CA Spectrum®

命令行界面用户指南

版本 9.3

ca technologies

CA Technologies 产品引用

本指南参考 CA Spectrum®。

联系技术支持

要获取在线技术帮助以及办公地址、主要服务时间和电话号码的完整列表，请联系技术支持：http://www.ca.com/worldwide。

目录

第 1 章： 命令行界面 (CLI) 简介

此部分包含以下主题：

概述

CA Spectrum 命令行界面 (CLI) 是核心 CA Spectrum 组件，随核心 CA Spectrum 产品一起安装。

您可以通过 OneClick 用户界面访问 CA Spectrum 数据并执行 CA Spectrum 操作。不过，如果您希望从命令行执行 CA Spectrum 操作，则可以使用 CLI。对于那些无法在 OneClick 中执行的任务，只能使用 CLI CA Spectrum 资源来完成。

CLI 是一款功能强大的工具，但它不能提供 OneClick 所具备的安全防卫功能，尤其是与建模相关的功能。CLI 只能供 CA Spectrum 管理员使用，因为只有他们才了解在网络建模架构中随意创建和销毁模型以及修改模型属性存在的潜在危害。

CLI 是一个灵活的产品。您可以使用系统（如 UNIX、DOS 和 Bash）提供的任何命令提示符打开 CLI 会话和发出命令。

CLI 命令

CLI 命令与 UNIX 命令类似，它们可与 UNIX 或 DOS 命令一起使用，尤其是和 grep（查找）、管道和重定向符号一起使用。然而，部分 CLI 命令可能会与同名的 UNIX 命令发生冲突。例如，CLI update 命令可能会与 UNIX update 命令发生冲突。

为避免冲突，请在 vnmsh 目录中使用 **./update**。使用脚本时，应该使用 CLI 命令的完整路径名称，例如 *<$SPECROOT>***/vnmsh/update**。

注意：CLI update 命令始终会提供响应 - 要么确认更新成功，要么显示更新失败的消息。使用 update 命令时，如果您未收到 CLI 的响应，则键入“**which update**”。系统可能按以下方式响应：

```
/etc/update
```

详细信息：

命令说明 (p. 25)

在 Shell 脚本中使用 CLI

CLI 命令可以集成到 shell 脚本或菜单系统中，为您提供了更强大、更多样的方法来访问 CA Spectrum 数据。

每个 CLI 命令会将报告命令是否执行成功的输出发送给标准错误。命令执行成功时预期返回的正常输出结果会发送给标准输出。每个命令在执行成功时还会生成返回代码零；如果失败，则生成非零错误代码。使用 CLI 命令的 shell 脚本借助返回代码即可确定每个命令是否成功并据此确定后续操作。

CLI 组件

本指南对 CLI 组件的描述如下：

- 可执行命令
- 四个环境变量
- 维护与 SpectroSERVER 之间的通信的后台进程
- 集成 CLI 命令的一组示例 shell 脚本

详细信息：

命令说明 (p. 25)
CLI 环境变量 (p. 9)
CLI 本地服务器 (p. 11)
示例脚本 (p. 61)

CLI 环境变量

您可以为 CLI 设置以下四个环境变量：

CLIMNAMEWIDTH

显示模型名称。默认情况下，create、seek 和 show 命令各自可以显示的模型名称的最大长度为 16 个字符。但如果使用环境变量 CLIMNAMEWIDTH，最多可指定 1024 个字符来显示模型名称。例如，使用 C shell：

```
setenv CLIMNAMEWIDTH 32
```

您可以在 .login 文件或脚本中设置此变量，也可以在发出命令前设置。您可以在 CLI 会话中执行任意次设置或更改操作，具体次数取决于模型名称的长度。

CLISESSID

表示可在脚本中使用的 ID。将 CLISESSID 变量设置为 <$$>，用于表示正在运行的 shell 脚本的进程 ID。使用 cron 同时运行 CLI 脚本时必须使用此变量。例如，使用 bash shel：

```
CLISESSID=<$$>; export CLISESSID
```

此外，如果您是使用 bash shell（而非 DOS）在 Windows 上运行 CLI，则需要针对每个 CLI 会话为 CLISESSID 环境变量设置唯一值。您可以为每个 bash shell 分配唯一的时间戳，例如：

```
export CLISESSID='date +%s'
```

SPECROOT

显示警报或事件说明。如果指定了 -x 选项，执行 show alarms 或 show events 命令时就需要使用 SPECROOT 环境变量。如果能够定位该变量，就能从 SG-Support 目录树中获取警报或事件的说明，从而展开该命令的输出。

在 UNIX 中，您可以在登录 shell 中指定 SPECROOT 变量，并可以将此变量设置为 CA Spectrum home 目录。例如：

```
SPECROOT=/home/CA Spectrum; export SPECROOT
```

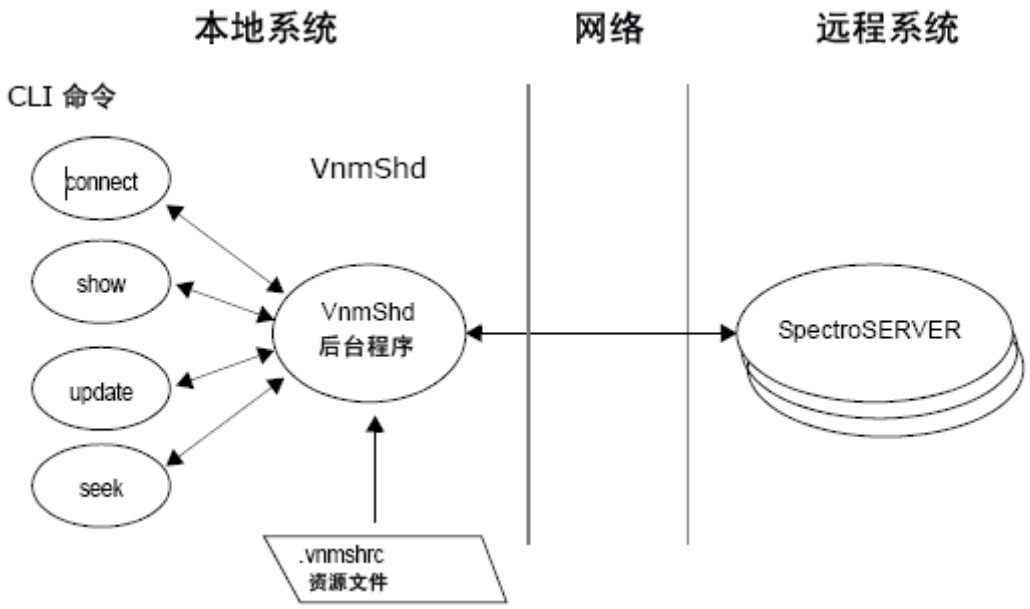
对于 Windows 系统，请参阅系统文档了解设置环境变量的详细信息。

CLIPATH

显示 *<$SPECROOT>*/vnmsh 目录的路径以及需要使用 CLI 命令的脚本。

CLI 体系结构

下图显示了 CLI 体系结构：



使用 .vnmshrc 的 CLI 本地服务器在启动时执行以下主要功能：

- 与 SpectroSERVER 保持持续的网络连接。每次执行命令时，CLI 本地服务器会阻止断开连接。不管有多少 CLI 用户连接到后台进程，此服务器始终会保持与 SpectroSERVER 的*单个*连接。从时间和资源使用情况等方面考虑，套接字连接和断开连接的代价昂贵。
- 维护每个 CLI 用户的状态信息。例如，current 和 setjump 命令需要通过 CLI 本地服务器来存储状态信息。current 命令会存储模型句柄和格局句柄，以供将来的命令使用。setjump 命令会存储文本字符串，用于在 CA Spectrum 格局中识别用户的当前位置。

启动文件

.vnmshrc 文件，即 CLI 本地服务器启动文件，位于 *<$SPECROOT>*/vnmsh 目录中。此文件包含几个控制 vnmsh 如何与 SpectroSERVER 进行通信的参数。下面的列表对这些参数进行了介绍：

vnm_hostname

指定要连接的 SpectroSERVER 的主机名。

client_handshake_timeout

指定设置连接时客户端等待服务器 ID 信息的时间（用毫秒表示）。

默认值：900

server_handshake_timeout

指定设置连接时服务器等待客户端 ID 信息的时间（用毫秒表示）。

默认值：900

connect_time_limit

指定等待与 SpectroSERVER 建立连接的最长时间（用毫秒表示）。

默认值：1000

listen_backlog

指定在等待前面的请求完成时队列中存储的对 SpectroSERVER 的客户端请求的数目。

默认值：10

vnm_tcp_port

指定当 vnmsh 是 SpectroSERVER 客户端时，vnmsh 用来与 SpectroSERVER 通信的 TCP 端口。

vsh_tcp_port

指定当 vnmsh 充当处理客户端请求（如 show、update 命令）的服务器时，vnmsh 侦听 TCP 消息的 TCP 端口。

debug_file

指定 CLI 写入错误消息的文件。

max_show_event_length

指定使用 **show events -x** 命令显示事件消息时所显示的最大字符数。

默认值：512

CLI 本地服务器

发出 connect 命令的第一个用户会自动启动该工作站上的 CLI 本地服务器（VnmShd 后台进程），并与 SpectroSERVER 建立连接。每个工作站只能运行一个 CLI 本地服务器，该后台进程只与 SpectroSERVER 建立一个连接。

在工作站上启动 CLI 本地服务器之后，连接到该工作站上的 CLI 的所有后续用户都将使用同一台 CLI 本地服务器。

详细信息：

CLI 环境变量 (p. 9)

错误检查

在 OneClick 中执行任务时，CA Spectrum 会强制实施特定规则。例如，这些规则可以控制您在不同视图中创建或移动设备模型时允许执行的操作。

CLI 不强制实施这些规则，因此无法执行任何错误检查。因此，CLI 允许用户创建模型，并在不执行错误检查的情况下将其置于他们希望放置的位置。如果尝试通过不符合其格式的方式使用 CLI 命令，则会出现错误。

第 2 章： 使用命令行界面

此部分包含以下主题：

在 UNIX 上启动 CLI 会话

在 UNIX 平台上，您可以从 shell 提示符下启动 CLI 会话。

注意：您可以使用脚本将 CLI 打包，以便将其发送给其他服务器。有关详细信息，请参阅 *CA Spectrum 分布式 SpectroSERVER 管理员指南*。

遵循这些步骤:

1. 启动需要连接的 SpectroSERVER。
2. 导航到 CA Spectrum 安装目录下的 vnmsh 目录：

   ```
   $ cd <$SPECROOT>/vnmsh
   ```

3. 打开连接：

   ```
   $ connect
   ```

   此时会连接到 CLI 会话。

在 Windows 上使用 DOS 提示符启动 CLI 会话

在 Windows 平台上，您可以从 DOS 提示符下启动 CLI 会话。

注意：您可以使用脚本将 CLI 打包，以便将其发送给其他服务器。有关详细信息，请参阅 *CA Spectrum 分布式 SpectroSERVER 管理员指南*。

本指南中提供的所有 UNIX（相对于 CLI）命令实例，都可在必要时替换成等效的 DOS 命令。例如，可使用 *find* 来替换 *grep*。

遵循这些步骤:

1. 对于 DOS 提示符，依次选择“开始”、“程序”、“命令提示符”。

   当 DOS 提示符出现时，即表示可以接受 CLI 命令。
2. 启动需要连接的 SpectroSERVER。
3. 导航到 CA Spectrum 安装目录下的 vnmsh 目录：

   ```
   $ cd <$SPECROOT>/vnmsh
   ```
4. 打开连接：

   ```
   $ connect
   ```

   此时会连接到 CLI 会话。

详细信息：

UNIX 与 DOS 的转换 (p. 83)

使用 Bash 提示符在 Windows 上启动 CLI 会话

在 Windows 平台上，还可以使用 bash shell 提示符启动 CLI 会话。

注意：您可以使用脚本将 CLI 程序打包，以便将其发送给其他服务器。有关详细信息，请参阅 *CA Spectrum 分布式 SpectroSERVER 管理员指南*。

遵循这些步骤:

1. 依次单击“开始”、“程序”和“命令提示符”。

   此时会显示 DOS 提示符。
2. 在 DOS 提示符下，键入 **bash**。
3. 依次单击“开始”、“运行”，然后键入 **bash -login**。

   您可以使用 bash shell 提示符启动 CLI 会话。
4. 启动需要连接的 SpectroSERVER。
5. 导航到 CA Spectrum 安装目录下的 vnmsh 目录：

   ```
   $ cd <$SPECROOT>/vnmsh
   ```
6. 打开连接：

   ```
   $ connect
   ```

   此时会连接到 CLI 会话。

用法示例

以下示例演示了如何使用 CLI 命令来完成 CA Spectrum 中的常见任务。

创建用户模型

用户模型为用户提供访问 CA Spectrum 的权限。用户由登录 ID 来标识。

注意：启动 CLI 会话之前，确认已经创建用户模型，并且启动了要连接的 SpectroSERVER。

遵循这些步骤:

1. 连接到 SpectroSERVER。

   ```
   $ cd <$SPECROOT>/vnmsh
   $ ./connect
   ```

   您将连接到 SpectroSERVER。

   注意：如果您在连接时遇到问题，请检查错误消息。有关详细信息，请参阅错误消息 (p. 65)。

2. 为您希望使用 show 命令创建的模型类型确定模型类型句柄。在本例中，它是用户类型的模型。输入以下命令：

   ```
   $ ./show types | grep User
   ```

   注意：“./”非常重要。一些 UNIX 系统使用 show 命令来读取电子邮件。如果“.”不是用户环境中的第一个路径，则需要使用“./”。

   此时会显示包含字符串“User”的模型类型列表，用户模型类型会列在最前面。

   ```
   句柄        名称           标志
   0x10004     User           V,I,D
   0x1040a     UserGroup      V,I,D
   0x1040f     DefUserGroup   V,I,N,U,R
   0xaa000d    GenSwUserPort  V,I,D
   0xf000d     ForeUserAgen   V,I,D
   0xaf000c    ForeUserApp    V,I,D
   ```

