname> ] \
      PHYSICAL_VOLUMES="<disk>[,<disk#2>,...]"
```

注：如果对现有镜像池执行 *add* 操作，那么指定物理卷将添加至该镜像池。

```
      clmgr modify mirror_pool <pool_name> \
      [ VOLUME_GROUP=<vgname> ]\
      [ NAME=<new_pool_name> ] \
      [ MODE={sync|async} ] \
      [ FORCE_SYNC={false|true} ] \
      [ ASYNC_CACHE_LV=<lvname> ] \
      [ ASYNC_CACHE_HW_MARK=## ]

clmgr query mirror_pool [ <pool_name>[,<pool#2>,...] ]
clmgr delete mirror_pool <pool_name>,[,<pool#2>,...]| ALL }\
      [ VOLUME_GROUP=<vgname> ]
clmgr delete mirror_pool <pool_name> \
      [ VOLUME_GROUP=<vgname> ] \
      PHYSICAL_VOLUMES="<disk>[,<disk#2>,...]"
```

注：当为 delete 操作指定物理卷时，将从镜像池移除磁盘的列表。如果移除所有磁盘，那么将移除镜像池。

注： *mirror_pool* 的别名是 mp 和 pool。

EFS

```
clmgr add efs \
      MODE=ldap \
      [ PASSWORD=<password> ]
clmgr add efs \
      MODE=shared_fs \
      VOLUME_GROUP=<vgname> \
      SERVICE_IP=<service_ip> \
      [ PASSWORD=<password> ]

clmgr modify efs \
      MODE={ldap|shared_fs} \
      [ VOLUME_GROUP=<vgname> ] \
      [ SERVICE_IP=<service_ip> ] \
      [ PASSWORD=<password> ]

clmgr query efs
clmgr delete efs
```

Report

```
clmgr view report [<report>] \
      [ FILE=<PATH_TO_NEW_FILE> ] \
      [ TYPE={text|html} ]

clmgr view report {nodeinfo|rginfo|lvinfo|
      fsinfo|vginfo|dependencies} \
      [ TARGETS=<target>[,<target#2>,...] ] \
      [ FILE=<PATH_TO_NEW_FILE> ] \
      [ TYPE={text|html} ]

clmgr view report cluster \
      TYPE=html \
      [ FILE=<PATH_TO_NEW_FILE> ] \
      [ COMPANY_NAME="<BRIEF_TITLE>" ] \
      [ COMPANY_LOGO="<RESOLVEABLE_FILE>" ]

clmgr view report availability \
      [ TARGETS=<appctlr>[,<appctlr#2>,...] ] \
      [ FILE=<PATH_TO_NEW_FILE> ] \
      [ TYPE={text|html} ] \
      [ BEGIN_TIME="YYYY:MM:DD" ] \
     [ END_TIME="YYYY:MM:DD" ]
```

注： 当前支持的报告有：
basic、cluster、status、topology、applications、availability、events、nodeinfo、rginfo、networks、vginfo、lvinfo、fsinfo、dependencies 和 roha。其中某些报告提供了重叠信息，但是每个报告也提供了自身独有的信息。

值 appctlr 是 application_controller 的缩写。

MM 必须在 1 到 12 之间。DD 必须在 1 到 31 之间。

如果未提供 BEGIN_TIME，那么将为 END_TIME 之前的 30 天生成报告。

如果未提供 END_TIME，那么缺省值将为当前时间。

report 的别名是 re。

LDAP 服务器

以下语法用于为集群配置一个或多个 LDAP 服务器。

```
clmgr add ldap_server <server>[,<server#2>,...] \
      ADMIN_DN=<admin_distinguished_name> \
      PASSWORD=<admin_password> \
      BASE_DN=<suffix_distinguished_name> \
      SSL_KEY=<full_path_to_key> \
      SSL_PASSWORD=<SSL_key_password> \
      VERSION=<version> \
      DB2_INSTANCE_PASSWORD=<password> \
      ENCRYPTION_SEED=<seed> \
      [ SCHEMA=<schema_type> ] \
      [ PORT={636|###} ]
```

注： *ldap_server* 的别名是 ls。

以下语法用于添加一个或多个已向集群配置的 LDAP 服务器。

```
clmgr add ldap_server <server>[,<server#2>,...] \
      ADMIN_DN=<admin_distinguished_name> \
      PASSWORD=<admin_password> \
      BASE_DN=<suffix_distinguished_name> \
      SSL_KEY=<full_path_to_key> \
      SSL_PASSWORD=<SSL_key_password> \
      [ PORT={636|###} ]
```

注：如果指定了一个以上的服务器，那么这些服务器必须处于对等配置中，从而共享同一端口号。

```
clmgr query ldap_server
clmgr delete ldap_server
```

LDAP 客户机

```
clmgr add ldap_client \
      SERVERS=<LDAP_server>[,<LDAP_server#2>]\
      BIND_DN=<bind_distinguished_name> \
        PASSWORD=<LDAP_admin_password> \
      BASE_DN=<base_dn> \
      SSL_KEY=<full_path_to_key> \
        SSL_PASSWORD=<SSL_key_password> \
      [ PORT={636|###} ] \

clmgr query ldap_client
clmgr delete ldap_client
```

注： *ldap_client* 的别名是 lc。

用户

```
clmgr add/modify user <user_name> \
      [ REGISTRY={local|ldap} ] \
      [ RESOURCE_GROUP=<resource_group> ] \
      [ ID=### ] \
      [ PRIMARY=<group> ] \
      [ PASSWORD="{<password>|}" ] \
      [ CHANGE_ON_NEXT_LOGIN={true|false} ] \
      [ GROUPS=<group#1>[,<group#2>,...] ] \
      [ ADMIN_GROUPS=<group#1>[,<group#2>,...] ] \
      [ ROLES=<role#1>[,<role#2>,...] ] \
      [ SWITCH_USER={true|false} ] \
      [ SU_GROUPS={ALL|<group#1>[,<group#2>,...]} ] \
      [ HOME=<full_directory_path> ] \
      [ SHELL=<defined_in_/etc/shells> ] \
      [ INFO=<user_information> ] \
```

```
[ EXPIRATION=<MMDDhhmmyy> ] \
[ LOCKED={false|true} ] \
[ LOGIN={true|false} ] \
[ REMOTE_LOGIN={true|false} ] \
[ SCHEDULE=<range#1>[,<range#2>,...>] ] \
[ MAX_FAILED_LOGINS={#|0} ] \
[ AUTHENTICATION={compat|files|DCE|ldap} ] \
[ ALLOWED_TTYS=<tty#1>[,<tty#2>,...] ] \
[ DAYS_TO_WARN={#|0} ] \
[ PASSWORD_VALIDATION_METHODS=<meth#1>[,<meth#2>,...]]\
[ PASSWORD_FILTERS=<filter#1>[,<filter#2>,...] ] \
[ MIN_PASSWORDS=<number_of_passwords_before_reuse> ] \
[ REUSE_TIME=<weeks_before_password_reuse> ] \
[ LOCKOUT_DELAY=<weeks_btwn_expiration_and_lockout> ] \
[ MAX_PASSWORD_AGE={0..52} ] \
[ MIN_PASSWORD_LENGTH={0..8} ] \
[ MIN_PASSWORD_ALPHAS={0..8} ] \
[ MIN_PASSWORD_OTHERS={0..8} ] \
[ MAX_PASSWORD_REPEATED_CHARS={0..52} ] \
[ MIN_PASSWORD_DIFFERENT={0..8} ] \
[ UMASK=#### ] \
[ AUDIT_CLASSES=<class#1>[,<class#2>,...] ] \
[ TRUSTED_PATH={nosak|on|notsh|always} ] \
[ PRIMARY_AUTH={SYSTEM|.} ] \
[ SECONDARY_AUTH={NONE|SYSTEM|<token>;<user>} ] \
[ PROJECTS=<project#1>[,<project#2>,...] ] \
[ KEYSTORE_ACCESS={file|none} ] \
    [ ADMIN_KEYSTORE_ACCESS={file|none} ] \
    [ KEYSTORE_MODE={admin|guard} ] \
    [ ALLOW_MODE_CHANGE={false|true} ] \
[ KEYSTORE_ENCRYPTION={RSA_1024|RSA_2048|RSA_4096} ] \
    [ FILE_ENCRYPTION={AES_128_CBC|AES_128_EBC| \
                       AES_192_CBC|AES_192_ECB| \
                       AES_256_CBC|AES_256_ECB} ] \
[ ALLOW_PASSWORD_CHANGE={no|yes} ]
```

注：INFO 字段仅接受字母数字字符，包括空格、下划线 (_) 和连字符 (-)。

注：对于 *add* 操作，*REGISTRY* 指示在何处创建用户。对于 *modify*，它指示要更改指定用户的哪个实例。

注：SCHEDULE 定义允许用户登录此系统的时间。 SCHEDULE 值是项的逗号分隔列表，如下所示：

```
* [!][MMdd[-MMdd]]:hhmm-hhmm
* [!]MMdd[-MMdd][:hhmm-hhmm]
* [!][w[-w]]:hhmm-hhmm
* [!]w[-w][:hhmm-hhmm]
```

其中 *MM* 是月份编号（00 = 一月，11 = 十二月），*dd* 是月份中的日期，*hh* 是一天中的小时 (00 - 23)，*mm* 是小时中的分钟，而 *w* 是一周中的日期（0 = 星期日，6 = 星期六）。惊叹号可以用于指示不允许在指定时间范围进行访问。

可以将 MAX_FAILED_LOGINS、DAYS_TO_WARN、MIN_PASSWORDS 和 REUSE_TIME 设置为零以禁用这些功能。

可以将 LOCKOUT_DELAY 设置为 -1 以禁用这些功能。

```
 clmgr modify user {<user_name> | ALL_USERS} \
ALLOW_PASSWORD_CHANGE={no|yes}
```

注：*ALLOW_PASSWORD_CHANGE* 指示是否允许用户使用 C-SPOC 在整个集群中更改其密码。

```
      clmgr query user TYPE={AVAILABLE|ALLOWED}
clmgr query user RESOURCE_GROUP=<resource_group>
clmgr query user <user_name> \
      [ RESOURCE_GROUP=<resource_group> ]
clmgr delete user <user_name> \
      [ RESOURCE_GROUP=<resource_group> ] \
      [ REMOVE_AUTH_INFO={true|false} ]                    [ REGISTRY={files |LDAP} ]
```

组

