Library Compiler™ Tool Commands

Version O-2018.06, June 2018

SYNOPSYS®

Copyright Notice and Proprietary Information

Â© 2018 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at
http://www.synopsys.com/Company/Pages/Trademarks.aspx.
All other product or company names may be trademarks of their respective owners.

Free and Open-Source Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com

Copyright Notice for the Command-Line Editing Feature

Â© 1992, 1993 The Regents of the University of California. All rights reserved. This code is derived from software contributed to Berkeley by Christos Zoulas of Cornell University.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

   This product includes software developed by the University of California, Berkeley and its contributors.
4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright Notice for the Line-Editing Library

Â© 1992 Simmule Turner and Rich Salz. All rights reserved.

This software is not subject to any license of the American Telephone and Telegraph Company or of the Regents of the University of California.

Permission is granted to anyone to use this software for any purpose on any computer system, and to alter it and redistribute it freely, subject to the following restrictions:

1. The authors are not responsible for the consequences of use of this software, no matter how awful, even if they arise from flaws in it.
2. The origin of this software must not be misrepresented, either by explicit claim or by omission. Since few users ever read sources, credits must appear in the documentation.
3. Altered versions must be plainly marked as such, and must not be misrepresented as being the original software. Since few users ever read sources, credits must appear in the documentation.
4. This notice may not be removed or altered.

Copyright Notice for the Nonlinear Optimization Library

Â© 2011 by Massachusetts Institute of Technology.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

1. The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
2. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Contents

add_to_collection

Adds objects to a collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

```
collection add_to_collection
   [-unique]
   collection1
   object_spec
```

Data Types

collection1	collection
object_spec	list

ARGUMENTS

-unique

Indicates that duplicate objects are to be removed from the resulting collection. By default, duplicate objects are not removed.

collection1

Specifies the base collection to which objects are to be added. This collection is copied to the result collection, and objects matching *object_spec* are added to the result collection. The *collection1* option can be the empty collection (empty string), subject to some constraints, as explained in the DESCRIPTION section.

object_spec

Specifies a list of named objects or collections to add.

If the base collection is heterogeneous, only collections can be added to it.

If the base collection is homogeneous, the object class of each element in this list must be the same as in the base collection. If it is not the same class, it is ignored. From heterogeneous collections in the *object_spec*, only objects of the same class of the base collection are added. If the name matches an existing collection, the collection is used. Otherwise, the objects are searched for in the database using the object class of the base collection.

The *object_spec* has some special rules when the base collection is empty, as explained in the DESCRIPTION section.

DESCRIPTION

The **add_to_collection** command allows you to add elements to a collection. The result is a new collection representing the objects in the *object_spec* added to the objects in the base collection.

Elements that exist in both the base collection and the *object_spec*, are duplicated in the resulting collection. Duplicates are not removed unless you use the *-unique* option. If the *object_spec* is empty, the result is a copy of the base collection.

If the base collection is homogeneous, the command searches in the database for any elements of the *object_spec* that are not collections, using the object class of the base collection. If the base collection is heterogeneous, all implicit elements of the *object_spec* are ignored.

When the *collection1* argument is the empty collection, some special rules apply to the *object_spec*. If the *object_spec* is non-empty, there must be at least one homogeneous collection somewhere in the *object_spec* list (its position in the list does not matter). The first homogeneous collection in the *object_spec* list becomes the base collection and sets the object class for the function. The examples show the different errors and warnings that can be generated.

The **append_to_collection** command has similar semantics as the **add_to_collection** command; however, the **append_to_collection** command can be much more efficient in some cases. For more information about the command, see the man page.

For background on collections and querying of objects, see the **collections** man page.

EXAMPLES

The following example from PrimeTime uses the **get_ports** command to get all of the ports beginning with 'mode' and then adds the "CLOCK" port.

```
pt_shell> set xports [get_ports mode*]
{"mode[0]", "mode[1]", "mode[2]"}
pt_shell> add_to_collection $xports [get_ports CLOCK]
{"mode[0]", "mode[1]", "mode[2]", "CLOCK"}
```

The following example from PrimeTime adds the cell u1 to a collection containing the SCANOUT port.

```
pt_shell> set so [get_ports SCANOUT]
{"SCANOUT"}
pt_shell> set u1 [get_cells u1]
{"u1"}
pt_shell> set het [add_to_collection $so $u1]
{"u1"}
pt_shell> query_objects -verbose $het
{"port:SCANOUT", "cell:u1"}
```