3. 使用 show 命令列出用户模型类型的属性，并确定模型名称属性的属性 ID。在创建模型时您需要使用此属性 ID。输入以下命令：

   ```
   $ ./show attributes mth=0x10004 | grep -i name
   ```

此时会显示包含模型名称属性的用户模型类型属性的列表。

Id	名称	类型	标志
0x10000	Modeltype_Name	文本字符串	R,S,M
0x1006e	Model_Name	文本字符串	R,W,G,O,M,D
0x10074	User_Full_Name	文本字符串	R,W,O,D
0x1155f	gib_mtype_name	文本字符串	R,W,S,D
0x11560	gib_mtype_name_menu	文本字符串	R,W,S,D
0x11561	gib_model_name	文本字符串	R,W,D
0x11563	gib_model_name_menu	文本字符串	R,W,D
0x1197d	WatchNames	标记的八进制数	R,W,D

4. 使用带有模型类型句柄、模型名称的属性 ID 以及用户的值（登录 ID 名称）的 create 命令创建模型。在此例中，用户登录 ID 是 j_doe。输入以下命令：

   ```
   $ ./create model mth=0x10004 attr=0x1006e,val=j_doe
   ```

   当看到类似下例所示的系统消息时，即可确认模型创建完毕：

   ```
   created model handle = 0xbe0001b
   ```

   注意：这些示例中使用的所有句柄和 ID 都是虚构的。您创建的模型的模型句柄会有所不同；模型句柄由系统创建。

详细信息：

错误消息 (p. 65)

修改模型属性

本节提供了使用 CLI 命令更改模型属性值的示例。此外，此示例演示了如何更改在用户模型中创建的模型 (j_doe) 的团体字符串属性值。有关详细信息，请参阅创建用户模型 (p. 15)。

遵循这些步骤:

1. 确定 j_doe 模型句柄，然后将 j_doe 设为当前模型：
   a. $ ./show models | grep j_doe

      此时会显示与 j_doe 模型相关的以下信息：

      ```
      0xbe0001b        j_doe(Active)        0x10004        User
      ```

   b. $ ./current mh=0xbe0001b

      系统确认 j_doe 是当前模型：

      ```
      当前模型是 0xbe0001b
      当前格局是 0xbe00000
      ```

2. 确定团体字符串的属性 ID。

   注意：为简洁起见，此步骤将属性名称的已知部分（团体字符串）显示为 grep 命令的参数。如果不知道属性的名称，您可以显示并扫描该模型的所有属性，从而确定正确的属性名称及其属性 ID。

3. 输入以下命令：

   ```
   $ ./show attributes | grep -i community_string
   ```

   此时会显示属性 ID、属性名称和团体字符串值：

   ```
   0x1007a        User_Community_String        ADMIN,0
   ```

   CA Spectrum 会在创建用户模型时将 ADMIN,0 的默认值分配给所有用户模型。ADMIN,0 将 CA Spectrum 中的完全管理权限授予用户模型。

4. 使用 update 命令将 j_doe 模型的管理权限级别 ADMIN,0 更改为示例权限级别 ADMIN,5（只读）。输入以下命令：

   ```
   $ ./update attr=0x1007a,val=Subnet3,5
   ```

   此时会返回显示属性值更改的条目：

   ```
   Id         名称                    Iid     值
   0x1007a    User_Community_String           ADMIN,5
   ```

   由于 Iid 属性仅适用于列表属性，因此在此处没有值。有关详细信息，请参阅《*CA Spectrum 管理员指南*》。

一步创建和修改模型

本节中提供的示例说明如何使用单个命令字符串创建模型并将默认属性值替换为其他值。只有在尝试执行命令之前就已知晓为命令提供的相关模型标识符的值的情况下，才能执行本节中展示的复杂命令。

下例使用在“创建用户模型”和“修改模型属性”中介绍的参数值：

```
$ ./create model mth=0x10004 attr=0x1006e,val=j_doe attr=0x1007a,val=ADMIN,5
```

CLI 脚本文件示例 – 创建新用户

您可以在 Windows 平台上通过 bash 提示符执行集成 CLI 命令的 shell 脚本，这个过程与在 UNIX 上从 shell 命令提示符下执行命令一样简单。

下面的示例演示了如何使用脚本来创建 CA Spectrum 用户模型。

```
#
#  检查是否设置了 CLIPATH。如果未设置，我们必须创建 CLIPATH。
#
#  设置一个变量以指向 /install_area/vnmsh 目录，这样才能
#  找到需要的命令。
#
if [ -z "$CLIPATH" ]
then
     CLIPATH=/usr/data/spectrum/7.0/vnmsh
     export CLIPATH
fi
#
#  执行测试，确保 CLIPATH 指向有效的目录
#
if [ ! -d $CLIPATH ]
then
        echo "错误：找不到 $CLIPATH"
        echo "请查找 vnmsh 目录的正确路径，并将"
        echo "CLIPATH 环境变量设置为该值。"
        exit 0
fi
#
#  现在检查有多少个命令行参数。如果没有
#  任何参数，则回显用法消息。如果有一个参数，应该就是我们
#  创建新用户所需要的... 如果还有第二个参数，就可以
#  同时设置 Community_String。
#
#  此设置仅用于在本地系统上创建用户或者为 vnm_hostname 设置
#  .vnmshrc 文件指向的内容。还可以添加第三个字段，
#  以用于接受要连接的 vnm_hostname。
#
#  另外，可以选择使用 getopts shell 命令将“switches”解析到
#  脚本：使用 -n 解析名称、使用 -c 解析团体字符串，使用 -v 解析 vnm_hostname。
#
#  （注意：如果脚本是在 bourne shell (sh) 中执行，getopts 应当位于
#   /usr/bin/getopts 中。k-shell 内置有 getopts 函数）
#
if [ $# -eq 0 ]
then
        echo "用法:  $0 username [Community_String]"
        exit 1
```

```
elif [ $# -eq 1 ]
then
        command="attr=0x1006e,val=$1"
        flag=0
elif [ $# -eq 2 ]
then
        command="attr=0x1006e,val=$1 attr=0x1007a,val=$2"
        flag=1
fi

#
#  至此，应当完成了所有设置，可以创建新用户了。
#  必须做的第一件事是建立连接。
#
$CLIPATH/connect
#
#  现在检查连接的退出状态，看看我们是否已进入...
#
if [ $? -ne 0 ]
then
        echo "错误: 无法连接到 SpectroSERVER。$0 正在退出"
        exit 0
fi
#
#  好的，如果执行到这里，说明已建立了连接。我们现在尝试执行
#  create 命令。
#
$CLIPATH/create model mth=0x10004 $command
#
#  现在，再次检查退出状态，看看是否创建了模型。
#
if [ $? -ne 0 ]
then
        echo "错误: 无法创建新用户。$0 正在退出"
        exit 0

else
        echo -n "新用户 $1 已创建"
        if [ $flag -eq 1 ]
then
        echo " Community_String 已设置为 $2"
fi
        echo "已成功创建新模型... 正在退出。"
fi
$CLIPATH/disconnect
exit 1
```

生成事件报告

CLI 会保存一个事件列表，能够记录格局中最近发生的 2000 个事件。但是，如果格局中发生了大量事件，最近事件可能就是大约一小时内发生的事件。

通过在使用 show events 命令时带有 -x 选项，可以设置 SPECROOT 环境变量。以下命令是使用 CLI 运行事件报告的示例：

```
$ ./show events | more
$ SPECROOT=/home/spectrum; export SPECROOT
$ ./show events -x > event_rpt
```

模型切换

jump 和 setjump 命令在脚本中非常有用，可用于在不同模型之间来回切换。使用 setjump 命令，可以分配一个文本字符串来表示模型句柄及其相应的格局句柄。然后，可以使用带有该文本字符串的 jump 命令来检索该信息并作为当前模型句柄。例如：

- $ ./current mh=0xb6000f8

  当前模型是 0xb6000f8
  当前格局是 0xb600000

- $ ./setjump emme

  模型 0xb6000f8 和格局 0xb600000 已存储在 emme 下

- $ ./jump emme

  当前模型是 0xb6000f8
  当前格局是 0xb600000

创建故障排除者模型

您可以使用 CLI 创建故障排除者模型，然后将其与用户模型进行关联。创建并关联故障排除者模型之后，才能为其分配警报。这些模型随后才能接收必须调查的电子邮件通知并解决警报。

下面的过程介绍如何创建故障排除者模型并将其与用户模型进行关联。我们将以在“创建用户模型 (p. 15)”中创建的用户模型为例进行介绍。

遵循这些步骤:

1. 导航到 *<$SPECROOT>*/vnmsh 目录。
2. 在命令提示符（如使用 $ 提示符的 bash shell）下键入以下命令来连接 SpectroSERVER：

   ```
   $ ./connect
   ```

3. 确定故障排除者模型类型。输入以下命令：

   ```
   $ ./show types | grep -i trouble
   ```

   此时会返回故障排除者模型类型条目。

   ```
   0x10372        TroubleShooter        V,I
   ```

4. 确定故障排除者模型类型 EmailAddress 属性 ID。输入以下命令：

   ```
   $ ./show attributes mth=0x10372 | grep -i email
   ```

   此时会显示 EmailAddress 条目：

   ```
   0x11d24        EmailAddress          文本字符串      R,W,D
   ```

5. 使用 CLI create 命令创建故障排除者模型：

   ```
   $ ./create model mth=0x10372
   attr=0x1006e,val=j_doe_fixit
   attr=0x11d24,val=j_doe@aprisma.com
   ```

   当看到类似下例所示的系统消息时，即可确认模型创建完毕：

   ```
   已创建模型，句柄 = 0xbe0001c
   ```

   注意：这些示例中使用的所有句柄和 ID 都是虚构的。您创建的模型的模型句柄会有所不同。这取决于您的系统为它创建的句柄。

6. 使用 CLI create 命令在 j_doe 用户模型 (mh=0xbe0001b) 和 j_doe_fixit 故障排除者模型 (mh=0xbe0001c) 之间建立关联：

   ```
   $ ./create association rel=Is_Assigned lmh=0xbe0001b rmh=0xbe0001c
   ```

   当看到类似下例所示的系统消息时，即可确认已建立关联：

   ```
   $ create association: 成功
   ```

将警报分配给故障排除者模型

本节介绍如何使用 CLI 命令列出警报以及将警报分配给故障排除者模型。

遵循这些步骤:

1. 使用 show 命令列出警报。

   此步骤介绍如何才能只找出 alarm_severity 为“主要”的警报。

   ```
   $ ./show alarms | grep 主要
   ```

   此时会显示“主要”警报列表。例如：

```
7509   09/27/2000   14:46:44   0xd80008   0xa6000df   duncan    9E133_36   主要   否
7645   09/27/2000   14:47:16   0xd80008   0xa60025e   infinity  9H422_12   主要   否
7518   09/27/2000   14:47:01   0xd80008   0xa6000eb   rugone    9E132_15   主要   否
7979   09/27/2000   14:53:12   0xf40002   0xa600161   FDDI2     FddiMAC    主要   否
8018   09/27/2000   14:53:13   0xf40002   0xa6003da   FDDI FNB  FddiMAC    主要   否
7512   09/27/2000   14:46:47   0xd80008   0xa6000af   ruthere   9A426_02   主要   否
```

2. 选择要为其分配故障排除者的警报。在本示例中，选择的是针对 9A426-02 设备的警报 ID 7512。

3. 选择要分配给警报的故障排除者。在本示例中，选中了在“创建故障排除者模型 (p. 20)”中创建的 j_doe_fixit 故障排除者模型。

   注意：在下一步中，要在 update 命令中指定故障排除者，应使用故障排除者模型句柄 0xa600722，而不是故障排除者模型名称 j_doe_fixit。

4. 使用 update 命令将警报分配给故障排除者：

   ```
   $./update alarm aid=7512 assign=0xa600722
   ```

   当看到类似下例所示的系统消息时，即可确认已为警报分配故障排除者：

   ```
   $ update: 成功
   ```

   现在已经为由 j_doe_fixit 模型代表的用户分配了警报。此用户会收到警报分配的电子邮件通知。

创建全局集合

您可以创建一个全局集合，并可以在 CLI 会话中使用 GUID 设置搜索条件。唯一标识符 (GUID) 是创建全局集合的关键属性。必须使用 GUID，全局集合才能正常工作。您可以通过对 VNM 模型执行操作来获取 GUID。

注意：没有 GUID 的全局集合在 CLI 中是无效的。使用 OneClick 创建全局集合时，即会自动生成 GUID。有关详细信息，请参阅 *CA Spectrum IT 基础架构建模与管理 - 管理员指南*。

遵循这些步骤:

1. 输入以下命令：

   ```
   update action=(用于获取唯一标识符的操作) 0x10474 mh= (VNM 模型句柄)
   ```

   此时会生成 GUID。

   注意：获取新 GUID 的操作是 0x10474，该操作与全局集合模型类型一样。

2. 输入以下命令，以便创建使用 GUID 的全局集合：

   ```
   create model mth=(全局集合的模型类型) 0x10474 attr=(GUID) 0x12e56, val=(接收的前一值) 4a85b9af-0d52-1000-017f-0013727f8c0a
   ```

   此时会创建一个全局集合，并显示在导航窗格中的“全局集合”下。

示例：创建带或不带 XML 字符串的全局集合

输入以下命令，以便创建一个带或不带 XML 字符串的全局集合：

```
update action=(获取 GUID 的操作)0x10474 mh=(格局)

输出:
update action: 成功
响应有 1 个属性:
 0) Attribute 0x0 text: EXAMPLE GUID(4a85b9af-0d52-1000-017f-0013727f8c0a)

create model mth=(全局集合的模型类型)0x10474
attr=(GUID)0x12e56,val=(接收的前一值)4a85b9af-0d52-1000-017f-0013727f8c0a
attr=(dynamicCriteriaXML)0x12a6a,val=(XMLString)'<search-criteria><filtered
models><equals-ignore-case><model-name>sometext</model-name></equals-ignore-case></filtered-models></search-criteria>'
输出:
created model handle = New model handle(0x78101069)
```

注意：您可以使用 create 命令指定 dynamicCriteriaXML (0x12a6a) 属性，也可以稍后再更新此模型。

详细信息：

命令说明 (p. 25)

在 CLI 输出中抑制标头

要在 CLI 输出中抑制标头，您可以创建一个包含下节所提供的函数的文件，并可以在每个脚本的顶部引用此文件。

以下过程中的函数调用了 CLI 命令函数并从命令输出中剔除了标头。

遵循这些步骤:

1. 在您的脚本目录中创建一个名为 StripHeaders 的文件。
2. 在 StripHeaders 文件中包含以下函数：

```
tcreate() # 仅 create alarm 命令
    { # 和 create event 命令需要
    $CLIPATH/create $@ | tail +2
    }
tseek()
    {
      $CLIPATH/seek $@ | tail +2
    }
tshow()
    {
    $CLIPATH/show $@ | tail +2
    }
tupdate()
    {
    $CLIPATH/update $@ | tail +2
    }
```

3. 在 CLI 脚本的顶部包含名称 StripHeaders，如下所示：

```
. StripHeaders
```

4. 如果您想从 CLI 输出中剔除标头，只需调用 tcreate()、tseek()、tshow() 和 tupdate() 函数，而不必执行相应的 CLI 命令。

   例如，以下代码行会在执行 show models 命令时生成不包含 CLI 标头信息的输出：

```
tshow models
```

第 3 章： 命令说明

本章提供 CLI 命令和输出的相关说明。

此部分包含以下主题：

命令说明概述

使用 CLI，可以更改 CA Spectrum 知识库，而没有像 CA Spectrum 所具备的那样的安全措施。如果您指定的信息不正确，可能会导致系统崩溃或数据库损坏。因此，使用 create、destroy 或 update 命令时应该谨慎操作。

注意：可以使用 CLI 命令参数来创建和管理与 CLI 相关的响应时间测试。

有关详细信息，请参阅 *CA Spectrum Service Performance Manager 用户指南*。

ack alarm – 确认警报

ack alarm 命令可确认通过 landscape_handle 指定的格局中由 alarm_id 指定的警报。如果未指定 landscape_handle，此命令可确认当前格局中由 alarm_id 指定的警报。

确认模型的一个警报意味着您只确认该警报，而不会确认该模型的其他警报。

此命令采用以下格式：

```
ack alarm aid=<alarm_id> [lh=<landscape_handle>]
```

如果在输入的 ack alarm 命令中指定了有效的 alarm_id 和 landscape_handle，会显示以下消息：

```
ack alarm: 成功
```

示例：ack alarm

```
$ ack alarm aid=42 lh=0x400000
ack alarm: 成功
```

connect – 连接到 SpectroSERVER

connect 命令可将 CA Spectrum 命令行界面的用户连接到在主机系统 (*hostname*) 上运行的 SpectroSERVER。此命令*也*可将由 landscape_handle 指定的格局设置为当前格局。如果 CLI 本地服务器尚未运行，connect 命令会启动它。

此命令采用以下格式：

```
connect [<hostname>] [lh=<landscape_handle>][vnmsocket=<vnmsocket>]
```

hostname

（可选）如果未指定 hostname，此命令会将用户连接到在 CLI 资源文件 .vnmshrc 中指定的主机。

注意：CA Spectrum 命令行界面不支持 localhost 或 127.0.0.1 选项。要连接到本地主机，您可以指定实际的主机名，或不指定任何参数。

landscape_handle

（可选）如果未指定 landscape_handle，此命令会将指定主机名的格局设置为当前格局。

vnmsocket

（可选）如果未指定 vnmsocket，此命令会使用在 .vnmshrc 文件中指定的套接字连接到 SpectroSERVER。可以使用 vnmsocket 通过由 vnmsocket 定义的其他端口连接来连接到另一个 SpectroSERVER。

在 UNIX 中，CLI 本地服务器所报告的错误消息显示在控制台窗口中。在 Windows 上，这些错误显示在用户 bash shell 窗口中。

示例：connect

```
#! /usr/bin/sh
# 获取特定重要级别的警报
# 并设置 CLISESSID 的脚本示例

if [$# !=1]
then
    echo "用法: $0 <alarm severity>"
    exit 0
fi

CLISESSID=$$

$SPECROOT/vnmsh/connect
$SPECROOT/vnmsh/show alarms | grep -i $1
$SPECROOT/vnmsh/disconnect

exit 0
```

如果命令执行成功，则会显示以下消息：

```
connect: 成功连接到 <host_name>
当前格局是 <landscape_handle>
```

Hostname 是用户输入的 SpectroSERVER 主机或在 .vnmshrc 文件中指定的主机。landscape_handle 是用户输入的格局或主机的格局。

详细信息：

CLI 环境变量 (p. 9)
启动文件 (p. 10)

使用 connect 命令时的注意事项

使用 connect 命令时，务必考虑以下注意事项：

- 终端上的用户必须使用 connect 命令来启动通信。该用户还必须使用 disconnect 命令来终止与 SpectroSERVER 的通信。
- 一旦第一个用户输入了 connect 命令，CLI 本地服务器即会连接到 SpectroSERVER。
- 使用同一 CLI 本地服务器的其他 CLI 用户只能连接到初始 SpectroSERVER 的格局映射中的 SpectroSERVER。一旦所有用户断开连接之后，即可使用 connect 命令连接到不同格局映射中的 SpectroSERVER。

- 要成功连接到 SpectroSERVER，必须将使用 connect 命令的第一个用户定义为原始 SpectroSERVER 的 CA Spectrum 数据库中的用户。
- 在 bash shell 中运行 CLI 的 Windows 用户也必须定义 CLISESSID。
- 用于特殊用户的终端设备可以使用 ttyslot(3V) 函数来确定。
- Cron 脚本未附加到 ttyslot。因此，ttyslot 函数针对所有 cron 脚本都会返回 0。也就是说，将两个 CLI 脚本作为 cron 脚本同时运行时，看起来就像同一个 CLI 用户访问 CLI 本地服务器，这会导致不可预知的结果。因此，您必须在脚本顶部插入一行代码，以便导出环境变量 CLISESSID。为 CLISESSID 设置唯一的数值。CLI 现在可以区分不同的 cron 脚本。

  以下示例在脚本中定义了唯一的 CLI 会话 ID：

  ```
  CLISESSID=$$; export CLISESSID
  ```

- 此例将 CLISESSID 设置为运行脚本的 shell 的进程 ID。在 ttyslot 函数返回零时，CLI 使用 CLISESSID 识别用户。为每个 CLI 会话设置一次 CLISESSID。如果作为 cron 脚本运行的 CLI 脚本调用了其他 CLI 脚本，只有顶层脚本需要设置 CLISESSID 环境变量。除非您在脚本顶部调用新 shell (#!/bin/sh)，否则其他 CLI 脚本都会在同一进程 ID 下运行。要在其他脚本中调用新 shell，可以导出 CLISESSID，然后重新连接后再断开连接。
- 在一些环境或配置中，即使命令是从命令行输入的，ttyslot 函数也可能返回零。在这种情况下，connect 命令会返回以下错误：

  ```
  connect: 变量 CLISESSID 未设置
  ```

  在这种情况下，可从命令行设置 CLISESSID，也可从 .cshrc 或其他启动文件进行设置

- CLI 使用用户名和终端设备来标识每个 CLI 用户。CLI 会将通过终端设备同时运行多个脚本的用户视为同一个用户。如果一个脚本在后台运行而另一个脚本在前台运行，或者多个脚本在后台运行，CLI 可能会产生不可预知的结果。

  例如，脚本 A1 将当前模型设置为模型 A。由同一个用户通过相同终端设备运行的脚本 B1 会将当前模型设置为模型 B。如果脚本 A1 在模型 A 上执行 update 命令，也会在脚本 B1 的模型 B 上执行 update 命令。

  每次从特殊终端设备仅执行一个 CLI 会话。要同时运行多个 CLI 会话，请从单独的终端设备运行这些会话，或使用 at(1) 或 batch(1) 命令并针对每个会话将 CLISESSID 环境变量设置为唯一值来运行。

create – 创建对象

使用 create 命令创建对象。

注意：有关如何在安全域中创建模型的信息，请运行 ./create 查看用法语句。

此命令采用以下格式：

```
create  model ip=<IP Address | Low_IP-High_IP>
[sec_dom=Secure_Domain_Address][comm=Community_Name] [to=Time_Out]
[tc=Try_Count] [lh=landscape_handle] |
create  model mth=model_type_handle [attr=attribute_id,val=value ...]
[lh=landscape_handle] |
create  association rel=relation lmh=left_model_handle rmh=right_model_handle
create alarm [-nr] sev=alarm_severity cause=probable_cause_id [mh=model_handle]
|
create  event type=event_type text=event_text
[mh=model_handle|lh=landscape_handle]
```

详细信息：

CLI 环境变量 (p. 9)

create alarm

create alarm 命令可为模型句柄为 model_handle 的模型创建重要级别为 alarm_severity、原因为 probable_cause_id 的警报。有效的警报重要级别选项有：关键、主要、次要、正常、维护、已抑制或初始。默认情况下，新警报会替换现有警报。

如果在输入的 create alarm 命令中指定了有效的 alarm_severity、probable_cause_id 和 model_handle，会显示警报表中已创建的条目。创建时间采用 hh:mm:ss 格式显示。

示例：create

```
$ create alarm sev=关键 cause=0x10308 mh=0x400134

ID      日期            时间          可能原因 ID     模型句柄      模型名称      模型类型名称      重要级别      确
认
984     05/11/2000     12:33:27     0x10308      0x400134     12.84     Bdg_CSI_CN     关键      否
```

create association

create association 命令可在模型句柄分别为 left_model_handle 和 right_model_handle 的两个模型之间创建关系（关联）实例。

如果在输入的 create association 命令中指定了 left_model_handle 和 right_model_handle 之间的有效关系，则会显示以下消息：

```
create association: 成功
```

示例：create association

```
$ create association rel=Collects lmh=0x400009 rmh=0x400134
create association: 成功
```

create event

create event 命令可为由 model_handle 指定的模型创建类型为 event_type、包含文本 event_text 的事件。如果指定了 landscape_handle，则针对创建此事件的用户模型创建事件。

注意：在 CLI 的先前版本中，是针对格局模型创建事件。

如果未指定 model_handle 或 landscape_handle，则针对创建此事件的用户模型创建事件。如果指定了模型，则针对当前模型创建事件。CA Spectrum 中的部分事件缺少关联模型。例如，应用程序连接到 SpectroSERVER 时，没有模型与事件关联。

如果在输入的 create event 命令中指定了有效的 event_type、event_text 以及 model_handle 或 landscape_handle（如果有），相应条目就会显示在事件表中。创建时间采用 hh:mm:ss 格式显示。

event_type 命令（在 CA Spectrum 中也称为事件代码）是 4 字节的十六进制数。两个最重要的字节指定事件的开发者 ID（0001 表示 CA Spectrum 生成的事件代码），另外两个次要字节是唯一的事件标识符。并非所有事件类型都包含用户输入的文本。例如，在事件格式文件中包含变量 {S 0} 的事件类型就是这种事件类型。对于未包含用户输入文本的事件类型，event_text 参数会被忽略，但仍必须在命令行上提供该参数。

有关详细信息，请参阅 *《CA Spectrum 认证用户指南》*。

示例：create event

```
$ create event type=0x1061a text=“fan down” mh=0x40013

日期          时间        类型        模型句柄      模型名称      模型类型名称
05/11/2000    12:39:42    0x1061a     0x400134      12.84         Bdg_CSI_CNB20
```

create model

您可以随 create model 命令指定 IP 地址或模型类型句柄。不管指定其中的哪个参数，系统都会在由 landscape_handle 指定的格局中创建模型。如果未指定 landscape_handle，此命令会在当前格局中创建模型。

注意：只有创建用户模型时才需要 model_name 属性。

- 如果随 create model 命令指定了 IP 地址，系统会在指定的 ip_address 中查找对象并为其创建模型。创建的模型将具有该对象的所有属性，包括所有关联的子属性。例如，如果对象是集线器，create model 命令会创建一个包含所有端口的集线器模型。
- 您可以指定 IPv4 地址或 IPv6 地址。此命令不支持 IPv6 范围，也不支持设置属性 ID。
- 要同时创建多个模型，可以在 create model 命令中定义 IP 地址范围。指定 Low_IP 和 High_IP 参数，两者之间用“-”进行分隔。如果未指定 Community_Name，新创建的模型的类型为“可 ping”。如果指定了 Community_Name，则会使用合适的模型类型为设备建模。Try_Count 和 Time_Out 选项与 OneClick 中“按 IP 地址创建模型”对话框中的选项类似。
- 如果在 create model 命令中指定了模型类型句柄，系统会创建 model_type_handle 类型的模型。随后可以为创建的模型设置一个或多个属性值。
- 如果在 create model 命令中指定了模型类型句柄，还可以在同一个命令中指定多个属性。指定多对“attribute_id, value”并用空格对邻接对进行分隔。
- 应在 create model 命令中指定用户在使用 OneClick 创建特定类型的模型时指定的属性值。否则，在创建模型时，SpectroSERVER 内可能会出现推理处理程序错误。例如，使用 OneClick 创建 Hub_CSI_IRM3 模型时，会显示一个窗口，您可以在其中输入“模型名称”、“网络地址”、“团体字符串”的值。使用 create model 命令时也可以指定这些属性的值，以便通过 CLI 创建同一类型的模型。
- 如果在输入的 create model 命令中指定了有效的 model_type_handle 和“attribute_id, value”对（如果有），会显示所创建的模型句柄。
- 如果在输入的 create model 命令中指定了有效的 ip_address，会显示所创建的模型句柄。

示例：create model

```
$ create model mth=0x102d attr=0x12d7f,val=132.177.12.84
attr=0x1006e,val=12.84lh=0x400000
已创建模型，句柄 = 0x400134

$ create model ip=206.61.231.1-206.61.231.5

正在为 IP=206.61.231.1 创建模型
已创建模型，句柄 =  0x9a00259
正在为 IP=206.61.231.2 创建模型
create model: DCM 设备不可访问
正在为 IP=206.61.231.3 创建模型
create model: DCM 设备不可访问
正在为 IP=206.61.231.4 创建模型
create model: DCM 设备不可访问
正在为 IP=206.61.231.5 创建模型
已创建模型，句柄 =  0x9a0025a
```

注意：默认情况下，create 命令显示的模型名称长度不超过 16 个字符。但可以使用环境变量 CLIMNAMEWIDTH 来指定显示的模型名称可包含的字符数（最多可达 1024 个字符）。

current – 设置模型或格局

current 命令可将由 model_handle 指定的模型设置为当前模型。此命令也可将由 landscape_handle 指定的格局设置为当前格局。如果未指定 model_handle 和 landscape_handle，current 命令将显示当前模型句柄和当前格局句柄。