```
clmgr add group <group_name>
      [ REGISTRY={local(files)|LDAP} ]
      [ RESOURCE_GROUP=<resource_group> ] \
      [ ID=### ] \
      [ ADMINISTRATIVE={false|true} ] \
      [ USERS=<user#1>[,<user#2>,...] ] \
      [ ADMINS=<admin#1>[,<admin#2>,...] ] \
      [ PROJECTS=<project#1>[,<project#2>,...] ] \
      [ KEYSTORE_MODE={admin|guard} ] \
      [ KEYSTORE_ENCRYPTION={ RSA_1024|RSA_2048|RSA_4096} ] \
      [ KEYSTORE_ACCESS={file|none} ] \


clmgr modify group <group_name> \
      [ RESOURCE_GROUP=<resource_group> ] \
      [ ID=### ] \
      [ ADMINISTRATIVE={false|true} ] \
      [ USERS=<user#1>[,<user#2>,...] ] \
      [ ADMINS=<admin#1>[,<admin#2>,...] ] \
      [ PROJECTS=<project#1>[,<project#2>,...] \
      [ KEYSTORE_MODE={admin|guard} ] \
      [ KEYSTORE_ENCRYPTION={ RSA_1024|RSA_2048|RSA_4096} ] \
      [ KEYSTORE_ACCESS={file|none} ]
```

注：RG 选项对于本地定义的组是必需的。如果未提供 RG 选项，那么假定具有一个 LDAP 组。

```
clmgr query group RESOURCE_GROUP=<resource_group>
clmgr query group <group_name> \
      [ RESOURCE_GROUP=<resource_group> ]

clmgr delete group <group_name> \
      [ RESOURCE_GROUP=<resource_group> ] \
                       [ REGISTRY={files|LDAP} ]
```

注：RG 选项对于本地定义的组是必需的。*group* 的别名是 gp。

存储代理程序

```
clmgr add storage_agent <agent_name> \
      TYPE={ds8k_gm|xiv_rm} \
      ADDRESSES=<IP>[<IP#2>,...] \
      [ USER=<user_id> ] \
      [ PASSWORD=<password> ] \
      [ ATTRIBUTES=<NAME>@<VALUE>[,<NAME#2>@<VALUE#2>,...] ]
clmgr modify storage_agent <agent_name> \
      [ NAME=<new_agent_name> ] \
      [ ADDRESSES=<IP>[<IP#2>,...] ] \
      [ USER=<user_id> ] \
      [ PASSWORD=<password> ] \
      [ ATTRIBUTES=<NAME>@<VALUE>[,<NAME#2>@<VALUE#2>,...] ]
clmgr query storage_agent [ <agent>[,<agent#2>,...] ]
clmgr delete storage_agent {<agent>[,<agent#2>,...] | ALL}
```

注：*storage agent* 的别名是 sta。

存储系统

```
clmgr add storage_system <storage_system_name> \
      TYPE={ds8k_gm|xiv_rm} \
      SITE=<site> \
      AGENTS=<agent>[,<agent#2>,...] \
      VENDOR_ID=<identifier> \
      [ WWNN=<world_wide_node_name> ] \
      [ ATTRIBUTES=<NAME>@<VALUE>[,<NAME#2>@<VALUE#2>,...] ]
clmgr add storage_system <storage_system_name> \
      TYPE=ds8k_inband_mm \
      SITE=<site> \
      VENDOR_ID=<identifier> \
      [ WWNN=<world_wide_node_name> ] \
      [ ATTRIBUTES=<NAME>@<VALUE>[,<NAME#2>@<VALUE#2>,...] ]
clmgr add storage_system <storage_system_name> \
      TYPE=svc \
      SITE=<site> \
      ADDRESSES=<IP>[<IP#2>,...] \
      MASTER=<Master/Auxiliary> \
      PARTNER=<Remote Partner> \
      [ AGENTS=<agent>[,<agent#2>,...] ] \
      [ ATTRIBUTES=<NAME>@<VALUE>[,<NAME#2>@<VALUE#2>,...] ]

clmgr modify storage_system <storage_system_name> \
      [ NAME=<new_storage_system_name> ] \
      [ SITE=<site> ] \
      [ AGENTS=<agent>[,<agent#2>,...] ] \
      [ WWNN=<world_wide_node_name> ] \
      [ VENDOR_ID=<identifier> ] \
      [ ADDRESSES=<IP>[<IP#2>,...] ] \
      [ MASTER=<Master/Auxiliary> ] \
      [ PARTNER=<Remote Partner> ] \
      [ ATTRIBUTES=<NAME>@<VALUE>[,<NAME#2>@<VALUE#2>,...] ]

clmgr query storage_system [ <storage_system>[,<ss#2>,...] ]


clmgr -a VENDOR_ID query storage_system \
      TYPE={ds8k_gm|ds8k_inband_mm|xiv_rm}
```

注：以下查询将列出可用的供应商标识。

```
clmgr delete storage_system {<storage_system>[,<ss#2>,...] | ALL}
```

注：*storage system* 的别名是 sts。

镜像对

```
clmgr add mirror_pair <mirror_pair_name> \
      FIRST_DISK=<disk_1> \
      SECOND_DISK=<disk_2>
clmgr modify mirror_pair <mirror_pair_name> \
      [ NAME=<new_mirror_pair_name> ] \
      [ FIRST_DISK=<disk_1> ] \
      [ SECOND_DISK=<disk_2> ]
clmgr query mirror_pair [ <mirror_pair>[,<mp#2>,...] ]
clmgr delete mirror_pair {<mirror_pair>[,<mp#2>,...] | ALL}
```

注：*mirror_pair* 的别名是 mip。

镜像组

```
: HyperSwap user mirror groups
  clmgr add mirror_group <mirror_group_name> \
        TYPE=ds8k_inband_mm \
```

```
       MG_TYPE=user \
       VOLUME_GROUPS=<volume_group>[,<vg#2>,...] \
       DISKS=<raw_disk>[,<disk#2>,...] \
       [ HYPERSWAP_ENABLED={no|yes} ] \
       [ CONSISTENT={yes|no} ] \
       [ UNPLANNED_HS_TIMEOUT=## ] \
       [ HYPERSWAP_PRIORITY={medium|high} ] \
       [ RECOVERY={manual|auto} ] \
       [ RESYNC={manual|auto} ] \
       [ ATTRIBUTES=<NAME>@<VALUE>[,<NAME#2>@<VALUE#2>,...] ]

 clmgr modify mirror_group <mirror_group_name> \
       [ NAME=<new_mirror_group_name> ] \
       [ VOLUME_GROUPS=<volume_group>[,<vg#2>,...] ] \
       [ DISKS=<raw_disk>[,<disk#2>,...] ] \
       [ STORAGE_SYSTEMS=<storage_system>[,<ss#2>,...] ] \
       [ HYPERSWAP_ENABLED={no|yes} ] \
       [ CONSISTENT={yes|no} ] \
       [ UNPLANNED_HS_TIMEOUT=## ] \
       [ HYPERSWAP_PRIORITY={medium|high} ] \
       [ RECOVERY={manual|auto} ] \
       [ RESYNC={manual|auto} ] \
       [ ATTRIBUTES=<NAME>@<VALUE>[,<NAME#2>@<VALUE#2>,...] ]

: HyperSwap system mirror groups
  clmgr add mirror_group <mirror_group_name> \
       TYPE=ds8k_inband_mm \
       MG_TYPE=system \
       VOLUME_GROUPS=<volume_group>[,<vg#2>,...] \
       DISKS=<raw_disk>[,<disk#2>,...] \
       NODE=<node> \
       HYPERSWAP_ENABLED={no|yes} \
       [ CONSISTENT={yes|no} ] \
       [ UNPLANNED_HS_TIMEOUT=## ] \
       [ HYPERSWAP_PRIORITY={medium|high} ] \
       [ ATTRIBUTES=<NAME>@<VALUE>[,<NAME#2>@<VALUE#2>,...] ]

  clmgr modify mirror_group <mirror_group_name> \
       [ NAME=<new_mirror_group_name> ] \
       [ VOLUME_GROUPS=<volume_group>[,<vg#2>,...] ] \
       [ DISKS=<raw_disk>[,<disk#2>,...] ] \
       [ NODE=<node> ] \
       [ STORAGE_SYSTEMS=<storage_system>[,<ss#2>,...] ] \
       [ HYPERSWAP_ENABLED={no|yes} ] \
       [ CONSISTENT={yes|no} ] \
       [ UNPLANNED_HS_TIMEOUT=## ] \
       [ HYPERSWAP_PRIORITY={medium|high} ] \
       [ ATTRIBUTES=<NAME>@<VALUE>[,<NAME#2>@<VALUE#2>,...] ]

: HyperSwap repository mirror groups
  clmgr add mirror_group <mirror_group_name> \
       TYPE=ds8k_inband_mm \
       MG_TYPE=repository \
       SITE=<site> \
       NON_HS_DISK=<Non-HyperSwap_disk> \
       HS_DISK=<HyperSwap_disk> \
       [ HYPERSWAP_ENABLED={no|yes} ] \
       [ CONSISTENT={yes|no} ] \
       [ UNPLANNED_HS_TIMEOUT=## ] \
       [ HYPERSWAP_PRIORITY={medium|high} ] \
       [ RESYNC={manual|auto} ] \
       [ ATTRIBUTES=<NAME>@<VALUE>[,<NAME#2>@<VALUE#2>,...] ]

  clmgr modify mirror_group <mirror_group_name> \
       [ NAME=<new_mirror_group_name> ] \
       [ SITE=<node> ] \
       [ NON_HS_DISK=<non-HyperSwap_disk> ] \
```

```
       [ HS_DISK=<HyperSwap_disk> ] \
       [ STORAGE_SYSTEMS=<storage_system>[,<ss#2>,...] ] \
       [ HYPERSWAP_ENABLED={no|yes} ] \
       [ CONSISTENT={yes|no} ] \
       [ UNPLANNED_HS_TIMEOUT=## ] \
       [ HYPERSWAP_PRIORITY={medium|high} ] \
       [ RESYNC={manual|auto} ] \
       [ ATTRIBUTES=<NAME>@<VALUE>[,<NAME#2>@<VALUE#2>,...] ]

