CA Spectrum®

コマンド ライン インターフェース ユーザ ガイド

リリース 9.4

ca technologies

このドキュメント（組み込みヘルプ システムおよび電子的に配布される資料を含む、以下「本ドキュメント」）は、お客様への情報提供のみを目的としたもので、日本 CA 株式会社（以下「CA」）により随時、変更または撤回されることがあります。

CA の事前の書面による承諾を受けずに本ドキュメントの全部または一部を複写、譲渡、開示、変更、複本することはできません。 本ドキュメントは、CA が知的財産権を有する機密情報です。ユーザは本ドキュメントを開示したり、（i）本ドキュメントが関係する CA ソフトウェアの使用について CA とユーザとの間で別途締結される契約または (ii) CA とユーザとの間で別途締結される機密保持契約により許可された目的以外に、本ドキュメントを使用することはできません。

上記にかかわらず、本ドキュメントで言及されている CA ソフトウェア製品のライセンスを受けたユーザは、社内でユーザおよび従業員が使用する場合に限り、当該ソフトウェアに関連する本ドキュメントのコピーを妥当な部数だけ作成できます。ただし CA のすべての著作権表示およびその説明を当該複製に添付することを条件とします。

本ドキュメントを印刷するまたはコピーを作成する上記の権利は、当該ソフトウェアのライセンスが完全に有効となっている期間内に限定されます。 いかなる理由であれ、上記のライセンスが終了した場合には、お客様は本ドキュメントの全部または一部と、それらを複製したコピーのすべてを破棄したことを、CA に文書で証明する責任を負います。

準拠法により認められる限り、CA は本ドキュメントを現状有姿のまま提供し、商品性、特定の使用目的に対する適合性、他者の権利に対して侵害のないことについて、黙示の保証も含めいかなる保証もしません。 また、本ドキュメントの使用に起因して、逸失利益、投資損失、業務の中断、営業権の喪失、情報の喪失等、いかなる損害（直接損害か間接損害かを問いません）が発生しても、CA はお客様または第三者に対し責任を負いません。CA がかかる損害の発生の可能性について事前に明示に通告されていた場合も同様とします。

本ドキュメントで参照されているすべてのソフトウェア製品の使用には、該当するライセンス契約が適用され、当該ライセンス契約はこの通知の条件によっていかなる変更も行われません。

本ドキュメントの制作者は CA です。

「制限された権利」のもとでの提供：アメリカ合衆国政府が使用、複製、開示する場合は、FAR Sections 12.212、52.227-14 及び 52.227-19(c)(1)及び(2)、ならびに DFARS Section252.227-7014(b)(3) または、これらの後継の条項に規定される該当する制限に従うものとします。

Copyright © 2014 CA. All rights reserved. 本書に記載された全ての製品名、サービス名、商号およびロゴは各社のそれぞれの商標またはサービスマークです。

CA Technologies 製品リファレンス

このガイドでは、CA Spectrum® について説明します。

CA への連絡先

テクニカル サポートの詳細については、弊社テクニカル サポートの Web サイト（http://www.ca.com/jp/support/）をご覧ください。

目次

第 1 章: コマンド ライン インターフェース(CLI)入門 9

概要 9
CLI コマンド 10
シェル スクリプト内の CLI 10
CLI コンポーネント 11
CLI の環境変数 12
CLI アーキテクチャ 13
スタートアップ ファイル 14
CLI Local Server 15
エラー チェック 16

第 2 章: コマンド ライン インターフェースの操作 17

UNIX 上の CLI のセッションを開始します 17
DOS プロンプトを使用して Windows で CLI セッションを開始する 18
Bash プロンプトを使用して Windows で CLI セッションを開始する 19
使用例 19
ユーザ モデルの作成 20
モデル属性の変更 21
1 つの手順でのモデルの作成と変更 23
サンプル CLI スクリプト ファイル - 新規ユーザの作成 23
イベント レポート生成 25
モデル スイッチ 26
トラブルシュータ モデルの作成 26
トラブルシュータへのアラームの割り当て 28
グローバル コレクションの作成 29
CLI 出力でのヘッダの抑制 30

第 3 章: コマンド解説 33

コマンド解説の概要 33
ack alarm - アラームを確認する 34
connect–Connects to SpectroSERVER 34
connect コマンドを使用する場合の考慮事項 36
create - オブジェクトを作成する 38

create alarm 38
create association 39
create event 39
create model 40
current - モデルまたはランドスケープを設定する 42
destroy - オブジェクトを破棄する 44
destroy alarm 44
destroy association 45
destroy model 45
disconnect–Disconnects from SpectroSERVER 46
jump - 保存されたモデルまたはランドスケープにジャンプする 46
seek - モデルを検索する 47
setjump - モデルおよびランドスケープの保存 50
show - オブジェクトを表示する 52
show alarm 54
show association 55
show attributes 56
show children 60
show devices 60
show enumerations 60
show events 61
show inheritance 62
show landscapes 63
show models 63
show parents 64
show relations 65
show rules 65
show types 66
show watch 68
stopShd–Terminates CLI Local Server 69
update - モデルおよびモデル属性を更新する 71

付録 A: サンプル スクリプト 77

サンプル スクリプトの概要 77
active_ports スクリプト 78
app_if_security スクリプト 78
cli_script スクリプト 79
database_tally スクリプト 80
update_mtype スクリプト 80
active_ports スクリプト 81

付録 B: エラー メッセージ 83

付録 C: UNIX から DOS への変換 107

第 1 章: コマンド ライン インターフェース（CLI）入門

このセクションには、以下のトピックが含まれています。

概要 (P. 9)
CLI アーキテクチャ (P. 13)
エラー チェック (P. 16)

概要

CA Spectrum コマンド ライン インターフェース（CLI）はコア CA Spectrum コンポーネントであり、コア CA Spectrum 製品と共にインストールされます。

OneClick ユーザ インターフェースから、CA Spectrum のデータにアクセスし、CA Spectrum の操作を実行できます。ただし、コマンド ラインから CA Spectrum の操作を実行したい場合は、CLI を使用することもできます。OneClick で実行できないタスクの場合、ユーザが利用可能な CA Spectrum リソースは CLI のみです。

CLI は強力なツールですが、特にモデリングに関しては、OneClick が提供するような保護手段を提供しません。 CLI は、CA Spectrum 管理者が、誤ってモデルを作成および破壊したり、ネットワーク モデリング スキーム上のモデル属性を変更したりする潜在的な悪影響を理解した上で使用する必要があります。

CLI は柔軟なオプションです。 UNIX、DOS、および Bash などのシステムで利用可能な任意のコマンド プロンプトから CLI セッションを開いて、コマンドを実行できます。

CLI コマンド

CLI コマンドは UNIX コマンドに似ています。また、UNIX または DOS コマンド、特に grep（検索）、パイプ、シンボルのリダイレクトと共に使用できます。 ただし、一部の CLI コマンドは同じ名前の UNIX コマンドと競合する場合があります。 たとえば、CLI の update コマンドは UNIX の update コマンドと競合する場合があります。

競合を回避するには、vnmsh ディレクトリ内から **./update** を使用します。スクリプトを使用する場合は、CLI コマンドのフルパス名（たとえば <*$SPECROOT*>**/vnmsh/update**）を使用します。

注: CLI の update コマンドは、常に、更新が成功したという確認または更新が失敗したというメッセージのいずれかの応答を提供します。 update コマンドを使用したときに、CLI からの応答がない場合は、「**which update**」と入力します。 システムから次のような応答があります。

```
/etc/update
```

詳細情報:

コマンド解説 (P. 33)

シェル スクリプト内の CLI

CLI コマンドは、CA Spectrum データにアクセスするためのより強力で用途の広い方法を提供するために、シェル スクリプトやメニュー システムに組み込むことができます。

各 CLI コマンドは、標準エラーにコマンドの成功または失敗をレポートする出力を送信します。 ただし、コマンドが成功した結果として期待される通常の出力は、標準出力に送信されます。 コマンドはそれぞれ、成功した場合はゼロのリターン コードを、失敗した場合はゼロ以外のエラー コードを生成します。 リターン コードによって、CLI コマンドを使用するシェル スクリプトでは、各コマンドの成功または失敗に従って処理を続行できます。

CLI コンポーネント

このガイドでは、以下のような CLI コンポーネントについて説明します。

- 実行可能コマンド
- 4 つの環境変数
- SpectroSERVER との通信を維持するデーモン
- CLI コマンドを組み込んだサンプル シェル スクリプトのセット

詳細情報:

コマンド解説 (P. 33)
CLI の環境変数 (P. 12)
CLI Local Server (P. 15)
サンプル スクリプト (P. 77)

CLI の環境変数

CLI に対して以下の 4 つの環境変数を設定できます。

CLIMNAMEWIDTH

モデル名を表示します。デフォルトでは、create、seek、および show コマンドはそれぞれ、モデル名として最大 16 文字を表示します。 ただし、環境変数 CLIMNAMEWIDTH で、モデル名として最大 1024 文字を表示することを指定できます。 たとえば、C シェルを使用して、次のように指定します。

```
setenv CLIMNAMEWIDTH 32
```

この変数は、.login ファイル、スクリプト、または単にコマンドを発行する前に設定できます。 モデル名の長さに応じて、CLI セッションで何度でも設定または変更できます。

CLISESSID

スクリプトで使用するための ID を表します。 CLISESSID 変数を <$$> に設定します。これは、実行中のシェル スクリプトのプロセス ID を表します。 この変数は、cron を使用して CLI スクリプトを同時に実行する場合に必要です。 たとえば、bash シェルを使用して、次のように指定します。

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

また、DOS の代わりに bash シェルを使用して、Windows 上で CLI を実行する場合、CLISESSID 環境変数を各 CLI セッションで一意の値に設定する必要があります。 たとえば、各 bash シェルに一意のタイム スタンプを指定できます。

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

SPECROOT

アラームまたはイベントの説明を表示します。SPECROOT 環境変数は、show alarms または show events コマンドで、-x オプションを指定する場合に必要です。 SG-Support ディレクトリ ツリーが見つかり、コマンドからの出力を展開できる場合、この変数は SG-Support ディレクトリ ツリーからアラームまたはイベントの説明を取得します。

UNIX で、ログイン シェルで SPECROOT 変数を指定でき、この変数を CA Spectrum ホーム ディレクトリに設定できます。 例：

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

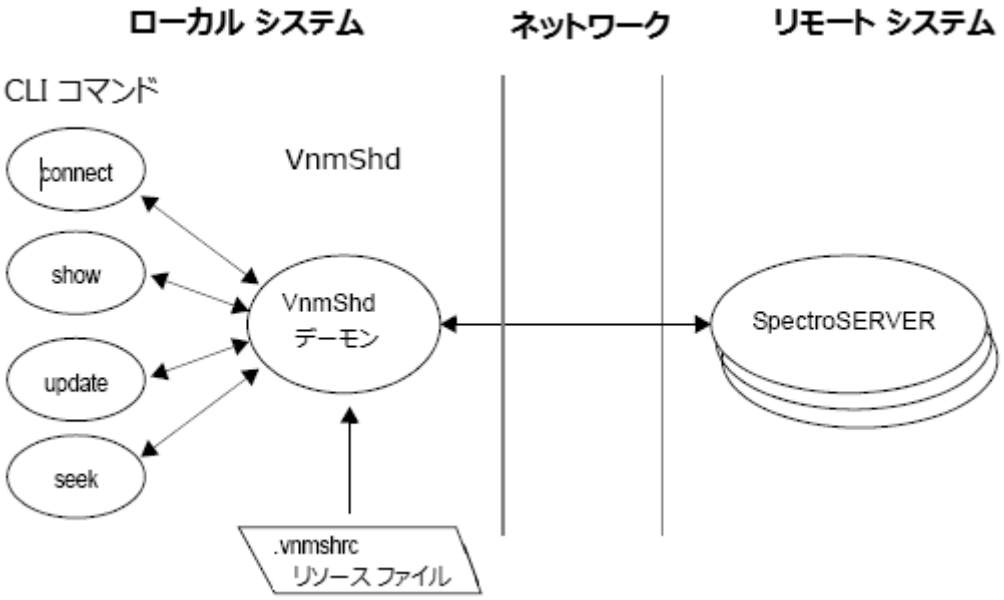
Windows では、環境変数の設定の詳細については、システムのドキュメントを参照してください。

CLIPATH

スクリプトで CLI を使用するために必要な、*<$SPECROOT>*/vnmsh ディレクトリのパスを表示します。

CLI アーキテクチャ

以下の図に、CLI アーキテクチャを示します。



CLI ローカル サーバは、スタートアップ時に .vnmshrc を使用して、以下の主要な機能を実行します。

- SpectroSERVER との一定のネットワーク接続の維持。 CLI ローカル サーバは、コマンドが実行されるたびに切断されることを防ぎます。 このサーバは、デーモンに接続される CLI のユーザの数にかかわらず、SpectroSERVER への*単一の*接続を維持します。 時間およびリソース使用状況に関する限り、ソケットの接続と切断はコストが高くなります。
- 各 CLI のユーザの状態情報の維持。 たとえば、'current' および setjump コマンドでは、CLI ローカル サーバで状態情報を格納する必要があります。 current コマンドは、後のコマンドで使用するために、モデル ハンドルおよびランドスケープ ハンドルを格納します。 setjump コマンドは、CA Spectrum ランドスケープ内でユーザの現在位置を識別するためのテキスト文字列を格納します。