用户设置当前模型时，CLI 会将包含此模型的格局设置为当前格局。用户设置当前格局时，CLI 会将当前模型设置为未定义。

系统会为连接到 CLI 本地服务器的每个会话分别维护单独的当前模型和当前格局值。

例如，current 命令只为指定的会话保留状态信息、当前模型和当前格局。

该命令采用以下格式：

```
current [mh=<model_handle>|lh=<landscape_handle>]
```

- 如果输入命令时指定了有效的 model_handle，则会显示以下消息：

  当前模型是 <model_handle>
  当前格局是 <current_landscape_handle>

- 如果输入命令时指定了有效的 landscape_handle，则会显示以下消息：

  当前模型未定义
  当前格局是 <landscape_handle>

- 如果未指定 model_handle 和 landscape_handle，则会显示以下消息：

  当前模型是 <current_model_handle>
  当前格局是 <current_landscape_handle>

- 如果未指定 model_handle 和 landscape_handle，并且未定义当前模型，则会显示以下消息：

  当前模型未定义
  当前格局是 <current_landscape_handle>

示例：current

```
$ current mh=0x400142
当前模型是 0x400142
当前格局是 0x400000

$ current lh=0x500000
当前模型未定义
当前格局是 0x500000

$ current
当前模型未定义
当前格局是 0x500000
```

注意：当前格局是由 connect 命令设置的，因此始终包含值。

destroy – 销毁对象

使用 destroy 命令销毁对象。该命令采用以下格式：

```
destroy model [-n] mh=model_handle |
destroy association [-n] rel=relation lmh=left_model_handle
rmh=right_model_handle|
destroy alarm [-n] aid=alarm_id [lh=landscape_handle]
```

-n

如果随 destroy 命令指定了 -n（无提示）选项，系统不会提示您确认。此选项在 CLI 脚本中非常有用。

除非指定了 -n 选项，否则始终会显示以下消息：

```
destroy model: 确定要执行此操作吗?
destroy association: 确定要执行此操作吗?
destroy alarm: 确定要执行此操作吗?
```

有效的回复有 y、yes、Y、Yes、n、no、N 和 No。

destroy alarm

销毁由 landscape_handle 指定的格局中由 alarm_id 指定的警报。除非指定了 -n 选项，否则 destroy alarm 会在销毁警报之前提示您确认。如果未指定 landscape_handle，此命令会销毁当前格局中由 alarm_id 指定的警报。使用 show alarms 命令为模型确定 alarm_id。

如果在输入的 destroy alarm 命令中指定了有效的 alarm_id 和 landscape_handle，会显示以下消息：

```
destroy alarm: 成功
```

示例：destroy alarm

```
$ destroy alarm aid=300
destroy alarm: 确定要执行此操作吗? y
destroy alarm: 成功
```

destroy association

销毁模型句柄分别为 left_model_handle 和 right_model_handle 的两个模型之间的关联（关系实例）。除非指定了 -n 选项，否则 destroy association 会在销毁关联之前提示您确认。

如果在输入的 destroy association 命令中指定了 left_model_handle 和 right_model_handle 之间的有效关系，会显示以下消息：

```
destroy association: 成功
```

示例：destroy association

```
$ destroy association rel=Lost_and_Found lmh=0x400001 rmh=0x40h0142
destroy association: 确定要执行此操作吗? y
destroy association: 成功
```

destroy model

销毁包含指定 model_handle 的模型。除非指定了 -n 选项，否则 destroy model 会在销毁模型之前提示您确认。

如果在输入的 destroy model 命令中指定了有效的 model_handle，会显示以下消息：

```
destroy model: 成功
```

示例：destroy model

```
$ destroy model mh=0xa600715

以下模型将被销毁：
Model_Handle      ->     0xa600715
Model_Type_Handle ->     0x10004
Model_Name        ->     garciaparra
Model_Type_Name   ->     User

destroy model: 确定要执行此操作吗？y
destroy model: 成功
```

disconnect – 断开与 SpectroSERVER 的连接

使用 disconnect 命令可以断开 CLI 用户与当前连接的 SpectroSERVER 之间的连接。

该命令采用以下格式：

```
disconnect
```

如果命令执行成功，则会显示以下消息，其中 hostname 是用户所连接的 SpectroSERVER 主机的名称：

```
disconnect: 成功从 <hostname> 或 <IP address> 断开 - 已连接 xx 小时 yy 分
```

详细信息：

stopShd – 终止 CLI 本地服务器 (p. 55)

jump – 跳转到保存的模型或格局

jump 命令可跳转到先前已保存的模型和格局。jump 命令可将当前模型和当前格局设置为由 setjump 命令保存在标签 text_string 下的模型和格局。如果未指定 text_string，则会显示前一个 setjump 命令所提供的 text_string 列表。

此命令采用以下格式：

```
jump [<text_string>]
```

- 如果在输入的 jump 命令中指定了事先已定义的有效 text_string，会显示新的当前模型和当前格局：

  ```
  当前模型是 <current_model_handle>
  当前格局是 <current_landscape_handle>
  ```

- 如果输入的 jump 命令不带 text_string，则会显示当前已定义的 text_string 的列表。例如：

  ```
  text_string1
  text_string2
  --
  ```

- 如果输入 jump 并且未定义新的当前模型，则会显示以下消息：

  ```
  当前模型未定义
  当前格局是 <current_landscape_handle>
  ```

示例：jump

```
$ jump tutorial
当前模型是 0x400142
当前格局是 0x400000
```

详细信息：

setjump – 保存模型和格局 (p. 39)

seek – 定位模型

使用 seek 命令定位模型。seek 命令可在由 landscape_handle 指定的格局中查找由 attribute_id 所指定的属性包含指定值的模型。如果未指定 landscape_handle，此命令会在当前格局中查找由 attribute_id 指定的属性包含指定值的模型。您还可以随 seek 命令使用通配符 (*)，以便查找包含指定子字符串的模型实例。如果输入空值，则可以查找没有名称的所有模型（如 attr=0x1006e）。

使用 seek 命令无法搜索包含一个字符的属性值。尝试执行这种搜索时会返回错误。

此命令采用以下格式：

```
seek [-i] [-s]  attr=attribute_id,val=value [lh=landscape_handle]
```

其中的选项可以采用任何顺序排列，例如 -i -s 或 -s -i。

-i

如果随 seek 命令指定了 -i（不区分大小写）选项，则会返回包含由 val 参数指定的模型信息（不区分大小写）的模型。

-s

如果随 seek 命令指定了 -s（允许子字符串）选项，则会返回由 val 参数指定的模型信息包含子字符串（如果适用）的模型。

如果在输入的 seek 命令中指定了有效的 attribute_id 和有效值，会采用以下格式显示所有匹配的模型：

```
模型句柄        模型名称    模型类型句柄            模型类型
模型句柄        名称         模型类型句柄            名称
```

如果找不到匹配的模型，则会显示以下消息：

```
seek: 找不到模型
```

注意：默认情况下，seek 命令显示的模型名称长度不超过 16 个字符。但可以使用环境变量 CLIMNAMEWIDTH 来指定显示的模型名称可包含的字符数（最多可达 1024 个字符）。

示例：seek

```
$ seek attr=0x1006e,val=CA Spectrum

模型句柄        模型名称    模型类型句柄    模型类型名称
0xb100018       spectrum    0x1004      User
0xb10008d       spectrum    0x820000    ScmConfig

$ seek attr=0x1006e,val=CA Spectrum

模型句柄        模型名称    模型类型句柄    模型类型名称
0xb100018       spectrum    0x820000    ScmConfig

$ seek attr=0x1006e,val=SPE
seek: 找不到模型

$ seek attr=0x1006e,val=spe lh=0xb100000
seek: 找不到模型

$ seek -i attr=0x1006e,val=CA Spectrum

模型句柄        模型名称    模型类型句柄    模型类型名称
0xb10018        spectrum    0x10004     User
0xb1008c        spectrum    0x820000    ScmConfig
0xb1008d        spectrum    0x820000    ScmConfig
```

```
$ seek -i -s attr=0x1006e,val=CA Spectrum

模型句柄      模型名称    模型类型句柄    模型类型名称
0xb10018      spectrum    0x10004    User
0xb10089      spectrum    0x820000   ScmConfig
0xb1008c      spectrum    0x820000   ScmConfig
0xb1008d      spectrum    0x820000   ScmConfig

$ seek -i -s attr=0x1006e,val=CA Spectrum lh=0xb100000

模型句柄      模型名称    模型类型句柄    模型类型名称
0xb10018      spectrum    0x10004    User
0xb10089      spectrum    0x820000   ScmConfig
0xb1008c      spectrum    0x820000   ScmConfig
0xb1008d      spectrum    0x820000   ScmConfig

$ seek -s attr=0x1006e,val=CA Spectrum

模型句柄      模型名称    模型类型句柄    模型类型名称
0xb10008c     spectrum    0x820000   ScmConfig

$ seek attr=0x110df,val=0.0.C.18

seek: 找不到模型

$ seek -s attr=0x110df,val=0.0.C.18

模型句柄      模型名称    模型类型句柄    模型类型名称
0xb100070     frog10       0x210022     Rtr_CiscoIGS
0xb100072     frog10_1     0x220011     Gen_IF_Port
0xb10005b     cisco rtr    0x210022     Rtr_CiscoIGS
0xb100070     frog10_2     0x220011     Gen_IF_Port
0xb100070     cisco rtr_1  0x220011     Gen_IF_Port
0xb100070     cisco rtr_2  0x220011     Gen_IF_Port

$ seek attr=0x1006e,val=spe*

模型句柄      模型名称    模型类型句柄    模型类型名称
0xb10018      spectrum         0x10004     User
0xb10089      spectrum         0x820000    ScmConfig
0xb1008d      spectrum         0x820000    ScmConfig

$ seek attr=0x1006e,val=

模型句柄      模型名称    模型类型句柄    模型类型名称
0xd00258      0x102c8                     Physical_Addr
0xd002f8      0x102c8                     Physical_Addr
0xd00368      0x820000                    ScmConfig
0xd00259      0x102c8                     Physical_Addr
0xd002f9      0x102c8                     Physical_Addr
0xd00301      0x102c8                     Physical_Addr
```

```
$ seek attr=0x12d7f,val=192.168.93.*

MHandle       MName              MTypeHnd      MTypeName
0x28000190    192.168.93.14      0xd0004       HubCSIEMME
0x28000190    192.168.93.14      0xd0004       HubCSIEMME
0x280001a0    192.168.93.14_Sy   0x23001c      System2_App
0x28000198    192.168.93.14_St   0x590006      RMONApp
0x28000191    192.168.93.14_A    0xd000a       CSIIfPort
0x280001a1    192.168.93.14_IC   0x230012      ICMP_App
0x28000199    192.168.93.14_E    0x590013      RMONEthProbe
0x280001a2    192.168.93.14_UD   0x230019      UDP2_App
0x280001c2    DLM App            0x830001      DLM_Agent
0x2800019a    192.168.93.14_E    0x590013      RMONEthProbe
0x28000192    192.168.93.14_B    0xd000a       CSIIfPort
0x2800019b    192.168.93.14_E    0x590013      RMONEthProbe
```

setjump – 保存模型和格局

setjump 命令可将当前模型句柄和当前格局保存在标签 text_string 下。用户随后可以使用带 text_string 的 jump 命令，将存储在 text_string 下的模型句柄和格局重新设置为当前模型句柄和当前格局。如果在两个 setjump 命令中使用了同一个 text_string，系统会提示用户验证。

系统会为连接到 CLI 本地服务器的每个会话分别维护单独的 setjump 值。setjump 命令只为指定的会话保留相关信息，即为会话分配的 setjump 文本字符串。

此命令采用以下格式：

```
setjump [-n] <text_string>
```

-n

如果随 setjump 命令指定了 -n（无提示）选项，那么在 text_string 之前已被使用的情况下系统不会提示。

- 如果在输入的 setjump 命令中指定了新的 <text_string> 而且当前模型已存在，则会显示以下消息：

  ```
  模型 <current_model_handle> 和格局
  <current_landscape_handle> 已存储在 <text_string> 下
  ```

  其中，<current_model_handle> 是当前模型的句柄，<current_landscape_handle> 是当前格局的句柄。

- 如果在输入的 setjump 命令中指定了新的 <text_string> 而且当前模型不存在，则会显示以下消息：

  模型未定义，格局 <current_landscape_handle>
  已存储在 <text_string> 下

- 如果在输入的 setjump 命令中指定了已在前一个 setjump 命令中定义的 text_string，则会显示以下消息：

  setjump model: <text_string> 已被使用。要覆盖吗?