: DS8000 Global Mirror and XIV mirror groups
  clmgr add mirror_group <mirror_group_name> \
       TYPE={ds8k_gm|xiv_rm} \
       MODE={sync|async} \
       RECOVERY={auto|manual} \
       [ STORAGE_SYSTEMS=<storage_system>[,<ss#2>,...] ] \
       [ VENDOR_ID=<vendor_specific_identifier> ] \
       [ ATTRIBUTES=<NAME>@<VALUE>[,<NAME#2>@<VALUE#2>,...] ]

  clmgr modify mirror_group <mirror_group_name> \
       [ NAME=<new_mirror_group_name> ] \
       [ MODE={sync|async} ] \
       [ RECOVERY={auto|manual} ] \
       [ STORAGE_SYSTEMS=<storage_system>[,<ss#2>,...] ] \
       [ VENDOR_ID=<vendor_specific_identifier> ] \
       [ ATTRIBUTES=<NAME>@<VALUE>[,<NAME#2>@<VALUE#2>,...] ]

: SVC mirror groups
  clmgr add mirror_group <mirror_group_name> \
       TYPE=svc \
       STORAGE_SYSTEMS=<MASTER_SVC>,<AUXILIARY_SVC> \
       MIRROR_PAIRS=<mirror_pair>[,<mirror_pair#2>,...] ] \
       [ MODE={sync|async} ] \
       [ RECOVERY={auto|manual} ]

  clmgr modify mirror_group <mirror_group_name> \
       [ NAME=<new_mirror_group_name> ] \
       [ STORAGE_SYSTEMS=<MASTER_SVC>,<AUXILIARY_SVC> ] \
       [ MIRROR_PAIRS=<mirror_pair>[,<mirror_pair#2>,...] ] \
       [ MODE={sync|async} ] \
       [ RECOVERY={auto|manual} ]

: Hitachi mirror groups
  clmgr add mirror_group <mirror_group_name> \
       TYPE=hitachi \
       VENDOR_ID=<device_group> \
       HORCM_INSTANCE=<instance> \
       [ MODE={sync|async} ] \
       [ RECOVERY={auto|manual} ] \
       [ HORCM_TIMEOUT=### ] \
       [ PAIR_EVENT_TIMEOUT=### ]

  clmgr modify mirror_group <mirror_group_name> \
       [ NAME=<new_mirror_group_name> ] \
       [ VENDOR_ID=<device_group> ] \
       [ HORCM_INSTANCE=<instance> ] \
       [ MODE={sync|async} ] \
       [ RECOVERY={auto|manual} ] \
       [ HORCM_TIMEOUT=### ] \
       [ PAIR_EVENT_TIMEOUT=### ]

: EMC mirror groups
  clmgr add mirror_group <mirror_group_name> \
       TYPE=emc \
       [ MG_TYPE={composite|device} ] \
       [ MODE={sync|async} ] \
       [ RECOVERY={auto|manual} ] \
       [ CONSISTENT={yes|no} ] \
```

```
        [ VENDOR_ID=<vendor_specific_identifier> ]

  clmgr modify mirror_group <mirror_group_name> \
       [ NAME=<new_mirror_group_name> ] \
       [ MG_TYPE={composite|device} ] \
       [ MODE={sync|async} ] \
       [ RECOVERY={auto|manual} ] \
       [ CONSISTENT={yes|no} ] \
       [ VENDOR_ID=<device_group> ]

: HyperSwap mirror groups
  clmgr {swap|view} mirror_group <mirror_group_name>[,<mg#2>,...] \
        [ NODE=<node_name> ]
  clmgr {swap|view} mirror_group \
        NODES=<node_name>[,<node#2>,...] \
        [ SYSTEM_GROUPS={yes|no} ]
  clmgr {swap|view} mirror_group \
        SITES=<site_name>[,<site#2>] \
        [ SYSTEM_GROUPS={yes|no} ] \
        [ REPOSITORY_GROUP={yes|no} ]
```

注： swap 和 view 属性仅对 DS 系列 Inband (HyperSwap®) 有效。

```
clmgr manage mirror_group refresh
      <mirror_group_name>[,<mg#2>,...] \
      [ NODE=<node_name> ]
clmgr manage mirror_group refresh \
      NODES=<node_name>[,<node#2>,...] \
      [ SYSTEM_GROUPS={yes|no} ]
clmgr manage mirror_group refresh \
      SITES=<site_name>[,<site#2>] \
      [ SYSTEM_GROUPS={yes|no} ] \
      [ REPOSITORY_GROUP={yes|no} ]

: All mirror groups
  clmgr query mirror_group [ <mirror_group>[,<mg#2>,...] ]
  clmgr delete mirror_group {<mirror_group>[,<mg#2>,...] | ALL}
```

注： *mirror_group* 的别名是 mig。

事件

```
cl clmgr add event <EVENT_NAME> \
         FILE=<EXECUTABLE_FILE> \
         [ DESCRIPTION=<EVENT_DESCRIPTION> ]
clmgr modify event <EVENT_NAME> \
         [ NAME=<NEW_EVENT_NAME> ] \
         [ FILE=<EXECUTABLE_FILE> ] \
         [ DESCRIPTION=<EVENT_DESCRIPTION> ]
clmgr modify event <BULTIN EVENT_NAME> \
         [ COMMAND=<COMMAND_OR_FILE> ] \
         [ NOTIFY_COMMAND=<COMMAND_OR_FILE> ] \
         [ RECOVERY_COMMAND=<COMMAND_OR_FILE> ] \
           [ RECOVERY_COUNTER=# ] \
         [ PRE_EVENT_COMMAND=<CUSTOM_EVENT> ] \
         [ POST_EVENT_COMMAND=<CUSTOM_EVENT> ]
clmgr query event [ <EVENT_NAME>[,<EVENT_NAME#2>,...] ]
         [ TYPE={CUSTOM|PREDEFINED|ALL} ]
clmgr delete event { <EVENT_NAME>[,<EVENT_NAME#2>,...] | ALL}
```

注： *event* 的别名是 ev。

HMC

```
clmgr add hmc <HMC> \
            [ TIMEOUT=<###> ] \
            [ RETRY_COUNT=<###> ] \
            [ RETRY_DELAY=<###> ] \
            [ NODES=<node>[,<node#2>,...] ] \
            [ SITES=<site>[,<site#2>,...] ] \
            [ CHECK_HMC=<Yes|No> ]
clmgr modify hmc <HMC> \
            [ TIMEOUT=<###> ] \
            [ RETRY_COUNT=<###> ] \
            [ RETRY_DELAY=<###> ] \
            [ NODES=<node>[,<node#2>,...] ] \
            [ SITES=<site>[,<site#2>,...] ] \
            [ CHECK_HMC=<Yes|No> ]

clmgr query hmc [<HMC>[,<HMC#2>,...]]
clmgr delete hmc {<HMC> | ALL}
```

注：clmgr delete 示例将移除与指定节点关联的指定 HMC 或所有 HMC。如果未指定任何节点，那么将移除所有节点。

CoD

```
clmgr add cod <APPCTRL> \
      [ USE_DESIRED="Yes|No"> ] \
      [ OPTIMAL_MEM=#.## ]  \
      [ OPTIMAL_CPU=# ]  \
      [ OPTIMAL_PU=#.## ] \
      [ OPTIMAL_VP=# ]

clmgr modify cod <APPCTRL> \
      [ USE_DESIRED="Yes|No"> ] \
      [ OPTIMAL_MEM=#.## ] \
      [ OPTIMAL_CPU=# ] \
      [ OPTIMAL_PU=#.## ] \
      [ OPTIMAL_VP=# ]
```

注：

1. 可使用此命令提供运行应用程序控制器所需资源的最佳级别。
2. 如果设置 USE_DESIRED=1，那么将使用预期 LPAR 概要文件级别，它提供应用程序控制器所需资源的最佳级别。
3. 如果设置 USE_DESIRED=0，那么您可更精确地进行控制，使用 OPTIMAL_MEM、OPTIMAL_CPU、OPTIMAL_PU 和 OPTIMAL_VP 值来配置应用程序控制器所需的资源级别。
4. 为应用程序控制器提供一定级别的资源允许 PowerHA SystemMirror 执行操作（DLPAR、On/Off CoD 和 EPCoD）以便为应用程序控制器提供最佳级别的资源。
5. 可通过使用 **clmgr verify cluster** 命令验证集群来检查提供级别。
6. *cod* 的别名为 *roha*、*dlpar* 和 *cuod*。

```
clmgr query cod [<APPCTRL> ]
clmgr delete cod {<APPCTRL> | ALL}
```

示例

在以下示例中，**clmgr** 命令的类属性不区分大小写。例如，在以下命令中，NODES 属性可以是 NODES、nodes 或 Nodes。

```
clmgr create cluster clMain NODES=nodeA,nodeB
```

1. 以下示例将创建一个 PowerHA SystemMirror Standard Edition for AIX 集群，该集群包含名为 nodeA 和 nodeB 的两个节点。该集群名为 haCL，并且具有名为 hdisk5 的存储库磁盘。环境要求为集群使用预先确定的多点广播地址 229.9.3.17。

```
clmgr create cluster haCL NODES=nodeA,nodeB \
      REPOSITORY=hdisk5 \
      CLUSTER_IP=229.9.3.17
clmgr sync cluster
```

注：只是因为环境需要多点广播地址，所以此示例中才需要 CLUSTER_IP 属性。如果未提供多点广播地址，那么系统将根据当时正在使用的地址选择一个地址。

2. 以下示例将使用缺省策略来创建标准（非并发）资源组。该资源组名为 db2RG，包含名为 access1 的服务 IP 地址，并包含名为 db2Controller 的应用程序控制器。该资源组管理名为 vg1 和 vg2 的两个非并发卷组。

```
clmgr add resource_group db2RG SERVICE_IP=access1 \
      APPLICATIONS=db2Controller \
      VOLUME_GROUP=vg1,vg2
clmgr sync cluster
```

3. 可以使用以下命令来检查集群中各种对象的状态。

```
clmgr -a STATE query cluster
clmgr -a STATE query node nodeA
clmgr -a STATE query resource_group rg1
```

注：

- STATE 类返回整个集群的逻辑最坏情况聚集。例如，如果包含四个节点的集群中的集群遇到错误，那么为整个集群返回的状态将报告为错误。
- 运行此命令所返回的值采用标准 ATTR=VALUE 格式。例如，如果集群脱机，那么返回的值为 STATE=OFFLINE。
- 通过使用 **-a** 标志，可以一次检索多个属性。例如，如果运行以下命令，那么您将同时获得集群的名称和状态：