スタートアップ ファイル

.vnmshrc ファイルは、CLI ローカル サーバのスタートアップ ファイルであり、*<$SPECROOT>*/vnmsh ディレクトリにあります。 このファイルには、vnmsh が SpectroSERVER とどのように通信するかを制御するいくつかのパラメータが含まれます。 これらのタスクについては、以下のリストで説明します。

vnm_hostname

接続する SpectroSERVER のホスト名を指定します。

client_handshake_timeout

接続をセットアップするとき、クライアントがサーバ ID 情報を待機するミリ秒数を指定します。

デフォルト： 900

server_handshake_timeout

接続をセットアップするとき、サーバがクライアント ID 情報を待機するミリ秒数を指定します。

デフォルト： 900

connect_time_limit

SpectroSERVER への接続を待機する最大ミリ秒数を指定します。

デフォルト： 1000

listen_backlog

前のクライアント要求が完了するまで待機する間、キューに保持される SpectroSERVER へのクライアント要求の数を指定します。

デフォルト： 10

vnm_tcp_port

vnmsh が SpectroSERVER クライアントであるときに、vnmsh が SpectroSERVER と通信するために使用している TCP ポートを指定します。

vsh_tcp_port

vnmsh が、show や update などのクライアント要求に対するサーバとして動作しているときに、vnmsh が TCP メッセージをリスンする TCP ポートを指定します。

debug_file

CLI がエラー メッセージを書き込むファイルを指定します。

max_show_event_length

show events -x コマンドを使用してイベント メッセージを表示するときに、表示される最大文字数を指定します。

デフォルト： 512

CLI Local Server

最初のユーザが connect コマンドを発行すると、そのワークステーション上で自動的に CLI Local Server（VnmShd デーモン）が起動され、SpectroSERVER への接続が確立されます。 ワークステーションごとに CLI ローカル サーバを 1 つだけ実行できます。また、そのデーモンは、SpectroSERVER への接続を 1 つだけ作成します。

CLI ローカル サーバがワークステーション上で起動された後、そのワークステーション上で CLI に接続する、それ以降のユーザはみな同じ CLI ローカル サーバを使用します。

詳細情報:

CLI の環境変数 (P. 12)

エラー チェック

ユーザが OneClick 内のタスクを実行するとき、CA Spectrum は特定のルールを適用します。 たとえば、ユーザが別のビューでデバイス モデルを作成するか移動させるとき、ルールによって許容できるアクションを制御します。

CLI はこれらのルールを適用しないので、エラー チェックを実行できません。 その結果、CLI によって、ユーザはモデルを作成し、エラー チェックを実行せずに、必要な場所に配置できます。 CLI コマンドの形式に一致しない方法で CLI コマンドを使用使用とすると、エラーが出力されます。

第 2 章: コマンド ライン インターフェースの操作

このセクションには、以下のトピックが含まれています。

UNIX 上の CLI のセッションを開始します (P. 17)
DOS プロンプトを使用して Windows で CLI セッションを開始する (P. 18)
Bash プロンプトを使用して Windows で CLI セッションを開始する (P. 19)
使用例 (P. 19)
イベント レポート生成 (P. 25)
モデル スイッチ (P. 26)
トラブルシューター モデルの作成 (P. 26)
グローバル コレクションの作成 (P. 29)
CLI 出力でのヘッダの抑制 (P. 30)

UNIX 上の CLI のセッションを開始します

UNIX プラットフォームでは、シェル プロンプトから CLI のセッションを開始できます。

注: スクリプトを別のサーバに送信できるように、スクリプトを使用して CLI をまとめることできます。 詳細については、「*CA Spectrum 分散 SpectroSERVER 管理者ガイド*」を参照してください。.

次の手順に従ってください:

1. 接続する SpectroSERVER を開始します。
2. CA Spectrum インストール ディレクトリ内の vnmsh ディレクトリに移動します。

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

3. 接続を開きます。

   ```
   $ connect
   ```

   CLI のセッションに接続されます。

DOS プロンプトを使用して Windows で CLI セッションを開始する

Windows プラットフォームでは、DOS プロンプトから CLI のセッションを開始できます。

注: スクリプトを別のサーバに送信できるように、スクリプトを使用して CLI をまとめることできます。 詳細については、「*CA Spectrum 分散 SpectroSERVER 管理者ガイド*」を参照してください。.

このガイドでは、UNIX（CLI に対立するものとしての）コマンドのすべてのインスタンスについて、必要に応じて同等の DOS コマンドに置き換えてください。 たとえば、*grep* の代わりに *find* を使用してください。

次の手順に従ってください:

1. DOS プロンプトを表示するには、［スタート］、［プログラム］、［コマンド プロンプト］を選択します。

   DOS プロンプトが表示され、CLI コマンドを受け付けることができます。

2. 接続する SpectroSERVER を開始します。
3. CA Spectrum インストール ディレクトリ内の vnmsh ディレクトリに移動します。

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

4. 接続を開きます。

   ```
   $ connect
   ```

   CLI のセッションに接続されます。

詳細情報:

UNIX から DOS への変換 (P. 107)

Bash プロンプトを使用して Windows で CLI セッションを開始する

Windows プラットフォームでは、bash シェル プロンプトから CLI のセッションを開始することもできます。

注: スクリプトを別のサーバに送信できるように、スクリプトを使用して CLI プログラムをまとめることできます。 詳細については、「*CA Spectrum 分散 SpectroSERVER 管理者ガイド*」を参照してください。.

次の手順に従ってください:

1. ［スタート］、［プログラム］、および［コマンド プロンプト］をクリックします。

   DOS プロンプトが表示されます。

2. DOS プロンプトから、「**bash**」と入力します。

3. ［スタート］、［ファイル名を指定して実行］をクリックし、「**bash -login**」と入力します。

   bash シェル プロンプトから CLI のセッションを開始できます。

4. 接続する SpectroSERVER を開始します。

5. CA Spectrum インストール ディレクトリ内の vnmsh ディレクトリに移動します。

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

6. 接続を開きます。

   ```
   $ connect
   ```

   CLI のセッションに接続されます。

使用例

以下の例は、CA Spectrum で CLI コマンドを共通のタスクに使用する方法を示します。

ユーザ モデルの作成

User モデルはユーザに CA Spectrum へのアクセスを提供します。ユーザはログイン ID によって識別されます。

注: CLI のセッションを開始する前に、User モデルが作成され、接続する SpectroSERVER が開始されていることを確認します。

次の手順に従ってください:

1. SpectroSERVER ネットワークに接続します。

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

   SpectroSERVER に接続されます。

   注: 接続時に問題が発生する場合は、エラー メッセージを確認してください。 詳細については、「エラー メッセージ (P. 83)」を参照してください。

2. show コマンドを使用して、作成するモデル タイプのモデル タイプ ハンドルを決定します。 この場合は、User タイプのモデルです。 次のコマンドを入力します。

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

   注: 「./」は重要です。 一部の UNIX システムでは、電子メールの読み取りに show コマンドを使用します。 「.」がユーザ環境内の最初のパスでない場合、「./」が必要です。

   文字列「User」が含まれるモデル タイプのリストが表示され、最初に User モデル タイプがリストで表示されます。

   ```
   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. show コマンドを User モデル タイプに使用して、属性をリスト表示し、モデル名属性に対して属性 ID を決定します。 モデルを作成するには、この属性 ID が必要です。 次のコマンドを入力します。

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

モデル名属性を含めて User モデル タイプ属性のリストが表示されます。

```
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. create コマンドをモデル タイプ ハンドル、モデル名の属性 ID、およびユーザの値（ログイン ID）と共に使用してモデルを作成します。 この例では、ユーザ ログイン ID は j_doe です。次のコマンドを入力します。

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

   以下の例のようなシステム メッセージは、モデルが作成されたことを確認します。

   ```
   作成されたモデル ハンドル = 0xbe0001b
   ```

   注: この例で使用されるすべてのハンドルおよび ID は架空のものです。作成したモデルのモデル ハンドルとは異なります。モデル ハンドルはシステムによって作成されます。

詳細情報:

エラー メッセージ (P. 83)

モデル属性の変更

このセクションでは、CLI コマンドを使用して、モデル属性の値を変更する例を示します。 具体的には、この例は、User モデルで作成されたモデル（j_doe）のコミュニティ文字列属性値を変更する方法を示します。 詳細については、「ユーザ モデルの作成 (P. 20)」を参照してください。

次の手順に従ってください:

1. j_doe モデル ハンドルを特定し、次に、現在のモデルとして j_doe を設定します。

   a. $ ./show models | grep j_doe

      j_doe モデルに関する以下の情報が表示されます。

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

   b. $ ./current mh=0xbe0001b

      システムは、j_doe が現在のモデルであることを確認します。

      ```
      現在のモデル: 0xbe0001b
      現在のランドスケープ: 0xbe00000
      ```

2. コミュニティ文字列属性の ID を特定します。

   注: 簡潔さのために、この手順では、grep コマンドの引数として、属性名の既知の部分（コミュニティ文字列）を示します。 属性の名前がわからない場合、モデルの属性をすべて表示してスキャンし、正しい属性名および属性 ID を決定できます。

3. 次のコマンドを入力します。

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

   属性 ID、属性名、およびコミュニティ文字列値が表示されます。

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

   CA Spectrum は、ユーザ モデルの作成時に、すべてのユーザ モデルにデフォルト値の ADMIN,0 を割り当てます。ADMIN,0 は、CA Spectrum 内の完全な管理者権限をユーザ モデルに与えます。

4. update コマンドを使用して、j_doe モデルの管理者権限レベルを、ADMIN,0 からサンプルの管理者権限レベル ADMIN,5（読み取り専用）に変更します。 次のコマンドを入力します。

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

   属性値の変更を示すエントリが返されます。

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

   lid 属性は、属性をリストする場合にのみ適用されるので、ここには値がありません。 詳細については、「*CA Spectrum* 管理者ガイド」を参照してください。

1 つの手順でのモデルの作成と変更

このセクションでは、単一のコマンド文字列で、モデルを作成し、デフォルトの属性値を別の値に置換する例を示します。 このセクションで示されているタイプの複雑なコマンドを実行できるのは、コマンドを実行する前に、コマンドで指定する関連するモデル識別子の値がわかっている場合だけです。

以下の例では、「ユーザ モデルの作成」と「モデル属性の変更」で紹介したパラメータ値を使用します。

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

サンプル CLI スクリプト ファイル - 新規ユーザの作成

UNIX 上のシェル コマンド プロンプトからコマンドを実行するのと同様に、Windows プラットフォームの bash プロンプトから、CLI コマンドを組み込んだシェル スクリプトを実行できます。

この以下の例では、CA Spectrum ユーザ モデルを作成するためにどのようにスクリプトを使用できるかを示します。

```
#
# CLIPATH が設定されているかどうかを確認します。 設定されていない場合は、作成する必要があります。
#
# /install_area/vnmsh ディレクトリを指すように変数を設定して、
# 必要なコマンドを見つけることができるようにします。
#
if [ -z "$CLIPATH" ]
then
     CLIPATH=/usr/data/spectrum/7.0/vnmsh
     export CLIPATH
fi
```

```
#
#  CLIPATH が有効なディレクトリを指すことをテストします
#
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
#
#  ここでコマンド ライン引数の数を確認します。  引数がない場合は、
#  使用法のメッセージを表示します。  引数が 1 つある場合、必要な処理は新規ユーザを
#  作成することだけです...  2 番目の引数がある場合は、
#  Community_String も同時に設定できます。
#
#  この設定は、ローカル システムまたは .vnmshrc ファイルで vnm_hostname と
#  指定されているシステムでユーザを作成する場合にのみ適用されます。  3 番目のフィールドを追加
して
#  接続先の vnm_hostname を受け付けることもできます。
#
#  必要に応じて、getopts シェル コマンドを使用して、スクリプトに対する「スイッチ」を
#  解析できます。-n は名前、-c はコミュニティ文字列、-v は vnm_hostname です。
#
#（注: スクリプトを bourne シェル（sh）で実行する場合は、getopts を
#  /usr/bin/getopts に置く必要があります。  k シェルでは、getopts 関数が組み込まれています）
#
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

#
#  これで設定は完了したので、新規ユーザの作成に進みます。
#  最初に必要なことは接続です。
#
$CLIPATH/connect
```

```
#
#  ここで、接続の終了ステータスを確認して、接続していたかどうかを確認してみましょう...
#
if [ $? -ne 0 ]
then
        echo "ERROR:  could not connect to SpectroSERVER.  $0 exiting"
        exit 0
fi
#
#  準備ができたので、接続してみます。  create コマンドを
#  実行してみましょう。
#
$CLIPATH/create model mth=0x10004 $command
#
#  もう一度終了ステータスを確認して、実際にモデルが作成されたかどうかを確認します。
#
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
```

イベント レポート生成

CLI は、ランドスケープ上で発生する 2000 個の最新のイベントのリストを維持します。 ただし、多くのイベントがランドスケープ上で発生する場合、最新のイベントはおよそ 1 時間前のものになります。

show events コマンドで -x オプションを使用する場合、SPECROOT 環境変数を設定できます。 以下のコマンドは、CLI を使用してイベント レポートを実行する例です。

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

モデル スイッチ

jump および setjump コマンドは、異なるモデル間を頻繁に移動するスクリプトで役立ちます。 setjump コマンドでは、モデル ハンドルおよび対応するランドスケープ ハンドルを表すためにテキスト文字列を割り当てることができます。 次に、jump コマンドでそのテキスト文字列を使用して、現在のモデル ハンドルとしてその情報を取得できます。 例：

