

### 一般的な CLI コマンド

**alias:** コマンドにユーザ定義の別名を付ける

```
alias alias-name defn-body
定義済みエイリアスの一覧
alias [ alias-name ]
```

**capture:** コマンド出力を文字列にキャプチャ

```
capture [ -out | -err | -both ]
[ -f filename ] command
```

**dlappend:** リスト要素を TotalView 変数に追加

```
dlappend variable-name value [ ... ]
```

**dset:** CLI の状態変数を生成、変更、閲覧

```
dset debugger-var value
CLI の状態変数を閲覧
dset [ debugger-var ]
```

CLI の状態変数にデフォルト値を設定

```
dset -set_as_default debugger-var value
```

**dunset:** 特定の CLI 変数をデフォルト値に戻す

```
dunset debugger-var
全ての CLI 変数をデフォルト変数に戻す
dunset -all
```

**help:** ヘルプ情報を表示する

```
help [ topic ]
```

**stty:** ターミナルのプロパティを設定

```
stty [ stty-args ]
```

**unalias:** 特定のエイリアスを削除

```
unalias alias-name
全てのエイリアスを削除
unalias -all
```

### CLI の初期化と終了、プロセスを CLI に追加

**dattach:** プロセスにアタッチして管理下に置く

```
dattach [ -g gid ] [ -r hname ] [ -rank num ]
[ -replay | -no_replay ]
[ -go | -halt ]
```

```
[ -ask_attach_parallel
| -no_attach_parallel ]
[ -c corefile-name ] [ -e filename ]
[ pid-list ]
[ -parallel_attach_subset
subset_specification ]
```

**ddetach:** プロセスからデタッチする

```
ddetach
```

**ddlopen:** 共有ライブラリをロードする

```
ddlopen [ -now | -lazy ] [ -local | -global ]
```

```
[ -mode int ] filespec
```

共有ライブラリの情報を表示

```
ddlopen [ -list dll-ids... ]
```

**dgroups:** グループにメンバー追加

```
dgroups -add [ -g gid ] [ id-list ]
```

グループを消去

```
dgroups -delete [ -g gid ]
```

スレッド/プロセスとグループの共通部分を抜き出す

```
dgroups -intersect [ -g gid ] [ id-list ]
```

グループの情報を表示

```
dgroups [ -list ] [ pattern ]
```

新しいグループを作成

```
dgroups -new [ thread_or_process ]
```

```
[ -g gid ] [ id-list ]
```

グループからメンバーを削除

```
dgroups -remove [ -g gid ] [ id-list ]
```

**dkill:** プロセスの実行を終了

```
dkill [ -remove ]
```

**dload:** デバッグ情報をロード

```
dload [ -g gid ] [ -mpi starter ] [ -r hname ]
```

```
[ -replay | -no_replay ]
```

```
[ -env variable=value ]... [ -e ]
```

```
executable
```

```
[ -parallel_attach_subset
```

```
subset_specification ]
```

**drerun:** プロセスをリスタート

```
drerun [ cmd_arguments ] [ < infile ]
```

```
[ > [ > ] [ & ] outfile ]
```

```
[ 2> [ > ] errfile ]
```

**drun:** プロセスをスタート/リスタート

```
drun [ cmd_arguments ] [ < infile ]
```

```
[ > [ > ] [ & ] outfile ]
```

```
[ 2> [ > ] errfile ]
```

**dsession:** セッションをロード

```
dsession [ -load session_name ]
```

**exit:** CLI セッションを終了

```
exit [ -force ]
```

**quit:** CLI セッションを終了

```
quit [ -force ]
```

### プログラムの情報表示

**dcalltree:** パラレルバックトレースを表示

```
dcalltree [ -data pbv_data_array ]
[ -show_details ] [ -hide_backtrace ]
[ -sort columns ] [ -saveAsCSV filename ]
[ -saveAsDot filename ]
```

**dtdown:** コールスタックを下に移動

```
dtdown [ num-levels ]
```

**dexamine:** メモリの内容を表示

```
dexamine [ -column_count cnt ]
[ -count cnt ] [ -data_only ]
[ -show_chars ] [ -string_length len ]
[ -format fmt ] [ -memory_info ]
[ -wordsize size ]
variable_or_expression
```