  ```
  clmgr -a STATE,NAME query cluster
  ```

4. 所有操作、类和属性都可以缩写为显式命名的别名或使其唯一的最少数目的字符。以下示例显示一些完整命令，各命令下方显示该命令的缩写版本。

   - ```
     clmgr query resource_group
     clmgr q rg
     ```
   - ```
     clmgr modify node mynode PERSISTENT_IP=myIP NETWORK=myNet
     clmgr mod node mynode pe=myIP netw=myNet
     ```
   - ```
     clmgr online node nodeA
     clmgr start node nodeA
     ```

注：这些操作、类和属性的缩写旨在供您以交互方式对集群使用 **clmgr** 命令时使用。尽管这些缩写可以用在脚本中，但是请避免在脚本中使用这些缩写，因为它们无法提供简单易读的代码。

5. 在命令行中为 **clmgr** 命令提供了帮助信息。如果您不清楚要运行的完整命令，那么您可以尽可能多地输入所知内容，帮助信息就会显示。例如，如果提供无效的对象或值作为命令的一部分，那么帮助信息将仅显示有效的对象或值。运行以下命令作为示例，以查看命令行中如何显示不同的帮助信息。

```
clmgrclmgr view
clmgr view report
clmgr view report -h
```

注： **-h** 标志将请求特定操作的所有有效选项的列表，只能将其用在对象类之后或一组选项对之后。此标志是 **clmgr** 命令中唯一不需要紧接在 clmgr 命令之后放置的标志。
```

以下示例描述 clmgr 命令的一些常见使用方案。已测试所有示例。请将值替换为对于您环境有效的值。以下任务是这些方案的基础，下文中进行了详细描述。

- 创建集群
- 创建资源组
- 检查当前状态
- 查看所有属性和设置
- 根据某些过滤器或条件显示对象
- 使 clmgr 命令更易于使用
- 获取 clmgr 命令的即时帮助

示例：创建标准集群

详细信息：

此集群是具有两个节点的标准集群，并且没有任何相关联的站点。集群名称为 DB2_cluster，节点名为 DBPrimary 和 DBBackup。将在名为 hdisk5 的磁盘上创建存储库磁盘。

例如：

```
1. clmgr create cluster DB2_cluster NODES=DBPrimary,DBBackup \
                                    REPOSITORY=hdisk5
2. clmgr sync cluster
```

注释：

- 存储库磁盘在运行 **clmgr** 命令的节点上进行解析。您可以采用 PVID 或 UUID 格式指定存储库磁盘。
- 未指定脉动信号类型。因此，集群使用缺省类型（即，单点广播通信）。
- **clmgr** 命令不区分大小写。您可以将存储库属性指定为 REPOSITORY、Repository 或 repository。

示例：创建延伸集群

详细信息：

此集群是一个名为 Oracle_cluster 的延伸集群。此集群具有四个节点，名称分别为 Ora1、Ora2、Ora3 和 Ora4。此集群具有两个站点，名称分别为 Ora_Primary 和 Ora_Secondary。名为 Ora_Primary 的站点将管理名为 Ora1 和 Ora2 的节点。名为 Ora_Secondary 的站点将管理名为 Ora3 和 Ora4 的节点。将在名为 hdisk5 的磁盘上创建存储库磁盘。此集群使用多点广播通信作为脉动信号类型。

例如：

```
1. clmgr create cluster Oracle_cluster \
         NODES=Ora1,Ora2,Ora3,Ora4 \
         TYPE=SC \
         REPOSITORY=hdisk5 \
         HEARTBEAT_TYPE=multicast
2. clmgr add site Ora_Primary NODES=Ora1,Ora2
3. clmgr add site Ora_Secondary NODES=Ora3,Ora4
4. clmgr sync cluster
```

Comment:

存储库磁盘在运行 **clmgr** 命令的节点上进行解析。您可以采用 PVID 或 UUID 格式指定存储库磁盘。

示例：创建链接集群

详细信息：

此集群是一个名为 SAP-cluster 的链接集群。此集群具有四个节点，名称分别为 SAP-A1、SAP-A2、SAP-B1 和 SAP-B2。此集群具有两个站点，名称分别为 SAP_Active 和 SAP_Backup。名为 SAP_Active 的站点将管理名为 SAP-A1 和 SAP-A2 的节点。名为 SAP_Backup 的站点将管理名为 SAP-B1 和 SAP-B2 的节点。SAP_Active 站点上的存储库磁盘的名称为 hdisk5。SAP_Backup 站点上的存储库磁盘的名称为 hdisk11。此集群使用单点广播通信作为脉动信号类型。

例如：

1.
```
clmgr create cluster SAP-cluster \
      NODES=SAP-A1,SAP-A2,SAP-B1,SAP-B2 \
      TYPE=LC \
      HEARTBEAT_TYPE=unicast
```
2.
```
clmgr add site SAP_Active NODES=SAP-A1,SAP-A2 REPOSITORY=hdisk5
```
3.
```
clmgr add site SAP_Backup NODES=SAP-B1,SAP-B2 REPOSITORY=hdisk11
```
4.
```
clmgr sync cluster
```

注释：

- 链接集群要求每个站点都具有存储库磁盘。您必须确定每个站点的存储库磁盘。
- 存储库磁盘在 **clmgr** 命令能够与其通信的第一个节点上进行解析。对于链接集群，为每个站点定义的第一个节点是 **clmgr** 命令尝试与其通信的站点。在此示例中，hdisk5 存储库磁盘在 SAP-A1 节点上进行解析，hdisk11 存储库磁盘在 SAP-B1 节点上进行解析。
- 您可以采用 PVID 或 UUID 格式指定存储库磁盘。

示例：创建资源组

详细信息：

此资源组将是使用缺省策略的标准（非并发）资源组，并将命名为 db2RG。该资源组将包含名为 access1 的服务 IP 地址，并包含名为 db2Controller 的应用程序控制器。此外，该资源组还将管理名为 vg1 和 vg2 的两个卷组，这两个卷组都不是并发卷组。

示例：

-
```
clmgr add resource_group db2RG SERVICE_IP=access1 \
      APPLICATIONS=db2Controller \
      VOLUME_GROUP=vg1,vg2
```
-
```
clmgr sync cluster
```

示例：检查当前状态

详细信息：

在很多时候，确切了解给定对象所处的状态以便能够采取适当的操作，这一点很重要。使用 clmgr，就可以通过 query 操作来实现此目的。

示例：

-
```
clmgr -a STATE query cluster
```
-
```
clmgr -a STATE query site siteA
```
-
```
clmgr -a STATE query node nodeA
```
-
```
clmgr -a STATE query resource_group rg1
```

注释：

- 对于 site 和 cluster 类，所返回的 STATE 是成员节点的逻辑最坏情况聚集。例如，在包含四个节点的集群中，即使只有一个节点遇到错误，整个集群的状态也将报告为 ERROR。
- 返回的值将采用标准 ATTR=VALUE 格式，例如 STATE=OFFLINE。如果您仅需要值，那么可以将几个其他标志与 **-a** 有效组合以实现此目的。使用标志组合 **-cSa** 将仅返回 VALUE，例如 OFFLINE。此做法每次将仅对单个值起作用。
- 可以使用 **-a** 标志一次检索多个属性，例如，**-a NAME,STATE**。此外，**-a** 标志不区分大小写 (-a Name,state)，并且支持通配符 (-a N*)。

示例：查看所有属性和设置

详细信息：

PowerHA SystemMirror 是这样一个产品：一旦已安装并全面测试该产品，就通常不再与该产品主动交互，直至发生问题或需要进行某种维护。发生此类情况时，就需要能够查看集群的内容以及所有设置。使用 clmgr，可通过 query 操作来实现此目的，同时可以选择请求特定格式，即冒号分隔格式或 XML。以下命令示例使用资源组，但原则对于所有对象类而言都相同。

示例：

- `clmgr query resource_group`
- `clmgr query resource_group rg1,rg2`
- `clmgr -c query resource_group rg1,rg2`
- `clmgr -x query resource_group rg1,rg2`
- `clmgr -v query resource_group`
- `clmgr -cv query resource_group`
- `clmgr -xv query resource_group`

注释：

- 当 query 命令中未提供任何目标对象，并且未使用详细标志 **-v** 时，将显示对象的简单列表。
- 当 query 命令中提供了一个或多个目标对象时，将显示这些对象的所有已知属性或设置。这将覆盖 **-v** 标志。
- 当 **-v** 标志与 query 命令结合使用时，将显示指定类的所有已知对象的所有已知属性或设置。
- 当显示详细属性或设置时，缺省情况下它们以 ATTR=VALUE 格式显示，每行显示一条。如果提供 **-c**，那么将使用冒号分隔的格式在一行中显示所有值。如果提供 **-x**，那么将以简单 XML 格式显示所有属性和值。

示例：根据某些过滤器或条件显示对象

详细信息：

以下情况并不罕见：为给定类（例如资源组）定义大量对象，或在给定类中定义大量设置。这样，要查找真正需要的信息有时就变得具有挑战性。幸运的是，clmgr 提供了用于为 query 操作指定过滤条件的功能来解决此问题。

示例：

- `clmgr query file_collection FILE="*rhosts*"`
- `clmgr query resource_group CURRENT_NODE=`get_local_nodename``

注释：

- 第一个示例显示用于查找包含特定值或设置的对象的一种简单方法；在此例中，该对象是包含名为 rhosts 的文件的文件集合（请注意此处支持通配符）。
- 第二个示例显示一个很好的实际示例，它说明如何查找与动态值匹配的对象。在此例中，示例显示如何获取本地节点上当前正在运行的所有资源组的列表。
- 可以将此过滤功能与 **-a** 标志组合使用，以提供非常强大而灵活的数据检索。

示例：使 clmgr 更易于使用

详细信息：

clmgr 中的所有内容都不区分大小写，这有助于减少令人沮丧的输入错误。此外，所有操作、类和属性或选项都可以缩写为显式命名的别名（例如，将 start 用作 online 的别名，或将 resource_group 缩写为 rg）或使其唯一的最少数目的字母。以下几对命令在功能方面完全相同。

示例：