  有效的回复有 y、yes、Y、Yes、n、no、N 和 No。

示例：setjump

```
$ current mh=0x400142
当前模型是 0x400142
当前格局是 0x400000
$ setjump -n tutorial
模型 0x400142 和格局 0x400000 已存储在 tutorial 下
```

show – 显示对象

要显示对象，请使用 show 命令。

此命令采用以下格式：

```
show models [mhr=low_model_handle-high_model_handle]
        [mth=model_type_handle][mname=model_name][lh=landscape_handle] |
devices [lh=landscape_handle]|
landscapes |
types [mthr=low_mth-high_mth] [mtname=mt_name]
                [flags=V|I|D|N|U|R] [lh=landscape_handle] |
relations [lh=landscape_handle] |
associations [mh=model_handle] |
parents [rel=relation] [mh=model_handle] |
children [rel=relation] [mh=model_handle] | attributes [-e]
                [attr=attribute_id[,iid=instance_id][,next]...|
                [attrr=low_attr-high_attr] [attrname=attr_name]]
                [mh=model_handle] |
attributes [-c] [-e]
                [attr=attribute_id[,iid=instance_id][,next]...|
                [attrr=low_attr-high_attr] [attrname=attr_name]]
                [mh=model_handle]|
attributes mth=model_type_handle [attrr=low_attr-high_attr]
                [attrname=attr_name]          [flags=E|R|W|S|T|G|O|M|D|P|L|V]
                [lh=landscape_handle] |
alarms [-a] [-x] [-t] [-s]
               [mh=model_handle|lh=landscape_handle] |
events [-x] [ -a | -n no_events ]
                [mh=model_handle|lh=landscape_handle] |
inheritance mth=model_type_handle [lh=landscape_handle] |
rules rel=relation [lh=landscape_handle] |
enumerations [attr=attribute_id] [mth=model_type_handle]
                [lh=landscape_handle] |
watch [mh=model_handle]
```

-a

如果指定了 -a（全部）选项，show alarms 命令不会进行任何屏蔽，而会显示所有关键、主要、次要、维护、已抑制或初始警报。

-x

如果指定了 -x（扩展）选项（并且设置了变量 $SPECROOT），show alarms 命令输出的结尾处会显示可能的原因文本。show events 命令的输出会显示事件格式。show events -x 命令显示的字符数由 .vnmshrc 资源文件参数控制。

max_show_event_length

如果您使用仅支持 SpectroSERVER 的工作站运行 CLI，-x 选项不会提供正常警报原因或事件格式信息，这是因为 SG-Support 目录中没有 CsPCause 和 CsEvFormat 文件。可能出现的错误消息有：

- 未提供原因信息（与 show alarms 关联）
- 未提供事件格式信息（与 show events 关联）

要修复此问题，请将 SG-Support/CsPCause 和 SG-Support/CsEvFormat 目录及文件复制到 SpectroSERVER 工作站上的 <*$SPECROOT*>/SG-Support 目录中。

默认值：512

-e

如果指定了 -e（枚举）选项，show attributes 命令的输出中会显示数据库枚举字符串。

-c

如果指定了 -c（读取最新内容）选项，则会将“Read Mode”属性设置为“Read Most Current”。此模式使用最新用户界面轮询提供的属性值，界面轮询每 5 秒钟更新一次。如果未设置此标志，则使用“Read Most Available”模式。此模式使用最近一次 CA Spectrum 轮询存储在数据库中的最新值。轮询频率取决于用户定义的时间间隔。

-n

如果指定了 -n（事件数量）选项，show events 命令的输出中会显示指定数量的事件。

-t

如果指定了 -t（故障单 ID）选项，show alarms 命令的输出中会显示“故障单 ID”字段。

-s

如果指定了 -s（影响重要级别）选项，show alarms 命令的输出中会显示“影响重要级别”字段。

要在 show alarms 和 show events 命令中指定 -x 选项（显示可能的警报原因消息和扩展的事件消息），必须在本地服务器上安装 OneClick，并将 SPECROOT 环境变量设置为 Spectrum 支持根目录的路径。例如，如果 SG-Support 文件位于 /usr/spectrum/SG-Support 中，则将 SPECROOT 设置为 /usr/spectrum。

show alarm

show alarms 命令可显示与由 model_handle 指定的模型相关的所有警报。或者，也可以只显示与由 landscape_handle 指定的格局中的每个模型相关的、重要级别最高的警报（重要级别为关键、主要或次要的警报）。如果指定了 landscape_handle，show alarms 会屏蔽包含初始、已抑制或维护警报的所有模型。因此，只会显示包含关键、主要或次要警报的模型。如果既未指定 model_handle，也未指定 landscape_handle，show alarms 也会执行屏蔽，只显示与当前格局中每个模型相关的、重要级别最高的警报（重要级别为关键、主要或次要的警报）。

“确认”字段指出警报是否已得到确认。此字段的可能值为“是”和“否”。“陈旧”字段指出警报是否过时。此字段的可能值为“是”和“否”。“分配”和“状态”字段分别显示警报故障排除者信息和警报状态。警报创建时间采用 hh:mm:ss 格式显示。

show alarms 命令采用以下格式显示信息：

```
ID    日期                  时间                  可能原因 ID     模型句柄   模型名称   模型类型名称   重要级别   确
认   陈旧   分配   状态
id    mm/dd/yyyy   hh:mm:ss     原因 ID              句柄            名称            名称                  重
要级别   确认   陈旧   分配   状态
```

如果随 show alarms 指定 -x 选项，则会在最后一个警报后显示一个包含原因代码和可能原因文本消息的表。例如：

0x10402 DUPLICATE PHYSICAL ADDRESS0x10302 SpectroSERVER 已与此设备失去联系。

注意：show 命令显示的模型名称长度不超过 16 个字符。但可以使用环境变量 CLIMNAMEWIDTH 来指定显示的模型名称可包含的字符数（最多可达 1024 个字符）。

示例：show alarms

show alarms 命令采用以下格式显示信息：

```
$ show alarms lh=0x110000
```

```
ID      日期          时间            可能原因 ID     模型句柄      模型名称      模型类型名称   重要级别     确
认   陈旧 分配          状态
928  05/11/2000   02:33:22     0x10c04         0x110000c   infinity         VNM                  关
键              否         否       McDonald     正在处理
```

详细信息：

命令说明 (p. 25)
CLI 环境变量 (p. 9)

show association

show associations 命令可显示为由 model_handle 指定的模型定义的所有实例化关系（关联）。如果未指定 model_handle，show associations 将显示当前模型的所有实例化关系。

show associations 命令采用以下格式显示信息：

左侧模型句柄 左侧模型名称 关系 右侧模型句柄 右侧模型名称
句柄 名称 关系 句柄 名称

示例：show associations

show associations 命令采用以下格式显示信息：

```
$ show associations mh=0x400141
```

```
左侧模型句柄 左侧模型名称 关系 右侧模型句柄 右侧模型名称
0x400001 LostFound Lost_and_Found 0x400141 12.77-bridge
```

show attributes

show attributes 命令可为由 model_handle 指定的模型显示由 attr=attribute_id 指定的属性。如果未指定 attribute_id，show attributes 将列出由 model_handle 指定的模型的所有属性及属性值。

如果未指定 model_handle，show attributes 将显示当前模型的所有可用属性。您可以使用 attr=low_attr-high_attr 指定属性范围。显示特定模型的单个属性或属性列表时，可在 instance_id 中指定属性的实例 ID。instance_id 必须是用句点分隔的一系列正整数。只能为列表属性指定实例 ID。列表属性是指设置了列表标志的属性。

以下规则适用于列表属性：

- 要显示列表属性的所有属性值和实例 ID，请不要随 attribute_id 输入 instance_id。只需输入 attribute_id。
- 要显示列表属性的第一个属性值和实例 ID，请在 attribute_id 之后输入以下命令：

  ```
  ,next
  ```

- 要显示列表属性的特定属性值和实例 ID，请随 attribute_id 输入 instance_id。
- 要显示列表属性的某特定实例 ID 后的下一个属性值和实例 ID，请在 instance_id 之后输入以下命令：

  ```
  ,next
  ```

- 显示模型的所有属性时不能指定实例 ID，其原因如下：
  - 实例 ID 仅适用于列表属性（如集线器的主板和端口属性）
  - 模型的某特定属性的实例 ID 可能与同一模型内其他属性的实例 ID 有所不同。
- show attributes 命令可为由 landscape_handle 指定的格局中的 model_type_handle 显示所有属性（按 ID、名称、类型和标志）。如果未指定 landscape_handle，此命令会显示当前格局中定义的所有模型类型。“标志”字段会列出当前设置的每个属性标志的缩写（用逗号分隔）。如果未设置某标志，列表中就不会显示其缩写。

  以下列表显示了属性标志及其缩写：
  - External（外部）= E
  - Readable（可读）= R
  - Writable（可写）= W
  - Shared（已共享）= S
  - List（列表）= T
  - Guaranteed（已保证）= G
  - Global（全局）= O
  - Memory（内存）= M
  - Database（数据库）= D
  - Polled（已轮询）= P
  - Logged（已记录）= L
  - Preserve Value（保留值）= V

注意：有关属性标志更详细的说明，请参阅*模型类型编辑器用户指南*。

show attributes 命令采用以下格式显示信息：

```
ID      名称      Iid      值
id      名称      iid      值
```

show attributes mth 命令采用以下格式显示信息：

```
ID      名称      类型      标志
id      名称      类型      标志
```

示例：show attributes

```
$ show attributes mh=0xcd00011

Id         名称                        Iid     值
0xd0000    Modeltype_Name                      User
0x10000    Modeltype_Handle                    0x10004
0x10004    Contact_Status                      1
0x10009    Security_String                     ADMIN
0x1000a    Condition                           0

$ show attributes -e mh=0xcd00011

Id         名称                        Iid     值
0xd0000    Modeltype_Name                      User
0x10000    Modeltype_Handle                    0x10004
0x10004    Contact_Status                      Established
0x10009    Security_String                     ADMIN
0x1000a    Condition                           Normal

$ show attributes -e attr=0x1000-0x11fff attrname=status mh=0xcd00011

Id         名称                        Iid     值
0x10004    Contact_Status                      Established
0x110ed    Dev_Contact_Status                  2
0x111a56   ContactStatusEventSwitc             FALSE

$ show attributes attr=0x1006e mh=0x400165

Id         名称                        Iid     值
0x1006e    Model_Name                          142.77

$ show attributes attr=0x100d4 mh=0x400165

Id         名称                        Iid     值
0x100d4    If_Out_Ucast_Pkts             1     1169585
0x100d4    If_Out_Ucast_Pkts             2     1227557
0x100d4    If_Out_Ucast_Pkts             3     1227557
0x100d4    If_Out_Ucast_Pkts             4     8624873

$ show attributes attr=0x100d4,next mh=0x400165

Id         名称                        Iid     值
0x100d4    If_out_Ucast_Pkts             1     1169589

$ show attributes attr=0x100d4,iid=2 mh=0x400165

Id         名称                        Iid     值
0x100d4    If_Out_Ucast_Pkts             2     1227569

$ show attributes attr=0x100d4,iid=2,next mh=0x400165

Id         名称                        Iid     值
0x100d4    If_out_Ucast_Pkts             3     1227573
```

```
$ show attributes mth=0x10004 lh=0xd00000

Id          名称                          类型              标志
0xd0000     namingTree                    Group ID           S,D
0x10000     Modeltype_Name          Text String       R,S,M,K
0xd0200     upsBatteryCapacityInteger                      E,R

$ show attributes mth=0x3d0002 attrname=port

Id          名称                          类型              标志
0x10023     Agent_Port                    Integer            R,W,M,D
0x112e3     IF_Port_Types                 Octet String       R,W,S,D
0x11554     Create_IF_Port                Boolean            R,S,D
0x11d28     PortLinkDownEventCode         Counter            R,S,D
0x11d29     PortLinkUpEventCode           Counter            R,S,D
0x11d3d     support_ICMP                  Boolean            R,W,D
0x11d41     Poll_Linked_Ports             Boolean            R,W,M,D
0x11e24     TelnetPortNum                 Integer            R,W,G,D

$ show attributes mth=0x3d0002 attrname=port flags=rwmd

Id          名称                          类型              标志
0x10023     Agent_Port                    Integer            R,W,M,D
0x11d41     Poll_Linked_Ports             Boolean            R,W,M,D

$ show attributes -e attrname=port mh=0xcd00023

ID          名称                          Iid       值
0x10023     Agent_Port                              161
0x112e3     IF_Port_Types                           11.0.22.0
0x11554     Create_IF_Port                          TRUE
0x11d28     PortLinkDownEventCode                   66312
0x11d29     PortLinkUpEventCode                     66313
0x11d3d     support_ICMP                            TRUE
0x11d41     Poll_Linked_Ports                       TRUE
0x11e24     TelnetPortNum                           0
```

show children

show children 命令可显示与由 model_handle 指定的模型相关的子项。如果未指定关系，show children 会显示所有关系中的子项。如果未指定 model_handle，此命令会显示当前模型的子项。

show children 命令采用以下格式显示信息：

```
模型句柄    模型名称    模型类型句柄    模型类型名称    关系
句柄              名称              句柄                    名称                    关
系
```

示例：show children

```
$ show children mh=0x400009

模型句柄      名称      模型类型句柄     模型类型名称          关系
0x40000d    12.84    0x100d6          Bdg_CSI_CNB2    Collects
```

show devices

show devices 命令可显示由 landscape_handle 指定的格局中所有设备模型的列表。

show devices 命令采用以下格式显示信息：

```
模型句柄     模型名称     模型类型句柄     模型类型名称
句柄              名称              句柄                        名称
```

示例：show devices

```
$ show devices lh=0x400000

模型句柄      模型名称             模型类型句柄      模型类型名称
0x1005c0    10.253.32.101   0x3d002              GnSNMPDev
0x100030    10.253.2.10       0x2c60021           RstonesSwRtr
```

show enumerations

show enumerations 命令可显示与指定的枚举值构成映射的枚举字符串值。

show enumerations 命令采用以下格式显示信息：

```
ID    字符串    值
ID    字符串    值
```

show enumerations mth 命令采用以下格式显示信息：

```
模型句柄     字符串    值
HandLE      字符串    值
```

示例：show enumerations

```
$ show enumerations attr=0x10004

ID              字符串          值
0x10004    已丢失          0
0x10004    已建立          1
0x10004    初始            2

$ show enumerations mth=0x10004

ID              字符串          值
0x10004    已丢失          0
0x10004    已建立          1
0x10004    初始            2
```

show events

show events 命令可显示由 model_handle 指定的模型的相关事件，或者由 landscape_handle 指定的格局中的所有模型的相关事件。默认情况下，show events 命令显示由 model_handle 或 landscape_handle 指定的模型相关的 2000 个最新事件。如果指定了 -a 选项，此命令最多可显示由 model_handle 或 landscape_handle 指定的模型相关的 10,000 个事件。

如果使用 explicit no_events 语句指定了 -n 选项，则可以显示由 model_handle 或 landscape_handle 指定的模型相关的指定数量的事件。如果未指定 model_handle 和 landscape_handle，此命令会显示当前格局中所有模型的相关事件。如果指定了 -x 选项，CLI 会显示解释事件类型的文本消息。事件时间采用 hh:mm:ss 格式显示。

show events 命令采用以下格式显示信息：

```
日期                时间              类型      模型句柄      模型名称      模型类型名称
mm/dd/yyyy    hh:mm:ss    类型    句柄          名称              名称
```

如果随 show events 指定了 -x 选项，显示的事件没有固定的格式。下面是一个典型输出示例：

```
Thur 11 May, 2000 - 8:04:01 - 为类型为 LAN_802_3 的 AntLAN 设备生成的警报数为 10。
当前条件为初始 (默认)。
(事件 [00010701])
```

