CA Spectrum®

命令列介面使用者指南

9.3 版

CA Technologies 產品參考資料

本指南涉及 CA Spectrum®。

連絡技術支援

如需線上技術協助及完整的地址清單、主要服務時間以及電話號碼，請洽「技術支援」，網址爲：http://www.ca.com/worldwide。

目錄

第 1 章： 命令列介面 (CLI) 簡介

本節包含以下主題：

總覽

CA Spectrum 命令列介面 (CLI) 是一個核心 CA Spectrum 元件，會隨核心 CA Spectrum 產品一起安裝。

您可以從 OneClick 使用者介面存取 CA Spectrum 資料並可執行 CA Spectrum 作業。 但若您偏好從命令列執行 CA Spectrum 作業，即可使用 CLI。 對於無法在 OneClick 中執行的工作，CLI 是您唯一可用的 CA Spectrum 資源。

CLI 是功能強大的工具，但未提供 OneClick 所提供的安全措施 (尤其是與模型化相關的安全措施)。 CLI 必須由 CA Spectrum 管理員使用，他很瞭解在網路模型結構描述上隨意建立和終結模型及修改模型屬性的可能有害影響。

CLI 是一個彈性選項。 您可以開啓 CLI 工作階段並從系統 (例如 UNIX、DOS 和 Bash) 上任何可用的命令提示字元發出命令。

CLI 命令

CLI 命令類似於 UNIX 命令，而且可以搭配 UNIX 或 DOS 命令使用，尤其是 grep (find)、pipes 和 redirect 符號。 但是，有些 CLI 命令可能會與同名的 UNIX 命令衝突。 例如，CLI 更新命令可能會與 UNIX 更新命令衝突。

若要避免衝突，請使用 vnmsh 目錄中的 **./update**。使用指令檔時，請使用 CLI 命令的完整路徑名稱，例如，<*$SPECROOT*>**/vnmsh/update**。

附註：CLI 更新命令永遠都會提供回應，不是更新成功的確認就是更新失敗的訊息。如果您在使用更新命令時，沒有收到來自 CLI 的回應，請輸入 "**which update**"。 系統的可能回應如下：

```
/etc/update
```

更多資訊：

命令說明 (位於 p. 29)

Shell 指令檔中的 CLI

CLI 命令可以併入 shell 指令檔或功能表系統中，爲您提供更加強大和多元的 CA Spectrum 資料存取方法。

每個 CLI 命令都會傳送輸出，報告命令的成功或失敗以至標準錯誤。 但是，預期在命令成功時產生的正常輸出會傳送至標準輸出。 此外，每個命令會在成功時產生值爲零的傳回碼，而在失敗時產生非零的錯誤碼。 傳回碼可讓使用 CLI 命令的 shell 指令檔根據各命令的成功或失敗繼續執行。

CLI 元件

本指南中的 CLI 元件如下所述：

- 可執行的命令
- 四個環境變數
- 與 SpectroSERVER 保持通訊的精靈
- 併入 CLI 命令的範例 shell 指令檔集合

更多資訊：

CLI 環境變數 (位於 p. 11)
CLI 本機伺服器 (位於 p. 14)
命令說明 (位於 p. 29)
範例指令檔 (位於 p. 69)

CLI 環境變數

您可以爲 CLI 設定下列四個環境變數：

CLIMNAMEWIDTH

顯示模型名稱。 依預設，create、seek 和 show 命令顯示的模型名稱最多爲 16 個字元。 但是，利用環境變數 CLIMNAMEWIDTH，您可以指定顯示最多 1024 個字元的模型名稱。 例如，使用 C shell：

```
setenv CLIMNAMEWIDTH 32
```

您可以在 .login 檔案中、在指令檔中設定此變數，或只要在發出命令前設定此變數即可。 您可以根據模型名稱的長度，在 CLI 工作階段中設定或變更任意次數。

CLISESSID

表示要用於指令檔中的 ID。 將 CLISESSID 變數設定爲 <$$>，以表示執行中 shell 指令檔的程序 ID。使用 cron 來同時執行 CLI 指令檔時，需要有此變數。 例如，使用 bash shell：

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

此外，如果您在使用 bash shell (而非 DOS) 的 Windows 上執行 CLI，則需要將 CLISESSID 環境變數設定爲各 CLI 工作階段的唯一值。 您可以提供唯一的時間戳記給每個 bash shell，例如：

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

SPECROOT

顯示警報或事件說明。 若已指定 -x 選項，則 show alarms 或 show events 命令需要 SPECROOT 環境變數。 此變數會從 SG-Support 目錄樹狀結構取得警報或事件的說明 (找得到的話)，以便擴展命令的輸出。

在 UNIX 上，您可以在登入 shell 中指定 SPECROOT 變數，並可將此變數設定爲 CA Spectrum 主目錄。 例如：

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

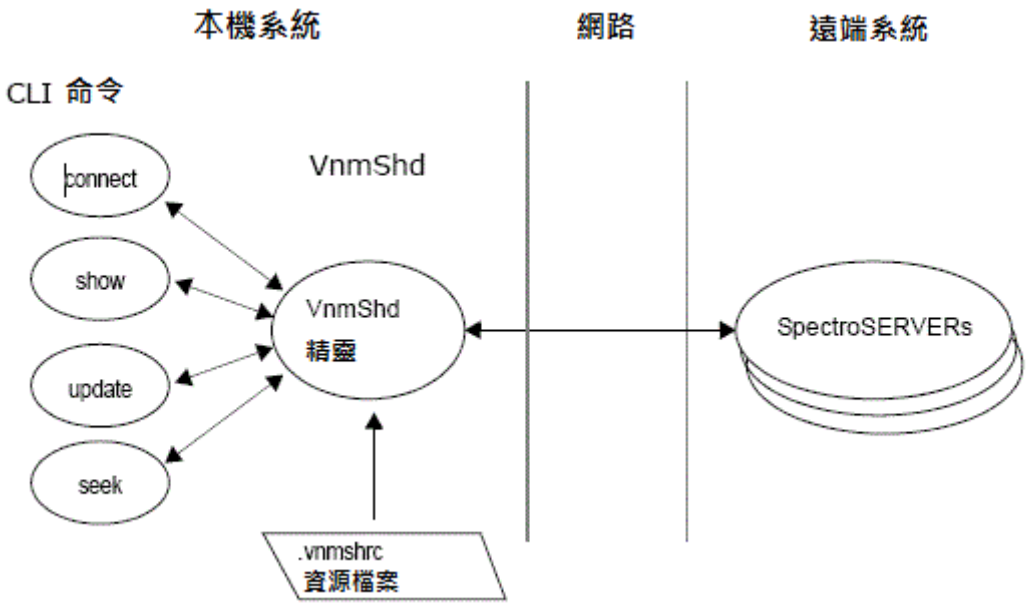
在 Windows 上，請參閱系統文件中有關設定環境變數的詳細資訊。

CLIPATH

顯示 *<$SPECROOT>*/vnmsh 目錄的路徑以及使用 CLI 命令所需的指令檔。

CLI 架構

下圖描述 CLI 架構：



CLI 本機伺服器 (在啓動時使用 .vnmshrc) 會執行下列主要功能：

- 與 SpectroSERVER 維持固定網路連線。 CLI 本機伺服器會在每次執行命令時防止連線中斷。 不論連線到精靈的 CLI 使用者有多少，此伺服器都會與 SpectroSERVER 維持*單*一連線。 就相關的時間和資源使用量而言，通訊端連線和中斷連線所費不貲。
- 維護每個 CLI 使用者的狀態資訊。例如，'current' 和 setjump 命令需要 CLI 本機伺服器來存放狀態資訊。current 命令可存放模型控制代碼和範圍控制代碼，以便用於未來的命令。 setjump 命令可存放文字字串，以識別使用者在 CA Spectrum 範圍中的目前位置。

啓動檔案

.vnmshrc 檔案 (CLI 本機伺服器啓動檔案) 位於 *<$SPECROOT>*/vnmsh 目錄中。此檔案包含數個參數，可控制 vnmsh 與 SpectroSERVER 的通訊方式。這些參數會在下列清單中說明：

vnm_hostname

指定要連線之 SpectroSERVER 的主機名稱。

client_handshake_timeout

指定在設定連線時，用戶端等待伺服器 ID 資訊的毫秒數。

預設值：900

server_handshake_timeout

指定在設定連線時，伺服器等待用戶端 ID 資訊的毫秒數。

預設值：900

connect_time_limit

指定等待連線至 SpectroSERVER 的毫秒數上限。

預設值：1000

listen_backlog

指定在等待先前的用戶端要求完成時，佇列中保留之對 SpectroSERVER 的用戶端要求數。

預設值：10

vnm_tcp_port

指定當 vnmsh 是 SpectroSERVER 用戶端時，vnmsh 用來與 SpectroSERVER 通訊的 TCP 連接埠。

vsh_tcp_port

指定當 vnmsh 當作用戶端要求 (例如 show、update) 的伺服器時，vnmsh 用於接聽 TCP 訊息的 TCP 連接埠。

debug_file

指定 CLI 寫入錯誤訊息的檔案。

max_show_event_length

指定當 **show events -x** 命令用於顯示事件訊息時的顯示字元數上限。

預設值：512

CLI 本機伺服器

發出 connect 命令的第一個使用者會在該工作站上自動啓動 CLI 本機伺服器 (VnmShd 精靈) 並建立對 SpectroSERVER 的連線。 每個工作站只可以執行一個 CLI 本機伺服器，而且該精靈只會對 SpectroSERVER 進行一個連線。

在工作站上啓動 CLI 本機伺服器之後，所有連線至該工作站上 CLI 的後續使用者都會使用相同的 CLI 本機伺服器。

更多資訊：

CLI 環境變數 (位於 p. 11)

錯誤檢查

當您在 OneClick 中執行工作時，CA Spectrum 會強制實施某些規則。 例如，規則會控制當您在不同的檢視中建立或移動裝置模型時允許的動作。

CLI 不會強制實施這些規則，而且無法執行任何錯誤檢查。 因此，CLI 可讓使用者建立模型並放置在所需之處，而無需執行錯誤檢查。 如果您嘗試以不符合格式的方式使用 CLI 命令，就會出現錯誤。

第 2 章： 使用命令列介面

本節包含以下主題：

在 UNIX 上啟動 CLI 工作階段

在 UNIX 平台上，您可以從 shell 提示啟動 CLI 工作階段。

附註：您可以使用指令檔來封裝 CLI，以便傳送到其他伺服器。 如需詳細資訊，請參閱《*CA Spectrum*》*分散式 SpectroSERVER 管理指南*。

請依循下列步驟:

1. 啟動您要連線的 SpectroSERVER。
2. 導覽至 CA Spectrum 安裝目錄中的 vnmsh 目錄：

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

3. 開啓連線：

   ```
   $ connect
   ```

   您已連線至 CLI 工作階段。

在使用 DOS 提示的 Windows 上啟動 CLI 工作階段

在 Windows 平台上，您可以從 DOS 提示啟動 CLI 工作階段。

附註：您可以使用指令檔來封裝 CLI，以便傳送到其他伺服器。 如需詳細資訊，請參閱《*CA Spectrum*》*分散式 SpectroSERVER 管理指南*。

如需本指南中 UNIX (相對於 CLI) 命令的所有例項，必要時替代對等的 DOS 命令。 例如，使用 *find* 而非 *grep*。

請依循下列步驟:

1. 如需 DOS 提示，請選取 [開始]、[所有程式]、[命令提示字元]。

   DOS 提示隨即出現，準備接受 CLI 命令。
2. 啓動您要連線的 SpectroSERVER。
3. 導覽至 CA Spectrum 安裝目錄中的 vnmsh 目錄：

   ```
   $ cd <$SPECROOT>/vnmsh
   ```
4. 開啓連線：

   ```
   $ connect
   ```

   您已連線至 CLI 工作階段。

更多資訊：

UNIX 與 DOS 的轉換 (位於 p. 95)

在使用 Bash 提示的 Windows 上啓動 CLI 工作階段

在 Windows 平台上，您也可以從 bash shell 提示啓動 CLI 工作階段。

附註：您可以使用指令檔來封裝 CLI 程式，以便傳送到其他伺服器。 如需詳細資訊，請參閱《*CA Spectrum*》*分散式 SpectroSERVER 管理指南*。

請依循下列步驟:

1. 按一下 [開始]、[所有程式] 和 [命令提示字元]。

   DOS 提示隨即出現。
2. 在 DOS 提示中，輸入 **bash**。
3. 按一下 [開始]、[執行]，然後輸入 **bash -login**。

   您可以從 bash shell 提示啓動 CLI 工作階段。
4. 啓動您要連線的 SpectroSERVER。

5. 導覽至 CA Spectrum 安裝目錄中的 vnmsh 目錄：

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

6. 開啓連線：

   ```
   $ connect
   ```

   您已連線至 CLI 工作階段。

使用方式範例

下列範例示範如何將 CLI 命令使用於 CA Spectrum 中的常見工作。

建立使用者模型

使用者模型可供使用者存取 CA Spectrum。 使用者是依據登入 ID 進行識別。

附註：啓動 CLI 工作階段之前，請先確認已建立使用者模型並已啓動要連線的 SpectroSERVER。

請依循下列步驟:

1. 連線至 SpectroSERVER。

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

   您已連線至 SpectroSERVER。

   附註：如果您有連線問題，請驗證錯誤訊息。 如需詳細資訊，請參閱錯誤訊息 (位於 p. 75)。

2. 針對您想建立的模型類型，使用 show 命令判斷模型類型控制代碼。 在此例中，這是「使用者」類型的模型。 輸入此命令：

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

   附註：“./” 很重要。 某些 UNIX 系統使用 show 命令來讀取電子郵件。 如果 “.” 不是使用者環境中的第一個路徑，則需要 "./"。

此時會出現包含 'User' 字串的模型類型清單，而且「使用者」模型類型會列在最前面。

```
Handle     Name          Flags
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「使用者」模型類型的屬性，並判斷模型名稱屬性的屬性 ID。 您需要此屬性 ID 才能建立模型。 輸入此命令：

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

   包含模型名稱屬性的「使用者」模型類型屬性清單隨即出現。

   ```
   Id        Name                Type          Flags
   0x10000   Modeltype_Name       Text String   R,S,M
   0x1006e   Model_Name           Text String   R,W,G,O,M,D
   0x10074   User_Full_Name       Text String   R,W,O,D
   0x1155f   gib_mtype_nameText   String        R,W,S,D
   0x11560   gib_mtype_name_menu  Text String   R,W,S,D
   0x11561   gib_model_name       Text String   R,W,D
   0x11563   gib_model_name_menu  Text String   R,W,D
   0x1197d   WatchNames          Tagged Octet  R,W,D
   ```

4. 使用 create 命令搭配模型類型控制代碼、模型名稱的屬性 ID 以及使用者的值 (登入 ID 名稱) 以建立模型。 在此範例中，使用者登入 ID 爲 j_doe。 輸入此命令：

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

   類似下列命令的系統訊息會確認已建立模型：

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

   附註：這些範例中使用的所有控制代碼和 ID 都是虛構的。 您建立之模型的模型控制代碼會不同；模型控制代碼是由系統所建立。

更多資訊：

錯誤訊息 (位於 p. 75)

修改模型屬性

這一節提供的範例說明如何使用 CLI 命令來變更模型屬性的值。 此範例特別示範如何針對使用者模型中建立的模型 (j_doe) 以變更社群字串屬性值。 如需詳細資訊，請參閱建立使用者模型 (位於 p. 17)。

請依循下列步驟:

1. 判斷 j_doe 模型控制代碼，然後將 j_doe 設定爲目前的模型：
   a. $ ./show models | grep j_doe

      有關 j_doe 模型的下列資訊隨即出現：

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

   b. $ ./current mh=0xbe0001b

      系統確認 j_doe 爲目前模型：

      ```
      current model is 0xbe0001b
      current landscape is 0xbe00000
      ```

2. 判斷社群字串屬性的 ID。

   附註：爲了簡明扼要，此步驟將屬性名稱的已知部份 (社群字串) 顯示爲 grep 命令的引數。 如果您不知道屬性名稱，可以顯示並掃描模型的所有屬性，以判斷正確的屬性名稱及其屬性 ID。

3. 輸入此命令：

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

   屬性 ID、屬性名稱和社群字串值隨即出現：

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

   建立使用者模型時，CA Spectrum 會將預設值 ADMIN,0 指派給所有使用者模型。ADMIN,0 會將 CA Spectrum 的完整管理權限授與給使用者模型。

4. 使用 update 命令，將 j_doe 模型的管理權限層級從 ADMIN,0 變更爲範例權限層級 ADMIN,5 (唯讀)。 輸入此命令：

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

   系統會傳回可顯示屬性值變更的項目：

   ```
   Id        Name                    Iid    Value
   0x1007a   User_Community_String          ADMIN,5
   ```

   Iid 屬性僅套用至清單屬性，所以此處沒有它的值。 如需詳細資訊，請參閱《*CA Spectrum 管理指南*》。

在一個步驟中建立與修改模型

這一節提供的範例說明如何在單一命令字串中，建立模型並以其他值取代預設屬性值。 只有在嘗試執行本節中顯示之類型的複雜命令之前，您已知道針對此命令提供的相關模型識別碼的值時，才只可以執行該命令。

以下範例使用「建立使用者模型和修改模型屬性」中介紹的參數值：

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

更多資訊：

建立使用者模型 (位於 p. 17)
修改模型屬性 (位於 p. 19)

CLI 指令檔檔案範例 - 建立新使用者

您可以在 Windows 平台中執行可從 bash 提示併入 CLI 命令的 shell 指令檔，就好像在 UNIX 上從 shell 命令提示執行此命令。

以下範例示範如何將指令檔用於建立 CA Spectrum 使用者模型。

```
#
#  Check to see if CLIPATH is set.  If it is not then we will have to create it.
#
#  Setup a variable to point to the /install_area/vnmsh directory so we can
#  find the commands we need.
#
if [ -z "$CLIPATH" ]
then
     CLIPATH=/usr/data/spectrum/7.0/vnmsh
     export CLIPATH
fi
```

```
#
#  Test to make sure the CLIPATH points to a valid directory
#
if [ ! -d $CLIPATH ]
then
        echo "ERROR: could not find $CLIPATH"
        echo "Please find the correct path to the vnmsh directory and set"
        echo "the CLIPATH environment variable to it."
        exit 0
fi
#
#  Now check to see how many command line arguments there are.  If there are
#  none, then echo a usage message.  If there is one, that is all we really
#  need to create a new user...  If there is a second argument then we can
#  set the Community_String at the same time.
#
#  This setup is only for creating a user on the local system or what the
#  .vnmshrc file points to for the vnm_hostname.  A third field could be
#  added that accepts the vnm_hostname to connect to.
#
#  Optionally, the getopts shell command can be used to parse "switches" to
#  the script:  -n for name, -c for community string and -v for vnm_hostname.
#
#  (NOTE:  getopts should be located in /usr/bin/getopts if the script is
#   done in bourne shell (sh).  k-shell has a built in getopts function)
#
if [ $# -eq 0 ]
then
        echo "Usage:  $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

#
#  Okay, we should be all set now to go ahead and create the new user.
#  The first thing we have to do is connect.
#
$CLIPATH/connect
```

```
#
#  Now let's check the exit status of the connection to see if we got in...
#
if [ $? -ne 0 ]
then
        echo "ERROR:  could not connect to SpectroSERVER.  $0 exiting"
        exit 0
fi
#
#  Okay if we made it this far then we have a connection.  Let's try the
#  create command.
#
$CLIPATH/create model mth=0x10004 $command
#
#  Now we check the exit status again and see if we actually created a model.
#
if [ $? -ne 0 ]
then
        echo "ERROR:  could not create a new user.  $0 exiting"
        exit 0

else
        echo -n "New user $1 created"
        if [ $flag -eq 1 ]
then
        echo " Community_String was set to $2"
fi
        echo "Successfully created new model... exiting."
fi
$CLIPATH/disconnect
exit 1
```

事件報告產生

CLI 會保留一份清單，內含在某個範圍發生的 2000 個最近事件。 但是，如果在某個範圍中發生許多事件，則最近事件大約在一小時前發生。

您可以使用 -x 選項搭配 show events 命令，設定 SPECROOT 環境變數。 以下命令是使用 CLI 執行事件報告的範例：

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

模型交換

可以在不同模型中來回移動的指令檔中，jump 和 setjump 命令很實用。 setjump 命令可讓您指派文字字串，以表示模型控制代碼與其對應的範圍控制代碼。 然後您可以使用 jump 命令搭配該文字字串，擷取該資訊作爲目前的模型控制代碼。 例如：

- $ ./current mh=0xb6000f8

  ```
  current model is 0xb6000f8
  current landscape is 0xb600000
  ```

- $ ./setjump emme

  ```
  model 0xb6000f8 and landscape 0xb600000 stored under emme
  ```

- $ ./jump emme

  ```
  current model is 0xb6000f8
  current landscape is 0xb600000
  ```

建立疑難排解員模型

您可以使用 CLI 建立「疑難排解員」模型並使其與「使用者」模型產生關聯。 一旦建立並產生關聯，即可將警報指派給這些疑難排解員，而他們會收到電子郵件通知，通知他們必須調查警報並加以解決。

以下程序描述如何建立「疑難排解員」模型並使其與「使用者」模型產生關聯。 以在建立使用者模型 (位於 p. 17)中建立的「使用者」模型爲例。

請依循下列步驟:

1. 導覽至 <*$SPECROOT*>/vnmsh 目錄。
2. 在命令提示字元輸入以下命令 (包含 $ 提示之 bash shell 的範例)，以連線至 SpectroSERVER：

   ```
   $ ./connect
   ```

3. 判斷「疑難排解員」模型類型。 輸入此命令：

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

   隨即傳回「疑難排解員」模型類型項目。

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

4. 判斷「疑難排解員」模型類型 EmailAddress 屬性 ID。 輸入此命令：

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

   EmailAddress 項目隨即出現：

   ```
   0x11d24        EmailAddress         TextString    R,W,D
   ```

5. 使用 CLI create 命令建立「疑難排解員」模型：

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

   類似下列命令的系統訊息會確認已建立模型：

   ```
   $ created model handle = 0xbe0001c
   ```

   附註：範例中使用的所有控制代碼和 ID 都是虛構的。 您建立之模型的模型控制代碼會不同。 視您的系統的建立而定。

6. 使用 CLI create 命令，建立 j_doe 使用者模型 (mh=0xbe0001b) 與 j_doe_fixit 疑難排解員模型 (mh=0xbe0001c) 之間的關聯：

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

   類似下列命令的系統訊息會確認已建立關聯：

   ```
   $ create association successful
   ```

將警報指派給疑難排解員

這一節說明如何使用 CLI 命令列出警報，並將警報指派給疑難排解員。

請依循下列步驟:

1. 使用 show 命令列出警報。

   此步驟顯示如何只找出 alarm_severity 爲 MAJOR (主要) 的警報。

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

   「主要」警報清單隨即出現。 例如：

```
7509   2000/9/27   14:46:44   0xd80008   0xa6000df   duncan    9E133_36   MAJOR   No
7645   2000/9/27   14:47:16   0xd80008   0xa60025e   infinity  9H422_12   MAJOR   No
7518   2000/9/27   14:47:01   0xd80008   0xa6000eb   rugone    9E132_15   MAJOR   No
7979   2000/9/27   14:53:12   0xf40002   0xa600161   FDDI2     FddiMAC    MAJOR   No
8018   2000/9/27   14:53:13   0xf40002   0xa6003da   FDDI FNB  FddiMAC    MAJOR   No
7512   2000/9/27   14:46:47   0xd80008   0xa6000af   ruthere   9A426_02   MAJOR   No
```

2. 選取您要爲其指派疑難排解員的警報。 在此範例中，已選取 9A426-02 裝置的警報 ID 7512。

3. 選取要指派給該警報的疑難排解員。 在此範例中，已選取在建立疑難排解員模型 (位於 p. 23) 中建立的 j_doe_fixit 疑難排解員模型。

   附註：使用「疑難排解員」模型控制代碼 (0xa600722) 而非「疑難排解員」模型名稱 (j_doe_fixit)，在下一個步驟的 update 命令中指定疑難排解員。

4. 使用 update 命令將警報指派給疑難排解員：

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

   類似下列範例的系統訊息會確認已將疑難排解員指派給警報：

   ```
   $ update: successful
   ```

   現在已將以 j_doe_fixit 模型表示的人員指派給警報。 此人員會收到警報指派的電子郵件通知。

建立全域集合

您可以使用 GUID 建立全域集合，並可在 CLI 工作階段中設定搜尋準則。 唯一識別碼 (GUID) 是建立全域集合的重要屬性。 全域集合需要 GUID，才能正常運作。 您可以透過對 VNM 模型採取的動作取得 GUID。

附註：CLI 中沒有 GUID 的全域集合是無效的。 使用 OneClick 建立全域集合時，系統會自動建立 GUID。 如需詳細資訊，請參閱《*CA Spectrum*》*模型化和管理 IT 基礎架構管理指南*。

請依循下列步驟:

1. 輸入下列命令：

   ```
   update action=(取得唯一識別碼的動作) 0x10474 mh= (VNM 模型控制代碼)
   ```

   GUID 即建立完成。

   附註：取得新 GUID 的動作爲 0x10474，這與全域集合模型類型相同。

2. 輸入以下命令以使用 GUID 建立全域集合：

   ```
   create model mth=(全域集合的模型類型) 0x10474 attr=(GUID) 0x12e56, val=(您之前所收到的值) 4a85b9af-0d52-1000-017f-0013727f8c0a
   ```

   全域集合隨即建立並出現在全域集合之下的 [導覽] 窗格。

範例：建立包含或不含 XML 字串的全域集合

輸入以下命令以建立包含或不含 XML 字串的全域集合：