**dflush:** 保留された expression evaluation の先頭を除去

```
dflush
```

全ての保留された *dprint* を除去

```
dflush -all
```

*susp-eval-id* までの保留された *dprint*

*evaluations* を除去

```
dflush susp-eval-id
```

**dga:** global array 変数を表示

```
dga [ -lang lang_type ]
[ handle_or_name ] [ slice ]
```

**dlist:** 現在の位置周辺にあるコードを表示

```
dlist [ -n num-lines ]
```

指定された位置周辺にあるコードを表示

```
dlist source-loc [ -n num-lines ]
```

現在の PC 周辺のコードを表示

```
dlist -e [ -n num-lines ]
```

**dmstat:** メモリ使用状況を表示

```
dmstat
```

**dprint:** 変数や expression の値を表示

```
dprint [ -nowait ] [ -slice slice_expr ]
[ -stats [ -data ] ]
variable_or_expression
```

**dptsets:** P/T expression の Tcl 配列内にあるプロセスやスレッドの状態を表示

```
dptsets [ ptset_array ]
```

**dstatus:** プロセス/スレッドの状態を表示

```
dstatus
```

**dup:** コールスタックを上に移動

```
dup [ num-levels ]
```

**dwhat:** 変数や関数名、シンボル情報を表示

```
dwhat symbol-name
```

**dwwhere:** コールスタック内での位置を表示

```
dwwhere [ -level num-levels ] [ num-levels ]
[ -args ] [ -locals ] [ -registers ]
[ -show_image ] [ -noshow_pc ]
[ -noshow_fp ]
全てのコールスタックを表示(デフォルト)
dwwhere -all [ -args ] [ -locals ]
[ -registers ] [ -show_image ]
[ -noshow_pc ] [ -noshow_fp ]
```

### 実行を制御

**dcont:** 実行を再開し、停止するのを待つ

```
dcont
```

**dfocus:** 以降の CLI コマンドの対象を変更、または現在の値を返す

```
dfocus [ p/t-set ]
```

この P/T セットに対してコマンドを実行

```
dfocus p/t-set command
```

**dgo:** 対象のプロセスで実行を再開

```
dgo
```

**dhalt:** プロセスの実行を保留

```
dhalt
```

**dhistory:** ReplayEngine に関する操作

```
dhistory [ -info ] [ -enable ] [ -disable ]
[ -create_bookmark [ comment ] ]
[ -goto_bookmark ID ] [ -go_live ]
[ -show_bookmarks ]
[ -delete_bookmark ID ]
```

Replay の履歴をファイルに保存

```
[ -save [ recording-file ] ]
```

Deprecated なオプション

```
[ -get_time ] [ -go_time time ]
```

**dhold:** プロセスをホールド

```
dhold -process
```

スレッドをホールドする

```
dhold -thread
```

**dnnext:** ソースをステップ実行し、関数をまたぐ

```
dnnext [ -back ] [ num-steps ]
```

**dnnexti:** アセンブリをステップ実行し、関数をまたぐ

```
dnnexti [ -replay | -no_replay ] [ num-steps ]
```

**dout:** 現在の関数を呼び出した直後の位置まで実行

```
dout [ -back ] [ frame-count ]
```

**dstep:** ソースをステップ実行し、関数の中へ

```
dstep [ -back ] [ num-steps ]
```

**dstepi:** アセンブリをステップ実行し、関数の中へ

<b>dstepi</b> [ <b>-back</b> ] [ <i>num-steps</i> ]
<b>dunhold:</b> プロセスをリリース
<b>dunhold</b> <b>-process</b> スレッドをリリース
<b>dunhold</b> <b>-thread</b>
<b>duntil:</b> 特定の位置まで実行
<b>duntil</b> [ <b>-back</b> ] <i>line-number</i> あるアドレスまで実行
<b>duntil</b> [ <b>-back</b> ] <b>-address</b> <i>addr</i> ある関数に入るまで実行
<b>duntil</b> <i>proc-name</i>
<b>dwait:</b> プロセス停止までコマンド入力をブロック
<b>dwait</b>
<b>dworker:</b> スレッドを worker group に追加/削除
<b>dworker</b> { <i>number</i>   <i>boolean</i> }