示例：show events

```
$ show events lh=0x400000

日期          时间        类型          模型句柄     模型名称     模型类型名称
04/25/1999   13:27:38   0x10302   0x4000f9    1.3              Host_IBM
04/25/1999   13:27:38   0x10202   0x400131   qa1sgi        Host_SGI
```

```
$ show events -n

日期          时间                  类型                          模型句柄              模
型名称              模型类型名称
08/21/1999    11:30:02              0x100090xcd00067    els100-01.india   RMONApp
08/21/1999    11:25:33              0x100090xcd00067    els100-01.india   RMONApp
08/21/1999    11:20:17              0x100090xcd00067    els100-01.india   RMONApp
08/21/1999    11:15:52              0x100090xcd00067    els100-01.india   RMONApp
08/21/1999    11:10:27              0x100090xcd00067    els100-01.india   RMONApp
```

show inheritance

show inheritance 命令可显示由 landscape_handle 指定的格局中由 model_type_handle 指定的模型类型的继承信息。如果未指定 landscape_handle，则使用当前格局。此字段的可能值是“基础”或“派生”。

show inheritance 命令采用以下格式显示信息：

```
模型句柄    模型名称    标志    继承
句柄              名称            标志    继承
```

示例：show inheritance

```
$ show inheritance mth=0x1037b lh=0x400000

句柄          名称                  标志      继承
0x10000   Root                  V,D       基础
0x103ad   BanVinesFS     V,I,U     派生
```

show landscapes

show landscapes 命令可显示为每个 SpectroSERVER 定义的所有格局。所显示的格局映射是初始 SpectroSERVER 的映射。

show landscapes 采用以下格式显示信息：

```
SS 名称     优先级     端口     服务     左侧句柄
ss 名称      优先级     端口     服务     句柄
```

示例：show landscapes

```
$ show landscapes

SS 名称     优先级     端口          服务          左侧句柄
devsgi       10            0xbeef    0x10101    0x28000000
devibm     10            0xbeef    0x10101    0x11f00000
```

show models

show models 命令可显示由 landscape_handle 指定的格局中定义的所有模型。如果未指定 landscape_handle，此命令会显示当前格局中定义的所有模型。可使用以下命令指定模型句柄的范围：

```
mhr=low_model_handle-high_model_handle
```

可通过指定 mname=model_name 来搜索特定模型。

show models 命令可将用户模型识别为“(活动)”或“(非活动)”。如果用户模型状态是“(非活动)”，用户仍无法连接到服务器。如果用户模型状态是“(活动)”，则可连接到服务器。

show models 命令采用以下格式显示信息：

模型句柄	模型名称	模型类型句柄	模型类型名称
句柄	名称	句柄	名称

示例：show models

```
$ show models lh=0x400000

模型句柄     模型名称     模型类型句柄     模型类型名称
0x400004   World          0x10040          World
0x4000d9                  0x10020          AUI

$ show models mname=

模型句柄     模型名称        模型类型句柄        模型类型名称
0xcd00016   0x1120002    AppDataServer
0xcd00022   0x1006b        SnmpPif
0xcd00030   0x1028f         IcmpPif

$ show models mhr=0xcd00000-0xcd000ff mth=0x230018 mname=india lh=0xcd00000

模型句柄     模型名称                                  模型类型句柄     模型类型名称
0xcd000a3   hplaser.zeitnet.India.com    0x230018          TCP2_App
0xcd0002b   desire.zeitnet.India.com      0x230018          TCP2_App
```

show parents

show parents 命令可显示与由 model_handle 指定的模型相关的父项。如果未指定关系，此命令会显示所有关系的父项。如果未指定 model_handle，show parents 会显示当前模型的父项。

show parents 命令采用以下格式显示信息：

模型句柄	模型名称	模型类型句柄	模型类型名称	关系
句柄	名称	句柄	名称	关系

示例：show parents

```
$ show parents mh=0x40000d

模型句柄      模型名称                  模型类型句柄      模型类型名称 关系
0x400009    auto-lan-30x1003c              LAN_802_3     Collects
```

show relations

show relations 命令可显示由 landscape_handle 指定的格局中当前已定义的所有关系。如果未指定 landscape_handle，此命令会显示当前格局中定义的所有关系。

show relations 命令采用以下格式显示信息：

```
名称                    类型
关系名称        关系类型
```

示例：show relations

```
$ show relations

名称类型
Passes_Through MANY_TO_MANY
Lost_and_Found ONE_TO_MANY
Owns ONE_TO_MANY
Contains ONE_TO_MANY
```

show rules

show rules 命令可显示关系的规则。关系在由 landscape_handle 指定的格局中指定。如果未指定 landscape_handle，则使用当前格局。

show rules 命令采用以下格式显示信息：

```
左侧模型类型句柄      左侧模型类型名称      右侧模型类型句柄      右侧模型类型名称
句柄          名称          句柄          名称
```

示例：show rules

```
$ show rules rel=Owns lh=0x400000

左侧模型类型句柄      左侧模型类型名称      右侧模型类型句柄      右侧模型类型名称
0x102da       Org_Owns    0x10043       Site
0x102da       Org_Owns    0x210023      Rtr_CiscoMGSshow
```

show types

show types 命令可显示由 landscape_handle 指定的格局中当前已定义的所有模型类型。如果未指定 landscape_handle，此命令会显示当前格局中定义的所有模型类型。“标志”字段会列出当前设置的每个属性标志（共有 6 个）的缩写。如果未设置某标志，则不会在列表中显示其缩写。

以下列表显示了模型类型标志及其缩写：

- Visible（可视）= V
- Instantiable（可实例化）= I
- Derivable（可派生）= D
- No Destroy（不销毁）= N
- Unique（唯一）= U
- Required（必需）= R

show types 命令 [mth=low_mth-high_mth]] 可显示介于 low_mth 和 high_mth 之间的所有模型类型。

注意：有关模型类型标志的更多信息，请参见*模型类型编辑器用户指南*.

show types 命令采用以下格式显示信息：

```
句柄     名称     标志
句柄     名称     标志
```

示例：show types

```
$ show types lh=0x400000

句柄     名称           标志
0x10000   Root          V,D
0x10080   Gen_Rptr_Prt  V,D

$ show types mthr=0x10002-0x10008

句柄     名称             标志
0x10002   Network_Entity
0x10003   VNM              V,I,D,N,U,R
0x10004   User             V,I,D
0x10005   VIB
0x10007   DataRelay        V,D
```

```
$ show types mthr=0x210020-0x21002f mtname=Rtr_Cisco lh=0xcd00000

句柄       名称                标志
0x210020    Rtr_CiscoAGS       V,I,D
0x210021    Rtr_CiscoCGS       V,I,D
0x210022    Rtr_CiscoIGS       V,I,D
0x210023    Rtr_CiscoMGS       V,I,D
0x210024    Rtr_CiscoMIM       V,I,D
0x21002b    Rtr_Cisco2500      V,I,D
0x21002c    Rtr_CiscoMIM3T     V,I,D
0x21002d    Rtr_Cisco3000      V,I,D
0x21002e    Rtr_Cisco4000      V,I,D
0x21002f    Rtr_Cisco7000      V,I,D

$ show types flags=VIDNUR lh=0xcd00000

句柄         名称                标志
0x25e0000     MgmtInventory      V,I,D,N,U,R
0x10040       World              V,I,D,N,U,R
0x102cf       Top_Org            V,I,D,N,U,R
0x10003       VNM                V,I,D,N,U,R
0x102be       LostFound          V,I,D,N,U,R
0x25e0001     TopologyWrkSpc     V,I,D,N,U,R
0x10091       Universe           V,I,D,N,U,R
```

show watch

show watch 命令可列出由 model_handle 指定的模型的可用监视数据。

如果未指定 landscape_handle 和 model_handle，show 命令会使用以下默认值：

命令	默认
show alarms	当前格局
show associations	当前模型
show attributes	当前模型
show attributes mth	当前格局
show children	当前模型
show devices	当前格局
show enumerations	当前格局
show enumerations mth	当前格局
show events	当前格局

命令	默认
show inheritance	当前格局
show models	当前格局
show parents	当前模型
show relations	当前格局
show rules	当前格局
show types	当前格局
show watch	当前模型

注意：可以在“show alarms”和“show events”命令中指定 x 选项，以便显示警报的可能原因消息和扩展的事件消息。

请确认满足下列先决条件：

- 在本地服务器上安装了 OneClick。
- 环境变量 SPECROOT 已设置为根目录 (SG-Support) 的路径。

  例如，如果 SG-Support 文件位于 /usr/spectrum/SG-Support 中，则将 SPECROOT 设置为 /usr/spectrum。

show watch 命令采用以下格式显示信息：

```
监视 ID    监视名称    监视类型    监视状态
监视 ID    监视名称    监视类型    监视状态
```

示例：show watch

```
$ show watch mh=0xc600015

监视 ID      监视名称    监视类型    监视状态
0xffff0001   watch798     Calc          活动
```

stopShd – 终止 CLI 本地服务器

使用 stopShd 命令可以终止 CLI 本地服务器（VnmShd 后台进程）。stopShd 命令会断开所有 CA Spectrum CLI 用户与当前连接的 SpectroSERVER 的连接，并终止 CLI 本地服务器。此命令在断开用户连接并关闭服务器之前会提示用户确认。（您也可以使用 kill -2 命令关闭后台进程。）

此命令采用以下格式：

```
stopShd [-n]
```

-n

指定“无提示”。在 stopShd 命令中指定此选项可以禁用确认提示。

否则，始终会显示以下消息：

```
stopShd: n 个用户已连接，确定要执行此操作吗?
```

“n”表示连接的用户数，包括您自己在内。

有效的回复有 y、yes、Y、Yes、n、no、N 和 No。

如果命令执行成功，则会显示以下消息：

```
stopShd: 成功
```

当执行 stopShd 来终止 CLI 本地服务器时，系统控制台上会显示以下消息：

```
VnmShd: 已执行 stopShd。正在退出...
```

示例：stopShd

```
$ stopShd
stopShd: 2 个用户已连接，确定要执行此操作吗？y
stopShd: 成功
```

详细信息：

disconnect – 断开与 SpectroSERVER 的连接 (p. 35)

update – 更新模型和模型属性

使用 update 命令来更新模型和模型类型属性。

此命令采用以下格式：

```
update  [mh=modelhandle]
attr=attribute_id[,iid=instance_id],val=value
                [attr=attribute_id[,iid=instance_id],val=value...] |
                [mh=modelhandle]
                attr=attribute_id,iid=instance_id,remove
                [attr=attribute_id,iid=instance_id,remove...] |
                [-n] mth=model_type_handle |
attr=attribute_id,val=value [attr=attribute_id,val=value ... ]
                [lh=landscape_handle] |
alarm [-r] aid=alarm_id <assign=troubleshooter |
                status=status_text | ticket=troubleticketID |
                ack=(true|false)> [lh=landscape_handle] |
action=action_code [watch=watch_id] [mh=modelhandle]
```

-n

如果随 update 命令指定了 -n（无提示）选项，系统不会提示您确认。此选项在 CLI 脚本中非常有用。

-r

如果 update alarm 命令中使用了状态或票单参数，那么可以指定 -r（替换状态文本/替换故障单 ID）选项。如果使用了 -r 选项，现有的警报状态文本或警报故障单 ID 将替换为由状态参数或票单参数指定的文本。如果未使用 -r 选项，则会将新值附加到现有值。

action_code

- reconfig、0x1000e 或 65550 - 重新配置模型
- activate、0x00480003 或 4718595 - 激活监视
- deactivate、0x00480004 或 4718596 - 停用监视
- reconfigure_apps、0x210008 或 2162696 用于在 Cisco 和 Wellfleet 设备上重新配置应用程序模型
- reload_event_disp、0x000100a2 或 65698 用于更新 SpectroSERVER，以便包含对 EventDisp 和 AlertMap 文件所做的更改

注意：watch = *<watch_id>* 参数仅适用于以下操作：activate（或等效的十六进制 0x00480003）和 deactivate（或等效的十六进制 0x00480004）。

以下几点介绍了 update 命令的功能：

- update 命令可为由 model_handle 指定的模型或由 landscape_handle 指定的格局中模型类型为 model_type_handle 的所有模型更新由 attribute_id 值指定的属性。
- 通过指定多个用空格分隔的“attribute_id, value”对，即可使用一个 update 命令更新多个属性。
- remove 选项可从列表属性中删除指定的实例。
- 如果更新模型类型属性时未指定 landscape_handle，则使用当前格局。如果未指定 model_handle，则更新当前模型的指定属性。
- 请记住，更新模型类型属性时只会更新共享属性。共享属性是指设置了共享标志的属性。可以使用 show attributes 命令查看属性是否已共享。
- 诸如 User_Community_String 和 Model_Security_String 之类的安全敏感型属性可以通过 CLI 更新。不过，当前用户模型无法更新自己的 User_Security_String 或 Security_String，但可以更新其他模型的这些属性。
- update 命令还允许用户指定实例 ID 以便更改单个属性值。更新属性值列表时，可以为列表中的每个属性指定实例 ID。instance_id 是相应属性的实例 ID。instance_id 必须是正整数，或用句点分隔的正整数序列。
- 如果未指定实例 ID，update 命令会使用它为属性找到的第一个有效实例。如果找不到有效的实例，则会显示错误消息：

  `update: 找不到列表属性 <attr_id> 的有效实例`

- update alarm 命令可更新由 alarm_id 指定的警报，使其包含由故障排除者（故障排除者模型句柄或故障排除者名称）、状态文本、故障单 ID 或确认参数指定的值。要清除警报的故障排除者、状态文本或故障单 ID 参数的现有值，可以将相应的参数值留空（status=、ticket= 或 assign=）。landscape_handle 参数指定要从中查找警报的格局。
- update action 命令可在由 model_handle 指定的设备上执行由 action_code 指定的操作。使用 action_code reconfig，可以重新配置模型类型为 GnSNMPDev 或从 GnSNMPDev 继承的任何模型类型的任何设备。action_code 为 activate 或 deactivate 时，会更新由 model_handle 指定的设备上的监视状态。发送 activate 操作对象时，监视状态在从“初始”变为“活动”的这段时间内可能会出现短暂的延迟，具体取决于选定模型的内置智能。可以使用 show watch 命令获取指定为要更新其状态的监视的 watch_id。reconfigure_apps action_code 可为 Cisco 和 Wellfleet 设备模型更新应用程序模型类型。reload_event_disp action_code 可以更新 SpectroSERVER，以便包含对 EventDisp 或 AlertMap 文件所做的更改。