```
update action=(取得 GUID 的動作)0x10474 mh=(範圍)

輸出：
update action: successful
Response has 1 attributes:
 0) Attribute 0x0 text: EXAMPLE GUID(4a85b9af-0d52-1000-017f-0013727f8c0a)

create model mth=(全域集合的模型類型)0x10474
attr=(GUID)0x12e56,val=(您之前所收到的值)4a85b9af-0d52-1000-017f-0013727f8c0a
attr=(dynamicCriteriaXML)0x12a6a,val=(XMLString)'<search-criteria><filtered
models><equals-ignore-case><model-name>sometext</model-name></equals-ignore-c
ase></filtered-models></search-criteria>'
輸出：
created model handle = New model handle(0x78101069)
```

附註：您可以使用 create 命令指定 dynamicCriteriaXML (0x12a6a) 屬性，也可以稍後更新模型。

更多資訊：

命令說明 (位於 p. 29)

在 CLI 輸出中隱藏標頭

若要在 CLI 輸出中隱藏標頭，您可以建立包含下一節中提供之函數的檔案，並可在各指令檔的最上方參照此檔案。

以下程序中的函數會呼叫 CLI 命令，並去除命令輸出中的標頭資訊。

請依循下列步驟:

1. 在 scripts 目錄中建立名爲 StripHeaders 的檔案。
2. 在 StripHeaders 檔案中包含下列函數：

```
tcreate() # only needed for the createalarm
    { # and create event commands
    $CLIPATH/create $@ | tail +2
    }
tseek()
    {
      $CLIPATH/seek $@ | tail +2
    }
tshow()
    {
    $CLIPATH/show $@ | tail +2
    }
tupdate()
    {
    $CLIPATH/update $@ | tail +2
    }
```

3. 在 CLI 指令檔的最上方包含名稱 StripHeaders，如下所示：

```
. StripHeaders
```

4. 每當您想要移除 CLI 輸出中的標頭時，請呼叫 tcreate()、tseek()、tshow() 和 tupdate() 函數，而非對應的 CLI 命令。

   例如，下一行可產生不含 CLI 標頭資訊的 show models 命令輸出：

```
tshow models
```

第 3 章： 命令說明

本節提供 CLI 命令和輸出的說明。

本節包含以下主題：

命令說明概觀

不需利用 CA Spectrum 中的安全措施，您即可使用 CLI 變更 CA Spectrum 知識庫。如果您指定的資訊不正確，可能會導致系統當機或資料庫毀損。因此，當您繼續使用 create、destroy 或 update 命令時，請格外小心。

附註：將 CLI 命令參數使用於透過 CLI 建立和管理回應時間測試。

如需詳細資訊，請參閱《*CA Spectrum*》*Service Performance Manager 使用者指南*。

ack alarm - 認可警報

ack alarm 命令會認可在 landscape_handle 指定的範圍中，alarm_id 所指定的警報。若未指定 landscape_handle，則此命令會認可目前範圍中，alarm_id 所指定的警報。

認可模型的某一個警報，意味著您僅認可該警報，而不認可該模型的其他警報。

此命令的格式如下：

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

如果使用有效的 alarm_id 和有效的 landscape_handle 輸入 ack alarm，則會顯示下列訊息：

```
ack alarm: successful
```

範例：ack alarm

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

connect - 連線至 SpectroSERVER

connect 命令可讓 CA Spectrum 命令列介面的使用者連線至在主機系統 (*主機名稱*) 上執行的 SpectroSERVER。此命令也會將 landscape_handle 指定的範圍設定爲目前的範圍 。 如果尚未執行 CLI 本機伺服器，則 connect 命令會予以啓動。

此命令的格式如下：

```
connect [<主機名稱>] [lh=<landscape_handle>][vnmsocket=<vnm 通訊端>]
```

主機名稱

(選擇性) 若未指定主機名稱，則此命令會讓使用者連線至在 CLI 資源檔案 .vnmshrc 中指定的主機。

附註：CA Spectrum 命令列介面不支援 localhost 或 127.0.0.1 選項。 若要連線至 localhost，可以指定實際主機名稱或不指定任何參數。

landscape_handle

(選擇性) 若未指定 landscape_handle，則此命令會將目前範圍設定爲所指定主機名稱的範圍。

vnm 通訊端

(選擇性) 若未指定 vnm 通訊端，則此命令會使用在 .vnmshrc 檔案中指定的通訊端連線至 SpectroSERVER。 您可使用 vnm 通訊端來連線至不同連接埠連線上的其他 SpectroSERVER (由 vnm 通訊端所定義)。

在 Unix 上，CLI 本機伺服器所回報的錯誤訊息會顯示在主控台視窗中。 在 Windows 上，這些錯誤會顯示在使用者 bash shell 視窗中。

範例：connect

```
#! /usr/bin/sh
# A sample script to get alarms of a specific
# severity and set the CLISESSID

if [$# !=1]
then
    echo “Usage: $0 <alarm severity>”
    exit 0
fi

CLISESSID=$$

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

exit 0
```

如果命令成功，則會顯示以下訊息：

```
connect: successful hostname
current landscape is <landscape_handle>
```

Hostname 是使用者輸入的 SpectroSERVER 主機或是在 .vnmshrc 檔案中指定的主機。 landscape_handle 是使用者輸入之範圍或主機的範圍。

更多資訊：

啓動檔案 (位於 p. 13)
CLI 環境變數 (位於 p. 11)

使用 connect 命令時的注意事項

以下是使用 connect 命令時的重要注意事項：

- 終端機上的使用者必須使用 connect 命令來初始化通訊。 同一個使用者必須使用 disconnect 命令來終止與 SpectroSERVER 的通訊。
- 第一個使用者輸入 connect 命令後，CLI 本機伺服器就會連線至 SpectroSERVER。
- 使用相同 CLI 本機伺服器的其他 CLI 使用者，只可以連線至初始 SpectroSERVER 之範圍對應中的 SpectroSERVER。 所有使用者都中斷連線後，請使用 connect 命令連線至不同範圍對應中的 SpectroSERVER。
- 若要成功連線至 SpectroSERVER，必須將 connect 命令的第一個使用者定義爲原始 SpectroSERVER 的 CA Spectrum 資料庫中的使用者。
- 以 bash shell 執行 CLI 的 Windows 使用者也必須定義 CLISESSID。
- 特定使用者的終端機裝置是使用 ttyslot(3V) 函數判定的。
- Cron 指令檔並未附加到 ttyslot。 因此，ttyslot 函數會對所有 cron 指令檔傳回 0。 也就是，同時作爲 cron 指令檔執行的兩個 CLI 指令檔，會對 CLI 本機伺服器顯示爲一個 CLI 使用者，而導致無法預期的結果。 因此，您必須在指令檔的最上方插入一行，以匯出環境變數 CLISESSID。 將 CLISESSID 設定爲唯一數值。 CLI 現在可以區分不同的 cron 指令檔。

  以下範例定義指令檔中的唯一 CLI 工作階段 ID：

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

- 此範例將 CLISESSID 設定爲執行此指令檔之 shell 的程序 ID。 當 ttyslot 函數傳回零時，CLI 會使用 CLISESSID 來識別使用者。 爲每個 CLI 工作階段設定一次 CLISESSID。 如果作爲 cron 指令檔執行的 CLI 指令檔呼叫其他 CLI 指令檔，則只有最上層指令檔會設定 CLISESSID 環境變數。 除非在指令檔最上方叫用新的 shell (#!/bin/sh)，否則其他 CLI 指令檔會在相同的程序 ID 之下執行。 若要在其他指令檔中叫用新的 shell，請匯出 CLISESSID，然後連線後再次中斷連線。
- 在某些環境或配置中，即使是從命令列輸入此命令，ttyslot 函數仍可能傳回零。 在此例中，connect 命令會傳回下列錯誤：

  ```
  connect: variable CLISESSID not set
  ```

  在此情況下，請從命令列或從 .cshrc 或其他啓動檔案設定 CLISESSID。

- CLI 會利用使用者名稱和終端機裝置來識別每個 CLI 使用者。同時從終端機裝置執行多個指令檔的使用者會對 CLI 顯示爲相同使用者。如果某個指令檔正在背景執行，而另一個指令檔正在前景執行，或有多個指令檔同時在背景執行，則 CLI 可能會產生無法預期的結果。

  例如，指令檔 A1 會將目前模型設定爲模型 A。指令檔 B1 (由相同使用者從相同終端機裝置執行) 會將目前模型設定爲模型 B。如果指令檔 A1 在模型 A 上執行 update 命令，則也會在指令檔 B1 的模型 B 上執行 update 命令。

  一次只從特定終端機裝置執行一個 CLI 工作階段。若要同時執行多個 CLI 工作階段，請從個別的終端機裝置予以執行，或使用 at(1) 或 batch(1) 命令 (並將其 CLISESSID 環境變數設定成唯一值) 予以執行。

create - 建立物件

使用 create 命令建立物件。

附註：如需在安全的網域中建立模型的詳細資訊，請執行 ./create 以顯示使用方式陳述式。

此命令的格式如下：

```
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]
```

更多資訊：

CLI 環境變數 (位於 p. 11)

create alarm

create alarm 命令會爲具有 model_handle 的模型，建立嚴重性爲 alarm_severity，且原因爲 probable_cause_id 的警報。有效的警報嚴重性選項有：「重大」、「主要」、「次要」、「良好」、「維護」、「已隱藏」或「初始」。依預設，新警報會取代現有警報。

如果使用有效的 alarm_severity、有效的 probable_cause_id，和有效的 model_handle 輸入 create alarm，則會顯示警報表格中建立的項目。 建立時間會以 hh:mm:ss 格式顯示。

範例：create

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

ID     Date          Time        PCauseID    MHandle     MName    MTypeName     Severity    Ack
984    2000/5/11    12:33:27    0x10308     0x400134    12.84    Bdg_CSI_CN    CRITICAL    No
```

create association

create association 命令會建立具有 left_model_handle 的模型與具有 right_model_handle 的模型之間關係 (關聯) 的例項。

如果使用有效 left_model_handle 與 right_model_handle 之間的有效關係輸入 create association，則會顯示下列訊息：

```
create association: successful
```

範例：create association

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

create event

create event 命令會爲 model_handle 所指定的模型，建立類型爲 event_type 且文字爲 event_text 的事件。 若已指定 landscape_handle，則會爲已建立事件的使用者模型建立事件。

附註：在舊版的 CLI 中，系統會爲範圍模型建立事件。

若未指定 model_handle 或 landscape_handle，則會爲已建立事件的使用者模型建立事件。 或者，若已指定目前模型，則會爲目前模型建立事件。 CA Spectrum 中的某些事件缺少相關聯的模型。 例如，當應用程式連線至 SpectroSERVER 時，事件就沒有相關聯的模型。

如果使用有效的 event_type、有效的 event_text，和有效的 model_handle 或 landscape_handle (若存在) 輸入 create event，此項目就會顯示在事件表格中。 建立時間會以 hh:mm:ss 格式顯示。

event_type 命令 (在 CA Spectrum 中也稱爲事件代碼) 爲 4 個位元組的十六進位數字。 兩個最重要的位元組可指定事件的開發人員 ID (0001 代表 CA Spectrum 產生的事件代碼)，而兩個最不重要的位元組則是唯一的事件識別碼。 並非所有事件類型都包含使用者輸入的文字。 事件格式檔案中包含變數 {S 0} 的事件類型就屬於這類事件類型。 若是不包含使用者輸入之文字的事件類型，event_text 參數會被忽略，但仍必須出現在命令列上。