- ```
  clmgr query resource_group
  clmgr q rg
  ```
- ```
  clmgr modify node mynode PERSISTENT_IP=myIP NETWORK=myNet
  clmgr mod node mynode pe=myIP netw=net_ether_0
  ```
- ```
  clmgr online node nodeA
  clmgr start node nodeA
  ```

注释：

操作和类的缩写旨在供您在终端中以交互方式使用 clmgr 时使用。尽管这些缩写也可以用在脚本中，但是强烈建议在脚本中使用操作和类的完整名称。这样做可以提供更易读和易于使用的代码。

示例：获取 clmgr 的即时帮助

详细信息：

始终可以通过联机方式获取 clmgr 的帮助。但是，启动 Web 浏览器通常不方便，并且有时不切实际或甚至不可能完成。因此 clmgr 提供了尽可能多的内建帮助，这样您就能够立即获得所需的帮助。以下情况会提供一种帮助类型：需要已知的一组对象或值中的某个对象或值时。如果提供了无效的对象或值，那么不仅会显示相应的错误消息，还会显示对于该操作有效的对象或值的列表。这对于帮助您克服持续发生的输入错误而言非常有用。当您不确定需要什么操作、类或对象时，可从 clmgr 获取更多帮助。只需尽可能多地输入所知内容，clmgr 就会告知您接下来的所有可能值。然后，您只需选择其中的某个值就可以继续操作。尝试运行以下命令，以查看 clmgr 准备为您提供的帮助的一些示例。

示例：

- ```
  clmgr
  ```
- ```
  clmgr view
  ```
- ```
  clmgr view report
  ```
- ```
  clmgr view report -h
  ```

注释：

在命令行中，当在对象类或某组选项对之后提供 **-h** 标志时，此标志将请求此特定操作的所有有效选项的列表。此标志是 **clmgr** 命令中唯一不需要紧接在 **clmgr** 命令本身之后放置的标志。

相关信息:

资源组依赖关系
  ```

clpasswd 命令

用途

更改集群中或资源组中所有节点上的当前用户密码。

语法

```
clpasswd [-g resource group] user
```

描述

"集群密码"实用程序 (**clpasswd**) 使用户可以从单个节点中更改自身在集群中所有节点上的密码或者 PowerHA SystemMirror 管理员指定的资源组中的密码。要使用户能够跨集群节点更改其密码，PowerHA SystemMirror 管理员需要将不具有根特权的任何用户添加到允许更改其密码的用户列表中。

"集群密码"实用程序还可以替换 SMIT 快速路径 **cl_passwd** 中的 AIX 密码实用程序。

下表显示了基于用户授权以及活动的密码实用程序，用户密码在哪些位置中更改：

用户权限级别	当系统密码实用程序链接到 clpasswd 并且调用 /bin/passwd 时	当系统密码实用程序处于活动状态时
用户有权跨集群更改密码	将在所有集群节点上更改密码。	将在所有集群节点上更改密码。
用户无权跨集群更改密码	将仅在本地节点上更改密码。	将不更改密码。

标志

-g 指定用户可以在其中更改其密码的资源组的名称。将在指定资源组中的每个节点上更改密码。

user
要更改其密码的用户的用户名。

示例

```
clpasswd -g rg1 myusername
```

clRGinfo 命令

用途

创建报告以显示一个或多个指定资源组的位置和状态。

语法

```
clRGinfo [-h][-v][-s|-c][-t][-p][-a][-m][resgroup1] [resgroup2]...
```

描述

如果集群服务未在本地节点上运行，那么此 **clRGinfo** 命令标识集群服务在其上处于活动状态的节点，并从活动集群管理器获取资源组信息。如果使用此命令时未指定任何资源组，那么将显示有关所有已配置资源组的信息。

此命令的输出将显示资源组的全局状态以及本地节点上该资源组的特殊状态。

资源组的主要实例可处于下列其中一种状态：

联机 此资源组的所有资源处于活动状态。

错误 PowerHA SystemMirror 处理该资源组时发生了错误。

非受管
集群服务已停止（带有非受管选项）。

脱机 该资源组处于不活动状态。

集群事件进行时，资源组可处于以下过渡状态：

正在获取
正在激活该资源组的资源。

正在释放
正在释放该资源组的资源。

临时错误
发生了可恢复错误。

集群使用站点和复制资源时，包含已复制资源的资源组具有用于管理复制端点的主辅实例。**clRGinfo** 命令显示资源组的（辅助）实例的以下状态：

联机（辅助）
此资源组的所有辅助资源处于活动状态。

错误（辅助）
PowerHA SystemMirror 处理资源组的（辅助）资源时发生了错误。

非受管（辅助）
集群服务已停止（带有非受管选项）。

脱机（辅助）
资源组的辅助实例处于不活动状态。

获取（辅助）
正在激活该资源组的辅助资源。

释放（辅助）
正在释放该资源组的辅助资源。

临时错误（辅助）
PowerHA SystemMirror 处理资源组的辅助资源时发生了可恢复错误。

可对资源组配置依赖关系，这些依赖关系允许自动放置和管理与其他资源组相关的资源组。**clRGinfo** 命令显示带有父子关系的资源组以及具有位置依赖关系的资源组的以下状态：

脱机（因为父代脱机）
子资源组未处于活动状态，因为父资源组未处于活动状态。

脱机（因为故障转移）
发生了故障转移，该资源组未处于活动状态。

脱机（因为缺少节点）
未对集群中的节点标识该资源组。

脱机（因为目标脱机）
涉及与某个资源组的关系的资源组未处于活动状态，所配置依赖关系指示此资源组不能处于活动状态。

标志

-a 显示资源组的当前位置及其在发生集群事件之后的目标。在事件前脚本和事件后脚本中（特别是在具有依赖资源组的 PowerHA SystemMirror 集群中）使用此标志。当 PowerHA SystemMirror 处理依赖资源组时，可以通过 **rg_move** 事件一次移动多个资源组。

-c 显示冒号分隔格式的输出。

-h 显示使用情况消息。

-m 显示应用程序的状态。

-p 显示资源组的优先级覆盖位置信息。

-s 显示冒号分隔格式的输出。

-t 显示延迟计时器信息、所有延迟回退计时器以及当前在本地节点上处于活动状态的校正计时器。

注：仅当集群管理器在本地节点上处于活动状态时，才能使用此标志。

-v 显示详细输出。

示例

1. 以下示例显示在未指定任何标志参数的情况下运行 **clRGinfo** 命令的报告：

```
# clRGinfo
-----------------------------------------------------------------------------
Group Name     State                          Node
-----------------------------------------------------------------------------
VS_DATA_RG     ONLINE                         powerha53
               ONLINE                         powerha54
               ONLINE                         powerha63
               ONLINE                         powerha64

VS_REDO_RG     ONLINE                         powerha53
               ONLINE                         powerha54
               ONLINE                         powerha63
               ONLINE                         powerha64

RG1            ONLINE                         powerha53
               OFFLINE                        powerha54
               ACQUIRING                      powerha63
               OFFLINE                        powerha64
```

2. 以下示例显示在带有站点的集群中运行 **clRGinfo** 命令的报告。

```
# clRGinfo
-----------------------------------------------------------------------------
Group Name     State                          Node
-----------------------------------------------------------------------------
OASTRG         ONLINE                         als018022@site1
               ONLINE SECONDARY               alm194200@site2

VOTERG         ONLINE                         als018022@site1
               ONLINE SECONDARY               alm194200@site2
```

3. 以下示例显示运行 **clRGinfo -m** 命令的报告：

```
$ /usr/es/sbin/cluster/utilities/clRGinfo -m

Group Name     State           Application state        Node
-----------------------------------------------------------------
Group1         ONLINE          ONLINE MONITORED         merry

Application state could be any one of the below possible values:
    OFFLINE
    ONLINE FAILED
```

```
ONLINE FAILOVER
ONLINE MONITORED
ONLINE NOT MONITORED
ONLINE MONITOR FAILED
ONLINE MONITOR SUSPENDED
```

clRGmove 命令

用途

执行用户请求的 rg_move 事件以使资源组脱机或联机，或将资源组从一个节点移至另一个节点。

语法

```
clRGmove -g <groupname> -n <nodename> | -x -n <sitename> |-r | -a [-m | -u | -d] [-i] [-s true | false]
```

描述

可使用 **clRGmove** 以手动控制资源组的位置和状态。

可对非并行资源组执行以下任何操作：

- 使资源组从联机节点或联机（辅助）节点脱机。
- 使资源组组联机或联机（辅助）至特定节点。
- 将资源组从其当前托管节点移动到新位置。

可对并行资源组执行以下任何操作：

- 使资源组从组节点列表中的所有节点脱机。
- 使资源组从组节点列表中的一个节点脱机。
- 使资源组在组节点列表中的所有节点上联机。
- 使资源组在组节点列表中的一个节点上联机。

优先级覆盖位置

优先级覆盖位置覆盖资源组的所有其他节点策略和可能位置。

以下是非并行资源组的优先级覆盖位置。

- 对于使用 -n 标志显式指定目标（而不使用 -r 标志）的每个非并行资源组的移动，目标必须为优先级覆盖位置。如果再次手动移动资源组，那么您使用 -r 标志（而不是 -n 标志）显式指定位置之前，该优先级覆盖位置一直存在。
- 移动脱机资源组时，该资源组保持脱机直到您手动将其改回联机状态。如果手动将资源组改回联机状态并使用 -n 标志指定节点，那么该节点变为优先级覆盖位置。如果将资源组改为联机时使用了 -r 标志，那么将使用活动最高优先级节点，并且会从资源组中移除优先级覆盖位置。

以下是并行资源组的优先级覆盖位置：

- 使并行资源组在所有节点上脱机时，对于资源组中的所有节点，优先级覆盖位置处于 *OFFLINE* 状态。使并行资源组仅在一个节点上脱机时，该节点上的资源组的 *OFFLINE* 状态将添加至优先级覆盖位置列表。
- 使并行资源组在所有节点上联机时，对于资源组中的所有节点，将移除优先级覆盖位置。使并行资源组仅在一个节点上联机时，该节点上的资源组的 *OFFLINE* 状态将从优先级覆盖位置列表中移除。

对于所有资源组移动，可使用下列其中一个移动方法：

非持久性移动

持续至集群中的所有节点脱机。只要整个集群脱机，那么优先级覆盖位置会被忘记，并且集群重新联机时资源组继续完成正常行为。

持久移动

集群重新引导后持续。集群重新联机时，优先级覆盖位置保留。

限制

以下是 **clRGmove** 命令的限制：

- 一次只能使一个资源组联机或脱机。
- 使用命令行移动多个资源组时，必须确保请求合理。因此，建议使用 SMIT 接口移动资源组，因为它消除了发生错误的任何可能性。要使用 SMIT 接口移动资源组，请从命令行输入 smit cspoc 并选择**资源组和应用程序**。