## アクションポイント

**dactions:** アクションポイントについての情報を表示

```
dactions [ ap-id-list ] [ -at source-loc ]
[ -enabled | -disabled ]
[ -enabled_blocks | -disabled_blocks ]
[ -block_images ]
[ -block_lines ]
```

アクションポイントをファイルに保存

```
dactions -save [ filename ]
```

保存したアクションポイントをロード

```
dactions -load [ filename ]
```

**dbarrier:** バリアブレイクポイントのある位置に作成

```
dbarrier source -loc [ -pending ]
[ -stop_when_hit
{ group | process | none } ]
[ -stop_when_done
{ group | process | none } ]
```

バリアブレイクポイントのあるアドレスに作成

```
dbarrier -address addr [ -pending ]
[ -stop_when_hit
{ group | process | none } ]
[ -stop_when_done
{ group | process | none } ]
```

**dbreak:** ブレイクポイントを作成  
ソース内やアドレス内に

<b>dbreak</b> <i>loc</i> [ <b>-p</b>   <b>-g</b>   <b>-t</b> ] [ [ <b>-l</b> <i>lang</i> ] <b>-e</b> <i>expr</i> ] [ <b>-pending</b> ]
<b>dbreak</b> <b>-address</b> <i>addr</i> [ <b>-p</b>   <b>-g</b>   <b>-t</b> ] [ [ <b>-l</b> <i>lang</i> ] <b>-e</b> <i>expr</i> ] [ <b>-pending</b> ]
<b>ddelete:</b> いくつかのアクションポイントを削除
<b>ddelete</b> <i>action-point-list</i> 全てのアクションポイントを削除
<b>ddelete</b> <b>-a</b>
<b>ddisable:</b> アクションポイントを有効化
<b>ddisable</b> <i>action-point-list</i> <b>ddisable</b> <b>-a</b>

<b>denable:</b> アクションポイントを無効化
<b>denable</b> <i>action-point-list</i> <b>denable</b> <b>-a</b>
<b>dwatch:</b> 変数やアドレスにウォッチポイントを割り当てる
<b>dwatch</b> <i>variable</i> [ <b>-length</b> <i>byte-count</i> ] [ <b>-p</b>   <b>-g</b>   <b>-t</b> ] [ [ <b>-l</b> <i>lang</i> ] <b>-e</b> <i>expr</i> ] [ <b>-t</b> <i>type</i> ]
<b>dwatch</b> <b>-address</b> <i>addr</i> <b>-length</b> <i>byte-count</i> [ <b>-p</b>   <b>-g</b>   <b>-t</b> ] [ [ <b>-l</b> <i>lang</i> ] <b>-e</b> <i>expr</i> ] [ <b>-t</b> <i>type</i> ]

## 一部のプラットフォーム用

<b>dcuda:</b> CUDA スレッドのデバッグを許可
<b>dcuda</b> { <b>block</b> [( <i>x</i> , <i>y</i> )]   <b>thread</b> [( <i>x</i> , <i>y</i> , <i>z</i> )] } <b>dcuda</b> { <b>kernel</b> }
<b>dcuda</b> { <b>device</b> [< <i>n</i> >]   <b>sm</b> [< <i>n</i> >] } <b>warp</b> [< <i>n</i> >]   <b>lane</b> [< <i>n</i> >] }
<b>dcuda</b> { <b>info-system</b>   <b>info-device</b>   <b>info-sm</b>   <b>info-warp</b>   <b>info-lane</b> }
<b>dcuda</b> { <b>focus</b> ( <i>Bx</i> , <i>By</i> , <i>Bz</i> ),( <i>Tx</i> , <i>Ty</i> )   <b>hwfocus</b> < <i>D/S/W/L</i> > }

<b>spurs:</b> GDB の SPU Runtime System (SPU) ライブラリ使用時の情報を管理 変数 <code>OBJECT_SEARCH_PATH</code> にディレクトリ追加
<b>spurs</b> <b>add</b> [ <b>directory</b> <i>directory-list ...</i> ]