- $ ./current mh=0xb6000f8

  現在のモデル: `0xb6000f8`
  現在のランドスケープ: `0xb600000`

- $ ./setjump emme

  モデル `0xb6000f8` およびランドスケープ `0xb600000` は `emme` の下に保存されました

- $ ./jump emme

  現在のモデル: `0xb6000f8`
  現在のランドスケープ: `0xb600000`

トラブルシュータ モデルの作成

CLI を使用して、Troubleshooter モデルを作成して User モデルに関連付けることができます。 作成して関連付けると、これらのトラブルシュータにアラームを割り当てることができ、トラブルシュータはアラームを調査して解決する必要があるという電子メール通知を受信できるようになります。

以下手順では、TroubleShooter モデルを作成し、User モデルに関連付ける方法について説明します。「ユーザ モデルの作成 (P. 20)」で作成した User モデルを例として使用します。

次の手順に従ってください:

1. *<$SPECROOT>*/vnmsh ディレクトリに移動します。
2. コマンド プロンプトで以下のコマンドを入力することにより、SpectroSERVER に接続します（bash シェルの $ プロンプトの例）。

   ```
   $./connect
   ```

3. TroubleShooter モデル タイプを特定します。 次のコマンドを入力します。

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

   TroubleShooter モデル タイプ エントリが返されます。

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

4. TroubleShooter モデル タイプの EmailAddress 属性 ID を特定します。次のコマンドを入力します。

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

   EmailAddress エントリが表示されます。

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

5. CLI の create コマンドを使用して TroubleShooter モデルを作成します。

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

   以下の例のようなシステム メッセージは、モデルが作成されたことを確認します。

   ```
   $ 作成されたモデル ハンドル = 0xbe0001c
   ```

   注: 例で使用されるすべてのハンドルおよび ID は架空のものです。ユーザが作成したモデルのモデル ハンドルは異なります。これはユーザのシステムで作成されたものです。

6. CLI を使用して、j_doe User モデル（mh=0xbe0001b）と j_doe_fixit TroubleShooter モデル（mh=0xbe0001c）の間に関連付けを作成します。

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

   以下の例のようなシステム メッセージは、関連付けが作成されたことを確認します。

   ```
   $ create association successful
   ```

トラブルシュータへのアラームの割り当て

このセクションでは、CLI コマンドを使用して、アラームをリスト表示し、トラブルシュータにアラームを割り当てる方法について説明します。

次の手順に従ってください:

1. show コマンドを使用してアラームをリスト表示します。

   この手順では、alarm_severity が MAJOR であるアラームのみを検索する方法を示します。

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

   MAJOR アラームのリストが表示されます。 例 :

```
7509   09/27/2000   14:46:44   0xd80008   0xa6000df   duncan    9E133_36   MAJOR   No
7645   09/27/2000   14:47:16   0xd80008   0xa60025e   infinity  9H422_12   MAJOR   No
7518   09/27/2000   14:47:01   0xd80008   0xa6000eb   rugone    9E132_15   MAJOR   No
7979   09/27/2000   14:53:12   0xf40002   0xa600161   FDDI2     FddiMAC    MAJOR   No
8018   09/27/2000   14:53:13   0xf40002   0xa6003da   FDDI FNB  FddiMAC    MAJOR   No
7512   09/27/2000   14:46:47   0xd80008   0xa6000af   ruthere   9A426_02   MAJOR   No
```

2. トラブルシュータに割り当てるアラームを選択します。 この例では、9A426-02 デバイス用のアラーム ID 7512 が選択されています。

3. アラームに割り当てるためにトラブルシュータを選択します。 この例では、「Troubleshooter モデルの作成（P. 26）」で作成した j_doe_fixit TroubleShooter モデルが選択されています。

   注: 次の手順の update コマンドでは、トラブルシュータを指定するために、TroubleShooter モデル名（j_doe_fixit）ではなく、TroubleShooter モデル ハンドル（0xa600722）を使用します。

4. update コマンドを使用して、トラブルシュータにアラームを割り当てます。

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

   以下の例のようなシステム メッセージは、トラブルシュータにアラームが割り当てられたことを確認します。

   ```
   $ update: successful
   ```

   j_doe_fixit モデルによって表される人が、現在、アラームに割り当てられています。 この人は、アラーム割り当ての電子メール通知を受信します。

グローバル コレクションの作成

グローバル コレクションを作成し、GUID を使用して、CLI のセッションでの検索条件を設定できます。 一意の識別子（GUID）はグローバル コレクションを作成するための重要な属性です。 GUID はグローバル コレクションが正しく機能するのに必要です。 VNM モデルへのアクションを通じて GUID を取得できます。

注: CLI では、GUID のないグローバル コレクションは無効です。 OneClick を使用してグローバル コレクションを作成する場合、GUID は自動的に作成されます。 詳細については、「*CA Spectrum IT インフラストラクチャのモデリング/管理 - 管理者ガイド*」を参照してください。.

次の手順に従ってください:

1. 以下のコマンドを入力します。

   `update action=(一意の識別子を取得するアクション) 0x10474 mh= (VNM モデル ハンドル)`

   GUID が作成されます。

   注: 新しい GUID を取得するアクションは 0x10474 です。これはグローバル コレクション モデル タイプと同じです。

2. GUID を使用して、グローバル コレクションを作成する、以下のコマンドを入力します。

   `create model mth=(グローバル コレクションのモデル タイプ) 0x10474 attr=(GUID) 0x12e56, val=(受信した以前の値) 4a85b9af-0d52-1000-017f-0013727f8c0a`

   グローバル コレクションが作成され、［ナビゲーション］ペインのグローバル コレクションの下に表示されます。

例: XML 文字列を使用した場合と使用しない場合のグローバル コレクションの作成

XML 文字列を使用して、または使用しないでグローバル コレクションを作成するには、以下のコマンドを入力します。

```
update action=(GUID を取得するアクション)0x10474 mh=(ランドスケープ)
```

```
出力 :
update action: successful
レスポンスには 1 の属性があります:
 0) 属性 0x0 テキスト: EXAMPLE GUID(4a85b9af-0d52-1000-017f-0013727f8c0a)
```

```
create model mth=(グローバル コレクションのモデル タイプ)0x10474
attr=(GUID)0x12e56,val=(以前に受信した値)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>'
出力 :
created model handle = New model handle(0x78101069)
```

注: create コマンドで dynamicCriteriaXML（0x12a6a）属性を指定することも、後でモデルを更新することもできます。

詳細情報:

コマンド解説 (P. 33)

CLI 出力でのヘッダの抑制

CLI 出力でヘッダを抑制するために、以下のセクションで示す関数が含まれたファイルを作成し、各スクリプトの先頭でこのファイルを参照できます。

以下の手順で示す関数は、CLI コマンドを呼び出して、コマンドの出力からヘッダ情報を除去します。

次の手順に従ってください:

1. スクリプトディレクトリで StripHeaders という名前のファイルを作成します。
2. StripHeaders ファイルに以下の関数を入力します。

```
tcreate() # create alarm コマンドと
    { # create event コマンドでのみ必要
    $CLIPATH/create $@ | tail +2
    }
tseek()
    {
      $CLIPATH/seek $@ | tail +2
    }
tshow()
    {
    $CLIPATH/show $@ | tail +2
    }
tupdate()
    {
    $CLIPATH/update $@ | tail +2
    }
```

3. CLI スクリプトの先頭に、StripHeaders という名前を以下のように含めます。

```
. StripHeaders
```

4. CLI の出力からヘッダを除去する場合は常に、対応する CLI コマンドの代わりに、tcreate()、tseek()、tshow()、および tupdate() 関数を呼び出します。

   たとえば、以下の行は、CLI のヘッダ情報のない show models コマンドの出力を生成します。

```
tshow models
```

第 3 章: コマンド解説

この章では、CLI コマンドと出力について説明します。

このセクションには、以下のトピックが含まれています。

コマンド解説の概要 (P. 33)
ack alarm - アラームを確認する (P. 34)
connect–Connects to SpectroSERVER (P. 34)
create - オブジェクトを作成する (P. 38)
current - モデルまたはランドスケープを設定する (P. 42)
destroy - オブジェクトを破棄する (P. 44)
disconnect–Disconnects from SpectroSERVER (P. 46)
jump - 保存されたモデルまたはランドスケープにジャンプする (P. 46)
seek - モデルを検索する (P. 47)
setjump - モデルおよびランドスケープの保存 (P. 50)
show - オブジェクトを表示する (P. 52)
stopShd–Terminates CLI Local Server (P. 69)
update - モデルおよびモデル属性を更新する (P. 71)

コマンド解説の概要

CLI を使用すると、CA Spectrum で利用可能な保護手段を使用せずに、CA Spectrum ナレッジ ベースに変更を加えることができます。 正しくない情報を指定すると、システム クラッシュやデータベースの破損が生じる場合があります。 したがって、create、destroy、または update コマンドを使用する場合は、注意して続行してください。

注: CLI で応答時間のテストを作成および管理するために、CLI コマンド パラメータを使用します。

詳細については、「*CA Spectrum Service Performance Manager ユーザ ガイド*」を参照してください。.

ack alarm - アラームを確認する

ack alarm コマンドは、landscape_handle によって指定されたランドスケープで alarm_id によって指定されたアラームを確認します。landscape_handle が指定されていない場合、コマンドは現在のランドスケープで alarm_id によって指定されたアラームを確認します。

あるモデルの 1 つのアラームを確認することは、そのアラームのみを確認することを意味し、そのモデルの他のアラームを確認することを意味しません。

コマンドの形式は以下のとおりです。

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

ack alarm が有効な alarm_id および有効な landscape_handle を指定して入力された場合、以下のメッセージが表示されます。

```
ack alarm: successful
```

例 : ack alarm

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

connect–Connects to SpectroSERVER

connect コマンドは、CA Spectrum コマンド ライン インターフェースのユーザを、ホスト システム *hostname* 上で実行される SpectroSERVER に接続します。このコマンドは、landscape_handle によって指定されるランドスケープを現在のランドスケープに設定します。CLI ローカル サーバがまだ実行されていない場合、connect コマンドはこれを起動します。

コマンドの形式は以下のとおりです。

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

ホスト名

（オプション）hostname が指定されない場合、コマンドは CLI リソース ファイル .vnmshrc で指定されたホストにユーザを接続します。

注: CA Spectrum コマンド ライン インターフェースは localhost または 127.0.0.1 オプションをサポートしていません。 localhost に接続するには、実際のホスト名を指定するか、パラメータを何も指定しません。

landscape_handle

（オプション） landscape_handle が指定されない場合、コマンドは現在のランドスケープを、指定されたホスト名のランドスケープに設定します。

vnmsocket

（オプション）vnmsocket が指定されない場合、コマンドは .vnmshrc ファイルで指定されたソケットを使用して SpectroSERVER に接続します。 別のポート接続（vnmsocket によって定義される）で別の SpectroSERVER に接続する場合は vnmsocket を使用できます。

UNIX では、CLI ローカル サーバによってレポートされるエラー メッセージはコンソール ウィンドウに表示されます。Windows では、これらのエラーはユーザ bash シェル ウィンドウに表示されます。

例: connect

```
#! /usr/bin/sh
# 特定の重大度のアラームを取得し、
# CLISESSID を設定するサンプル スクリプト

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
```

コマンドが成功した場合、以下のメッセージが表示されます。

```
connect: successful hostname
現在のランドスケープ: <landscape_handle>
```

hostname は、ユーザが入力した SpectroSERVER ホスト、または .vnmshrc ファイルで指定されたホストです。 landscape_handle は、ユーザが入力したランドスケープまたはホストのランドスケープです。

詳細情報:

CLI の環境変数 (P. 12)
スタートアップ ファイル (P. 14)

connect コマンドを使用する場合の考慮事項

connect コマンドを使用する場合に重要な考慮事項を以下に示します。

- 端末上のユーザは connect コマンドを使用して通信を開始する必要があります。 同じユーザは、disconnect コマンドを使用して SpectroSERVER との通信を終了する必要があります。
- 最初のユーザが connect コマンドを入力すると、CLI ローカル サーバは SpectroSERVER に接続されます。
- 同じ CLI ローカル サーバを使用する他の CLI のユーザは、初期 SpectroSERVER のランドスケープ マップにある SpectroSERVER にのみ接続できます。 すべてのユーザが切断すると、connect コマンドを使用して、別のランドスケープ マップ内の SpectroSERVER に接続します。
- 正常に SpectroSERVER に接続するには、connect コマンドの最初のユーザは、元の SpectroSERVER の CA Spectrum データベースでユーザとして定義されている必要があります。
- bash シェル内で CLI を実行する Windows ユーザは、CLISESSID も定義する必要があります。
- 特定のユーザの端末デバイスは、ttyslot(3V) 関数を使用して特定されます。

- cron スクリプトは ttyslot に関連付けられていません。その結果、ttyslot 関数はすべての cron スクリプトについて 0 を返します。 すなわち、cron スクリプトとして同時に実行される 2 つの CLI スクリプトは、CLI ローカル サーバでは 1 人の CLI ユーザとして認識されるために、予測不能の結果に結びつきます。 そのため、環境変数 CLISESSID をエクスポートするための行を、スクリプトの先頭に挿入する必要があります。CLISESSID を一意の数値に設定します。 これで、CLI は異なる cron スクリプトを区別できます。