如需詳細資訊，請參閱《*CA Spectrum Certification 使用者指南*》。

範例：create event

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

Date          Time        Type       MHandle     MName    MTypeName
2000/5/11     12:39:42    0x1061a    0x400134    12.84    Bdg_CSI_CNB20
```

create model

您可以使用 IP 位址或模型類型控制代碼來指定 create model。 在任一情況下，系統會在 landscape_handle 所指定的範圍中建立模型。 若未指定 landscape_handle，則此命令會在目前範圍中建立模型。

附註：只有在建立「使用者」模型時才需要 model_name 屬性。

- 如果您使用 IP 位址指定 create model，則系統會尋找位於所指定 ip_address 的物件並爲其建立模型。 此模型具有該物件的所有內容，包含任何相關聯的子項。 例如，此物件若是中樞，則 create model 命令會利用其所有連接埠建立中樞的模型。
- 您可以指定 IPv4 位址或 IPv6 位址。 不支援 IPv6 範圍，而且此命令不支援屬性 ID 的設定。
- 若要同時建立多個模型，您可以利用 create model 命令定義 IP 位址範圍。指定 Low_IP 和 High_IP 參數並以 "-" 分隔。 若未指定 Community_Name，新建立的模型則爲 "Pingable" 類型。 若已指定 Community_Name，此裝置則會模型化成爲適當的模型類型。 Try_Count 和 Time_Out 選項類似於 OneClick [依據 IP 建立模型] 對話方塊的選項。
- 如果您使用模型類型控制代碼來指定 create model 命令，則系統會建立類型爲 model_type_handle 的模型。 您可以接著針對建立的模型，設定一或多個屬性的值。
- 當您使用模型類型控制代碼來指定 create model 命令時，您也可以在該命令中指定多個屬性。 指定多個「attribute_id、value」配對，並以空格分隔相鄰的配對。

- 使用者使用 OneClick 建立特定模型類型的模型時指定的屬性值，應在 create model 命令中加以指定。 否則，建立模型後，SpectroSERVER 中可能會在發生「推斷控制代碼」錯誤。 例如，使用 OneClick 建立 Hub_CSI_IRM3 模型時所顯示的視窗，可供您輸入 [模型名稱]、[網路位址]、[社群字串] 的值。 透過 CLI 使用 create model 命令建立相同類型的模型時，請指定這些屬性的值。
- 若使用有效的 model_type_handle 和有效的 attribute_id、值配對 (若存在) 指定 create model，則會顯示建立的模型控制代碼。
- 若使用有效的 ip_address 指定 create model，則會顯示建立的模型控制代碼。

範例：create model

```
$ create model mth=0x102d attr=0x12d7f,val=132.177.12.84
attr=0x1006e,val=12.84lh=0x400000
created model handle = 0x400134

$ create model ip=206.61.231.1-206.61.231.5

Creating model for IP=206.61.231.1
created model handle =  0x9a00259
Creating model for IP=206.61.231.2
create model: DCM device unreachable
Creating model for IP=206.61.231.3
create model: DCM device unreachable
Creating model for IP=206.61.231.4
create model: DCM device unreachable
Creating model for IP=206.61.231.5
created model handle =  0x9a0025a
```

附註：依預設，create 命令顯示的模型名稱最多 16 個字元。 但是，利用環境變數 CLIMNAMEWIDTH，您可以指定顯示不同字元數 (最多 1024 個) 的模型名稱。

current - 設定模型或範圍

current 命令會將 model_handle 指定的模型設定爲目前的模型。 或者，此命令會將 landscape_handle 指定的範圍設定爲目前的範圍。 若未指定 model_handle 和 landscape_handle，則 current 會顯示目前模型控制代碼和目前範圍控制代碼。

當使用者設定目前模型時，CLI 會將目前範圍設定爲含有此模型的範圍。 當使用者設定目前範圍時，CLI 會將目前模型設定爲「未定義」。

系統會針對已連線至 CLI 本機伺服器的每個工作階段，維護個別的目前模型和目前範圍值。

current 命令只會針對列舉該命令之類的工作階段，保留狀態資訊、目前模型和目前範圍。

這個命令具有下列格式：

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

- 若已將有效的 model_handle 指定爲輸入，則會顯示以下訊息：

  ```
  current model is <model_handle>
  current landscape is <current_landscape_handle>
  ```

- 若已將有效的 landscape_handle 指定爲輸入，則會顯示以下訊息：

  ```
  current model is undefined
  current landscape is <landscape_handle>
  ```

- 若未指定 model_handle 和 landscape_handle，則會顯示以下訊息：

  ```
  current model is <current_model_handle>
  current landscape is <current_landscape_handle>
  ```

- 若未指定 model_handle 和 landscape_handle 且未定義目前模型，則會顯示以下訊息：

  ```
  current model is undefined
  current landscape is <current_landscape_handle>
  ```

範例：current

```
$ current mh=0x400142
current model is 0x400142
current landscape is 0x400000

$ current lh=0x500000
current model is undefined
current landscape is 0x500000

$ current
current model is undefined
current landscape is 0x500000
```

附註：目前範圍是由 connect 命令所設定，所以永遠含有一個值。

destroy - 終結物件

使用 destroy 命令終結物件。 這個命令具有下列格式：

```
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

若已使用 destroy 命令指定 -n (不提示) 選項，則系統不會顯示確認提示。 在 CLI 指令檔中，這個選項很實用。

除非已指定 -n 選項，否則一律會顯示以下其中一項訊息：

```
destroy model: are you sure?
destroy association: are you sure?
destroy alarm: are you sure?
```

有效回應包含 y、yes、Y、Yes、n、no、N 和 No。

destroy alarm

終結在 landscape_handle 指定的範圍中，alarm_id 所指定的警報。 除非已指定 -n 選項，否則 destroy alarm 會在終結警報前提示您進行確認。 若未指定 landscape_handle，則此命令會終結目前範圍中，alarm_id 所指定的警報。 使用 show alarms 命令來判斷模型的 alarm_id。

如果使用有效的 alarm_id 和有效的 landscape_handle 輸入 destroy alarm 命令，則會顯示下列訊息：

```
destroy alarm: successful
```

範例：destroy alarm

```
$ destroy alarm aid=300
destroy alarm: are you sure? y
destroy alarm: successful
```

destroy association

終結具有 left_model_handle 的模型與具有 right_model_handle 的模型之間的關聯 (關係的例項)。 除非已指定 -n 選項，否則 destroy association 會在終結關聯前提示您進行確認。

如果使用有效 left_model_handle 與 right_model_handle 之間的有效關係輸入 destroy association，則會顯示下列訊息：

```
destroy association: successful
```

範例：destroy association

```
$ destroy association rel=Lost_and_Found lmh=0x400001 rmh=0x40h0142
destroy association: are you sure? y
destroy association: successful
```

destroy model

終結具有指定 model_handle 的模型。除非已指定 n 選項，否則 destroy model 會在終結模型前提示您進行確認。

若使用有效的 model_handle 輸入 destroy model，則會顯示下列訊息：

```
destroy model: successful
```

範例：destroy model

```
$ destroy model mh=0xa600715
```

將會終結以下模型：

```
Model_Handle      ->     0xa600715
Model_Type_Handle ->     0x10004
Model_Name        ->     garciaparra
Model_Type_Name   ->     User

destroy model: are you sure? y
destroy model: successful
```

disconnect–Disconnects from SpectroSERVER

使用 disconnect 命令中斷 CLI 使用者與目前連線之 SpectroSERVER 的連線。

這個命令具有下列格式：

```
中斷連線
```

如果此命令成功，則會顯示以下訊息，其中主機名稱是使用者所連線的 SpectroSERVER 主機名稱：

```
disconnect: successful from <主機名稱> or <IP 位址> - connected for xx hours, yy minutes
```

更多資訊：

stopShd–終止 CLI 本機伺服器 (位於 p. 61)

jump - 跳轉至已儲存的模型或範圍

jump 命令會跳轉至先前儲存的模型和範圍。 jump 命令會將目前模型和目前範圍設定爲由 setjump 命令儲存在 text_string 標籤之下的模型和範圍。 若未指定 text_string，則會顯示先前 setjump 命令中提供的 text_string 清單。

此命令的格式如下：

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

- 若使用先前定義的有效的 text_string 輸入 jump，則會顯示新的目前模型和目前範圍：

  ```
  current model is <current_model_handle>
  current landscape is <current_landscape_handle>
  ```

- 若輸入的 jump 沒有 text_string，則會顯示目前定義的 text_string 清單。例如：

  ```
  text_string1
  text_string2
  --
  ```

- 若已輸入 jump 但未定義新的目前模型，則會顯示以下訊息：

  ```
  current model is undefined
  current landscape is <current_landscape_handle>
  ```

範例：jump

```
$ jump tutorial
current model is 0x400142
current landscape is 0x400000
```

更多資訊：

setjump - 儲存模型和範圍 (位於 p. 44)

seek–尋找模型

使用 seek 命令尋找模型。 seek 命令會在 landscape_handle 指定的範圍中，尋找擁有 attribute_id 所指定屬性之指定值的模型。 若未指定 landscape_handle，此命令會在目前範圍中尋找擁有指定 attribute_id 之屬性值的模型。 您也可以使用萬用字元 (*) 搭配 seek，尋找含有指定子字串的模型例項。 如果您輸入 null 值，可以找到沒有名稱的所有模型 (例如，attr=0x1006e)。

您無法使用 seek 命令搜尋一個字元的屬性值。 嘗試進行這類搜尋會傳回錯誤。

此命令的格式如下：

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

選項的使用順序不拘，例如 -i -s 或 -s -i。

-i

若使用 seek 命令指定 -i (忽略大小寫區分) 選項，則會傳回使用 val 參數指定的模型資訊 (不區分大小寫)。

-s

若使用 seek 命令指定 -s (允許使用子字串) 選項，則使用 val 參數指定的模型資訊會與子字串 (若適用的話) 一起傳回。

若使用有效的 attribute_id 和有效值輸入 seek，則會以下列格式顯示所有相符模型：

```
MHandle      MName   MTypeHnd          MTypeName
modelhandle  name    modeltypehandle   name
```

若找不到相符模型，則會顯示以下訊息：

```
seek: no models found
```

附註：依預設，seek 命令顯示的模型名稱最多 16 個字元。 但是，利用環境變數 CLIMNAMEWIDTH，您可以指定顯示不同字元數 (最多 1024 個) 的模型名稱。

範例：seek

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

MHandle       MName      MTypeHnd   MTypeName
0xb100018     spectrum    0x1004     User
0xb10008d     spectrum    0x820000   ScmConfig

$ seek attr=0x1006e,val=CA Spectrum

MHandle       MName      MTypeHnd   MTypeName
0xb100018     spectrum    0x820000   ScmConfig

$ seek attr=0x1006e,val=SPE
seek: no models found

$ seek attr=0x1006e,val=spe lh=0xb100000
seek: no models found

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

MHandle       MName      MTypeHnd   MTypeName
0xb10018      spectrum    0x10004    User
0xb1008c      spectrum    0x820000   ScmConfig
0xb1008d      spectrum    0x820000   ScmConfig

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

MHandle       MName      MTypeHnd   MTypeName
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

MHandle       MName      MTypeHnd   MTypeName
0xb10018      spectrum    0x10004    User
0xb10089      spectrum    0x820000   ScmConfig
0xb1008c      spectrum    0x820000   ScmConfig
0xb1008d      spectrum    0x820000   ScmConfig

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

MHandle       MName      MTypeHnd   MTypeName
0xb10008c     spectrum    0x820000   ScmConfig
```

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

seek: no models found

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

MHandle       MName        MTypeHnd      MTypeName
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*

MHandle       MName           MTypeHnd    MTypeName
0xb10018      spectrum         0x10004     User
0xb10089      spectrum         0x820000    ScmConfig
0xb1008d      spectrum         0x820000    ScmConfig

$ seek attr=0x1006e,val=

MHandle       MName        MTypeHnd      MTypeName
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
```

更多資訊：

CLI 環境變數 (位於 p. 11)
show–顯示物件 (位於 p. 46)

setjump - 儲存模型和範圍

setjump 命令會將目前模型控制代碼和目前範圍控制代碼儲存在 text_string 標籤之下。 使用者稍後可以使用 jump 命令搭配 text_string，將目前模型控制代碼和目前範圍控制代碼設回儲存在 text_string 之下的控制代碼。如果相同的 text_string 用於兩個 setjump 命令中，則系統會提示使用者進行驗證。

系統會針對已連線至 CLI 本機伺服器的每個工作階段，維護個別的 setjump 值。 setjump 命令只會針對列舉該命令的工作階段，保留資訊 (也就是工作階段指派的 setjump 文字字串)。

此命令的格式如下：

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

-n

若已使用 setjump 命令指定 -n (不提示) 選項，則系統不會提示之前是否已使用該 text_string。

- 若使用新的 <text_string> 輸入 setjump 並已存在目前模型，則會顯示下列訊息：

  ```
  model <current_model_handle> and landscape
  <current_landscape_handle> stored under <text_string>
  ```

  其中 <current_model_handle> 是目前模型的控制代碼，而 <current_landscape_handle> 是目前範圍的控制代碼。

- 若使用新的 <text_string> 輸入 setjump 但目前模型不存在，則會顯示下列訊息：

  ```
  model undefined and landscape <current_landscape_handle>
  stored under <text_string>
  ```

- 若使用在先前 setjump 命令中定義的 text_string 輸入 setjump，則會顯示下列訊息：

  ```
  setjump model: <text_string> already used. Overwrite?
  ```

  有效回應包含 y、yes、Y、Yes、n、no、N 和 No。

範例：setjump

```
$ current mh=0x400142
current model is 0x400142
current landscape is 0x400000
$ setjump -n tutorial
model 0x400142 and landscape 0x400000 stored under tutorial
```