标志

-a 只能对并行资源组使用此标志。使用此标志以使资源组对组节点列表中的所有节点联机或脱机。使用 -n 标志以使并行资源组在单个节点上联机或脱机。

-d 使资源组脱机。不能将此标志与 -u 标志或 -m 标志配合使用。

-g 指定要通过以下格式移动的资源组的名称：

-g <groupname>
指定单个资源组名称。

-g "groupname1,groupname2,..."
指定多个资源组名称的逗号分隔列表。

-i 成功移动资源组后，运行 **clRGinfo** 命令。

-m 将资源组移至另一节点。不能将此标志与 -u 标志或 -d 标志配合使用。使用此标志以将多个联机资源组移至另一节点（一次一个节点）。

-n <nodename>
节点的名称，该节点包含已移动、变为联机或变为脱机的资源组。不能将此标志与 -r 标志或 -a 标志配合使用。如果节点名的前端具有 * 字符，那么该节点将配置为此资源组的最高优先级节点，并且该资源组移至另一节点。如果在节点中移动使用 * 字符标志的资源组，那么此移动会更改资源组的原始配置。

-n <sitename>
节点的名称，该节点包含在站点间移动的资源组。必须将此标志与 -x 标志配合使用。如果站点名的前端具有 * 字符，那么该站点将配置为此资源组的最高优先级站点，并且该资源组移至另一站点。如果在站点中移动使用 * 字符标志的资源组，那么此移动会更改资源组的原始配置。

-r 只能对非并行资源组使用此标志。使用对资源组将移至的目标节点可用的最高优先级节点。此标志移除要移动的资源组的优先级覆盖位置属性。仅当您将非并行资源组改为联机时或将非并行资源组移至另一节点时，才能使用此标志。不能将此标志与 -n 标志或 -a 标志配合使用。

-s true | false
指定要对资源组的主实例或辅助实例执行的操作（如果已定义站点）。使用此标志将资源组的主实例或辅助实例置于脱机或联机状态，或将其移至同一站点上的另一节点。可将此标志与 -r、-d、-u 和 -m 标志配合使用。

-s true
指定要对资源组的辅助实例执行的操作。

-s flase
指定要对资源组的主实例执行的操作。

-u 使资源组联机。不能将此标志与 -d 标志或 -m 标志配合使用。

-x 可使用此标志在站点内移动资源组。必须将此标志与 -n <sitename> 标志配合使用。

示例

1. 使脱机非并行资源组在名为 nodeB 的节点上联机：
   ```
   clRGmove -g rgA -n nodeB -u
   ```
2. 将联机非并行资源组移至名为 nodeB 的另一节点：
   ```
   clRGmove -g rgA -n nodeB -m
   ```
3. 将多个联机非并行资源组移至名为 nodeB 的另一节点：
   ```
   clRGmove -g "rgA,rgB,rgC" -n nodeB -m
   ```
4. 使联机非并行资源组在名为 nodeB 的节点上脱机：
   ```
   clRGmove -g rgA -n nodeB -d
   ```
5. 将联机非并行资源组移至活动最高优先级节点（该节点将移除另一 rg_move 事件产生的先前配置设置）：
   ```
   clRGmove -g *rgA -m -r
   ```
6. 使联机并行资源组在名为 nodeB 的节点上脱机：
   ```
   clRGmove -g rgA -n nodeB -d
   ```
7. 使联机并行资源组在所有节点上脱机：
   ```
   clRGmove -g rgA -a -d
   ```
8. 使脱机并行资源组在名为 nodeB 的节点上联机：
   ```
   clRGmove -g rgA -n nodeB -u
   ```
9. 使脱机并行资源组在所有节点上联机：
   ```
   clRGmove -g rgA -a -u
   ```
10. 将资源组移至名为 site2 的站点：
    ```
    clRGmove -s false -x -g rgA -n site2
    ```

相关参考:
第 37 页的『clmgr 命令』

---

clruncmd 命令

用途

使集群管理器恢复正常操作。

语法

```
clruncmd nodename
```

注： nodename 表示集群服务在其中处于活动状态的集群节点的名称。

描述

clruncmd 命令指示指定节点上的集群管理器在发生事件脚本故障后恢复事件处理。仅应在手动更正导致故障的原因后运行 **clruncmd** 命令。事件脚本故障发生后，失败事件的余下部分被跳过，系统继续处理事件队列中的下一个事件。必须手动执行发生事件故障后跳过的所有操作。

示例

要指示集群管理器对名称为 node1 的节点恢复正常操作，请输入：

```
clruncmd node1
```

相关参考:

第 37 页的『clmgr 命令』

---

clshowres 命令

用途

显示集群或节点的资源组信息。

语法

```
clshowres [-g group ] [-n nodename ] [-d odmdir ]
```

标志

-g group
要显示的资源组的名称。

-n nodename
在指定节点中搜索资源配置数据库。

-d odmdir
将 *odmdir*（而非缺省的 **/etc/objrepos**）指定为 ODM 对象存储库目录。

示例

1. 运行以下命令以列出集群的所有资源组信息。

   ```
   clshowres
   ```

2. 运行以下命令以列出 clam 节点的资源组信息。

   ```
   clshowres -n clam
   ```

---

clshowsrv 命令

用途

显示 PowerHA SystemMirror 子系统的状态。

语法

```
clshowsrv { -a | -v | subsystem ...}
```

描述

clshowsrv 命令显示 PowerHA SystemMirror 子系统的状态。状态包括子系统名称、组名、进程标识和状态。守护程序的状态可以是系统资源控制器 (SRC) 子系统反映的任何状态（活动、互操作、警告停止等等）。

标志

-a 显示所有 PowerHA SystemMirror 守护程序。

subsystem
显示指定 PowerHA SystemMirror 子系统的状态。此标志的有效值为：clstrmgrES、clinfoES 和 clcomd。如果指定多个子系统，那么必须使用空格来分隔条目。

-v 显示所有 RSCT、PowerHA SystemMirror 和可选 PowerHA SystemMirror 守护程序。

示例

1. 要显示所有 PowerHA SystemMirror 和 RSCT 子系统的状态，请输入：

```
clshowsrv -v
```

该命令将显示与以下示例相似的输出信息：

```
Local node: "hadev11" ("hadev11.aus.stglabs.ibm.com", "hadev11.aus.stglabs.ibm.com")
         Cluster services status:   "OFFLINE" ("ST_INIT")
         Remote communications:     "UP"
         Cluster-Aware AIX status:  "UP"

Remote node: "hadev12" ("hadev12.aus.stglabs.ibm.com", "hadev12")
         Cluster services status:   "OFFLINE" ("ST_INIT")
         Remote communications:     "UP"
         Cluster-Aware AIX status:  "UP"

Status of the RSCT subsystems used by PowerHA SystemMirror:
Subsystem  Group     PID   Status
 cthags           cthags           9371848      active
 ctrmc            rsct             11862036     active

Status of the PowerHA SystemMirror subsystems:
Subsystem  Group     PID   Status
 clstrmgrES       cluster          12124406     active

Status of the CAA subsystems:
Subsystem  Group     PID   Status
 clconfd          caa              10420354     active
 clcomd           caa              8912916      active
```

2. 要显示所有 PowerHA SystemMirror 子系统的状态，请输入：

```
clshowsrv -a
```

3. 要显示 clstrmgr 子系统的状态，请输入：

```
clshowsrv clstrmgrES
```

4. 要显示 clstrmgr 和 clinfo 子系统的状态，请输入：

```
clshowsrv clstrmgrES clinfo
```

相关参考:

第 37 页的『clmgr 命令』

---

clsnapshot 命令

用途

创建集群快照。快照是一组 ASCII 文件，这些文件包含 PowerHA SystemMirror 集群配置数据和状态信息。

语法

```
clsnapshot [-a] [-c] [-C] [-d description] [-e] [-f true|false] [-g] [-h]
[-i] [-l] [-m methodlist] -n filename [-N filename] [-o odmdir]
[-q] [-r] [-R] [s] [-t]
```

描述

clsnapshot 命令创建、修改或移除两个文件。第一个文件由文件扩展名 .odm 标识，包含当前 PowerHA SystemMirror ODM 类对象。可编写文件的简短描述。第二个文件带有扩展名 .info，包含对 PowerHA SystemMirror 集群故障诊断很有用的信息。

clsnapshot 命令在每个已配置节点上运行以获取特定于节点的信息。

可使用 **clsnapshot** 命令以对当前集群硬件应用快照。系统将运行验证实用程序，必须通过此验证，配置信息才会同步至集群节点。可使用 **-f** 标志强制应用快照，即使验证例程失败也是如此。

注：环境变量 *SNAPSHOTPATH* 包含引导至快照文件的路径。缺省情况下，此路径为 /usr/es/sbin/cluster/snapshots。

标志

-a 应用集群快照

-c 创建集群快照

-C 应用快照时，不要刷新活动集群资源。

-d text
已添加对快照的描述。

-e 将集群日志保存在快照中。将日志保存至快照可能会大幅增加快照文件大小。

-f true|false
验证失败时强制应用快照。

-g 生成用于保存快照的临时 ODM

-h 使用快照

-i 生成带 .info 扩展名的文件。

-l 列示快照文件。

-m methodlist
运行 methodlist 文件中列示的每个定制快照方法。

-n file
指定快照的名称。

-N file
指定快照的新名称。

-o odmdir
对 PowerHA SystemMirror ODM 类指定 ODM 目录 (ODMDIR)。

-r 移除快照。

-R 替换快照。

-s 显示快照。

-t 重置集群选项。

相关参考:

第 37 页的『clmgr 命令』

clsnapshotinfo 命令

用途

检索并显示某些 PowerHA SystemMirror 集群配置信息。

语法

```
clsnapshotinfo [-m <METHOD> [<METHOD#2> ...]]
```

描述

clsnapshotinfo 命令运行 PowerHA SystemMirror 和 AIX 命令以收集有关 PowerHA SystemMirror 集群的信息。**clsnapshotinfo** 命令仅从运行该命令的节点收集信息。该命令的输出将写至 STDOUT。通过 **clsnapshot** 命令运行 **clsnapshotinfo** 命令（自动进行）时，系统从集群中的所有节点收集信息，并且输出存储在扩展名为 .info 的快照文件中。