  以下の例は、スクリプト内で一意の CLI セッション ID を定義します。

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

- この例は、スクリプトを実行するシェルのプロセス ID として CLISESSID を設定します。 CLI は、ttyslot 関数がゼロを返すときに、ユーザを識別するために CLISESSID を使用します。 各 CLI のセッションに対して CLISESSID を 1 回設定します。 cron スクリプトとして実行している CLI のスクリプトが他の CLI のスクリプトを呼び出す場合、トップレベルのスクリプトのみが CLISESSID 環境変数を設定します。 スクリプトの先頭で新しいシェル（#!/bin/sh）を呼び出さない場合、他の CLI のスクリプトは同じプロセス ID の下で実行されます。他のスクリプトで新しいシェルを呼び出すには、CLISESSID をエクスポートし、接続して再度切断します。

- 一部の環境または設定では、コマンドがコマンド ラインから入力されたときでも、ttyslot 関数がゼロを返す場合があります。 このような場合、connect コマンドは以下のエラーを返します。

  ```
  connect: 変数 CLISESSID が設定されていません
  ```

  このような場合は、コマンド ラインから、または .cshrc やその他のスタートアップ ファイルから、CLISESSID を設定します。

- CLI では、ユーザ名と端末デバイスを使用して、各 CLI ユーザを識別します。 端末デバイスから同時に複数のスクリプトを実行するユーザは、同じユーザとして CLI に表示されます。 スクリプトがバックグラウンドで実行されており、別のスクリプトがフォアグラウンドで実行されている場合、または複数のスクリプトがバックグラウンドで実行されている場合、CLI は予測不能の結果をもたらす場合があります。

  たとえば、スクリプト A1 が現在のモデルをモデル A に設定するとします。スクリプト B1（同じ端末デバイスから同じユーザによって実行される）は、現在のモデルをモデル B に設定します。 スクリプト A1 がモデル A に対して update コマンドを実行する場合、update コマンドはスクリプト B1 のモデル B に対しても実行されます。

特定の端末デバイスからは、同時に 1 つの CLI セッションのみを実行します。 同時に複数の CLI のセッションを実行するには、個別の端末デバイスから実行するか、または CLISESSID 環境変数をそれぞれ一意の値に設定して、at(1) または batch(1) コマンドを使用して実行します。

create - オブジェクトを作成する

create コマンドを使用してオブジェクトを作成します。

注: セキュア ドメインでモデルを作成する方法の詳細については、./create を実行して使用法の説明を表示してください。

コマンドの形式は以下のとおりです。

```
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]
```

詳細情報:

CLI の環境変数 (P. 12)

create alarm

create alarm コマンドは、重大度が alarm_severity で原因が probable_cause_id のアラームを、model_handle のモデルについて作成します。 有効なアラームの重大度オプションは、CRITICAL、MAJOR、MINOR、OK、MAINTENANCE、SUPPRESSED、または INITIAL です。 デフォルトでは、新しいアラームは既存のアラームを置換します。

create alarm が、有効な alarm_severity、有効な probable_cause_id、および有効な model_handle を指定して入力された場合、アラーム テーブル内に作成されたエントリが表示されます。 作成時刻は hh:mm:ss 形式で表示されます。

例: create

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

ID      Date          Time        PCauseID    MHandle     MName    MTypeName     Severity    Ack
984     05/11/2000    12:33:27    0x10308     0x400134    12.84    Bdg_CSI_CN    CRITICAL    No
```

create association

create association コマンドは、left_model_handle のモデルと right_model_handle のモデルの間の関係（関連付け）のインスタンスを作成します。

create association が、有効な left_model_handle と right_model_handle の間の有効な関係を指定して入力された場合、以下のメッセージが表示されます。

```
create association: successful
```

例: create association

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

create event

create event コマンドは、model_handle によって指定されるモデルについて、タイプ event_type およびテキスト event_text を使用してイベントを作成します。 landscape_handle が指定される場合、イベントを作成したユーザ モデルについてイベントが作成されます。

注: CLI の旧バージョンでは、イベントはランドスケープ モデルに対して作成されていました。

model_handle または landscape_handle が指定されていない場合、イベントはイベントを作成したユーザ モデルについて作成されます。 または、現在のモデルが指定されている場合、イベントは現在のモデルについて作成されます。 CA Spectrum 内のいくつかのイベントには、関連付けられたモデルがありません。 たとえば、アプリケーションが SpectroSERVER に接続するとき、モデルはイベントに関連付けられません。

create event が、有効な event_type、有効な event_text、および有効な model_handle または landscape_handle（存在する場合）を指定して入力された場合、エントリはイベント テーブルに表示されます。 作成時刻は hh:mm:ss 形式で表示されます。

event_type コマンド（また CA Spectrum 内のイベント コードを指定した）は 4 バイト 16 進数です。 2 つの最上位バイトはイベントの開発者 ID（CA Spectrum で生成されたイベント コードの場合は 0001）であり、2 つの最下位バイトは一意のイベント識別子です。 すべてのイベント タイプにユーザが入力したテキストが含まれるとは限りません。 このようなイベント タイプの例として、イベント フォーマット ファイルに変数 {S 0} が含まれるものがあります。 ユーザが入力したテキストが含まれないイベント タイプについては、event_text パラメータは無視されますが、コマンド ライン上には存在している必要があります。

詳細については、「CA Spectrum 認定ユーザ ガイド」を参照してください。

例: create event

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

Date          Time        Type       MHandle     MName    MTypeName
05/11/2000    12:39:42    0x1061a    0x400134    12.84    Bdg_CSI_CNB20
```

create model

IP アドレスまたはモデル タイプ ハンドルを使用して、create model を指定できます。 いずれの場合も、システムは、landscape_handle によって指定されるランドスケープでモデルを作成します。 landscape_handle が指定されていない場合、コマンドは現在のランドスケープでモデルを作成します。

注: model_name 属性は、User モデルを作成する場合にのみ必要です。

- create model で IP アドレスを指定する場合、システムは指定された ip_address でオブジェクトを検索し、そのモデルを作成します。 モデルには、関連付けられた子を含め、そのオブジェクトのすべてのプロパティが含まれます。 たとえば、オブジェクトがハブである場合、create model コマンドはそのポートのすべてを持ったハブのモデルを作成します。
- IPv4 アドレスまたは IPv6 アドレスを指定できます。IPv6 範囲はサポートされていません。また、このコマンドは、属性 ID の設定をサポートしません。

- 一度に複数のモデルを作成するには、create model コマンドで IP アドレスの範囲を定義できます。 「-」で区切って、Low_IP と High_IP のパラメータを指定します。 Community_Name が指定されない場合、新しく作成されたモデルは「Ping 可能」タイプです。 Community_Name が指定された場合、デバイスは適切なモデル タイプにモデル化されます。 Try_Count と Time_Out のオプションは、OneClick の［IP アドレスでモデルを作成］ダイアログ ボックスのオプションに似ています。
- create model コマンドでモデル タイプ ハンドルを指定する場合、システムは model_type_handle タイプのモデルを作成します。 次に、作成されたモデルについて、1 つ以上の属性の値を設定できます。
- create model コマンドでモデル タイプ ハンドルを指定するときに、その 1 つのコマンドで複数の属性を指定することもできます。 スペースによって隣接するペアから各ペアを区切って、複数の「attribute_id, 値」のペアを指定します。
- OneClick を使用している間、特定のモデル タイプのモデルを作成するときにユーザが指定する属性値は、create model コマンドで指定してください。それ以外の場合、モデルが作成されるときに、SpectroSERVER 内で推論ハンドラ エラーが発生する可能性があります。 たとえば、OneClick を使用して、Hub_CSI_IRM3 モデルを作成するときに、モデル名、ネットワーク アドレス、コミュニティ文字列の値を入力できるウィンドウが表示されます。 CLI を使用して同じタイプのモデルを作成するには、create model コマンドを使用するときこれらの属性に対する値を指定します。
- create model で有効な model_type_handle と有効な attribute_id, value のペア（存在する場合）が指定された場合、作成されたモデル ハンドルが表示されます。
- create model で有効な ip_address が指定された場合、作成されたモデル ハンドルが表示されます。

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 デバイスに到達できません
Creating model for IP=206.61.231.3
create model: DCM デバイスに到達できません
Creating model for IP=206.61.231.4
create model: DCM デバイスに到達できません
Creating model for IP=206.61.231.5
created model handle =  0x9a0025a
```

注: デフォルトでは、create コマンドは、モデル名として最大 16 文字を表示します。 ただし、環境変数 CLIMNAMEWIDTH で、モデル名として表示される文字数に別の値（最大 1024 ）を指定できます。

current - モデルまたはランドスケープを設定する

current コマンドは、model_handle によって指定されるモデルを現在のモデルとして設定します。または、この command は landscape_handle によって指定されるランドスケープを現在のランドスケープとして設定します。model_handle と landscape_handle が指定されない場合、current は現在のモデル ハンドルおよび現在のランドスケープ ハンドルを表示します。

ユーザが現在のモデルを設定すると、CLI は現在のランドスケープをモデルが含まれるランドスケープに設定します。 ユーザが現在のランドスケープを設定すると、CLI は現在のモデルを未定義として設定します。

個別の現在のモデルおよび現在のランドスケープの値が、CLI ローカルサーバに接続される各セッションについて維持されます。

current コマンドは、たとえば、名前を指定したセッションについてのみ、状態情報、現在のモデル、および現在のランドスケープを保持します。

コマンドの形式は以下のとおりです。

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

- 有効な model_handle が入力として指定された場合、以下のメッセージが表示されます。

  現在のモデル: <model_handle>
  現在のランドスケープ: <current_landscape_handle>

- 有効な landscape_handle が入力として指定された場合、以下のメッセージが表示されます。

  現在のモデルが定義されていません
  現在のランドスケープ: <landscape_handle>

- model_handle と landscape_handle が指定されない場合、以下のメッセージが表示されます。

  現在のモデル: <current_model_handle>
  現在のランドスケープ: <current_landscape_handle>

- model_handle と landscape_handle が指定されず、現在のモデルが定義されていない場合、以下のメッセージが表示されます。

  現在のモデルが定義されていません
  現在のランドスケープ: <current_landscape_handle>

例: current

```
$ current mh=0x400142
現在のモデル: 0x400142
現在のランドスケープ: 0x400000

$ current lh=0x500000
現在のモデルが定義されていません
現在のランドスケープ: 0x500000

$ current
現在のモデルが定義されていません
現在のランドスケープ: 0x500000
```

注: 現在のランドスケープは connect コマンドによって設定されるので、常に値が含まれます。

destroy - オブジェクトを破棄する

オブジェクトを破棄するには、destroy コマンドを使用します。 コマンドの形式は以下のとおりです。

```
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

destroy コマンドで -n (プロンプトなし) オプションが指定される場合、システムは確認を求めません。 このオプションは CLI のスクリプトで便利です。

-n オプションが指定されない場合、以下のいずれかのメッセージが常に表示されます。

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

有効な応答は、y、yes、Y、Yes、n、no、N、および No です。

destroy alarm

landscape_handle によって指定されるランドスケープで alarm_id によって指定されたアラームを破棄します。 -n オプションが指定されない場合、destroy alarm はアラームを破棄する前にユーザに確認を求めます。 landscape_handle が指定されない場合、コマンドは現在のランドスケープで alarm_id によって指定されるアラームを破棄します。モデルの alarm_id を特定するには、show alarms コマンドを使用します。

destroy alarm コマンドが有効な alarm_id および有効な landscape_handle と共に入力された場合、以下のメッセージが表示されます。

```
destroy alarm: successful
```

例 : destroy alarm

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

destroy association

left_model_handle を持つモデルと right_model_handle を持つモデルの間で関連付け（関係のインスタンス）を破棄します。 -n オプションが指定されない場合、destroy association は関連付けを破棄する前にユーザに確認を求めます。

destroy association が、有効な left_model_handle と right_model_handle の間の有効な関係を指定して入力された場合、以下のメッセージが表示されます。

```
destroy association: successful
```

例：destroy association

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

destroy model

指定された model_handle を持つモデルを破棄します。 n オプションが指定されない場合、destroy model はモデルを破棄する前にユーザに確認を求めます。

destroy model が、有効な model_handle を指定して入力された場合、以下のメッセージが表示されます。

```
destroy model: successful
```

例：destroy model

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

以下のモデルは破棄されます:

```
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

現在接続されている SpectroSERVER から CLI ユーザを切断するには、disconnect コマンドを使用します。

コマンドの形式は以下のとおりです。

```
disconnect
```

コマンドが成功した場合、以下のメッセージが表示されます。hostname はユーザが接続されていた SpectroSERVER ホストの名前です。

```
disconnect: <hostname> または <IP address> での成功 - xx 時間 yy 分間接続されています
```

詳細情報:

stopShd–Terminates CLI Local Server (P. 69)

jump - 保存されたモデルまたはランドスケープにジャンプする

jump コマンドは以前に保存されたモデルおよびランドスケープにジャンプします。 jump コマンドは、現在のモデルと現在のランドスケープを、setjump コマンドによってラベル text_string の下で保存されたモデルとランドスケープに設定します。 text_string が指定されていない場合、前の setjump コマンドで指定された text_strings のリストが表示されます。

コマンドの形式は以下のとおりです。

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

- jump が以前に定義された有効な text_string と共に入力された場合、新しい現在のモデルおよび現在のランドスケープが表示されます。

  ```
  現在のモデル: <current_model_handle>
  現在のランドスケープ: <current_landscape_handle>
  ```

- jump が text_string なしで入力された場合、現在定義されている text_string のリストが表示されます。 例：