更多資訊：

current - 設定模型或範圍 (位於 p. 36)
jump - 跳轉至已儲存的模型或範圍 (位於 p. 40)

show–顯示物件

若要顯示物件，請使用 show 命令。

此命令的格式如下：

```
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

若已指定 -a (全部) 選項，則 show alarms 不會執行任何遮罩，而顯示所有的「重大」、「主要」、「次要」、「維護」、「已隱藏」和「初始」警報。

-x

若已指定 -x (展開) 選項 (並已設定 $SPECROOT 變數)，則 show alarms 命令的輸出會在輸出結尾處顯示可能原因的文字。show events 命令的輸出會顯示事件格式。 show events -x 命令所顯示的字元數是由 .vnmshrc 資源檔案參數所控制。

max_show_event_length

如果您使用僅限 SpectroSERVER 的工作站來執行 CLI，則 -x 選項不會提供正常警報原因或事件格式資訊，這是因爲 SG-Support 目錄中沒有 CsPCause 和 CsEvFormat 檔案。 可能的錯誤訊息如下：

- 沒有可用的原因資訊 (與 show alarms 相關聯)
- 沒有可用的事件格式資訊 (與 show events 相關聯)

若要解決此問題，請在 SpectroSERVER 工作站上將 SG-Support/CsPCause 和 SG-Support/CsEvFormat 目錄和檔案複製到 <*$SPECROOT*>/SG-Support 目錄。

預設值：512

-e

若已指定 -e (列舉) 選項，則 show attributes 命令的輸出會顯示資料庫列舉字串。

-c

若已指定 -c (讀取最近期) 選項，則 [讀取模式] 屬性會設定爲 [讀取最近期]。 此模式會使用來自最近使用者介面輪詢的屬性值 (每 5 秒更新一次)。若未設定此旗標，則會使用 [讀取最可用] 模式。此模式使用上次 CA Spectrum 輪詢最近儲存在資料庫中的值。 輪詢頻率爲使用者定義的間隔。

-n

若已指定 -n (事件數) 選項，則 show events 命令的輸出會顯示指定的事件數。

-t

若已指定 -t (問題工單 ID) 選項，則 show alarms 命令的輸出會顯示 [問題工單 ID] 欄位。

-s

若已指定 -s (影響嚴重性) 選項，則 show alarms 命令的輸出會顯示 [影響嚴重性] 欄位。

爲了讓 show alarms 和 show events 命令可與 -x 選項 (該選項會顯示警報的可能原因訊息和展開的事件訊息) 搭配使用，OneClick 必須安裝於本機伺服器上，而且 SPECROOT 環境變數必須設定爲 Spectrum Support 根目錄的路徑。例如，若 SG-Support 檔案位於 /usr/spectrum/SG-Support 中，請將 SPECROOT 設定爲 /usr/spectrum。

更多資訊：

啓動檔案 (位於 p. 13)
current - 設定模型或範圍 (位於 p. 36)
seek–尋找模型 (位於 p. 41)

show alarm

show alarms 命令可顯示 model_handle 所指定模型的所有警報。 或者，僅顯示 landscape_handle 所指定範圍中各模型的最嚴重警報 (若爲重大、主要或次要警報)。 若已指定 landscape_handle，則 show alarms 會遮蔽任何具有初始、已隱藏或維護警報的模型。 因此，只會顯示具有重大、主要或次要警報的模型。 若 model_handle 和 landscape_handle 均未指定，show alarms 也會執行遮罩，而且僅顯示目前範圍中各模型的最嚴重警報 (若爲重大、主要或次要警報)。

[認可] 欄位表示警報是否已獲認可。 此欄位的可能值爲 [是] 和 [否]。[過時] 欄位表示警報是否過時。 此欄位的可能值爲 [是] 和 [否]。[指派] 和 [狀態] 欄位分別顯示警報疑難排解員資訊和警報狀態。 警報建立時間會以 hh:mm:ss 格式顯示。

show alarms 命令會以下列格式顯示資訊：

```
Id  Date       Time     PCauseId  MHandle  MName  MTypeName  Severity  Ack  Stale  Assignment  Status
id  mm/dd/yyyy  hh:mm:ss  cause_id  handle   name   name       severity  ack  stale  assignment  status
```

若搭配使用 show alarms 與 -x 選項，則最後一個警報之後會顯示原因代碼和可能原因文字訊息。 例如：

0x10402 DUPLICATE PHYSICAL ADDRESS0x10302 SpectroSERVER has lost contact with this device.

附註：show 命令顯示的模型名稱最多 16 個字元。 但是，利用環境變數 CLIMNAMEWIDTH，您可以指定顯示不同字元數 (最多 1024 個) 的模型名稱。

範例：show alarms

show alarms 命令會以下列格式顯示資訊：

```
$ show alarms lh=0x110000
ID   Date        Time      PCauseId   MHandle     MName     MTypeName  Severity   Ack  Stale
Assignment  Status
928  2000/5/11  02:33:22  0x10c04    0x110000c   infinity  VNM        CRITICAL    No   No
McDonald    Working on it
```

更多資訊：

CLI 環境變數 (位於 p. 11)
命令說明 (位於 p. 29)

show association

show associations 命令顯示針對具有 model_handle 的模型定義的所有具現化關係 (關聯)。 若未指定 model_handle，則 show associations 會顯示目前模型的所有具現化關係。

show associations 命令會以下列格式顯示資訊：

```
LMHandle   LMName   Relation   RMHandle   RMName
handle     name     relation   handle     name
```

範例：show associations

show associations 命令會以下列格式顯示資訊：

```
$ show associations mh=0x400141

LMHandle   LMName     Relation        RMHandle   RMName
0x400001   LostFound  Lost_and_Found  0x400141   12.77-bridge
```

show attributes

show attributes 命令會針對具有 model_handle 的模型，顯示 attr=attribute_id 所指定的屬性。 若未指定 attribute_id，則 show attributes 會針對具有 model_handle 的模型，列出所有屬性與其值。

若未指定 model_handle，則 show attributes 會顯示目前模型的所有適用屬性。 您可以使用 attr=low_attr-high_attr 指定某個範圍的屬性。 顯示特定模型的單一屬性或屬性清單時，即可在 instance_id 中指定屬性的例項 ID。 instance_id 必須是以句號分隔的正整數序列。您只能針對清單屬性指定例項 ID。 清單屬性就是已設定清單旗標的屬性。

下列規則適用於清單屬性：

- 若要顯示清單屬性的所有屬性值與例項 ID，請勿將 instance_id 與 attribute_id 一同輸入。 僅輸入 attribute_id 即可。
- 若要顯示清單屬性的第一個屬性值與例項 ID，請在 attribute_id 之後輸入下列命令：

  ```
  ,next
  ```

- 若要顯示清單屬性的特定屬性值與例項 ID，請輸入 instance_id 和 attribute_id。
- 若要在清單屬性的特定例項 ID 之後顯示下一個屬性值與例項 ID，請在 instance_id 之後輸入下列命令：

  ```
  ,next
  ```

- 顯示模型的所有屬性時，無法指定例項 ID，原因如下：
  - 例項 ID 僅適用於清單屬性 (例如，中樞的電路板和連接埠屬性)
  - 模型的某些屬性之例項 ID 可能與相同模型內其他屬性的例項 ID 不同。
- show attributes 命令可顯示 landscape_handle 所指定範圍中，model_type_handle 的所有屬性 (依照 ID、名稱、類型和旗標)。 若未指定 landscape_handle，則此命令會顯示在目前範圍中定義的所有模型類型。 [旗標] 欄位會列出目前設定之各屬性旗標的縮寫 (以逗號分隔)。 若未設定旗標，其縮寫不會包含在清單中。

  以下清單包含屬性旗標與其縮寫：
  - 外部 = E
  - 可讀取 = R
  - 可寫入 = W
  - 共用 = S
  - 清單 = T
  - 保證 = G
  - 全域 = O

- 記憶體 = M
- 資料庫 = D
- 已輪詢 = P
- 已記錄 = L
- 保留值 = V

附註：如需屬性旗標的詳細說明，請參閱*模型類型編輯器 使用者指南*.

show attributes 命令會以下列格式顯示資訊：

```
Id     Name     Iid     Value
id     name     iid     value
```

show attributes mth 命令會以下列格式顯示資訊：

```
Id     Name     Type     Flags
id     name     type     flags
```

範例：show attributes

```
$ show attributes mh=0xcd00011

Id         Name                        Iid     Value
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         Name                        Iid     Value
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         Name                        Iid     Value
0x10004    Contact_Status                      Established
0x110ed    Dev_Contact_Status                  2
0x111a56   ContactStatusEventSwitc             FALSE
```

```
$ show attributes attr=0x1006e mh=0x400165

Id         Name                        Iid     Value
0x1006e    Model_Name                          142.77

$ show attributes attr=0x100d4 mh=0x400165

Id         Name                        Iid     Value
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         Name                        Iid     Value
0x100d4    If_out_Ucast_Pkts            1      1169589

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

Id         Name                        Iid     Value
0x100d4    If_Out_Ucast_Pkts            2      1227569

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

Id         Name                        Iid     Value
0x100d4    If_out_Ucast_Pkts            3      1227573

$ show attributes mth=0x10004 lh=0xd00000

Id         Name                        Type            Flags
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         Name                        Type            Flags
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         Name                      Type            Flags
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         Name                      Iid      Value
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 命令會顯示具有 model_handle 模型的相關子項。 若未指定關係，則 show children 會顯示所有關係中的子項。 若未指定 model_handle，則此命令會顯示目前模型的子項。

show children 命令會以下列格式顯示資訊：

```
MHandle    MName    MTypeHn   MTypeName   Relation
handle      name     handle    name        relation
```

範例：show children

```
$ show children mh=0x400009

MHandle    Name    MTypeHnd   MTypeName      Relation
0x40000d   12.84   0x100d6    Bdg_CSI_CNB2   Collects
```

show devices

show devices 命令會顯示 landscape_handle 所指定的範圍中所有裝置模型的清單。

show devices 命令會以下列格式顯示資訊：

```
MHandle   MName   MTypeHnd   MTypeName
Handle    Name    Handle     Name
```

範例：show devices

```
$ show devices lh=0x400000

MHandle     MName           MTypeHnd    MTypeName
0x1005c0    10.253.32.101   0x3d002     GnSNMPDev
0x100030    10.253.2.10     0x2c60021   RstonesSwRtr
```

show enumerations

show enumerations 命令會顯示指定之對應列舉值的列舉字串值對應。

show enumerations 命令會以下列格式顯示資訊：

```
Id   String   Value
id   string   value
```

show enumerations mth 命令會以下列格式顯示資訊：

```
MHandle   String   Value
HandLE    string   value
```

範例：show enumerations

```
$ show enumerations attr=0x10004

ID        String        Value
0x10004   Lost          0
0x10004   Established   1
0x10004   INITIAL       2

$ show enumerations mth=0x10004

ID        String        Value
0x10004   Lost          0
0x10004   Established   1
0x10004   INITIAL       2
```

show events

show events 命令會顯示在 landscape_handle 指定的範圍中，具有 model_handle 模型的事件或所有模型的事件。 show events 命令預設會針對 model_handle 或 landscape_handle 所指定的模型，顯示最近 2,000 個事件。若已指定 -a 選項，此命令會針對 model_handle 或 landscape_handle 所指定的模型，最多顯示 10,000 個事件。

若已使用明確 no_events 陳述式指定 -n 選項，則會針對 model_handle 或 landscape_handle 所指定的模型，顯示指定的事件數。 若 model_handle 和 landscape_handle 均未指定，則此命令會顯示目前範圍中所有模型的事件。 若已指定 -x 選項，則 CLI 會顯示文字訊息，以說明事件類型。 事件時間會以 hh:mm:ss 格式顯示。

show events 命令會以下列格式顯示資訊：

```
Date          Time        Type      MHandlE     MName     MTypeName
mm/dd/yyyy    hh:mm:ss    type      handle      name      name
```

如果搭配使用 show events 與 -x 選項，則顯示的事件沒有固定格式。 以下是典型輸出範例：

```
Thur 11 May, 2000 - 8:04:01 - Alarm number 10 generated for device AntLAN of type
LAN_802_3.
Current condition is INITIAL(DEFAULT).
(event [00010701])
```

範例：show events

```
$ show events lh=0x400000

Date          Time        Type       MHandle     MName     MTypeName
1999/4/25     13:27:38    0x10302    0x4000f9    1.3       Host_IBM
1999/4/25     13:27:38    0x10202    0x400131    qa1sgi    Host_SGI

$ show events -n

Date          Time        Type       MHandle              MName              MTypeName
1999/8/21     11:30:02              0x100090xcd00067     els100-01.india    RMONApp
1999/8/21     11:25:33              0x100090xcd00067     els100-01.india    RMONApp
1999/8/21     11:20:17              0x100090xcd00067     els100-01.india    RMONApp
1999/8/21     11:15:52              0x100090xcd00067     els100-01.india    RMONApp
1999/8/21     11:10:27              0x100090xcd00067     els100-01.india    RMONApp
```

show inheritance

show inheritance 命令會顯示在範圍控制代碼指定的範圍中，model_type_handle 所指定模型類型的模型類型繼承。 若未指定 landscape_handle，則會使用目前的範圍。 此欄位的可能值爲 [基礎] 或 [衍生]。

show inheritance 命令會以下列格式顯示資訊：

```
MHandle    MName    Flags    Inheritance
handle     name     flags    inheritance
```