建议您在 **clsnapshot** 命令中运行 **clsnapshotinfo** 命令以尽可能多地收集有关集群的信息。

标志

-m 指定一个或多个定制快照方法。这些方法的输出包含在 **clsnapshotinfo** 命令收集的整体数据中。

相关参考:

第 37 页的『clmgr 命令』

clstat 命令（ASCII 方式和 X Windows 方式）

注：本主题包含有关 **clstat** 命令的 ASCII 方式和 X Windows 方式的信息。

ASCII 方式

用途

集群状态监视器（ASCII 方式）。

语法

```
clstat [-c cluster ID | -n cluster name] [-i] [-r seconds] [-a] [-o][-s]
```

标志

-c cluster id
仅显示有关具有指定标识的集群的集群信息。如果指定的集群不可用，那么 clstat 将继续查找集群，直至找到集群或程序取消。如果使用 -i 选项，那么不能指定此标志。

-i 以交互方式运行 ASCII **clstat**。最初显示可由系统访问的所有集群的列表。用户必须选择要为其显示详细信息的集群。详细显示中提供了很多功能。

-n name
显示有关具有指定名称的集群的集群信息。如果使用 -i 选项，那么不能指定此标志。

-r seconds
在指定的秒数更新集群状态显示。缺省值为 1 秒；但是，仅当集群状态更改时，显示才会更新。

-a 使 **clstat** 以 ASCII 方式显示。

-o 提供集群状态和出口的单次快照。此标志可以用于在**定时**作业外部来运行 **clstat**。必须在指定 **-a** 的情况下运行；将忽略 **-i** 和 **-r** 选项。

-s 显示服务标签及其状态（启动或关闭）。

X Windows 方式

用途

集群状态监视器（X Windows 方式）。

语法

```
clstat [-a] [-c id | -n name ] [-r tenths-of-seconds ][-s]
```

标志

-a 以 ASCII 方式运行 **clstat**。

-c id
仅显示有关具有指定标识的集群的集群信息。如果指定的集群不可用，那么 **clstat** 将继续查找集群，直至找到集群或程序取消。如果使用 **-n** 选项，那么不能指定此标志。

-n name
仅显示有关具有指定名称的集群的集群信息。

-r tenths-of-seconds
clstat 实用程序更新显示的时间间隔。对于图形界面，此值将在数十秒内解析。缺省情况下，**clstat** 每 0.10 秒更新一次显示。

-s 显示服务标签及其状态（启动或关闭）。

示例

1. 运行以下命令以显示有关 mycluster 集群的集群信息。

   ```
   clstat -n mycluster
   ```

2. 以交互方式运行 ASCII clstat，允许多集群监视。

   ```
   clstat -i
   ```

   以下是 X Window System 显示上的按钮：

   Prev 显示上一个集群。

   Next 显示下一个集群。

   Name:Id
   刷新栏，按此栏可使 **clstat** 立即刷新。

   Quit 退出应用程序。

   Help 弹出帮助窗口显示 **clstat** 手册页面。

---

clstop 命令

用途

停止集群子系统。

语法

```
clstop { -f | -g | -gr } [-s] [-y] [ -N | -R | -B ]
```

描述

clclstop 停止本地节点上的集群服务并根据您指定的标志处理所有活动资源组。此命令可选择通过 /etc/inittab 文件中的条目禁止重新引导时自动启动。

标志

-f 强制关闭。集群守护程序终止，并且不运行任何本地过程。

-g 正常关闭并且不接管。

-gr
正常关闭，资源被此节点释放并由另一节点接管。守护程序正常终止，节点释放其资源，资源被另一节点接管。必须对"正常关闭并接管"选项指定节点列表。

-s 执行静默关闭。此标志不会通过 **wall** 命令广播关闭消息。缺省设置是广播。

-y
关闭集群节点前不要求操作员确认。此标志是缺省值。

-B 立即停止，并在后续系统重新启动时停止。

-N
立即关闭。

-R 在后续系统重新启动时停止并移除 /etc/inittab 文件中的该条目。

注： /etc/rc.shutdown 文件是一个可选文件，包含在执行关闭命令期间运行的命令。

示例

1. 关闭集群节点时，如果要通过使用"正常关闭"选项并释放资源而不在停止集群进程前向用户发送警告消息，请输入：

   ```
   clstop -gr -s -y
   ```

2. 要在所有集群节点上强制立即关闭该集群（不释放资源），并在停止集群进程前向用户广播警告消息，请输入：

   ```
   clstop -f -y
   ```

3. 关闭集群节点时，如果要使用"正常关闭"选项并释放资源（资源随后被接管），同时在停止集群进程前向用户广播警告消息，请输入：

   ```
   clstop -gr -y
   ```

相关参考:

第 37 页的『clmgr 命令』

---

cltopinfo 命令

用途

显示完整的拓扑信息：集群名称、网络总数、错过的脉动信号的总数以及集群中配置的节点。显示每个节点的所有已配置网络。显示每个网络的所有已配置接口。还显示已定义的所有资源组。

语法

```
cltopinfo [-c] [-i] [-n] [-w]
```

标志

-c 显示集群名称和安全方式（标准或增强型）

-i 显示集群中配置的所有接口。信息包括接口标签、其连接到的网络（如果适用）、IP 地址、子网掩码、节点名和设备名。

-n 显示集群中配置的所有节点。对于每个节点，将列出已定义的所有网络。对于每个网络，将列出已定义的所有接口以及服务 IP 标签别名的分发首选项（如果已定义）。

-w 显示集群中配置的所有网络。对于每个网络，将列出连接到网络的所有节点。对于每个节点，将列出已定义的所有接口以及服务 IP 标签别名的分发首选项（如果已定义）。

示例1

要显示集群中定义的所有节点和网络（节点 coffey1 和 lee1），请使用 **cltopinfo** 命令。以下集群是通过 IPv4 地址和 IPv6 地址配置的。输出类似以下内容：

```
Cluster Name: hacmp_full_ipv6
Cluster Connection Authentication Mode: Standard
Cluster Message Authentication Mode: None
Cluster Message Encryption: None
Use Persistent Labels for Communication: No
There are 2 node(s) and 2 network(s) defined

NODE coffey1:
   Network net_ether_01
            service_ipv4_2   1.8.4.2
      service_ipv6_1   fe80::c862:67ff:fe58:5646
            coffey1_boot3   1.4.6.4
      coffey1_boot1   1.2.4.4
Network net_ether_02
            service_ipv4_32   1.8.4.4
      service_ipv6_31  fe80::c862:67ff:fe58:5846
          coffey1_boot_v6  fe80::c872:67ff:fe59:8647
          coffey1_boot_v6  fe80::c872:678f:fe95:8683
NODE lee1:
   Network net_ether_01
            service_ipv4_2   1.8.4.2
      service_ipv6_1   fe80::c862:67ff:fe58:5646
            lee1_boot1   1.2.4.3
      lee1_boot3   1.4.6.3
Network net_ether_02
            service_ipv4_32   1.8.4.4
      service_ipv6_31  fe80::c862:67ff:fe58:5846
            lee1_boot_v6   fe80::c672:fe56:fe82:2345
            lee1_boot_v6   fe80::fe34:3456:f873:f345

Resource Group RG1
      Startup Policy    Online On Home Node Only
      Fallover Policy    Fallover To Next Priority Node In The List
      Fallback Policy    Fallback To Higher Priority Node In The List
      Participating Nodes    coffey1 lee1
      Service IP Label       service_ipv4_1
      Service IP Label       service_ipv4_31

Resource Group RG2
      Startup Policy    Online On Home Node Only
      Fallover Policy    Fallover To Next Priority Node In The List
```

```
     Fallback Policy    Fallback To Higher Priority Node In The List
     Participating Nodes    lee1 coffey1
     Service IP Label       service_ipv4_2
     Service IP Label       service_ipv4_32
```

示例 2

要显示集群名称和当前安全方式，请使用 **cltopinfo** 命令。输出类似以下内容：

```
# cltopinfo -c

 Cluster Name: c10
 Cluster Connection Authentication Mode: Standard
 Cluster Message Authentication Mode: None
 Cluster Message Encryption: None
 Use Persistent Labels for Communication: No
```

示例 3

要显示集群中定义的所有节点，请使用 **cltopinfo** 命令。以下集群是通过 IPv4 地址和 IPv6 地址配置的。输出类似以下内容：

```
# cltopinfo -n

 NODE abby:
      Network net_ether_01
    abby_en1stby    192.168.121.7
    abby_en0boot    192.168.120.7
      Network net_ether_02
abby_boot1_v6  fe80::c872:67ff:fe59:8647
abby_boot2_v6  fe80::c872:678f:fe95:8683
      Network net_rs232_01
      Network net_rs232_02
    abby_tty0_01    /dev/tty0

 NODE polly:
      Network net_ether_01
    polly_en0boot    192.168.120.9
    polly_en1stby    192.168.121.9
    polly_en2boot    192.168.122.9
      Network net_ether_02
polly_boot1_v6  fe80::c672:fe56:fe82:2345
polly_boot2_v6  fe80::fe34:3456:f873:f345
      Network net_rs232_01
      Network net_rs232_02
    polly_tty0_01    /dev/tty0
```

示例 4

要显示集群中定义的所有网络，请使用 **cltopinfo** 命令。以下集群是通过 IPv4 地址和 IPv6 地址配置的。输出类似以下内容：

```
# cltopinfo -w

 Network net_ether_01
      NODE abby:
    abby_en1stby    192.168.121.7
    abby_en0boot    192.168.120.7
      NODE polly:
    polly_en0boot    192.168.120.9
    polly_en1stby    192.168.121.9
    polly_en2boot    192.168.122.9

 Network net_ether_02
      NODE abby:
```

```
abby_boot1_v6  fe80::c872:67ff:fe59:8647
abby_boot2_v6  fe80::c872:678f:fe95:8683
      NODE polly:
polly_boot1_v6  fe80::c672:fe56:fe82:2345
polly_boot2_v6  fe80::fe34:3456:f873:f345

 Network net_rs232_01
      NODE abby:
      NODE polly:

 Network net_rs232_02
      NODE abby:
    abby_tty0_01    /dev/tty0
      NODE polly:
    polly_tty0_01    /dev/tty0
```