  ```
  text_string1
  text_string2
  --
  ```

- jump が入力され、新しい現在のモデルが未定義の場合、以下のメッセージが表示されます。

  ```
  現在のモデルが定義されていません
  現在のランドスケープ: <current_landscape_handle>
  ```

例: jump

```
$ jump tutorial
現在のモデル: 0x400142
現在のランドスケープ: 0x400000
```

詳細情報:

setjump - モデルおよびランドスケープの保存 (P. 50)

seek - モデルを検索する

モデルを検索するには seek コマンドを使用します。 seek コマンドは、attribute_id で指定された属性の値を所有する、landscape_handle で指定されたランドスケープ内のモデルを検索します。 landscape_handle が指定されていない場合、コマンドは指定された attribute_id を持つ属性の値を所有する、現在のランドスケープ内のモデルを検索します。 また、seek でワイルドカード（*）を使用して、指定されたサブストリングが含まれるモデルのインスタンスを検索することもできます。 NULL 値を入力する場合、名前がないモデル（たとえば attr=0x1006e）をすべて検索できます。

seek コマンドを使用して、1 文字の属性値を検索することはできません。そのような検索を試行するとエラーが返されます。

コマンドの形式は以下のとおりです。

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

オプションは任意の順序（たとえば -i -s または -s -i）で使用できます。

-i

-i（大文字と小文字の区別を無視する）オプションが seek コマンドで指定される場合、val パラメータで指定されるモデル情報が大文字と小文字に関係なく返されます。

-s

-s（サブストリングを許可する）オプションが seek コマンドで指定される場合、該当する場合、val パラメータで指定されるモデル情報がサブストリング付きで返されます。

seek が有効な attribute_id および有効な値と共に入力される場合、一致するモデルはすべて以下の形式で表示されます。

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

一致するモデルが見つからない場合、以下のメッセージが表示されます。

```
seek: no models found
```

注: デフォルトでは、seek コマンドは、モデル名として最大 16 文字を表示します。ただし、環境変数 CLIMNAMEWIDTH で、モデル名として表示される文字数に別の値（最大 1024 ）を指定できます。

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
```

setjump - モデルおよびランドスケープの保存

setjump コマンドは、ラベル text_string の下に現在のモデル ハンドルおよび現在のランドスケープ ハンドルを保存します。 ユーザは、後で jump コマンドと共に text_string を使用して、現在のモデル ハンドルと現在のランドスケープ ハンドルを text_string に格納されたものに設定できます。 同じ text_string が 2 つの setjump コマンドで使用されている場合、ユーザは検証を求められます。

個別の setjump の値が、CLI ローカル サーバに接続される各セッションについて維持されます。 setjump コマンドは、名前を指定したセッションについてのみ、情報（すなわちセッションに割り当てられた setjump テキスト文字列）を保持します。

コマンドの形式は以下のとおりです。

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

-n

-n（プロンプトなし）オプションが setjump コマンドで指定される場合、text_string が以前に使用されている場合、システムはプロンプトを表示しません。

- setjump が新しい <text_string> と共に入力され、現在のモデルが存在する場合、以下のメッセージが表示されます。

  モデル <current_model_handle> およびランドスケープ
  <current_landscape_handle> は <text_string> の下に保存されました

  <current_model_handle> は現在のモデルのハンドルで、<current_landscape_handle> は現在のランドスケープのハンドルです。

- setjump が新しい <text_string> と共に入力され、現在のモデルが存在しない場合、以下のメッセージが表示されます。

  モデルが定義されていません。ランドスケープ <current_landscape_handle>
  が <text_string> の下に保存されました

- setjump が以前 setjump コマンドですでに定義されている text_string と共に入力された場合、以下のメッセージが表示されます。

  setjump model: <text_string> already used. Overwrite?

  有効な応答は、y、yes、Y、Yes、n、no、N、および No です。

例: setjump

```
$ current mh=0x400142
現在のモデル: 0x400142
現在のランドスケープ: 0x400000
$ setjump -n tutorial
モデル 0x400142 およびランドスケープ 0x400000 は tutorial の下に保存されました
```

show - オブジェクトを表示する

オブジェクトを表示するには、show コマンドを使用します。

コマンドの形式は以下のとおりです。

```
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] [-d]
                [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

-a（すべて）オプションが指定された場合、show alarms はマスキングを実行せず、CRITICAL、MAJOR、MINOR、MAINTENANCE、SUPPRESSED、および INITIAL アラームをすべて表示します。

-x

-x（展開する）オプションが指定され（、変数 $SPECROOT が設定され）た場合、show alarms コマンドの出力は出力の最後に想定される原因のテキストを表示します。show events コマンドの出力はイベント形式を表示します。 show events -x コマンドによって表示される文字の数は、.vnmshrc リソース ファイル パラメータによって制御されます。

-d

-d オプションが指定された場合、show alarms コマンドの出力は、リスト内のアラームごとに「タイトル」を表示します。また、このオプションは、-x オプションが表示する内容をすべて表示します。

max_show_event_length

SpectroSERVER 専用のワークステーションを使用して CLI を実行している場合、-x オプションを指定しても通常のアラームの原因またはイベント形式を提供しません。これは、CsPCause ファイルと CsEvFormat ファイルが SG-Support ディレクトリに存在しないためです。次のようなエラー メッセージが表示される可能性があります。

- No cause information available（show alarms に関連付けられる）
- No event format information available（show events に関連付けられる）

この問題を解決するには、SpectroSERVER ワークステーション上の <*$SPECROOT*>/SG-Support ディレクトリに SG-Support/CsPCause および SG-Support/CsEvFormat ディレクトリおよびファイルをコピーします。

デフォルト： 512

-e

-e（列挙）オプションが指定された場合、show attributes コマンドの出力はデータベース列挙文字列を表示します。

-c

-c（Read Most Current）オプションが指定された場合、属性 Read Mode は Read Most Current に設定されます。このモードは、最新のユーザ インターフェース ポーリングからの属性値を使用します。この値は 5 秒ごとに更新されます。このフラグが設定されていない場合、Read Most Available モードが使用されます。 このモードは、最後の CA Spectrum ポーリングによってデータベースに格納された最新の値を使用します。ポーリング周期はユーザ定義の間隔です。

-n

-n（イベントの数）オプションが指定された場合、show events コマンドの出力は指定された数のイベントを表示します。

-t

-t（トラブル チケット ID）オプションが指定された場合、show alarms コマンドの出力はトラブル チケット ID フィールドを表示します。

-s

-s（影響重大度）オプションが指定された場合、show alarms コマンドの出力は影響重大度フィールドを表示します。

show alarms および show events コマンドで -x オプションを指定して、アラームの想定される原因や展開されたイベント メッセージを表示するには、ローカル サーバに OneClick がインストールされ、SPECROOT 環境変数が spectrum サポート ルート ディレクトリに設定されている必要があります。 たとえば、SG-Support ファイルが/usr/spectrum/SG-Support にある場合は、SPECROOT を /usr/spectrum に設定します。

show alarm

show alarms コマンドは、model_handle で指定されるモデルのアラームをすべて表示します。 または、landscape_handle で指定されるランドスケープの各モデルについて、最も重大なアラームのみ（アラームが CRITICAL、MAJOR、または MINOR である場合）を表示します。 landscape_handle が指定されている場合、show alarms は INITIAL、SUPPRESSED、または MAINTENANCE アラームを持つすべてのモデルをマスキングします。 その結果、CRITICAL、MAJOR、または MINOR アラームを持つモデルのみが表示されます。 model_handle も landscape_handle も指定されていない場合も、show alarms はマスキングを実行し、現在のランドスケープの各モデルについて、最も重大なアラームのみ（アラームが CRITICAL、MAJOR、または MINOR である場合）を表示します。

Ack フィールドは、アラームが確認されたかどうかを示します。 このフィールドに対する可能な値は Yes と No です。Stale フィールドは、アラームが古くなっているかどうかを示します。 このフィールドに対する可能な値は Yes と No です。 Assignment と Status のフィールドはアラーム トラブルシューター情報およびアラーム ステータスをそれぞれ表示します。 アラーム作成時刻は hh:mm:ss 形式で表示されます。

show alarms コマンドは、以下の形式で情報を表示します。

```
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
```

show alarms を -x オプションと共に使用する場合、原因コードと想定される原因のテキスト メッセージのテーブルが最後のアラームの後に表示されます。 例 :

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

注: デフォルトでは、show コマンドは、モデル名として最大 16 文字を表示します。 ただし、環境変数 CLIMNAMEWIDTH で、モデル名として表示される文字数に別の値（最大 1024 ）を指定できます。

例 : show alarms

show alarms コマンドは、以下の形式で情報を表示します。

```
$ show alarms lh=0x110000
ID   Date        Time      PCauseId   MHandle     MName     MTypeName  Severity   Ack  Stale
Assignment  Status
928  05/11/2000  02:33:22  0x10c04    0x110000c   infinity  VNM        CRITICAL   No   No
McDonald    Working on it
```

詳細情報:

コマンド解説 (P. 33)
CLI の環境変数 (P. 12)

show association

show associations コマンドは、model_handle を持つモデルについて定義され、インスタンス化された関係（関連付け）をすべて表示します。 model_handle が指定されていない場合、show associations は現在のモデルについてインスタンス化された関係をすべて表示します。

show associations コマンドは、以下の形式で情報を表示します。

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

例 : show associations

show associations コマンドは、以下の形式で情報を表示します。

```
$ show associations mh=0x400141

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

show attributes

show attributes コマンドは、model_handle を持つモデルについて、sttr=attribute_id で指定された属性を表示します。 attribute_id が指定されていない場合、show attributes は model_handle を持つモデルについて、すべての属性とその値をリスト表示します。

model_handle が指定されていない場合、show attributes は現在のモデルに適用可能なすべての属性をすべて表示します。attr=low_attr-high_attr を使用して属性の範囲を指定できます。 特定のモデルの単一の属性または属性のリストを表示するときに、属性のインスタンス ID を instance_id で指定できます。 instance_id は、ピリオドで区切られた正の整数のシーケンスである必要があります。 インスタンス ID は、リスト属性についてのみ指定できます。 リスト属性はリスト フラグ セットがある属性です。

以下のルールがリスト属性に適用されます。

- リスト属性のすべての属性値およびインスタンス ID を表示するには、attribute_id と共に instance_id を入力しないでください。attribute_id のみを入力します。
- リスト属性の最初の属性値およびインスタンス ID を表示するには、attribute_id の後に以下のコマンドを入力します。

  ```
  ,next
  ```

- リスト属性の特定の属性値およびインスタンス ID を表示するには、attribute_id と共に instance_id を入力します。
- リスト属性の特定のインスタンス ID の後に次の属性値およびインスタンス ID を表示するには、instance_id の後に以下のコマンドを入力します。

  ```
  ,next
  ```

- 以下の理由で、モデルの属性をすべて表示するときには、インスタンス ID を指定できません。
  - インスタンス ID はリスト属性にのみ適用されます（たとえばハブのボードおよびポート属性）。
  - モデルの特定の属性のインスタンス ID は、同じモデル内の他の属性のインスタンス ID と異なる場合があります。
- show attributes コマンドは landscape_handle によって指定されたランドスケープで model_type_handle について、すべての属性（ID、名前、タイプ、およびフラグによる）を表示します。 landscape_handle が指定されていない場合、このコマンドは現在のランドスケープで定義されているモデル タイプをすべて表示します。 Flags フィールドは、現在設定されている、各属性フラグの省略形を（カンマで区切って）リスト表示します。 フラグが設定されていない場合、その省略形はリストに含まれません。

  以下のリストには属性フラグおよびそれらの省略形が含まれます。
  - External = E
  - Readable = R
  - Writable = W
  - Shared = S
  - List = T
  - Guaranteed = G
  - Global = O
  - Memory = M
  - Database = D
  - Polled = P
  - Logged = L
  - Preserve Value = V

注: 属性フラグの詳細については、「*Model Type Editor ユーザガイド*」を参照してください.

show attributes コマンドは、以下の形式で情報を表示します。

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

show attributes mth コマンドは、以下の形式で情報を表示します。

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

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 コマンドは model_handle を持つモデルに関連する子を表示します。関係が指定されていない場合、show children はすべての関係の子を表示します。 model_handle が指定されていない場合、コマンドは現在のモデルの子を表示します。

show children コマンドは、以下の形式で情報を表示します。

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

例 : show children

```
$ show children mh=0x400009

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

show devices

show devices コマンドは、landscape_handle によって指定されるランドスケープですべてのデバイス モデルのリストを表示します。

show devices コマンドは、以下の形式で情報を表示します。

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

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 コマンドは、指定された列挙値に対応する、列挙文字列値マッピングを表示します。

show enumerations コマンドは、以下の形式で情報を表示します。

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

show enumerations mth コマンドは、以下の形式で情報を表示します。

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

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 コマンドは、model_handle を持つモデルのイベントまたは landscape_handle で指定されるランドスケープ内のすべてのモデルのイベントを表示します。 デフォルトでは、show events コマンドは、model_handle または landscape_handle で指定されるモデルについて最新の 2,000 のイベントを表示します。 -a オプションが指定された場合、このコマンドは model_handle または landscape_handle によって指定されるモデルについて最大数である 10,000 のイベントを表示します。

-n オプションが明示的な no_events ステートメントで指定される場合、指定された数のイベントが model_handle または landscape_handle によって指定されるモデルについて表示されます。 model_handle も landscape_handle も指定されない場合、このコマンドは現在のランドスケープのすべてのモデルについてイベントを表示します。-x オプションが指定された場合、CLI はイベント タイプについて説明するテキスト メッセージを表示します。 イベント時刻は hh:mm:ss 形式で表示されます。

show events コマンドは、以下の形式で情報を表示します。

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

show events を -x オプションと共に使用する場合、表示されるイベントには固定フォーマットがありません。 以下に、一般的な出力の例を示します。