範例：show inheritance

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

Handle    Name        Flags    Inheritance
0x10000   Root        V,D      Base
0x103ad   BanVinesFS  V,I,U    Derived
```

show landscapes

show landscapes 命令顯示爲每個 SpectroSERVER 定義的所有範圍。顯示的範圍對應就是初始 SpectroSERVER 的對應。

show landscapes 會以下列格式顯示資訊：

```
SSName    Precedence    Port    Service    LHandle
ssname    precedence    port    service    handle
```

範例：show landscapes

```
$ show landscapes

SSName    Precedence    Port    Service   LHandle
devsgi    10            0xbeef  0x10101   0x28000000
devibm    10            0xbeef  0x10101   0x11f00000
```

show models

show models 命令會顯示在 landscape_handle 所指定範圍中定義的所有模型。 若未指定 landscape_handle，則此命令會顯示在目前範圍中定義的所有模型。 以下命令可指定模型控制代碼範圍：

```
mhr=low_model_handle-high_model_handle
```

指定 mname=model_name 即可搜尋特定模型。

show models 命令可將使用者模型識別爲 (使用中) 或 (非使用中)。 如果使用者模型狀態爲 (非使用中)，則使用者尚無法連線至伺服器。 當使用者模型狀態爲 (使用中) 時，使用者即可連線至伺服器。

show models 命令會以下列格式顯示資訊：

```
MHandle    MName    MTypeHnd    MTypeName
handle     name     handle      name
```

範例：show models

```
$ show models lh=0x400000

MHandle    MName    MTypeHnd    MTypeName
0x400004   World    0x10040     World
0x4000d9            0x10020     AUI

$ show models mname=

MHandle       MName        MTypeHnd      MTypeName
0xcd00016   0x1120002    AppDataServer
0xcd00022   0x1006b      SnmpPif
0xcd00030   0x1028f      IcmpPif

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

MHandle     MName                        MTypeHnd    MTypeName
0xcd000a3   hplaser.zeitnet.India.com    0x230018    TCP2_App
0xcd0002b   desire.zeitnet.India.com     0x230018    TCP2_App
```

show parents

show parents 命令會顯示具有 model_handle 模型的相關父項。 若未指定關係，則會顯示所有關係中的父項。 若未指定 model_handle，則 show parents 會顯示目前模型的父項。

show parents 命令會以下列格式顯示資訊：

```
MHandle    MName    MTypeHnd    MTypeName    Relation
handle     name     handle      name         relation
```

範例：show parents

```
$ show parents mh=0x40000d

MHandle    MName              MTypeHnd    MTypeName      Relation
0x400009   auto-lan-30x1003c              LAN_802_3    Collects
```

show relations

show relations 命令會顯示在 landscape_handle 所指定範圍中目前定義的所有關係。 若未指定 landscape_handle，則此命令會顯示在目前範圍中定義的所有關係。

show relations 命令會以下列格式顯示資訊：

```
Name             Type
relation_name    relation_type
```

範例：show relations

```
$ show relations

Name Type
Passes_Through MANY_TO_MANY
Lost_and_Found ONE_TO_MANY
Owns ONE_TO_MANY
Contains ONE_TO_MANY
```

show rules

show rules 命令會顯示關係的規則。 此關係指定於 landscape_handle 指定的範圍中。 若未指定 landscape_handle，則會使用目前的範圍。

show rules 命令會以下列格式顯示資訊：

```
LMTHandle    LMTName    RMTHandle    RMTName
handle       name       handle       name
```

範例：show rules

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

LMTHandle    LMTName    RMTHandle    RMTName
0x102da      Org_Owns   0x10043      Site
0x102da      Org_Owns   0x210023     Rtr_CiscoMGSshow
```

show types

show types 命令會顯示在 landscape_handle 所指定範圍中目前定義的所有模型類型。 若未指定 landscape_handle，則此命令會顯示在目前範圍中定義的所有模型類型。 [旗標] 欄位會列出目前設定之六個屬性旗標的縮寫。 若未設定旗標，其縮寫不會包含在清單中。

以下清單包含模型類型旗標與其縮寫：

- 可見 = V
- 可具現化 = I
- 可衍生 = D
- 不終結 = N
- 唯一 = U
- 必要 = R

show types 命令 [mth=low_mth-high_mth] 會顯示 low_mth 與 high_mth 之間範圍內的所有模型類型。

附註：如需模型類型旗標的詳細資訊，請參閱*模型類型編輯器 使用者指南*.

show types 命令會以下列格式顯示資訊：

```
Handle    Name    Flags
handle    name    flags
```

範例：show types

```
$ show types lh=0x400000

Handle    Name          Flags
0x10000   Root          V,D
0x10080   Gen_Rptr_Prt  V,D

$ show types mthr=0x10002-0x10008

Handle    Name             Flags
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

Handle      Name             Flags
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

Handle        Name             Flags
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 命令會列出 model_handle 所指定模型的適用監看資料。

若未指定 landscape_handle 和 model_handle，show 命令會使用下列預設值：

命令	預設值
show alarms	current landscape
show associations	current model
show attributes	current model
show attributes mth	current landscape
show children	current model
show devices	current landscape
show enumerations	current landscape
show enumerations mth	current landscape

命令	預設值
show events	current landscape
show inheritance	current landscape
show models	current landscape
show parents	current model
show relations	current landscape
show rules	current landscape
show types	current landscape
show watch	current model

附註：'show alarms' 和 'show events' 命令均可搭配 x 選項使用，以顯示警報的可能原因訊息和展開的事件訊息。

驗證下列先決條件：

- OneClick 已安裝於本機伺服器上。
- 環境變數 SPECROOT 已設定爲根目錄 (SG-Support) 的路徑。

  例如，若 SG-Support 檔案位於 /usr/spectrum/SG-Support 中，請將 SPECROOT 設定爲 /usr/spectrum。

show watch 命令會以下列格式顯示資訊：

```
Watch_Id   Watch_Name   Watch_Type   Watch_Status
watch_id   watch_name   watch_type   watch_status
```

範例：show watch

```
$ show watch mh=0xc600015

Watch_Id     Watch_Name   Watch_Type   Watch_Status
0xffff0001   watch798     Calc         Active
```

stopShd–終止 CLI 本機伺服器

使用 stopShd 命令來終止 CLI 本機伺服器 (VnmShd 精靈)。 stopShd 命令會中斷所有 CA Spectrum CLI 使用者與目前連線之 SpectroSERVER 的連線，並終止 CLI 本機伺服器。 此命令會在中斷使用者連線並關閉伺服器前，提示使用者進行確認。 (您也可以使用 kill -2 命令來關閉精靈。)

此命令的格式如下：

```
stopShd [-n]
```

-n

指定「不提示」。 將此選項放入 stopShd 命令中，以停用確認提示。

否則，一律會顯示以下訊息：

```
stopShd: n users are connected, are you sure?
```

'n' 表示已連線的使用者數目 (包含您自己)。

有效回應包含 y、yes、Y、Yes、n、no、N、No。

如果命令成功，則會顯示以下訊息：

```
stopShd: successful
```

當 stopShd 終止 CLI 本機伺服器時，系統主控台上會顯示以下訊息：

```
VnmShd: stopShd executed. Exiting...
```

範例：stopShd

```
$ stopShd
stopShd: 2 users are connected, are you sure? y
stopShd: successful
```

更多資訊：

disconnect–Disconnects from SpectroSERVER (位於 p. 39)

update–更新模型類型和模型類型屬性

使用 update 命令來更新模型和模型類型屬性。

此命令的格式如下：

```
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

若已使用 update 命令指定 -n (不提示) 選項，則系統不會顯示確認提示。在 CLI 指令檔中，這個選項很實用。

-r

使用狀態或工單引數時，可以使用 update alarm 命令指定 -r (取代狀態文字/取代問題工單 ID) 選項。若已使用 -r 選項，則會以狀態引數或工單引數所指定的文字取代現有警報狀態文字或警報問題工單 ID。若未使用 -r 選項，則會將新的值附加到現有的值。

action_code

- reconfig、0x1000e 或 65550 可重新配置模型
- activate、0x00480003 或 4718595 可啓動監看
- deactivate、0x00480004 或 4718596 可停用監看
- reconfigure_apps、0x210008 或 2162696 可在 Cisco 和 Wellfleet 裝置上重新配置應用程式模型
- reload_event_disp、0x000100a2 或 65698 可使用 EventDisp 和 AlertMap 檔的變更來更新 SpectroSERVER

附註：watch = *<watch_id>* 參數僅適用於下列動作：activate (或十六進位對等項目 0x00480003) 和 deactivate (或十六進位對等項目 0x00480004)。

下列各點描述 update 命令的功能：

- update 命令會針對 landscape_handle 所指定範圍中，具有 model_handle 的模型，或模型類型爲 model_type_handle 的所有模型，更新 attribute_id 值所指定的屬性。
- 指定多個「attribute_id、value」配對 (以空格分隔相鄰的配對)，即可使用一個 update 命令更新多個屬性。
- remove 選項可移除從清單屬性指定的例項。
- 若未指定 landscape_handle，則更新模型類型屬性時會使用目前的範圍。若未指定 model_handle，則會更新目前模型的已指定屬性。
- 當您更新模型類型屬性時，請記得只能更新共用屬性。 共用屬性就是已設定共用旗標的屬性。使用 show attributes 命令來查看是否已共用屬性。
- 可以透過 CLI 更新安全性相關屬性 (例如 User_Community_String 和 Model_Security_String)。 但是，目前使用者模型不能更新自己的 User_Security_String 或 Security_String，但可以更新其他模型的這些屬性。
- update 命令還可讓使用者在變更單一屬性值時指定例項 ID。更新屬性值清單時，可以針對清單上的每個屬性指定例項 ID。 instance_id 是對應屬性的例項 ID。instance_id 必須是一個正整數，或以點分隔的正整數序列。
- 若未指定例項 ID，則 update 命令會使用它爲屬性找到的第一個有效例項。 若找不到有效例項，則會顯示錯誤訊息：

  ```
  update: no valid instance for list attribute <attr_id>
  ```

- update alarm 命令會使用「疑難排解員」所指定的值 (「疑難排解員」模型控制代碼或「疑難排解員」名稱)、status_text、troubleticketID 或 ack 參數，更新 alarm_id 所指定的警報。 若要清除疑難排解員、狀態文字或問題工單 ID 的現有警報值，您可以將適當參數設定成沒有值 (status=、ticket= 或 assign=)。 landscape_handle 參數會指定將會找到警報的範圍。

- update action 命令可在 model_handle 指定的裝置上執行 action_code 所指定的動作。 利用 action_code 重新配置，可以重新配置模型類型 GnSNMPDev 的任何裝置，或重新配置任何繼承自 GnSNMPDev 之模型類型的任何裝置。 activate 或 deactivate action_code 可更新指定 model_handle 之裝置上的監看狀態。 若已傳送 activate 動作物件，則監看狀態從 INITIAL 變更爲 ACTIVE 之間的時間可能會有短暫延遲 (視選取的模型中內建的情報而定)。 使用 show watch 命令可以取得預定更新狀態的監看 watch_id。 reconfigure_apps 的 action_code 可更新 Cisco 和 Wellfleet 裝置模型的應用程式模型類型。 reload_event_disp 的 action_code 可使用 EventDisp 或 AlertMap 檔的變更來更新 SpectroSERVER。
- 使用 update action 命令時，請格外小心。 若您使用 CLI 命令的方式不正確，可能會毀損 SpectroSERVER 資料庫。 例如，不小心重新配置重要路由器，可能會對您的網路造成無法預期的結果。
- 若使用有效的 model_handle 或有效的 model_type_handle、有效的 attribute_id 和有效值輸入 update，則會以下列格式顯示修改後的屬性與其值：

  ```
  Id  Name    Value
  Id  Name    Value
  ```

- 如果在更新指定之模型類型的模型時，您未使用 -n 選項，則會顯示下列確認訊息：

  ```
  update: all models of this type will be updated, are you sure?
  ```

  有效回應包含 y、yes、Y、Yes、n、no、N、No。
- 如果 update alarm 命令成功，則會顯示以下訊息：

  ```
  update:successful
  ```

- 如果 update action 命令成功，則會顯示以下訊息：

  ```
  update action: successful
  ```

範例：Update

- 在以下範例中，具有 instance_id 的 update 命令用於停用模型控制代碼 0x4001f6 所代表中樞的電路板 5 上的連接埠 7：

```
$ update mh=0x4001f6 attr=0x10ee0,iid=5.7,val=1
Id        Name              Iid       Value
0x10ee0   CsPortAdminState            1
```

- 在以下範例中，update 命令會與 remove 選項搭配使用，以從特定模型 (mh) 的 deviceIPAddressList (attr) 中移除 IP 位址 (iid)。