## その他のコマンド

<b>dassign:</b> スカラー変数の値を変更
<b>dassign</b> <i>target value</i>
<b>dcache:</b> リモートライブラリのキャッシュを削除
<b>dcache</b> <b>-flush</b>
<b>dcheckpoint:</b> チェックポイントを作成

<b>dcheckpoint</b> [ <i>after_checkpointing</i> ] SGI IRIX 上で
[ <b>-by</b> <i>process_set</i> ] [ <b>-no_park</b> ] [ <b>-force</b> ] [ <b>-no_preserve_ids</b> ] [ <b>-ask_attach_parallel</b>   <b>-no_attach_parallel</b> ] <i>name</i>
IBM AIX 上で
<b>dcheckpoint</b> [ <b>-delete</b>   <b>-halt</b> ]
<b>dheap:</b> メモリデバッグ用のコマンド
<b>drestart:</b> チェックポイントをリスタート
AIX 上で
<b>drestart</b> [ <b>-halt</b> ] [ <b>-g</b> <i>gid</i> ] [ <b>-r</b> <i>host</i> ] [ <b>-no_same_hosts</b> ]
SGI 上で
<b>drestart</b> [ <i>process-state</i> ] [ <b>-no_unpark</b> ] [ <b>-g</b> <i>gid</i> ] [ <b>-r</b> <i>host</i> ] [ <b>-ask_attach_parallel</b>   <b>-no_attach_parallel</b> ] [ <b>-no_preserve_ids</b> ] <i>checkpoint-name</i>

## NameSpace コマンド

**actionpoint:** Actionpoint のプロパティ取得/設定

```
TV::actionpoint action [ obj-id ] [ other-args ]
```

**dec2hex:** 0 進数を 16 進数に変換

```
TV::dec2hex number  
Reference Guide 参照
```

**dll:** 共有ライブラリの管理

```
TV::dll action [ dll-id-list ] [ other-args ]
```

**errorCodes:** 全てのエラーコードのタグを返す

```
TV::errorCodes  
エラー情報を返すか raise する  
TV::errorCodes number_or_tag  
[ -raise [ message ] ]
```

**expr:** `dprint` `-nowait` の値を操作する

```
TV::expr action [ susp-eval-id ] [ other-args ]
```

**focus\_groups:** フォーカス下のグループのリスト

```
TV::focus_groups
```

**focus\_processes:** フォーカス下のプロセスのリスト

```
TV::focus_processes [ -all | -group | -process | -thread ]
```

**focus\_threads:** フォーカス下のスレッドのリスト

```
TV::focus_threads [ -all | -group | -process | -thread ]
```

**group:** グループのプロパティを設定/取得

```
TV::group action [ object-id ] [ other-args ]
```

**hex2dec:** 10 進数に変換

```
TV::hex2dec number
```

**process:** プロセスのプロパティを設定/取得

```
TV::process action [ object-id ] [ other-args ]
```

**read\_symbols:** シンボル読み込み

ライブラリから

```
TV::read_symbols -lib lib-name-list
```

特定のスタックフレームに関連したライブラリから

```
TV::read_symbols -frame [ number ]
```

バックトレース内の全フレームから

```
TV::read_symbols -stack
```

**respond:** コマンドへのレスポンスを提供

```
TV::respond response command
```

**scope:** 内部スコープのプロパティを設定/取得

```
TV::scope action [ object-id ] [ other-args ]
```

**source\_process\_startup:** プロセスを

ロードする時に.tvd ファイルを"Sources"する

```
TV::source_process_startup process_id
```

**symbol:** TotalView のシンボル情報を取得/設定

```
TV::symbol action [ object-id ] [ other-args ]
```

**thread:** スレッドのプロパティを取得/設定

```
TV::thread action [ object-id ] [ other-args ]
```

**type:** 型のプロパティを取得/設定

```
TV::type action [ object-id ] [ other-args ]
```

**type\_transformation:** type

transformation を作成し、そのプロパティを examine する

```
TV::type_transformation action  
[ object-id ] [ other-args ]
```