```
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])
```

例: show events

```
$ show events lh=0x400000

Date         Time       Type      MHandle    MName    MTypeName
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

Date         Time       Type      MHandle             MName             MTypeName
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 コマンドは、landscape_handle によって指定されるランドスケープで model_type_handle によって指定されるモデル タイプについて、モデル タイプ継承を表示します。 landscape_handle が指定されていない場合、現在のランドスケープが使用されます。 このフィールドに対する可能な値は Base または Derived です。

show inheritance コマンドは、以下の形式で情報を表示します。

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

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 コマンドは、各 SpectroSERVER に対して定義されているランドスケープをすべて表示します。 表示されるランドスケープ マップは初期 SpectroSERVER のマップです。

show landscapes は、以下の形式で情報を表示します。

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

例: show landscapes

```
$ show landscapes

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

show models

show models コマンドは、landscape_handle によって指定されたランドスケープで定義されているモデルをすべて表示します。 landscape_handle が指定されていない場合、このコマンドは現在のランドスケープで定義されているモデルをすべて表示します。 以下のコマンドによって、モデル ハンドルの範囲を指定できます。

```
mhr=low_model_handle-high_model_handle
```

mname=model_name を指定することにより、特定のモデルを検索できます。

ユーザ モデルは、show models コマンドによって、（Active）または（Not Active）のいずれかとして識別されます。ユーザ モデル ステータスが（Not Active）である場合、ユーザはまだサーバに接続できません。 ユーザ モデル ステータスが（Active）になると、ユーザはサーバに接続できます。

show models コマンドは、以下の形式で情報を表示します。

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

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 コマンドは model_handle を持つモデルに関連する親を表示します。 関係が指定されていない場合、このコマンドはすべての関係の親を表示します。 model_handle が指定されていない場合、show parents は現在のモデルの親を表示します。

show parents コマンドは、以下の形式で情報を表示します。

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

例: show parents

```
$ show parents mh=0x40000d

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

show relations

show relations コマンドは、landscape_handle によって指定されたランドスケープで現在定義されている関係をすべて表示します。 landscape_handle が指定されていない場合、このコマンドは現在のランドスケープで定義されている関係をすべて表示します。

show relations コマンドは、以下の形式で情報を表示します。

```
Name            Type
relation_name   relation_type
```

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 コマンドは、関係のルールを表示します。 関係は、landscape_handle によって指定されるランドスケープで指定されます。 landscape_handle が指定されない場合、現在のランドスケープが使用されます。

show rules コマンドは、以下の形式で情報を表示します。

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

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 コマンドは、landscape_handle によって指定されるランドスケープで現在定義されているモデル タイプをすべて表示します。 landscape_handle が指定されていない場合、このコマンドは現在のランドスケープで定義されたモデル タイプをすべて表示します。 Flags フィールドは、現在設定されている 6 つの各属性フラグの省略形をリスト表示します。 フラグが設定されていない場合、その省略形はリストに含まれていません。

以下のリストにはモデル タイプ フラグおよびそれらの省略形が含まれます。

- Visible = V
- Instantiable = I
- Derivable = D
- No Destroy = N
- Unique = U
- Required = R

表示タイプ コマンド [mth=low_mth-high_mth] は、low_mth と high_mth の間の範囲内のモデル タイプをすべて表示します。

注: モデル タイプ フラグの詳細については、「*Model Type Editor ユーザ ガイド*」を参照してください。.

show types コマンドは、以下の形式で情報を表示します。

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

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 コマンドは、model_handle によって指定されるモデル用の適用可能なウォッチ データをリスト表示します。

landscape_handle と model_handle が指定されていない場合、show コマンドは以下のデフォルトを使用します。

コマンド	デフォルト
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
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

注: show alarms コマンドと show events コマンドでは、x オプションを使用することによって、アラームの想定される原因メッセージおよび展開されたイベント メッセージを表示できます。

以下の前提条件を確認します。

- ローカル サーバに OneClick がインストールされていること。
- 環境変数 SPECROOT がルート ディレクトリ（SG-Support）のパスに設定されていること。

  たとえば、SG-Support ファイルが/usr/spectrum/SG-Support にある場合は、SPECROOT を /usr/spectrum に設定します。

show watch コマンドは、以下の形式で情報を表示します。

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

例: show watch

```
$ show watch mh=0xc600015

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

stopShd–Terminates CLI Local Server

CLI Local Server（VnmShd デーモン）を終了するには、stopShd コマンドを使用します。 stopShd コマンドは、現在接続されている SpectroSERVER から CA Spectrum CLI のユーザをすべて切断し、CLI ローカル サーバを終了します。 このコマンドは、ユーザを切断し、サーバをシャットダウンする前に、ユーザに確認を求めます （また、kill -2 コマンドを使用して、デーモンをシャットダウンすることもできます）。

コマンドの形式は以下のとおりです。

```
stopShd [-n]
```

-n

「プロンプトなし」を指定します。確認プロンプトを無効にするには、stopShd コマンドにこのオプションを含めます。

このオプションを指定しない場合は、以下のメッセージが常に表示されます。

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

「n」は、ユーザ自身を含めた接続ユーザ数を表します。

有効な応答は、y、yes、Y、Yes、n、no、N、No です。

コマンドが成功した場合、以下のメッセージが表示されます。

```
stopShd: successful
```

stopShd が CLI ローカル サーバを終了すると、以下のメッセージがシステム コンソールに表示されます。

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

例: stopShd

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

詳細情報:

disconnect–Disconnects from SpectroSERVER (P. 46)

update - モデルおよびモデル属性を更新する

モデルとモデル タイプ属性を更新するには、update コマンドを使用します。

コマンドの形式は以下のとおりです。

```
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

update コマンドで -n（プロンプトなし）オプションが指定される場合、システムは確認を求めません。 このオプションは CLI のスクリプトで便利です。

-r

-r（ステータス テキスト置換/トラブル チケット ID 置換）オプションは、status または ticket 引数を使用するときに、update alarm コマンドと共に指定できます。 -r オプションを使用する場合、既存のアラーム ステータス テキストまたはアラーム トラブル チケット ID は、status 引数または ticket 引数によって指定されたテキストと置換されます。 -r オプションを使用しない場合、新しい値は既存の値に追加されます。

action_code

- モデルを再設定する場合は、reconfig、0x1000e、または 65550
- ウォッチをアクティブ化する場合は、activate、0x00480003、または 4718595
- ウォッチを非アクティブ化する場合は、deactivate、0x00480004、または 4718596

- Cisco および Wellfleet デバイス上でアプリケーション モデルを再設定する場合は、reconfigure_apps、0x210008、または 2162696
- EventDisp と AlertMap への変更内容で SpectroSERVER を更新する場合は、reload_event_disp、0x000100a2、または 65698

注: watch = *<watch_id>* パラメータは、次のアクションについてのみ適用できます。activate（または対応する 16 進数値 0x00480003）および deactivate（対応する 16 進数値 0x00480004）。

以下のポイントで update コマンドの機能を説明します。

- update コマンドは、landscape_handle によって指定されたランドスケープで、model_handle を持つモデルまたはモデル タイプが model_type_handle であるすべてのモデルについて、attribute_id 値で指定された属性を更新します。
- 複数の attribute_id, value のペアを、スペースで隣接するペアと区切って指定することによって、1 つの update コマンドで複数の属性を更新できます。
- remove オプションは、リスト属性から指定されたインスタンスを削除します。
- モデル タイプ属性を更新するときに、landscape_handle が指定されていない場合、現在のランドスケープが使用されます。 model_handle が指定されていない場合、現在のモデルの指定された属性が更新されます。
- モデル タイプ属性を更新しているときに、共有属性のみを更新できることに注意してください。 共有属性は共有フラグが設定された属性です。 show attributes コマンドを使用して、属性が共有されているかどうか確認できます。
- User_Community_String や Model_Security_String などセキュリティ上重要な属性は CLI によって更新できます。 ただし、現在のユーザ モデルはそれ自身の User_Security_String または Security_String を更新できず、他のモデルのものを更新できます。
- 単一の属性値を変更するときに、update コマンドではユーザがインスタンス ID を指定できます。属性値のリストを更新するとき、インスタンス ID は、リスト上の各属性に対して指定できます。 instance_id は対応する属性のインスタンス ID です。 instance_id は正の整数、またはドットで区切られた正の整数のシーケンスである必要があります。

- インスタンス ID が指定されていない場合、update コマンドは属性について最初に見つかった有効なインスタンスを使用します。 有効なインスタンスが見つからない場合、エラー メッセージが表示されます。

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

- update alarm コマンドは、Troubleshooter（トラブルシュータ モデル ハンドルまたはトラブルシュータ名）、status_text、troubleticketID、または ack パラメータによって指定された値を使用して、alarm_id によって指定されたアラームを更新します。 既存のアラームのトラブルシュータ、ステータス テキスト、またはトラブル チケット ID の値をクリアするには、適切なパラメータが値を持たないように設定できます（status=、ticket=、または assign=）。 landscape_handle パラメータは、アラームが検索されるランドスケープを指定します。

- update action コマンドは、model_handle によって指定されたデバイス上で、action_code によって指定されたアクションを実行します。 action_code reconfig によって、モデル タイプ GnSNMPDev、または GnSNMPDev を継承するすべてのモデル タイプの任意のデバイスを再設定できます。 action_code activate または deactivate は、指定された model_handle のデバイス上のウォッチ ステータスを更新します。アクティブ化アクション オブジェクトが送信されるときに、選択されたモデルに組み込まれているインテリジェンスによって、ウォッチ ステータスが INITIAL から ACTIVE に変わるときに短い遅延が発生する場合があります。ステータスの更新が予定されていたウォッチの watch_id は、show watch コマンドの使用により取得できます。 reconfigure_apps action_code は、Cisco および Wellfleet デバイス モデルのアプリケーション モデル タイプを更新します。 reload_event_disp action_code は、EventDisp または AlertMap ファイルに対する変更内容で SpectroSERVER を更新します。

- update action コマンドを使用するときには注意が必要です。 すべての CLI コマンドと同様に、このコマンドを誤って使用すると、SpectroSERVER データベースが破損する可能性があります。 たとえば、不注意にクリティカルなルータを再設定すると、ネットワークで予測不能の結果を引き起こす可能性があります。

- 有効な model_handle または有効な model_type_handle、有効な attribute_id および有効な値と共に update を入力した場合、変更される属性およびその値は以下の形式で表示されます。

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

- 指定されたモデル タイプのモデルを更新するときに、-n オプションを使用しない場合、以下の確認メッセージが表示されます。

  ```
  update: all models of this type will be updated, are you sure?
  ```
  有効な応答は、y、yes、Y、Yes、n、no、N、No です。

- update alarm コマンドが成功した場合、以下のメッセージが表示されます。

  ```
  update:successful
  ```

- update action コマンドが成功した場合、以下のメッセージが表示されます。

  ```
  update action: successful
  ```

例: update

- 以下の例では、instance_id と共に update コマンドを使用して、モデル ハンドル 0x4001f6 によって表されるハブのボード 5 に搭載されたポート 7 を無効にします。

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

- 以下の例では、update コマンドを remove オプションと共に使用して、特定のモデル（mh）の deviceIPAddressList（属性）から IP アドレス（iid）を削除します。

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

- 以下の例では、update コマンドを使用して、モデル タイプ ハンドル 0x10059 によって表されるモデル タイプのすべてのモデルで AutoPlaceStartX という名前の属性を更新します。

  ```
  $ 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
  ```

- 以下の例では、update alarm コマンドを使用して、アラームのトラブルシュータの割り当てを更新します。

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

- 以下の例では、update alarm コマンドを使用して、アラーム ステータスを更新します。 -r オプションを使用して、既存のステータスに上書きします。

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

- 以下の例では、update alarm コマンドを使用して、アラームのトラブル チケット ID を更新します。 -r オプションを使用して、トラブル チケット ID の既存の値を上書きします。

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

- 以下の例では、update alarm コマンドを使用して、トラブル チケット ID の既存の値をクリアします。

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

- 以下の例では、update alarm コマンドを使用して、アラームを確認します。

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

- 以下の例では、update コマンドを使用して、User_Community_String の更新を制限します。

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

- 以下の例では、update action コマンドを使用して、Cisco ルータを再設定します。

```
$ 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
```

付録 A: サンプル スクリプト

サンプル スクリプトの概要

CLI に含まれているサンプル スクリプトは、CLI コマンドを UNIX シェル スクリプトに組み込んで、CLI セッションを自動化できるようにする方法を示します。これらのスクリプトや、スクリプトに含まれる関数の中には、ユーザ自身の作業に役立つものもあります。

CLI には以下のスクリプトおよびスクリプトについて説明した Readme ファイルが *<$SPECROOT>*/vnmsh/sample_scripts ディレクトリに含まれています。

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

CLI のスクリプトで作業するときには、以下の前提条件を確認します。

- 各スクリプトには CLIPATH という名前の内部変数があります。 スクリプトを使用するには、CLIPATH 変数を CLI の実行可能ファイルが格納されているディレクトリのパス名に設定します。
- CLIPATH 変数、およびパス名である他の環境変数は、スクリプトがどのように実行されるかに応じて *フル* パス名または *相対* パス名を指定できます。cron スクリプトとしてサンプル スクリプトを実行する場合は、CLIPATH およびその他の環境変数に対して *フル* パス名を使用します。 それ以外の場合は、これらの変数に *相対* パス名を使用できます。