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

- 在以下範例中，update 命令用於更新模型類型控制代碼 0x10059 所代表模型類型的所有模型上名爲 AutoPlaceStartX 的屬性。

```
$ update mth=0x10059 attr=0x118f2,val=100 lh=0x400000
update: all models of this type will be updated, are you sure? y
Id    Name              Value
Id    AutoPlaceStartX   100
```

- 在以下範例中，update alarm 命令用於更新警報疑難排解員指派。

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

- 在以下範例中，update alarm 命令用於更新警報狀態。 -r 選項用於覆寫現有狀態。

```
$ update alarm -r aid=928 status='Working on it'
update: successful
```

- 在以下範例中，update alarm 命令用於更新警報問題工單 ID。 -r 選項用於覆寫問題工單 ID 的現有值。

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

- 在以下範例中，update alarm 命令用清除問題工單 ID 的現有值。

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

- 在以下範例中，update alarm 命令用於認可警報。

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

- 在以下範例中，update 命令用於限制 User_Community_String 的更新。

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

- 在以下範例中，update action 命令用於重新配置 Cisco 路由器。

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

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

Watch_Id      MHandle      Watch_Status
0xffff0001    0xc600015    INITIAL

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

Watch_Id      MHandle      Watch_Status
0xffff0001    0xc600015    INACTIVE
```

附錄 A： 範例指令檔

範例指令檔概觀

CLI 隨附的範例指令檔示範如何將 CLI 命令併入 UNIX shell 指令檔中，以便自動執行 CLI 工作階段。 您會發現某些指令檔或這些指令檔中的某些函數對於您的工作很有幫助。

CLI 在 <*$SPECROOT*>/vnmsh/sample_scripts 目錄中包含下列指令檔，以及描述這些指令檔的讀我檔案：

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

使用 CLI 指令檔時，請檢閱下列先決條件：

- 每個指令檔都具有名爲 CLIPATH 的內部變數。 若要使用指令檔，請將 CLIPATH 變數設定爲 CLI 可執行檔所在目錄的路徑名稱。
- 根據指令檔的執行方式而定，CLIPATH 變數和其他屬於路徑名稱的環境變數可以是*完整*或*相對*路徑名稱。當您將範例指令檔當作 cron 指令檔執行時，使用 CLIPATH 和其他環境變數的*完整*路徑名稱。 否則，您可以使用這些變數的*相對*路徑名稱。
- 除了 update_mtype 以外，您可以將所有 CLI 指令檔當作 cron 指令檔執行。

  附註：請勿將 update_mtype 當作 cron 指令檔執行，因爲它會提示使用者輸入資料。
- 當您執行 CLI 指令檔時，請在 .vnmshrc 檔案中指定 vnm_hostname 變數的正確名稱。

active_ports 指令檔

使用 active_ports 指令檔識別 IRM2 中樞的各電路板的所有連接埠，以及識別各電路板上的使用中連接埠。

active_ports 指令檔會將名爲 hub_name 之中樞的報告放入名爲 output_file 的檔案中。此報告會列出各電路板的所有連接埠。報告的 ON 欄中的星號 (*) 表示使用中連接埠。

此指令檔的格式如下：

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

app_if_security 指令檔

使用 app_if_security 指令檔來更新 CA Spectrum 資料庫中所有介面和應用程式模型中的 Security_String 屬性值。 app_if_security 指令檔會從上層模型複製此屬性值，以進行更新。 如果接收者模型 (子項) 已具有 Security_String 屬性值，或父項*沒有* Security_String 屬性值，則此指令檔不會更新任何模型。更新模型安全性字串後，管理員即可使用此指令檔來更新模型子項的安全性字串。

此指令檔的格式如下：

```
app_if_security
```

cli_script 指令檔

當您提供資料檔作爲輸入時，請使用 cli_script 以批次模式執行大多數的 CLI 命令。 CLI 範例資料檔 (名爲 datafile) 包含表示要執行之命令的參數，以及要傳遞至命令的必要參數。 此指令檔會驗證每個命令是否執行成功，而且會維護執行階段記錄。

此指令檔的其中一個優點就是您可以使用名稱 (而非控制代碼) 建立批次檔。例如，您可以使用模型類型名稱，而非使用十六進位模型類型控制代碼。 檔案因此變得更加容易建立和讀取，而且當您想對所建立的模型執行後續動作時，您就會發現真正的優點。 您不需將十六進位模型控制代碼指派給模型，即可依照名稱參照模型。

此指令檔的格式如下：