示例 5

要显示集群中定义的所有接口，请使用 **cltopinfo** 命令。输出类似以下内容：

```
# cltopinfo -i
IP Label NetworkType   Node  Address If   Netmask   Pefixlenth
===================   ====  ======= ==== =======   ==========
abby_en1stby    net_ether_01  ether  abby  192.168.121.7  en2  255.255.255.0
abby_en0boot    net_ether_01  ether  abby  192.168.120.7  en1  255.255.255.0
abby_boot1_v6   net_ether_02  ether  abby  fe80::c872     en3           64
abby_boot2_v6   net_ether_02  ether  abby  fe80::c672     en4           64
abby_tty0_01    net_rs232_02  rs232  abby  /dev/tty0      tty0
polly_en0boot   net_ether_01  ether  polly 192.168.120.9  en1  255.255.255.0
polly_en1stby   net_ether_01  ether  polly 192.168.121.9  en2  255.255.255.0
polly_en2boot   net_ether_01  ether  polly 192.168.122.9  en3  255.255.255.0
polly_boot1_v6  net_ether_02  ether  polly fe80::c072     en4           64
polly_boot2_v6  net_ether_02  ether  polly fe80::c172     en5           64
polly_tty0_01   net_rs232_02  rs232  polly /dev/tty0      tty0
```

---

clvaryonvg 命令

用途

使卷组联机。

语法

```
clvaryonvg [-F] [-f] [-n] [-p] [-s] [-o] <vg>
```

描述

clvaryonvg 命令被设计为 AIX 操作系统中的 **varyonvg** 命令的替代项。此命令在卷组上执行一些检查，以确定调用 AIX **varyonvg** 命令前是否对卷组进行了任何更改。如果自上次卷组在本地联机后进行了任何更改，那么将导出该卷组，然后在卷组联机前导入该卷组。此进程验证所有节点的卷组内容是否一致。

如果卷组更新期间发生了系统故障，那么卷组可能变为对节点不可视。系统实现以下机制以防御系统故障：

- 导出卷组前，将在 /usr/es/sbin/cluster/etc/vg 目录中创建名为 <*VG*>.replay 的文件，其中 *VG* 是卷组的名称。此文件是包含一组命令的 shell 脚本，这些命令用于在卷组变为对节点不可视或不存在时复原卷组。如果卷组不存在，那么 <*VG*>.replay 文件中的命令将下次使用 **clvaryonvg** 命令时自动运行。
- 如果重放文件未修正该问题，那么可在 hacmp.out 文件中查看这些消息。这些消息说明如何手动复原卷组。hacmp.out 文件中的消息还可在 /usr/es/sbin/cluster/etc/vg/<*VG*>.desc 文件中找到，其中 *VG* 是卷组的名称。失败重放文件的副本放置在 /var/tmp 目录中。

标志

-f 将该标志传递至 **importvg** 命令或 **varyonvg** 命令。

-F 通过在卷组上使用 **exportvg** 命令或 **importvg** 命令并强制忽略时间戳记来强制更新。

-n 禁止卷组内旧物理分区的同步。此标志将传递至 **varyonvg** 命令。

-p 指定所有物理卷必须可供使用 **clvaryonvg** 命令。

-s 使卷组仅在系统管理方式下可用。

-o 完成后使卷组保持脱机。完成过程包含执行所有完整性检查以及（如果需要）运行 **importvg** 命令和 **exportvg** 命令。

示例

1. 要激活标签为 *vg03* 的卷组，请输入：

   ```
   clvaryonvg vg03
   ```

2. 要在标签为 *vg03* 的卷组上更新节点的信息，请输入：

   ```
   clvaryonvg -F vg03
   ```

---

get_local_nodename 命令

用途

检索本地节点的名称。

语法

```
get_local_nodename
```

描述

显示本地节点的名称。

示例

要显示本地节点的名称，请输入：

```
get_local_nodename
```

---

halevel 命令

用途

显示系统上安装的 PowerHA SystemMirror 版本、发行版、修改级别和 Service Pack 级别。

语法

```
halevel [-h|-?] [-s] [-x]
```

描述

如果从 PowerHA SystemMirror 客户机运行此命令，那么此命令不会正常工作。必须从 PowerHA SystemMirror 服务器节点运行此命令。

标志

-h | -?
显示帮助信息。

-s 显示 Service Pack 级别。

-x 启用调试 (ksh set -x)

示例

1. 要显示 PowerHA SystemMirror 版本、发行版和修改级别，请输入：

   ```
   halevel
   ```

2. 要显示 PowerHA SystemMirror 版本、发行版、修改级别和 Service Pack 级别，请输入：

   ```
   halevel -s
   ```

3. 要显示所有集群节点上的 PowerHA SystemMirror 版本、发行版、修改级别和 Service Pack 级别，请输入：

   ```
   /usr/es/sbin/cluster/cspoc/cli_on_cluster -S halevel -s
   ```

---

rc.cluster 命令

用途

使用 **rc.cluster** 命令可设置操作系统环境并跨集群节点启动集群守护程序。

注：与特定标志关联的参数必须在紧邻标志的后面指定。PowerHA SystemMirror

语法

```
| rc.cluster [-boot] [b] [-i | -I] [-N | -R | -B] [-M | -A] [-r] [-v] [-x] [-C interactive|yes]
```

标志

-boot
已启用 IPAT 时，将服务网络接口配置为在其引导地址上。

-i 使用缺省选项来启动集群信息 (**clinfoES**) 守护程序。

-I 在启用陷阱的情况下启动集群信息 (**clinfoES**) 守护程序。

-b 广播启动。

-N
立即启动守护程序（无 **inittab** 文件更改）。

-R 仅在系统重新启动时启动 PowerHA SystemMirror 守护程序。会将 PowerHA SystemMirror 启动命令添加到 **inittab** 文件。

-B 立即启动守护程序并将 PowerHA SystemMirror 条目添加到 **inittab** 文件。

-C 指定要用于在发生问题时执行更正操作的方式。指定 **yes** 以自动更正问题。指定 **interactive** 以在运行每个更正操作前显示提示。

-M 以手动资源获取方式来启动集群服务。如果您希望手动使资源组联机，请使用此选项。

-A 以自动资源获取方式来启动集群服务。如果您希望在集群启动时自动将资源组联机，请使用此选项。这是缺省选项。

-r 在强制关闭之后重新获取集群资源。如果您在集群强制关闭期间更改了任何集群资源（IP 标签、磁盘和应用程序）的状态，请使用此选项。

-v 在启动期间忽略验证错误 (auto ver sync)

-x 激活 NFS 交叉装配。

示例

要启动集群以及 **clinfo** 服务并广播事件，请运行以下命令：

```
rc.cluster -boot -N -i
```

声明

本信息是为在美国国内供应的产品和服务而编写的。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您所在区域当前可获得的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务的操作，由用户自行负责。

IBM 可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并不意味着授予用户使用这些专利的任何许可。您可以用书面形式将许可查询寄往：

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

有关双字节字符集 (DBCS) 信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往：

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION"按现状"提供本出版物，不附有任何种类的（无论是明示的还是默示的）保证，包括但不限于默示的有关非侵权、适销和适用于某种特定用途的保证。某些管辖区域在某些事务中不允许免除明示或默示的保证，因此本声明可能不适用于您。

本信息可能包含技术方面不够准确的地方或印刷错误。本信息将定期更改；这些更改将编入本信息的新版本中。IBM 可以随时对本出版物中描述的产品和/或程序进行改进和/或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而不必对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的：(i) 使其能够在独立创建的程序和其它程序（包括本程序）之间进行信息交换，以及 (ii) 使其能够对已经交换的信息进行相互使用，请与下列地址联系：

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本文档中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际程序许可协议或任何同等协议中的条款提供。

所引用性能数据和客户示例仅供参考。实际性能结果可能会因具体配置和操作条件而有所不同。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

所有 IBM 的价格均是 IBM 当前的建议零售价，可随时更改而不另行通知。经销商的价格可与此不同。

本信息仅用于规划目的。在所描述的产品上市之前，此处的信息会有更改。

本信息包括日常业务运作中使用的数据和报告的示例。为了尽可能完整地说明这些示例，示例中可能会包括个人、公司、品牌和产品的名称。所有这些名称均是虚构的，如与实际的人名或商业企业的名称有任何相似之处，纯属巧合。

版权许可：

本信息包含源语言形式的样本应用程序，用以阐明在不同操作平台上的编程技术。如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。这些示例尚未在所有条件下经过全面测试。因此，IBM 不能担保或暗示这些程序的可靠性、可维护性或功能。这些实例程序"按现状"提供，不附有任何种类的保证。对于因使用样本程序所引起的任何损害，IBM 概不负责。

这些样本程序的每一个副本或其任何部分或任何衍生产品都必须包括如下的版权声明：

©（贵公司的名称）（年）。

此部分代码是根据 IBM Corp. 公司的样本程序衍生出来的。

© Copyright IBM Corp.（请在此处输入年份）。

隐私策略注意事项

IBM 软件产品 （"软件产品"，其中包括作为服务解决方案的软件）可能使用 cookie 或其他技术来收集产品使用信息，以帮助改进最终用户体验、定制与最终用户的交互或实现其他目的。在许多情况下，软件产品不会收集任何个人可标识信息。我们的某些软件产品可以帮助您收集个人可标识信息。如果此软件产品使用 cookie 来收集个人可标识信息，那么会在下面列出有关此产品使用 cookie 的特定信息

此软件产品不会使用 cookie 或其他技术来收集个人可标识信息。

如果为此软件产品部署的配置使您能够作为客户通过 cookie 和其他技术从最终用户收集个人可标识信息，那么您应该向您自己的法律顾问咨询有关适用于这种数据收集（其中包括对于通知和同意的任何需求）的任何法律。

有关为这些目的使用各种技术（其中包括 cookie）的更多信息，请参阅 "IBM 隐私策略"（网址为 http://www.ibm.com/privacy）和"IBM 在线隐私声明"（网址为 http://www.ibm.com/privacy/details）中标题为"cookie、Web 信标和其他技术"和"软件产品和 Software-as-a 服务"（网址为 http://www.ibm.com/software/info/product-privacy）的部分。

商标

IBM、IBM 徽标和 ibm.com 是 International Business Machines Corp. 在全世界许多管辖区域注册的商标或注册商标。其他产品和服务名称可能是 IBM 或其他公司的商标。当前最新的 IBM 商标列表在以下 Web 站点提供版权和商标信息 (www.ibm.com/legal/copytrade.shtml)。

UNIX 是 The Open Group 在美国和其他国家或地区的注册商标。

索引

C

R

Printed in China