- 使用 update action 命令时务必小心谨慎。与任何 CLI 命令一样，如果您对此命令使用不当，可能会损坏 SpectroSERVER 数据库。例如，随意重新配置关键路由器可能会导致网络出现不可预知的后果。
- 如果在输入 update 命令中指定了有效的 model_handle 或 model_type_handle、有效的 attribute_id 及其值，则会采用以下格式显示修改过的属性及其值：

```
ID  名称    值
ID  名称    值
```

- 如果您在更新指定模型类型的模型时没有使用 -n 选项，则会显示以下确认消息：

```
update: 将更新此类型的所有模型，确定要执行此操作吗？
有效的回复有 y、yes、Y、Yes、n、no、N 和 No。
```

- 如果 update alarm 命令执行成功，则会显示以下消息：

```
update: 成功
```

- 如果 update action 命令执行成功，则会显示以下消息：

```
update action: 成功
```

示例：Update

- 在以下示例中，指定了 instance_id 的 update 命令用于禁用由模型句柄 0x4001f6 表示的集线器的主板 5 上的端口 7：

```
$ update mh=0x4001f6 attr=0x10ee0,iid=5.7,val=1
ID        名称              Iid       值
0x10ee0   CsPortAdminState            1
```

- 在以下示例中，随 update 命令指定了 remove 选项，用于从特定模型 (mh) 的 deviceIPAddressList (attr) 中删除 IP 地址 (iid)。

```
$ update mh=0xc600018 attr=0x12a53,iid=10.253.8.65,remove
update: 成功
```

- 在以下示例中，update 命令用于更新由模型类型句柄 0x10059 表示的模型类型的所有模型上名为 AutoPlaceStartX 的属性。

```
$ update mth=0x10059 attr=0x118f2,val=100 lh=0x400000
update: 将更新此类型的所有模型，确定要执行此操作吗？y
ID    名称              值
ID    AutoPlaceStartX   100
```

- 在以下示例中，update alarm 命令用于更新警报的故障排除者分配。

```
$ update alarm aid=928 assign=0xa600722
update: 成功
```

- 在以下示例中，update alarm 命令用于更新警报状态。-r 选项用于覆盖现有的状态。

```
$ update alarm -r aid=928 status='正在处理'
update: 成功
```

- 在以下示例中，update alarm 命令用于更新警报故障单 ID。-r 选项用于覆盖故障单 ID 的现有值。

```
$ update alarm -r aid=928 ticket='Ax1009'
update: 成功
```

- 在以下示例中，update alarm 命令用于清除故障单 ID 的现有值。

```
$ update alarm aid=928 ticket=
update: 成功
```

- 在以下示例中，update alarm 命令用于确认警报。

```
$ update alarm aid=928 ack=TRUE
update: 成功
```

- 在以下示例中，update 命令用于限制 User_Community_String 的更新。

```
$ update mh=0x9a000ff attr=0x1007a,val=AA,11
update: 成功
```

- 在以下示例中，update action 命令用于重新配置 Cisco 路由器。

```
$ update action=reconfig mh=0xc600030
update action: 成功

$ update action=activate watch=0xffff0001 mh=0xc600015

监视 ID        模型句柄        监视状态
0xffff0001     0xc600015     初始

$ update action=0x480004 watch=0xffff0001 mh=0xc600015

监视 ID        模型句柄        监视状态
0xffff0001     0xc600015     非活动
```

附录 A： 示例脚本

示例脚本概述

CLI 中包含的示例脚本演示了如何将 CLI 命令集成到 UNIX shell 脚本中，以便自动化处理 CLI 会话。您会发现其中的某些脚本或脚本中的某些功能对您的工作大有益处。

CLI 的 *<$SPECROOT>*/vnmsh/sample_scripts 目录中包含以下脚本以及用于说明脚本的自述文件：

- active_ports
- app_if_security
- cli_script
- database_tally
- update_mtype
- octet_to_ascii.pl

使用 CLI 脚本之前，请先回顾一下先决条件：

- 每个脚本都包含名为 CLIPATH 的内部变量。为了使用脚本，需将 CLIPATH 变量设置为 CLI 可执行文件所在目录的路径名。
- 为 CLIPATH 变量和其他环境变量设置的路径名可以是*完整*路径名，也可以是*相对*路径名，具体取决于脚本的运行方式。若要将示例脚本作为 cron 脚本运行，请为 CLIPATH 和其他环境变量设置*完整*路径名。否则，可以为这些变量设置*相对*路径名。
- 您可以将除 update_mtype 之外的所有 CLI 脚本作为 cron 脚本运行。

  注意：不要将 update_mtype 作为 cron 脚本运行，因为它会提示用户输入。
- 运行 CLI 脚本时，请为 .vnmshrc 文件中 vnm_hostname 变量指定正确的名称。

active_ports 脚本

使用 active_ports 脚本可以标识 IRM2 集线器上每个主板的所有端口以及每个主板上的活动端口。

active_ports 脚本会针对名为 hub_name 的集线器提供报告并存放在名为 output_file 的文件中。此报告会列出每个主板上的所有端口。此报告的 ON 列上显示星号 (*) 表示端口处于活动状态。

此脚本采用以下格式：

```
active_ports <hub_name> <output_file>]
```

app_if_security 脚本

使用 app_if_security 脚本可以更新 CA Spectrum 数据库中所有接口和应用程序模型的 Security_String 属性值。app_if_security 脚本通过从父模型复制属性值来执行更新。如果接收方模型（子项）已经包含 Security_String 属性的值，或者父模型*不包含* Security_String 属性的值，那么该脚本不会更新任何模型。在更新模型安全字符串之后，管理员可以使用此脚本来更新模型子项的安全字符串。

此脚本采用以下格式：

```
app_if_security
```

cli_script 脚本

在 cli_script 脚本中使用数据文件作为输入时，可以采用批处理模式执行大部分 CLI 命令。名为 datafile 的 CLI 示例数据文件包含指示要执行的命令的各种开关以及要传递给命令的必要参数。脚本会确认每个命令是否成功执行，并维护一个运行时日志。

此脚本的优势之一就是可以使用名称而不是句柄来创建批处理文件。例如，您可以使用模型类型名称，而不使用十六进制的模型类型句柄。这使得文件更易于创建和读取，而且当您希望针对创建的模型执行一些后续操作时，就更加体现出了这样做的强大优势。现在不必为模型分配十六进制的模型句柄，而是可以按名称引用模型。

此脚本采用以下格式：

```
cli_script datafile
```

cli_script 使用两个文件，即 datafile 和 clean.awk，这些文件也位于 sample_scripts 目录中。

datafile

包含 cli_script 的输入。它包含当前在 cli_script 中实现的每个 CLI 命令。有关此文件的格式和语法说明，请参阅 cli_script 标头信息。

clean.awk

包含在执行时使用的输入。.awk 文件用于设置在控制台上显示的数据格式。

使用 cli_script 时，应考虑以下几点：

- 记得将示例 datafile 中的“虚拟”网络地址 (255.255.255.255) 更改为实际地址。
- 如果将 cli_script 移至其他目录，则必须将环境变量 SPECROOT 更新为支持根目录 (SG-SUPPORT)。

  例如，如果 SG-SUPPORT 文件位于 /usr/spectrum/SG-Support 中，则将 SPECROOT 设置为 /usr/spectrum。

database_tally 脚本

使用此脚本可以确定数据库中的每种类型目前包含多少个模型。管理员可能会在评估系统性能时发现此脚本非常有用。此脚本可显示所有模型类型的列表以及数据库中每种类型的模型数目。

此脚本采用以下格式：

```
database_tally <vnm-name>
```

update_mtype 脚本

使用此脚本可以更新某种模型类型的所有模型的特定属性。如果属性是模型类型的共享属性，那么脚本不会更新该模型的属性。此脚本的优势之一是，您可以在提示符中使用模型和属性名称，而不必使用它们的十六进制 ID 句柄。

注意：将 CLIMNAMEWIDTH 环境变量设置为 256。设置较大的值可防止模型名称被截断，如果被截断，运行脚本时会导致匹配失败。

此脚本采用以下格式：

update_mtype <*model_name*> \ <*model_typename*>[<*attribute_name*> | <*attribute_id*> <*value*>]

model_name | model_type_name

为完成属性更新的模型类型的模型指定模型名称或模型名称的一部分。您可以在命令中指定该模型类型的任何模型。

脚本随后会显示模型名称中包含您输入的模型名称参数的所有模型类型的列表。脚本会提示您从列表中选择模型类型。

如果您使用模型名称，而不是模型类型名称，脚本会更新名称中包含在命令行上或提示符中输入的字符串的所有模型。如上所述，在这种情况下，给定模型类型的所有模型都不会更新。

注意：建议在提示符中使用模型名称或模型类型名称，而不要使用句柄。

attribute_name | attribute_id

如果开始时没有指定这些参数，此脚本在运行时会提示输入属性名称或属性 ID。此时，您必须指定属性名称或属性名称的一部分。脚本随后会提示您从包含您输入的文本的属性列表中选择。因此，您可以在事先不知道十六进制的模型类型句柄或属性句柄的情况下运行整个脚本。

详细信息：

CLI 环境变量 (p. 9)

active_ports 脚本

使用此脚本可以将 Octet_String 格式 XX.YY.ZZ 转换为其 ASCII 字符串表示形式。

附录 B： 错误消息

ack alarm: <alarm_id>: 无效的警报 ID

原因：

您输入的警报 ID 无效。

操作：

重新输入带有有效 alarm_id 的 ack alarm 命令。

ack alarm: <landscape_handle>: 无效的格局句柄

原因：

为警报输入的格局句柄无效。

操作：

重新输入带有有效 landscape_handle 的 ack alarm 命令。

command: 未能与 VnmShd 建立连接，请先运行 connect

原因：

您在运行 connect 命令之前尝试运行了其他命令。

操作：

使用 connect 命令启动 CLI 会话。

connect: 自 <date/time> 以来已连接到 <hostname>

原因：

无需尝试连接， 您已连接到 SpectroSERVER 主机。

操作：

无。

connect: 无法打开资源文件 <pathname>/.vnmshrc

原因：

connect 命令找不到 CLI 资源文件 .vnmshrc。

操作：

.vnmshrc 资源文件必须位于 connect 命令所在的同一目录中。

详细信息：

启动文件 (p. 10)

connect: 只能连接到 <hostname> 格局映射中的 SpectroSERVER - 其他用户已连接

原因：

已使用 connect 命令连接到特定 SpectroSERVER。您只能连接到原始 SpectroSERVER 的格局映射中的 SpectroSERVER。

操作：

无

connect: 错误: 没有名为 <username> 的 CA Spectrum 用户

原因：

第一个使用 connect 命令的用户未定义为 CA Spectrum 用户。

操作：

以 CA Spectrum 用户身份重新连接到 SpectroSERVER。

connect: <hostname> 未响应或不允许访问

原因：

connect 命令无法连接到 SpectroSERVER，因为主机名不正确、SpectroSERVER 未运行或者该用户没有用户模型。

操作：

确认主机名正确、SpectroSERVER 正在运行并且该用户具有用户模型。

connect: <landscape_handle>: 无效的格局句柄

原因：

用户指定的 landscape_handle 对指定主机名无效，或者您的 VNM 无法访问该句柄。

操作：

确认 landscape_handle 对指定主机名有效，并且您的 VNM 可以访问该句柄。

connect: 不兼容的 SpectroSERVER <version>

原因：

用户正在尝试连接到与 CLI 版本不兼容的 SpectroSERVER 主机。

操作：

更新您使用的 CLI 版本。

connect: CLISESSID 的 <value> 无效

原因：

在 cron 脚本内使用 connect 命令，或者窗口化系统为 ttyslot 返回 0，而环境变量 CLISESSID 被设置为非数值。

操作：

在 cron 脚本外使用 connect 命令，并将 CLISESSID 设置为数值。

connect: 变量 <CLISESSID> 未设置

原因：

您试图在未事先设置 CLISESSID 环境变量的情况下在 cron 脚本内使用 connect 命令。

操作：

在 cron 脚本内使用 connect 命令时，应设置环境变量 CLISESSID。

create: 用户没有创建警报的权限

原因：

您没有创建警报的权限。

操作：

确认您的用户权限。

create: 用户没有创建关联的权限

原因：

您没有创建关联的权限。

操作：

确认您的用户权限。

create: 用户没有创建事件的权限

原因：

您没有创建事件的权限。

操作：

确认您的用户权限。

create: 用户没有创建模型的权限

原因：

您没有创建模型的权限。

操作：

确认您的用户权限。

create alarm: <probable_cause_id>: 无效的警报原因

原因：

输入的 create alarm 命令中指定的 probable_cause_id 无效。

操作：

重新输入带有有效 probable_cause_id 的 create alarm 命令。

create alarm: <alarm_severity>: 无效的警报重要级别

原因：

输入的 create alarm 命令中指定的 alarm_severity 无效。

操作：

重新输入带有有效 alarm_severity 的 create alarm 命令。

create alarm: <model_handle>: 无效的模型句柄

原因：

输入的 create alarm 命令中指定的 model_handle 无效。

操作：

重新输入带有有效 model_handle 的 create alarm 命令。

create association: <left_model_handle>: 无效的模型句柄

原因：

输入的 create association 命令中指定的 left_model_handle 无效。

操作：

重新输入带有有效 left_model_handle 的 create association 命令。

create association: 模型属于不同格局

原因：

输入的 create association 命令中指定的 left_model_handle 和 right_model_handle 属于不同格局。

操作：

使用属于同一格局的两个句柄。

create association: rel=<relation>: 无效的关系

原因：

输入的 create association 命令中指定的关系无效。

操作：

重新输入带有有效关系的 create association 命令。

create association: <right_model_handle>: 无效的模型句柄

原因：

输入的 create association 命令中指定的 right_model_handle 无效。

操作：

重新输入带有有效 right_model_handle 的 create association 命令。

create event: <event_type>: 无效的事件类型

原因：

输入的 create event 命令中指定的 event_type 无效。

操作：

重新输入带有有效 event_type 的 create event 命令。

create event: <landscape_handle>: 未知格局

原因：

输入的 create event 命令中指定的 landscape_handle 无效。

操作：

重新输入带有有效 landscape_handle 的 create event 命令。

create event: <model_handle>: 无效的模型句柄

原因：

输入的 create event 命令中指定的 model_handle 无效。

操作：

重新输入带有有效 model_handle 的 create event 命令。

create model: <attribute_id>: 无效的属性 ID

原因：

未创建模型，因为输入的 create model 命令中指定的 attribute_id 无效。

操作：

重新输入带有有效 attribute_id 的 create model 命令。

create model: DCM 设备不可访问

原因：

未创建模型，因为输入的 create model 命令中指定的 ip_address 无效。DCM（设备通信管理器）发出此错误消息。

操作：

重新输入带有有效 ip_address 的 create model 命令。

create model: 超出了设备限制

原因：

未创建模型，因为分支管理器 SpectroSERVER（上限为 50 个设备模型）或站点管理器 SpectroSERVER（上限为 250 个设备模型）包含的设备模型数已达到所能包含的最大数目。