```
cli_script datafile
```

cli_script 會使用同樣位於 sample_scripts 目錄中的兩個檔案 (datafile 和 clean.awk)。

資料檔案

包含 cli_script 的輸入。 還包含目前在 cli_script 中實作的每個 CLI 命令。 請參閱 cli_script 標頭資訊，以取得此檔案的格式和語法相關指示。

clean.awk

包含用於執行中的輸入。 .awk 檔案用於格式化對主控台顯示的資料。

使用 cli_script 時，請考量下列幾點：

- 請記得將範例 datafile 中的「虛設的」Network_Address (255.255.255.255) 變更爲真實位址。
- 如果您將 cli_script 移至其他目錄，則必須更新 SPECROOT 環境變數才能支援根目錄 (SG-SUPPORT)。

  例如，若 SG-SUPPORT 檔案位於 /usr/spectrum/SG-Support 中，請將 SPECROOT 設定爲 /usr/spectrum。

database_tally 指令檔

使用此指令檔來判斷目前資料庫中各個類型的模型有多少。 管理員在評估系統效能時，會發現這個指令檔很實用。 此指令檔會顯示資料庫中所有模型類型的清單以及各個模型類型的模型數目。

此指令檔的格式如下：

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

update_mtype 指令檔

使用此指令檔來更新某個模型類型之所有模型的特定屬性。 如果該屬性是模型類型的共用屬性，則指令檔不會更新模型的屬性。 此指令檔的其中一個優點就是您可以在提示字元使用模型和屬性名稱，而非使用十六進位 ID 控制代碼。

附註：請將 CLIMNAMEWIDTH 環境變數設定爲 256。 較大的值可防止截斷模型名稱，以免在執行指令檔時發生比對錯誤。

此指令檔的格式如下：

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

model_name | model_type_name

針對已完成屬性更新之模型類型的模型，指定模型名稱 (或部份的模型名稱)。 您可以在命令中指定此模型類型的任何模型。

此指令檔接著顯示所有模型類型的清單，而這些模型類型具有名稱含有您輸入之模型名稱引數的模型。 指令檔會要求您從清單中選取模型類型。

如果您使用模型名稱而非模型類型名稱，則指令檔會更新名稱包含在命令列或在提示字元輸入之字串的所有模型。 在此情況下，並不會如上所述更新指定模型類型的所有模型。

附註：我們建議在提示字元使用模型名稱或模型類型名稱，而非使用控制代碼。

attribute_name | attribute_id

如果最初並未指定這些引數，則指令檔會在執行時提示指定屬性名稱或屬性 ID。 此時，您必須指定屬性名稱或部份的屬性名稱。 然後，指令檔會要求您從含有所輸入文字的屬性清單中進行選取。 因此，您不需事先知道十六進位模型類型控制代碼或屬性控制代碼，即可執行整個指令檔。

更多資訊：

CLI 環境變數 (位於 p. 11)

active_ports 指令檔

使用此指令檔將 Octet_String 格式 XX.YY.ZZ 轉換爲其 ASCII 字串表示法。

附錄 B： 錯誤訊息

ack alarm：<alarm_id>：警報 ID 無效

原因：

您輸入的警報 ID 無效。

動作：

使用有效的 alarm_id，再次輸入 ack alarm 命令。

ack alarm: <landscape_handle>: invalid landscape handle

原因：

您爲警報輸入的範圍控制代碼無效。

動作：

使用有效的 landscape_handle，再次輸入 ack alarm 命令。

command：無法連線至 VnmShd，請先執行命令

原因：

您嘗試在執行 connect 命令之前執行其他命令。

動作：

透過 connect 命令開始 CLI 工作階段。

connect: already connected to <主機名稱> since <日期/時間>

原因：

沒有必要嘗試連線。 您已經連線至 SpectroSERVER 主機。

動作：

無。

connect: cannot open resource file <路徑名稱>/.vnmshrc

原因：

connect 命令找不到 CLI 資源檔 .vnmshrc。

動作：

.vnmshrc 資源檔必須位於與 connect 命令相同的目錄中。

更多資訊：

啓動檔案 (位於 p. 13)

connect: can only connect to SpectroSERVERs in <主機名稱> landscape map - other user(s) already connected

原因：

connect 命令已經用於連線到特定 SpectroSERVER。您只可以連線到位於原始 SpectroSERVER 的範圍對應中的 SpectroSERVER。

動作：

無

connect: ERROR: No such CA Spectrum user as <使用者名稱>

原因：

connect 命令的第一個使用者並未定義爲 CA Spectrum 使用者。

動作：

以 CA Spectrum 使用者身分重新連線至 SpectroSERVER。

connect: <主機名稱> not responding or not permitting access

原因：

因爲主機名稱不正確、SpectroSERVER 不在執行中，或使用者沒有使用者模型，所以 connect 命令無法連線至 SpectroSERVER。

動作：

驗證主機名稱是否正確、SpectroSERVER 是否正在執行中，以及使用者是否具有使用者模型。

connect: <landscape_handle>: invalid landscape handle

原因：

使用者指定的 landscape_handle 對於指定的主機名稱無效，或您的 VNM 無法存取控制代碼。

動作：

驗證 landscape_handle 是否對於指定的主機名稱有效，以及驗證您的 VNM 是否可以存取控制代碼。

connect: incompatible SpectroSERVER <版本>

原因：

使用者正嘗試連線至其版本與 CLI 版本不相容的 SpectroSERVER 主機。

動作：

更新您所使用的 CLI 版本。

connect: invalid <值> for CLISESSID

原因：

connect 命令已使用於 cron 指令檔中，或視窗化系統對 ttyslot 傳回 0，且 CLISESSID 環境變數已設定爲非數值。

動作：

在 cron 指令檔之外使用 connect 命令，並將 CLISESSID 設定爲數值。

connect: variable <CLISESSID> not set

原因：

您已嘗試在 cron 指令檔中使用 connect 命令，而未先設定 CLISESSID 環境變數。

動作：

在 cron 指令檔中使用 connect 時，設定 CLISESSID 環境變數。

create: user not permitted to create alarm

原因：

不允許您建立警報。

動作：

驗證您的使用者權限。

create: user not permitted to create association

原因：

不允許您建立關聯。

動作：

驗證您的使用者權限。

create: user not permitted to create event

原因：

不允許您建立事件。

動作：

驗證您的使用者權限。

create: user not permitted to create model

原因：

不允許您建立模型。

動作：

驗證您的使用者權限。

create alarm: <probable_cause_id>: invalid alarm cause

原因：

使用無效的 probable_cause_id 輸入 create alarm 命令。

動作：

使用有效的 probable_cause_id 重新輸入 create alarm 命令。

create alarm: <alarm_severity>: invalid alarm severity

原因：

使用無效的 alarm_severity 輸入 create alarm 命令。

動作：

使用有效的 alarm_severity 重新輸入 create alarm 命令。

create alarm: <model_handle>: invalid model handle

原因：

使用無效的 model_handle 輸入 create alarm 命令。

動作：

使用有效的 model_handle 重新輸入 create alarm 命令。

create association: <left_model_handle>: invalid model handle

原因：

使用無效的 left_model_handle 輸入 create association 命令。

動作：

使用有效的 left_model_handle 重新輸入 create association 命令。

create association: models belong to different landscapes

原因：

使用位於不同範圍的 left_model_handle 和 right_model_handle 輸入 create association 命令。

動作：

兩個控制代碼均使用相同的範圍。

create association: rel=<關係>: invalid relation

原因：

使用無效的關係輸入 create association 命令。

動作：

使用有效的關係重新輸入 create association 命令。

create association: <right_model_handle>: invalid model handle

原因：

使用無效的 right_model_handle 輸入 create association 命令。

動作：

使用有效的 right_model_handle 重新輸入 create association 命令。

create event: <event_type>: invalid event type

原因：

使用無效的 event_type 輸入 create event 命令。

動作：

使用有效的 event_type 重新輸入 create event 命令。

create event: <landscape_handle>: unknown landscape

原因：

使用無效的 landscape_handle 輸入 create event 命令。

動作：

使用有效的 landscape_handle 重新輸入 create event 命令。

create event: <model_handle>: invalid model handle

原因：

使用無效的 model_handle 輸入 create event 命令。

動作：

使用有效的 model_handle 重新輸入 create event 命令。

create model: <attribute_id>: invalid attribute id

原因：

因爲使用無效的 attribute_id 輸入 create model 命令，所以未建立模型。

動作：

使用有效的 attribute_id 重新輸入 create model 命令。

create model: DCM device unreachable

原因：

因爲使用無效的 ip_address 輸入 create model 命令，所以未建立模型。 DCM (裝置通訊管理員) 發出此錯誤訊息。

動作：

使用有效的 ip_address 重新輸入 create model 命令。

create model: Device limit exceeded

原因：

因爲分支管理員 SpectroSERVER (50 個裝置模型限制) 或網站管理員 SpectroSERVER (250 個裝置模型限制) 包含它可容納的最大裝置模型數目，所以未建立模型。

動作：

驗證 SpectroSERVER 上的裝置模型數目是否尚未達到規定的限制。 如果已達到，您必須刪除某些裝置模型，然後重新輸入 create model 命令。

create model: <landscape_handle>: invalid landscape handle

原因：

因爲使用無效的 landscape_handle 輸入 create model 命令，所以未建立模型。

動作：

使用有效的 landscape_handle 重新輸入 create model 命令。

create model: <model_type_handle>: invalid model type handle

原因：

因爲使用無效的 model_type_handle 輸入 create model 命令，所以未建立模型。

動作：

使用有效的 model_type_handle 重新輸入 create model 命令。

create model: <value>: invalid value

原因：

因爲使用無效值輸入 create model 命令，所以未建立模型。

動作：

使用有效值重新輸入 create model 命令。

create model: <value>: No community name

原因：

在建立要求中提供的社群名稱不正確。

動作：

使用有效的社群名稱重新輸入 create 命令。

current: <model_handle>: invalid model handle current model is <current_model_handle>

原因：

指定的 model_handle 無效，所以未變更目前模型和目前範圍。

動作：
如果您想修改目前模型和目前範圍，請重新輸入有效的 model_handle。

current: <landscape_handle>: invalid landscape handle current landscape is <current_landscape_handle>

原因：

因爲指定的 landscape_handle 無效，所以未變更目前模型和目前範圍。

動作：

如果您想修改目前模型和目前範圍，請重新輸入有效的 landscape_handle。

current: <landscape_handle>: not responding or not permitting access current model is <current_model_handle>

原因：

已指定 landscape_handle，此範圍的 OneClick 已關閉，或使用者在該範圍內沒有使用者模型。

動作：

驗證有問題之範圍的 OneClick 是否在執行中；驗證您在有問題的範圍內是否有使用者模型；然後重新輸入 landscape_handle。

current: <landscape_handle>: not responding or not permitting access current landscape is <current_landscape_handle>

原因：

已指定 model_handle，含有該模型之範圍的 SpectroSERVER 已關閉，或使用者在該範圍內沒有使用者模型。

動作：

驗證有問題之範圍的 SpectroSERVER 是否在執行中；驗證您在有問題的範圍內是否有使用者模型；然後重新輸入 model_handle。

destroy: user not permitted to destroy alarm

原因：

不允許您終結警報。

動作：

驗證您的使用者權限。

destroy: user not permitted to destroy association

原因：

不允許您終結關聯。

動作：

驗證您的使用者權限。

destroy: user not permitted to destroy model

原因：

不允許您終結模型。

動作：

驗證您的使用者權限。

destroy alarm: aid=<alarm_id>: invalid alarm id

原因：

使用無效的 alarm_id 輸入 destroy alarm 命令。

動作：

使用有效的 alarm_id 重新輸入 destroy alarm 命令。

destroy alarm: <landscape_handle>: invalid landscape handle

原因：

使用無效的 landscape_handle 輸入 destroy alarm 命令。

動作：

使用有效的 landscape_handle 重新輸入 destroy alarm 命令。

destroy association: rel=<關係>: invalid relation

原因：

使用無效的關係輸入 destroy association 命令。

動作：

使用有效的關係重新輸入 destroy association 命令。

destroy association: <left_model_handle>: invalid model handle

原因：

使用無效的 left_model_handle 輸入 destroy association 命令。

動作：

使用有效的 left_model_handle 重新輸入 destroy association 命令。

destroy association: <right_model_handle>: invalid model handle

原因：

使用無效的 right_model_handle 輸入 destroy association 命令。

動作：

使用有效的 right_model_handle 重新輸入 destroy association 命令。

destroy association: association does not exist between given models

原因：

嘗試終結兩個不存在的模型之間的關聯。

動作：

驗證屬於您嘗試終結之關聯的兩個模型是否存在。

destroy association: models belong to different landscapes

原因：

使用位於不同範圍的 left_model_handle 和 right_model_handle 輸入 destroy association 命令。

動作：

使用來自相同範圍的 left_model_handle 和 right_model_handle，重新輸入 destroy association 命令。

destroy model: <model_handle>: invalid model handle

原因：

使用無效的 model_handle 輸入 destroy model 命令。

動作：

使用有效的 model_handle 重新輸入 destroy model 命令。

disconnect: failed

原因：

disconnect 命令失敗。

動作：

重試 disconnect 命令。

disconnect: failed to connect with VnmShd, please run connect first

原因：

嘗試在 CLI 本機伺服器未執行的情況下執行 disconnect。

動作：

無。 您已經中斷連線。

disconnect: not connected

原因：

因爲使用者未連線至 SpectroSERVER，所以 disconnect 命令失敗。

動作：

無。 您已經中斷連線。

jump: <text_string>: text string not defined

jump:<text_string>: text string not defined

其中 text_string1、text_string2... 是目前定義的文字字串。

原因：

使用未定義的 text_string 輸入 jump 命令。

動作：

使用任何已定義的文字字串，重新輸入 jump 命令。

<路徑名稱>/VnmShd: not found

<路徑名稱>/VnmShd: not found

connect: failed

其中路徑名稱代表 CLI 嘗試在其中執行 VnmShd 的目錄路徑。

原因：

connect 命令找不到 CLI 本機伺服器。

動作：

確定 VnmShd 和 connect 位於相同目錄，然後重新輸入 connect 命令。

Please connect first

原因：

在 connect 執行之後，您執行了 disconnect 或 stopShd，然後嘗試執行其他命令。

動作：

先重新發出 connect 命令。

seek: <attribute_id>: invalid attribute id

原因：

使用無效的 attribute_id 輸入 seek 命令。

動作：

使用有效的 attribute_id 重新輸入 seek 命令。

seek: <錯誤>: attribute not keyed

原因：

使用未鍵入屬性的 attribute_id 輸入 seek 命令。

動作：

使用已鍵入屬性的 attribute_id 重新輸入 seek 命令。

seek: <value>: invalid value

原因：

使用無效值輸入 seek 命令。

動作：

使用有效值重新輸入 seek 命令。

show attributes: <attribute_id>: non list attribute

原因：

使用非清單 attribute_id 的 instance_id 輸入 show attributes 命令。

動作：

使用清單屬性 ID 的 instance_id 重新輸入 show attributes 命令。

show attributes: <attribute_id>: invalid attribute id

原因：

使用無效的 attribute_id 輸入 show attributes 命令。

動作：

使用有效的 attribute_id 重新輸入 show attributes 命令。

show attributes: <instance_id>: invalid instance id

原因：

使用無效的 instance_id 輸入 show attributes 命令。 如果 instance_id 不是由非負整數的序列所組成，或不是針對指定的屬性而存在，則爲無效。

動作：

使用有效的 instance_id 重新輸入 show attributes 命令。

show attributes: <model_type_handle>: invalid model type handle

原因：

使用無效的 model_type_handle 輸入 show attributes 命令。

動作：

使用有效的 model_type_handle 重新輸入 show attributes 命令。

show: <landscape_handle>: invalid landscape handle

原因：

使用無效的 landscape_handle，輸入採用選擇性 landscape_handle 的 show 命令。

動作：

使用有效的 landscape_handle 重新輸入 show 命令。

show: <model_handle>: invalid model handle

原因：

使用無效的 model_handle，輸入採用選擇性 model_handle 的 show 命令。

動作：

使用有效的 model_handle 重新輸入 show 命令。

show: no current model defined

原因：

已輸入採用選擇性 model_handle 的 show associations 命令，但未指定 model_handle，且未定義目前模型。

動作：

重新輸入同時包含 model_handle 和目前模型的 show associations 命令。

show alarms: no cause information available

原因：

已搭配使用 show alarms 命令與 -x 選項，但無法取得含有可能原因文字訊息的 CA Spectrum 警報檔案。

動作：

爲了讓 show alarms 命令可與 -x 選項 (該選項會顯示警報的可能原因訊息和展開的事件訊息) 搭配使用，OneClick 必須安裝於本機伺服器上，而且 SPECROOT 環境變數必須設定爲 Support 根目錄的路徑 (SG-Support)。 例如，若 SG-Support 檔案位於下列目錄：

/usr/spectrum/SG-Support，請將 SPECROOT 設定爲 /usr/spectrum。

show children: <關係>: invalid relation

原因：

使用無效的關係輸入 show children 命令。

動作：

使用有效的關係重新輸入 show children 命令。

show events: no event format information available

原因：

已搭配使用 show events 命令與 -x 選項，但無法取得含有事件格式文字訊息的 CA Spectrum 事件檔案。

動作：

爲了讓 show events 命令可與 -x 選項 (該選項會顯示警報的可能原因訊息和展開的事件訊息) 搭配使用，OneClick 必須安裝於本機伺服器上，而且 SPECROOT 環境變數必須設定爲 Support 根目錄的路徑 (SG-Support)。 例如，若 SG-Support 檔案位於下列目錄：

/usr/spectrum/SG-Support，請將 SPECROOT 設定爲 /usr/spectrum

show parents: <關係>: invalid relation

原因：

使用無效的關係輸入 show parents 命令。

動作：

使用有效的關係重新輸入 show parents 命令。

show rules: <關係>: invalid relation

原因：

使用無效的關係輸入 show rules 命令。

動作：

使用有效的關係重新輸入 show rules 命令。

show inheritance: <model_type_handle>: invalid model type handle

原因：

使用無效的 model_type_handle 輸入 show inheritance 命令。

動作：

使用有效的 model_type_handle 重新輸入 show inheritance 命令。

stopShd: VnmShd not running

原因：

嘗試在 CLI 本機伺服器未執行中的情況下執行 stopShd。

動作：

啓動 CLI 本機伺服器。

stopShd: failed

原因：

stopShd 命令失敗。

動作：

再次嘗試連線，然後執行 stopShd。 如果沒有作用，請手動刪除 VnmShd 程序。

update: <attribute_id>: Attribute not writable

原因：

因爲嘗試更新不可寫入的模型屬性，所以未發生更新。

動作：

驗證您要更新的模型屬性是否可寫入，然後重新輸入 update 命令。

update: <attribute_id>: invalid attribute id

原因：

因爲使用無效的 attribute_id 輸入 update 命令，所以未發生更新。

動作：

使用有效的 attribute_id 重新輸入 update 命令。

update: <attribute_id>: non shared attribute

原因：

update 命令已用於模型類型，並已輸入非共用屬性的 attribute_id。

動作：

爲此模型類型重新輸入 update 命令，但這次使用共用屬性的 attribute_id。

update: <instance_id>: invalid instance id

原因：

因爲使用無效的 instance_id 輸入 update 命令，所以未發生更新。

動作：

使用有效的 instance_id 重新輸入 update 命令。

update: <landscape_handle>: invalid landscape handle

原因：

因爲使用無效的 landscape_handle 輸入 update 命令，所以未發生更新。

動作：

使用有效的 landscape_handle 重新輸入 update 命令。

update: <model_handle>: invalid model handle

原因：

因爲使用無效的 model_handle 輸入 update 命令，所以未發生更新。

動作：

使用有效的 model_handle 重新輸入 update 命令。

update: <model_type_handle>: invalid model type handle

原因：

因爲使用無效的 model_type_handle 輸入 update 命令，所以未發生更新。

動作：

使用有效的 model_type_handle 重新輸入 update 命令。

update: <value>: invalid value

原因：

因爲使用無效的值輸入 update 命令，所以未發生更新。

動作：

使用有效值重新輸入 update 命令。

update: <action_code>: invalid action code

原因：

因爲使用無效的 action_code 輸入 update 命令，所以未發生更新。

動作：

使用有效的 action_code 重新輸入 update 命令。

VnmShd: Error: Failed to connect to SpectroSERVER

原因：

CLI 本機伺服器無法連線至 SpectroSERVER。

動作：

驗證 SpectroSERVER 是否在執行中。

VnmShd: Error: Lost connection with SpectroSERVER

原因：

CLI 本機伺服器偵測到其連線的 SpectroSERVER 已終止，因而終止。

動作：

重新啓動 SpectroSERVER，然後執行 CLI 本機伺服器。

附錄 C： UNIX 與 DOS 的轉換

在 UNIX 平台上，CLI 命令通常會在終端機視窗中搭配 UNIX 命令使用。 在 Windows 平台上，您可以在原生 DOS 視窗中搭配使用 CLI 命令與 DOS 命令。本附錄列出常用的 UNIX 命令與其對等 DOS 命令。

附註：本附錄主要作爲 UNIX 和 DOS 命令的快速參考，而非詳盡的清單。 如需命令與其功能的詳細資訊，請參閱 UNIX、DOS 或 Windows 文件。

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

UNIX	DOS
goto (csh)	goto
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