- update_mtype を除き、cron スクリプトとしてすべての CLI スクリプトを実行できます。

  注: update_mtype は、ユーザに入力を求めるので、cron スクリプトとして実行しないでください。

- CLI のスクリプトを実行するときには、.vnmshrc ファイルで vnm_hostname 変数として正しい名前を指定します。

active_ports スクリプト

active_ports スクリプトは、IRM2 ハブの各ボードのポートをすべて識別し、各ボード上のアクティブなポートを識別するために使用します。

active_ports スクリプトは、output_file で指定したファイルに hub_name で指定したハブについてのレポートを生成します。このレポートは、各ボードのポートをすべてリスト表示します。レポートの ON 列のアスタリスク（*）は、どのポートがアクティブであるかをユーザに示します。

このスクリプトの形式は以下のとおりです。

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

app_if_security スクリプト

app_if_security スクリプトは、CA Spectrum データベース内のすべてのインターフェースとアプリケーション モデル内の Security_String 属性値を更新するために使用します。 app_if_security スクリプトは親モデルから属性値をコピーすることにより更新します。 受信者モデル（子）に Security_String 属性の値がすでに存在する場合や、親に Security_String 属性値がない場合、このスクリプトはモデルを更新しません。 モデル セキュリティ文字列を更新した後で、管理者はこのスクリプトを使用して、モデルの子のセキュリティ文字列を更新できます。

このスクリプトの形式は以下のとおりです。

```
app_if_security
```

cli_script スクリプト

cli_script は、入力としてデータ ファイルを提供するときに、ほとんどの CLI コマンドをバッチ モードで実行するために使用します。 datafile で指定する CLI サンプルデータ ファイルには、実行するコマンドおよびまたコマンドに渡す必要があるパラメータを示すスイッチが含まれます。 このスクリプトは、コマンドがそれぞれ正常に実行されたことを確認し、実行時のログをメンテナンスします。

このスクリプトの利点の 1 つは、ハンドルの代わりに名前を使用して、バッチ ファイルを作成できるということです。 たとえば、16 進のモデル タイプ ハンドルではなく、モデル タイプ名を使用できます。 これによりファイルの作成や読み取りが容易になりますが、本当の利点は、ユーザが作成したモデル上で後続のアクションを実行するときにあります。 モデルへの 16 進のモデル ハンドルを割り当てる代わりに、名前によってモデルを参照できます。

このスクリプトの形式は以下のとおりです。

```
cli_script datafile
```

cli_script では、datafile と clean.awk の 2 つのファイルを使用します。 これらのファイルも sample_scripts ディレクトリにあります。

datafile

cli_script の入力が含まれています。 cli_script で現在実装されている各 CLI コマンドが含まれます。 このファイルの形式および構文の説明については、cli_script ヘッダ情報を参照してください。

clean.awk

実行で使用される入力が含まれています。 .awk ファイルは、コンソールに表示されるデータの形式を指定するために使用されます。

cli_script を使用するときは、以下の点を考慮します。

- サンプル datafile 内の「ダミー」のネットワーク アドレス（255.255.255.255）を実際のアドレスに変更します。
- 別のディレクトリに cli_script を移動する場合は、環境変数 SPECROOT をサポート ルート ディレクトリ（SG-SUPPORT）に更新する必要があります。

  たとえば、SG-SUPPORT ファイルが/usr/spectrum/SG-Support にある場合は、SPECROOT を /usr/spectrum に設定します。

database_tally スクリプト

このスクリプトは、現在データベースにある各タイプのモデルの数を特定するために使用します。 管理者がシステム パフォーマンスを評価するときに、このスクリプトが役立つ場合があります。 スクリプトは、データベース内のすべてのモデル タイプ、および各モデル タイプのモデルの数のリストを表示します。

このスクリプトの形式は以下のとおりです。

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

update_mtype スクリプト

このスクリプトは、モデル タイプのすべてのモデルについて特定の属性を更新するために使用します。 属性がモデル タイプの共有属性である場合、このスクリプトはモデルの属性を更新しません。 このスクリプトの利点の 1 つは、プロンプトで、16 進の ID ハンドルではなく、モデル名と属性名を使用できるということです。

注: CLIMNAMEWIDTH 環境変数を 256 に設定してください。 値を大きくすることによって、モデル名の切り捨てられるために、スクリプトを実行するときに誤って一致することを防止できます。

このスクリプトの形式は以下のとおりです。

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

model_name | model_type_name

属性の更新が実行されるモデル タイプのモデルの、モデル名、またはモデル名の一部を指定します。コマンドでモデル タイプの任意のモデルを指定できます。

次に、スクリプトは、入力したモデル名引数が含まれる名前を持つモデルがあるすべてのモデル タイプのリストを表示します。スクリプトは、リストからモデル タイプを選択するようユーザに求めます。

モデル タイプ名の代わりにモデル名を使用する場合、スクリプトはコマンド ラインまたはプロンプトで入力された文字列を含む名前を持つモデルをすべて更新します。この場合、指定されたモデル タイプのすべてのモデルは、前に説明したように更新されるとは限りません。

注: プロンプトでは、モデル名またはモデル タイプ名を使用し、ハンドルは使用しないことをお勧めします。

attribute_name | attribute_id

最初にこれらの引数を指定しない場合、スクリプトは、実行時に属性名または属性 ID の指定を求めます。現時点では、属性名または属性名の一部のいずれかを指定する必要があります。 次に、スクリプトは、入力したテキストが含まれる属性のリストから選択するようにユーザに求めます。 そのため、16 進のモデル タイプ ハンドルまたは属性ハンドルが事前にわからない場合でも、スクリプト全体を実行できます。

詳細情報:

CLI の環境変数 (P. 12)

active_ports スクリプト

このスクリプトは、Octet_String 形式 XX.YY.ZZ を ASCII 文字列表現に変換するために使用します。

付録 B: エラー メッセージ

ack alarm: <alarm_id>: 無効なアラーム ID です

原因:

入力したアラーム ID は無効です。

処置:

有効な alarm_id を使用して、再度 ack alarm コマンドを入力します。

ack alarm: <landscape_handle>: 無効なランドスケープ ハンドルです

原因:

アラームに対して入力したランドスケープ ハンドルは無効です。

処置:

有効な landscape_handle を使用して、再度 ack alarm コマンドを入力します。

command: VnmShd との接続に失敗しました。最初に接続を実行してください

原因:

connect コマンドを実行する前に、他のコマンドを実行しようとしました。

処置:

connect コマンドを使用して CLI のセッションを開始します。

connect: <date/time> 以降、すでに <hostname> に接続されています

原因:

接続の試行は不要です。すでに SpectroSERVER ホストに接続されています。

処置:

なし。

connect: リソース ファイル <pathname>/.vnmshrc を開けません

原因:

connect コマンドで CLI リソース ファイル .vnmshrc を見つけることができませんでした。

処置:

.vnmshrc リソース ファイルは、connect コマンド自体と同じディレクトリにある必要があります。

詳細情報:

スタートアップ ファイル (P. 14)

connect: <hostname> ランドスケープ マップにある SpectroSERVER にのみ接続できます - 他のユーザはすでに接続されています

原因:

connect コマンドは、すでに特定の SpectroSERVER に接続するために使用されています。 元の SpectroSERVER のランドスケープ マップにある SpectroSERVER にのみ接続できます。

処置:

なし

connect: エラー: CA Spectrum ユーザ「<username>」が存在しません

原因:

connect コマンドの最初のユーザが CA Spectrum ユーザとして定義されていません。

処置:

CA Spectrum ユーザとして SpectroSERVER に再接続します。

connect: <hostname> が応答していないか、アクセスが許可されていません

原因:

ホスト名が正しくないか、SpectroSERVER が実行されていないか、またはユーザにユーザ モデルがないために、connect コマンドで SpectroSERVER に接続できません。

処置:

ホスト名が正しく、SpectroSERVER が実行されており、ユーザにユーザ モデルがあることを確認します。

connect: <landscape_handle>: 無効なランドスケープ ハンドルです

原因:

ユーザによって指定された landscape_handle は、指定されたホスト名で有効ではありません。または、ハンドルはユーザの VNM によってアクセスできません。

処置:

landscape_handle が指定されたホスト名で有効であることを確認し、ユーザの VNM によってハンドルにアクセスできることを確認します。

connect: incompatible SpectroSERVER <version>

原因:

ユーザは、バージョンが CLI のバージョンと互換性がない SpectroSERVER ホストに接続しようとしています。

処置:

使用している CLI のバージョンを更新します。

connect: CLISESSID の <value> が無効です

原因:

connect コマンドが cron スクリプト内で使用されたか、ウィンドウ システムが ttyslot として 0 を返し、環境変数 CLISESSID が非数値に設定されています。

処置:

cron スクリプトの外部で connect コマンドを使用し、CLISESSID を数値に設定します。

connect: 変数 <CLISESSID> が設定されていません

原因:

最初に CLISESSID 環境変数を設定せずに、cron スクリプト内で connect コマンドを使用しようとしました。

処置:

cron スクリプト内で connect を使用する場合、環境変数 CLISESSID を設定します。

create: ユーザにアラーム作成の権限がありません

原因:

アラームを作成することが許可されていません。

処置:

ユーザ権限を確認します。

create: ユーザに関連付け作成の権限がありません

原因:

関連付けを作成することは許可されていません。

処置:

ユーザ権限を確認します。

create: ユーザにイベント作成の権限がありません

原因:

イベントを作成することは許可されていません。

処置:

ユーザ権限を確認します。

create: ユーザにモデル作成の権限がありません

原因:

モデルを作成することは許可されません。

処置:

ユーザ権限を確認します。

create alarm: <probable_cause_id>: 無効なアラーム原因コード

原因:

create alarm コマンドは無効な probable_cause_id を指定して入力されました。

処置:

有効な probable_cause_id を指定して create alarm コマンドを再入力します。

create alarm: <alarm_severity>: 無効なアラーム重大度です

原因:

create alarm コマンドは無効な alarm_severity を指定して入力されました。

処置:

有効な alarm_severity を指定して create alarm コマンドを再入力します。

create alarm: <model_handle>: 無効なモデル ハンドルです

原因:

create alarm コマンドは無効な model_handle を指定して入力されました。

処置:

有効な model_handle を指定して create alarm コマンドを再入力します。

create association: <left_model_handle>: 無効なモデル ハンドルです

原因:

create association コマンドは無効な left_model_handle を指定して入力されました。

処置:

有効な left_model_handle を指定して create association コマンドを再入力します。

create association: models belong to different landscapes

原因:

create association コマンドは、異なるランドスケープにある left_model_handle と right_model_handle を指定して入力されました。

処置:

同じランドスケープを両方のハンドルに使用します。

create association: rel=<relation>: 無効な関係です

原因:

create association コマンドは無効な関係を指定して入力されました。

処置:

有効な関係を指定して create association コマンドを再入力します。

create association: <right_model_handle>: 無効なモデル ハンドルです

原因:

create association コマンドは無効な right_model_handle を指定して入力されました。

処置:

有効な right_model_handle を指定して create association コマンドを再入力します。

create event: <event_type>: 無効なイベント タイプです

原因:

create event コマンドは無効な event_type を指定して入力されました。

処置:

有効な event_type を指定して create event コマンドを再入力します。

create event: <landscape_handle>: 不明なランドスケープです

原因:

create event コマンドは無効な landscape_handle を指定して入力されました。

処置:

有効な landscape_handle を指定して create event コマンドを再入力します。

create event: <model_handle>: 無効なモデル ハンドルです

原因:

create event コマンドは無効な model_handle を指定して入力されました。

処置:

有効な model_handle を指定して create event コマンドを再入力します。

create model: <attribute_id>: 無効な属性 ID です

原因:

create model コマンドが無効な attribute_id を指定して入力されたので、モデルは作成されません。

処置:

有効な attribute_id を指定して create model コマンドを再入力します。

create model: DCM デバイスに到達できません

原因:

create model コマンドが無効な ip_address を指定して入力されたので、モデルは作成されませんでした。 DCM（デバイス通信マネージャ）はこのエラー メッセージを発行します。

処置:

有効な ip_address を指定して create model コマンドを再入力します。

create model: デバイス制限を超えています

原因:

ブランチ マネージャ SpectroSERVER（デバイス モデルの制限は 50）またはサイト マネージャ SpectroSERVER（デバイス モデルの制限は 250）に、格納できる最大数のデバイス モデルが格納されているので、モデルは作成されませんでした。

処置:

SpectroSERVER 上のデバイス モデルの数が規定された制限に達していないことを確認します。 制限に達している場合は、いくつかを削除した後、create model コマンドを再入力します。

create model: <landscape_handle>: 無効なランドスケープ ハンドルです

原因:

create model コマンドが無効な landscape_handle を指定して入力されたので、モデルは作成されませんでした。

処置:

有効な landscape_handle を指定して create model コマンドを再入力します。

create model: <model_type_handle>: 無効なモデル タイプ ハンドルです

原因:

create model コマンドが無効な model_type_handle を指定して入力されたので、モデルは作成されません。

処置:

有効な model_type_handle を指定して create model コマンドを再入力します。

create model: <value>: 無効な値です

原因:

create model コマンドが無効な値を指定して入力されたので、モデルは作成されません。

処置:

有効な値を指定して create model コマンドを再入力します。

create model: <value>: コミュニティ名がありません

原因:

create 要求で指定されたコミュニティ名が正しくありませんでした。

処置:

有効なコミュニティ名を指定して create コマンドを再入力します。

current: <model_handle>: 無効なモデル ハンドルです。現在のモデル: <current_model_handle>

原因:

無効な model_handle が指定されたので、現在のモデルおよび現在のランドスケープは変更されません。

処置:

現在のモデルおよび現在のランドスケープを変更する場合は、有効な model_handle を再入力します。

current: <landscape_handle>: 無効なランドスケープ ハンドルです。現在のランドスケープ: <current_landscape_handle>

原因:

無効な landscape_handle が指定されたので、現在のモデルおよび現在のランドスケープは変更されません。

処置:

現在のモデルおよび現在のランドスケープを変更する場合は、有効な landscape_handle を再入力します。

current: <landscape_handle>: 応答していないか、アクセスが許可されていません。現在のモデル: <current_model_handle>

原因:

landscape_handle が指定されましたが、ランドスケープの OneClick はダウンしていたか、ランドスケープにユーザのユーザ モデルがありませんでした。

処置:

問題のランドスケープの OneClick が実行されていることを確認します。問題のランドスケープにユーザ モデルがあることを確認します。その後、landscape_handle を再入力します。

current: <landscape_handle>: 応答していないか、アクセスが許可されていません。現在のランドスケープ: <current_landscape_handle>

原因:

model_handle が指定されましたが、そのモデルが含まれるランドスケープの SpectroSERVER がダウンしていたか、ランドスケープにユーザのユーザモデルがありませんでした。

処置:

問題のランドスケープの SpectroSERVER が実行されていることを確認します。問題のランドスケープにユーザ モデルがあることを確認します。その後、model_handle を再入力します。

destroy: ユーザにアラーム破棄の権限がありません

原因:

アラームを破棄することが許可されていません。

処置:

ユーザ権限を確認します。

destroy: ユーザに関連付け破棄の権限がありません

原因:

関連付けを破棄することが許可されていません。

処置:

ユーザ権限を確認します。

destroy: ユーザにモデル破棄の権限がありません

原因:

モデルを破棄することが許可されていません。

処置:

ユーザ権限を確認します。

destroy alarm: aid=<alarm_id>: 無効なアラーム ID です

原因:

destroy alarm コマンドは無効な alarm_id を指定して入力されました。

処置:

有効な alarm_id を指定して destroy alarm コマンドを再入力します。

destroy alarm: <landscape_handle>: 無効なランドスケープ ハンドルです

原因:

destroy alarm コマンドは無効な landscape_handle を指定して入力されました。

処置:

有効な landscape_handle を指定して destroy alarm コマンドを再入力します。

destroy association: rel=<relation>: 無効な関係です

原因:

destroy association コマンドは無効な関係を指定して入力されました。

処置:

有効な関係を指定して destroy association コマンドを再入力します。

destroy association: <left_model_handle>: 無効なモデル ハンドルです

原因:

destroy association コマンドは無効な left_model_handle を指定して入力されました。

処置:

有効な left_model_handle を指定して destroy association コマンドを再入力します。

destroy association: <right_model_handle>: 無効なモデル ハンドルです

原因:

destroy association コマンドは無効な right_model_handle を指定して入力されました。

処置:

有効な right_model_handle を指定して destroy association コマンドを再入力します。

destroy association: 指定されたモデル間に関連付けが存在しません

原因:

存在しない 2 つのモデル間の関連付けを破棄しようとしました。

処置:

破棄しようとしている関連付けに属する 2 つのモデルの存在を確認します。

destroy association: models belong to different landscapes

原因:

destroy association コマンドは、異なるランドスケープにある left_model_handle と right_model_handle を指定して入力されました。

処置:

同じランドスケープにある left_model_handle および right_model_handle を使用して、destroy association コマンドを再入力します。

destroy model: <model_handle>: 無効なモデル ハンドルです

原因:

destroy model コマンドは無効な model_handle を指定して入力されました。

処置:

有効な model_handle を指定して destroy model コマンドを再入力します。

disconnect: 失敗しました

原因:

disconnect コマンドが失敗しました。

処置:

再度 disconnect コマンドを試みます。

disconnect: VnmShd との接続に失敗しました。最初に接続を実行してください

原因:

CLI ローカル サーバが実行されていないときに、disconnect を実行しようとしました。

処置:

なし。 すでに切断されています。

disconnect: 接続していません

原因:

ユーザが SpectroSERVER に接続されていなかったため、disconnect コマンドは失敗しました。

処置:

なし。 すでに切断されています。

jump: <text_string>: テキスト文字列が定義されていません

jump:<text_string>: **テキスト文字列が定義されていません**

`text_string1`、`text_string2...` は現在定義されている文字列です。

原因:

jump コマンドは未定義の text_string を指定して入力されました。

処置:

定義されたテキスト文字列のいずれかを使用して、jump コマンドを再入力します。

<pathname>/VnmShd: 見つかりません

<pathname>/VnmShd: 見つかりません

connect: 失敗しました

pathname は、CLI が VnmShd を実行することを試みたディレクトリへのパスを表します。

原因:

connect コマンドで CLI ローカル サーバを見つけることができませんでした。

処置:

VnmShd と connect が同じディレクトリにあることを確認してから、connect コマンドを再入力します。

最初に接続してください

原因:

connect を実行した後、disconnect または stopShd を実行してから、別のコマンドを実行しようとしました。

処置:

connect コマンドを最初に再発行します。

seek: <attribute_id>: 無効な属性 ID です

原因:

seek コマンドは無効な attribute_id を指定して入力されました。

処置:

有効な attribute_id を指定して seek コマンドを再入力します。

seek: <error>: attribute not keyed

原因:

seek コマンドは、キー設定されていなかった属性の attribute_id を指定して入力されました。

処置:

キー設定された属性の attribute_id を使用して seek コマンドを再入力します。

seek: <value>: 無効な値

原因:

seek コマンドは無効な値を指定して入力されました。

処置:

有効な値を指定して seek コマンドを再入力します。

show attributes: <attribute_id>: リスト属性ではありません

原因:

show attributes コマンドは、リスト属性ではない attribute_id の instance_id を指定して入力されました。

処置:

リスト属性である attribute_id の instance_id を指定して show attributes コマンドを再入力します。

show attributes: <attribute_id>: 無効な属性 ID です

原因:

show attributes コマンドは無効な attribute_id を指定して入力されました。

処置:

有効な attribute_id を指定して show attributes コマンドを再入力します。

show attributes: <instance_id>: 無効なインスタンス ID です

原因:

show attributes コマンドは無効な instance_id を指定して入力されました。instance_id は、負でない整数のシーケンスから構成されない場合、またはそれが指定された属性に対して存在しない場合は無効です。

処置:

有効な instance_id を指定して show attributes コマンドを再入力します。

show attributes: <model_type_handle>: 無効なモデル タイプ ハンドルです

原因:

show attributes コマンドは無効な model_type_handle を指定して入力されました。

処置:

有効な model_type_handle を指定して show attributes コマンドを再入力します。

show: <landscape_handle>: 無効なランドスケープ ハンドルです

原因:

オプションの landscape_handle を使用する show コマンドが、無効な landscape_handle を指定して入力されました。

処置:

有効な landscape_handle を指定して show コマンドを再入力します。

show: <model_handle>: 無効なモデル ハンドルです

原因:

オプションの model_handle を使用する show コマンドが、無効な model_handle を指定して入力されました。

処置:

有効な model_handle を指定して show コマンドを再入力します。

show: 現在のモデルが定義されていません

原因:

オプションの model_handle を使用する show associations コマンドが入力されましたが、model_handle は指定されておらず、現在のモデルも定義されていませんでした。

処置:

model_handle と現在のモデルの両方を含めて、show associations コマンドを再入力します。

show alarms: no cause information available

原因:

show alarms コマンドが -x オプションと共に使用されましたが、想定される原因のテキスト メッセージが含まれる CA Spectrum アラーム ファイルが利用可能ではありません。

処置:

show alarms コマンドで -x オプションを指定して、アラームの想定される原因や展開されたイベント メッセージを表示するには、ローカル サーバに OneClick がインストールされ、環境変数 SPECROOT が Support ルート ディレクトリ (SG-Support) に設定されている必要があります。たとえば、SG-Support ファイルが以下のディレクトリにある場合：

/usr/spectrum/SG-Support、SPECROOT を /usr/spectrum に設定します。

show children: <relation>: 無効な関係です

原因:

show children コマンドは無効な関係を指定して入力されました。

処置:

有効な関係を指定して show children コマンドを再入力します。

show events: no event format information available

原因:

show events コマンドが -x オプションと共に使用されましたが、イベント形式のテキスト メッセージが含まれる CA Spectrum イベント ファイルが利用可能ではありません。

処置:

show events コマンドで -x オプションを指定して、アラームの想定される原因や展開されたイベント メッセージを表示するには、ローカル サーバに OneClick がインストールされ、環境変数 SPECROOT が Support ルート ディレクトリ (SG-Support) に設定されている必要があります。たとえば、SG-Support ファイルが以下のディレクトリにある場合：

/usr/spectrum/SG-Support、SPECROOT を /usr/spectrum に設定します。

show parents: <relation>: 無効な関係です

原因:

show parents コマンドは無効な関係を指定して入力されました。

処置:

有効な関係を指定して show parents コマンドを再入力します。

show rules: <relation>: 無効な関係です

原因:

show rules コマンドは無効な関係を指定して入力されました。

処置:

有効な関係を指定して show rules コマンドを再入力します。

show inheritance: <model_type_handle>: 無効なモデル タイプ ハンドルです

原因:

show inheritance コマンドは無効な model_type_handle を指定して入力されました。

処置:

有効な model_type_handle を指定して show inheritance コマンドを再入力します。

stopShd: VnmShd が実行中ではありません

原因:

CLI ローカル サーバが実行されていないときに、stopShd を実行しようとしました。

処置:

CLI ローカル サーバを起動します。

stopShd: 失敗しました

原因:

stopShd コマンドが失敗しました。

処置:

再度接続してみて、次に stopShd を実行します。 これで問題が解決しない場合、VnmShd プロセスを手動で強制終了します。

update: <attribute_id>: Attribute not writable

原因:

書き込み可能ではないモデル属性を更新しようとしたので、更新は発生しませんでした。

処置:

更新するモデル属性が書き込み可能であることを確認した後、update コマンドを再入力します。

update: <attribute_id>: 無効な属性 ID です

原因:

update コマンドが無効な attribute_id を指定して入力されたので、更新は発生しませんでした。

処置:

有効な attribute_id を指定して update コマンドを再入力します。

update: <attribute_id>: 共有属性ではありません

原因:

update コマンドがモデル タイプに対して使用され、共有属性ではない attribute_id が入力されました。

処置:

今回は共有属性の attribute_id を使用して、モデル タイプに対して update コマンドを再入力します。

update: <instance_id>: 無効なインスタンス ID です

原因:

update コマンドが無効な instance_id を使用して入力されたので、更新は発生しませんでした。

処置:

有効な instance_id を指定して update コマンドを再入力します。

update: <landscape_handle>: 無効なランドスケープ ハンドルです

原因:

update コマンドが無効な landscape_handle を指定して入力されたので、更新は発生しませんでした。

処置:

有効な landscape_handle を指定して update コマンドを再入力します。

update: <model_handle>: 無効なモデル ハンドルです

原因:

update コマンドが無効な model_handle を指定して入力されたので、更新は発生しませんでした。

処置:

有効な model_handle を指定して update コマンドを再入力します。

update: <model_type_handle>: 無効なモデル タイプ ハンドルです

原因:

update コマンドが無効な model_type_handle を指定して入力されたので、更新は発生しませんでした。

処置:

有効な model_type_handle を指定して update コマンドを再入力します。

update: <value>: 無効な値

原因:

update コマンドが無効な値を指定して入力されたので、更新は発生しませんでした。

処置:

有効な値を指定して update コマンドを再入力します。

update: <action_code>: 無効なアクション コードです

原因:

update コマンドが無効な action_code を指定して入力されたので、更新は発生しませんでした。

処置:

有効な action_code を指定して update を再入力します。

VnmShd: エラー: SpectroSERVER に接続できませんでした

原因:

CLI ローカル サーバは SpectroSERVER への接続に失敗しました。

処置:

SpectroSERVER が実行されていることを確認します。

VnmShd: エラー: SpectroSERVER との接続が切断されました

原因:

CLI ローカル サーバは、接続されていた SpectroSERVER が終了したことを検出して終了します。

処置:

SpectroSERVER を再起動し、CLI ローカル サーバを実行します。

付録 C: UNIX から DOS への変換

UNIX プラットフォームでは、CLI コマンドは、通常、ターミナル ウィンドウで UNIX コマンドと共に使用されます。 Windows プラットフォームでは、ネイティブ DOS ウィンドウで DOS コマンドと共に CLI コマンドを使用できます。 この付録では、一般的に用いられている UNIX コマンドと、それらに相当する DOS コマンドのリストを示します。

注: この付録は UNIX コマンドと DOS コマンドのクイック リファレンスであり、すべてを網羅したリストではありません。 コマンドとその機能の詳細については、使用している UNIX、DOS、または Windows のドキュメントを参照してください。

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

UNIX	DOS
format	fdisk
format->analyze	scandisk
fsck	chkdsk
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