操作：

确认 SpectroSERVER 上设备模型的数目未达到指定的限制。如果已达到限制，您可能需要删除一些设备模型，然后重新输入 create model 命令。

create model: <landscape_handle>: 无效的格局句柄

原因：

未创建模型，因为输入的 create model 命令中指定的 landscape_handle 无效。

操作：

重新输入带有有效 landscape_handle 的 create model 命令。

create model: <model_type_handle>: 无效的模型类型句柄

原因：

未创建模型，因为输入的 create model 命令中指定的 model_type_handle 无效。

操作：

重新输入带有有效 model_type_handle 的 create model 命令。

create model: <value>: 无效值

原因：

未创建模型，因为输入的 create model 命令中指定的值无效。

操作：

重新输入带有有效值的 create model 命令。

create model: <value>: 无团体名称

原因：

create 请求中提供的团体名称不正确。

操作：

重新输入带有有效团体名称的 create 命令。

current: <model_handle>: 无效的模型句柄，当前模型是 <current_model_handle>

原因：

指定了无效的 model_handle，因此当前模型和当前格局未更改。

操作：
若要修改当前模型和当前格局，请重新输入有效的 model_handle。

current: <landscape_handle>: 无效的格局句柄，当前格局是 <current_landscape_handle>

原因：

指定了无效的 landscape_handle，因此当前模型和当前格局未更改。

操作：

若要修改当前模型和当前格局，请重新输入有效的 landscape_handle。

current: <landscape_handle>: 无响应或没有访问权限，当前模型是 <current_model_handle>

原因：

指定了 landscape_handle，但格局的 OneClick 已关闭，或者用户在该格局上没有用户模型。

操作：

确认提到的格局的 OneClick 正在运行；确认您在提到的格局上有用户模型；然后重新输入 landscape_handle。

current: <landscape_handle>: 无响应或没有访问权限，当前格局是 <current_landscape_handle>

原因：

指定了 model_handle，但包含此模型的格局的 SpectroSERVER 出现故障，或者用户在该格局上没有用户模型。

操作：

确认提到的格局的 SpectroSERVER 正在运行；确认您在提到的格局上有用户模型；然后重新输入 model_handle。

destroy: 用户没有销毁警报的权限

原因：

您没有销毁警报的权限。

操作：

确认您的用户权限。

destroy: 用户没有销毁关联的权限

原因：

您没有销毁关联的权限。

操作：

确认您的用户权限。

destroy: 用户没有销毁模型的权限

原因：

您没有销毁模型的权限。

操作：

确认您的用户权限。

destroy alarm: aid=<alarm_id>: 无效的警报 ID

原因：

输入的 destroy alarm 命令中指定的 alarm_id 无效。

操作：

重新输入带有有效 alarm_id 的 destroy alarm 命令。

destroy alarm: <landscape_handle>: 无效的格局句柄

原因：

输入的 destroy alarm 命令中指定的 landscape_handle 无效。

操作：

重新输入带有有效 landscape_handle 的 destroy alarm 命令。

destroy association: rel=<relation>: 无效的关系

原因：

输入的 destroy association 命令中指定的关系无效。

操作：

重新输入带有有效关系的 destroy association 命令。

destroy association: <left_model_handle>: 无效的模型句柄

原因：

输入的 destroy association 命令中指定的 left_model_handle 无效。

操作：

重新输入带有有效 left_model_handle 的 destroy association 命令。

destroy association: <right_model_handle>: 无效的模型句柄

原因：

输入的 destroy association 命令中指定的 right_model_handle 无效。

操作：

重新输入带有有效 right_model_handle 的 destroy association 命令。

destroy association: 给定模型之间不存在关联

原因：

尝试销毁两个模型之间不存在的关联。

操作：

确认两个模型之间存在您尝试销毁的关联。

destroy association: 模型属于不同格局

原因：

输入的 destroy association 命令中指定的 left_model_handle 和 right_model_handle 属于不同格局。

操作：

重新输入使用属于同一格局的 left_model_handle 和 right_model_handle 的 destroy association 命令。

destroy model: <model_handle>: 无效的模型句柄

原因：

输入的 destroy model 命令中指定的 model_handle 无效。

操作：

重新输入带有有效 model_handle 的 destroy model 命令。

disconnect: 失败

原因：

disconnect 命令失败。

操作：

重试 disconnect 命令。

disconnect: 未能与 VnmShd 建立连接，请先运行 connect

原因：

尝试运行 disconnect 时 CLI 本地服务器未运行。

操作：

无。您已断开连接。

disconnect: 未连接

原因：

disconnect 命令失败，因为用户未连接到 SpectroSERVER。

操作：

无。您已断开连接。

jump: <text_string>: 文本字符串未定义

jump:<text_string>: 文本字符串未定义

其中的 text_string1、text_string2... 是当前已定义的文本字符串。

原因：

输入的 jump 命令中指定的 text_string 未定义。

操作：

重新输入使用已定义的任意文本字符串的 jump 命令。

<pathname>/VnmShd: 找不到

<pathname>/VnmShd: 找不到

connect: 失败

其中的 pathname 表示 CLI 尝试从中执行 VnmShd 的目录路径。

原因：

connect 命令找不到 CLI 本地服务器。

操作：

确保 VnmShd 和 connect 位于相同目录中，然后重新输入 connect 命令。

请先连接

原因：

执行 connect 命令后，您运行了 disconnect 或 stopShd，然后又尝试运行其他命令。

操作：

先重新发出 connect 命令。

seek: <attribute_id>: 无效的属性 ID

原因：

输入的 seek 命令中指定的 attribute_id 无效。

操作：

重新输入带有有效 attribute_id 的 seek 命令。

seek: <error>: 属性未使用键控

原因：

输入的 seek 命令中指定的属性的 attribute_id 未使用键控。

操作：

重新输入带有键控属性的 attribute_id 的 seek 命令。

seek: <value>: 无效值

原因：

输入的 seek 命令中指定的值无效。

操作：

重新输入带有有效值的 seek 命令。

show attributes: <attribute_id>: 非列表属性

原因：

输入的 show attributes 命令中指定了非列表 attribute_id 的 instance_id。

操作：

重新输入带有列表 attribute_id 的 instance_id 的 show attributes 命令。

show attributes: <attribute_id>: 无效的属性 ID

原因：

输入的 show attributes 命令中指定的 attribute_id 无效。

操作：

重新输入带有有效 attribute_id 的 show attributes 命令。

show attributes: <instance_id>: 无效的实例 ID

原因：

输入的 show attributes 命令中指定的 instance_id 无效。如果 instance_id 不是由一系列非负整数组成或者不在指定的属性中，则无效。

操作：

重新输入带有有效 instance_id 的 show attributes 命令。

show attributes: <model_type_handle>: 无效的模型类型句柄

原因：

输入的 show attributes 命令中指定的 model_type_handle 无效。

操作：

重新输入带有有效 model_type_handle 的 show attributes 命令。

show: <landscape_handle>: 无效的格局句柄

原因：

输入的 show 命令中使用了可选的 landscape_handle，但指定的 landscape_handle 无效。

操作：

重新输入带有有效 landscape_handle 的 show 命令。

show: <model_handle>: 无效的模型句柄

原因：

输入的 show 命令中使用了可选的 model_handle，但指定的 model_handle 无效。

操作：

重新输入带有有效 model_handle 的 show 命令。

show: 未定义当前模型

原因：

输入的 show associations 命令中使用了可选的 model_handle，但未指定 model_handle，也未定义当前模型。

操作：

重新输入包含 model_handle 和当前模型的 show associations 命令。

show alarms: 未提供原因信息

原因：

在 show alarms 命令中指定了 -x 选项，但没有包含可能原因文本消息的 CA Spectrum 警报文件可用。

操作：

要在 show alarms 命令中使用 -x 选项（显示可能的警报原因消息和扩展的事件消息），必须在本地服务器上安装 OneClick，并将 SPECROOT 环境变量设置为支持根目录 (SG-Support) 的路径。例如，如果 SG-Support 文件位于以下目录中：

/usr/spectrum/SG-Support，则将 SPECROOT 设置为 /usr/spectrum。

show children: <relation>: 无效的关系

原因：

输入的 show children 命令中指定的关系无效。

操作：

重新输入带有有效关系的 show children 命令。

show events: 无事件格式信息可用

原因：

输入的 show events 命令中指定了 -x 选项，但没有包含事件格式文本消息的 CA Spectrum 事件文件可用。

操作：

要在 show events 命令中使用 -x 选项（显示可能的警报原因消息和扩展的事件消息），必须在本地服务器上安装 OneClick，并将 SPECROOT 环境变量设置为支持根目录 (SG-Support) 的路径。例如，如果 SG-Support 文件位于以下目录中：

/usr/spectrum/SG-Support，则将 SPECROOT 设置为 /usr/spectrum。

show parents: <relation>: 无效的关系

原因：

输入的 show parents 命令中指定的关系无效。

操作：

重新输入带有有效关系的 show parents 命令。

show rules: <relation>: 无效的关系

原因：

输入的 show rules 命令中指定的关系无效。

操作：

重新输入带有有效关系的 show rules 命令。

show inheritance: <model_type_handle>: 无效的模型类型句柄

原因：

输入的 show inheritance 命令中指定的 model_type_handle 无效。

操作：

重新输入带有有效 model_type_handle 的 show inheritance 命令。

stopShd: VnmShd 未运行

原因：

尝试运行 stopShd 时 CLI 本地服务器未运行。

操作：

启动 CLI 本地服务器。

stopShd: 失败

原因：

stopShd 命令失败。

操作：

重试连接，然后执行 stopShd。如果不起作用，请手动终止 VnmShd 进程。

update: <attribute_id>: 属性不可写

原因：

未执行更新，因为尝试更新的模型属性不可写。

操作：

确认您要更新的模型属性可写，然后重新输入 update 命令。

update: <attribute_id>: 无效的属性 ID

原因：

未执行更新，因为输入的 update 命令中指定的 attribute_id 无效。

操作：

重新输入带有有效 attribute_id 的 update 命令。

update: <attribute_id>: 非共享属性

原因：

针对模型类型执行 update 命令时，输入了非共享属性的 attribute_id。

操作：

重新输入针对模型类型的 update 命令，这次使用共享属性的 attribute_id。

update: <instance_id>: 无效的实例 ID

原因：

未执行更新，因为输入的 update 命令中指定的 instance_id 无效。

操作：

重新输入带有有效 instance_id 的 update 命令。

update: <landscape_handle>: 无效的格局句柄

原因：

未执行更新，因为输入的 update 命令中指定的 landscape_handle 无效。

操作：

重新输入带有有效 landscape_handle 的 update 命令。

update: <model_handle>: 无效的模型句柄

原因：

未执行更新，因为输入的 update 命令中指定的 model_handle 无效。

操作：

重新输入带有有效 model_handle 的 update 命令。

update: <model_type_handle>: 无效的模型类型句柄

原因：

未执行更新，因为输入的 update 命令中指定的 model_type_handle 无效。

操作：

重新输入带有有效 model_type_handle 的 update 命令。

update: <value>: 无效值

原因：

未执行更新，因为输入的 update 命令中指定的一个或多个值无效。

操作：

重新输入带有有效值的 update 命令。

update: <action_code>: 无效的操作代码

原因：

未执行更新，因为输入的 update 命令中指定的 action_code 无效。

操作：

重新输入带有有效 action_code 的 update 命令。

VnmShd: 错误: 无法连接到 SpectroSERVER

原因：

CLI 本地服务器无法连接到 SpectroSERVER。

操作：

确认 SpectroSERVER 正在运行。

VnmShd: 错误: 失去与 SpectroSERVER 的连接

原因：

如果 CLI 本地服务器检测到所连接的 SpectroSERVER 已经终止，它也会随之终止。

操作：

重新启动 SpectroSERVER，然后运行 CLI 本地服务器。

附录 C： UNIX 与 DOS 的转换

在 UNIX 平台上，通常可在终端窗口中将 CLI 命令与 UNIX 命令配合使用。在 Windows 平台上，可在本机 DOS 窗口中将 CLI 命令与 DOS 命令配合使用。此附录列出了常用的 UNIX 命令以及与之等效的 DOS 命令。

注意：此附录只是为 UNIX 和 DOS 命令提供一个快速参考，不能作为详细的参考清单。有关各种命令及其功能的详细信息，请参阅您的 UNIX、DOS 或 Windows 文档。

UNIX	DOS
#	rem
cat	type
cd	cd
chdir	chdir
clear	cls
cmp、diff	comp、fc
cp	copy
cp -r	xcopy
cpio、dump、tar、ufsdump	cpio、dump、tar、ufsdump
cpio、restore、tar、ufsrestore	restore
csh、sh	command
date	date、time
echo	echo
ed	edlin
exit	exit
exportfs、share	share
fdformat、format	format
format	fdisk
format->analyze	scandisk
fsck	chkdsk
goto (csh)	goto

UNIX	DOS
grep	find
if	if
ln -s	subst
lp、lpr	print
ls	dir
ls -l	attrib
man	help
mkdir	md、mkdir
more	more
mv	move、ren、rename
print (sh)	echo
rm	del、erase
rm -r	deltree
rmdir	rd、rmdir
set path= (csh)， PATH= (sh)	path
set prompt= (csh)， PS1= (sh)	prompt
set var= (csh)， var= (sh)	set
shift	shift
showrev	ver
sort	sort
stty	mode
textedit，vi	edit
uncompress，unpack	expand