The following examples show how the **add_to_collection** command behaves when the base collection is empty. Adding two empty collections yields the empty collection. Adding an implicit list of only strings or heterogeneous collections to the empty collection generates an error message, because no homogeneous collections are present in the *object_spec* list. Finally, as long as one homogeneous collection is present in the *object_spec* list, the command succeeds, even though a warning message is generated. The example uses the variable settings from the previous example.

```
pt_shell> sizeof_collection [add_to_collection "" ""]
0

pt_shell> set A [add_to_collection "" [list a $het c]]
Error: At least one homogeneous collection required for argument 'object_spec'
        to add_to_collection when the 'collection' argument is empty (SEL-014)

pt_shell> add_to_collection "" [list a $het $sp]]
Warning: Ignored all implicit elements in argument 'object_spec'
         to add_to_collection because the class of the base collection
         could not be determined (SEL-015)
{"SCANOUT", "u1", "SCANOUT"}
```

SEE ALSO

```
append_to_collection(2)
collections(2)
query_objects(2)
remove_from_collection(2)
sizeof_collection(2)
```

alias

Creates a pseudo-command that expands to one or more words, or lists current alias definitions.

SYNTAX

```
string alias
[name]
[def]
```

Data Types

```
name      string
def       string
```

ARGUMENTS

name

Specifies a name of the alias to define or display. The name must begin with a letter, and can contain letters, underscores, and numbers.

def

Expands the alias. That is, the replacement text for the alias name.

DESCRIPTION

The **alias** command defines or displays command aliases. With no arguments, the **alias** command displays all currently defined aliases and their expansions. With a single argument, the **alias** command displays the expansion for the given alias name. With more than one argument, an alias is created that is named by the first argument and expanding to the remaining arguments.

You cannot create an alias using the name of any existing command or procedure. Thus, you cannot use **alias** to redefine existing commands.

Aliases can refer to other aliases.

Aliases are only expanded when they are the first word in a command.

EXAMPLES

Although commands can be abbreviated, sometimes there is a conflict with another command. The following example shows how to use **alias** to get around the conflict:

```
prompt> alias q quit
```

The following example shows how to use **alias** to create a shortcut for commonly-used command invocations:

```
prompt> alias include {source -echo -verbose}

prompt> alias rt100 {report_timing -max_paths 100}
```

After the previous commands, the command **include script.tcl** is replaced with **source -echo -verbose script.tcl** before the command is interpreted.

The following examples show how to display aliases using **alias**. Note that when displaying all aliases, they are in alphabetical order.

```
prompt> alias rt100
rt100  report_timing -max_paths 100

prompt> alias
include  source -echo -verbose
q  quit
rt100  report_timing -max_paths 100
```

SEE ALSO

unalias(2)

align_lib_objects

Sort and match multiple(>=2) Tcl collections based on one or more attributes, resulting in a list of collection and each collection is N-tuple (paired) list of grouped object from each input collection.

SYNTAX

```
collection align_lib_objects
   [-matching match_rule_list]
   collection1 collection2 ... collectionN
```

Data Types

```
collection1          collection
collection2          collection
collectionN          collection
match_rule_list      list
```

ARGUMENTS

collection1 collection2 ...collectionN

Specifies multiple (>=2) collections to be aligned. Only following library object types are supported.

```
library
cell
pin
pg_pin
timing
internal_power
leakage_power
ccsn_stage
```

-matching *match_rule_list*

Specifies a list of one or more application or user-defined attributes to use as align keys. following are two kinds of format for *match_rule_list*

```
-matching {attrib1 attrib2 ...}
-matching { + {attrib1 attrib2 ...} - {attrib3 attrib4 ...}}
```

The first format is to explictly provide attribute list to be used as align keys. The second format is to add or delete attributes from default list. * If the option is not specified * the object in the collection has name, the command uses name as align key. following class types are affected. * library * cell * pin * pg_pin

* the object in the collection does not have name, the command uses following default attributes as align keys.

```
*******************************************************************************
class type        *                    default match rule list                   *
*******************************************************************************
timing            *   related_pin    * when           * timing_sense * timing_type*
*******************************************************************************
internal_power    *   related_pin    * related_pg_pin  * when          *           *
*******************************************************************************
ccsn_stage        *   when           * related_node    *               *           *
*******************************************************************************
leakage_power     *   related_pg_pin * when            *               *           *
*******************************************************************************
```

DESCRIPTION

You can use the **align_lib_objects** command to sort and match objects in collections based on one or more attributes. NULL is for mismatching elements.

EXAMPLES

The following exmaple aligns a collection of pins and use pin name as key. the result is six collections and each collection has two pins.

```
lc_shell>set cell1 test/BUF/*
test/BUF/*
lc_shell>set cell2 test/CKGT/*
test/CKGT/*
lc_shell>set pinlist1 [get_lib_pins $cell1]
{"test/BUF/A", "test/BUF/X"}
lc_shell>set pinlist2 [get_lib_pins $cell2]
{"test/CKGT/CK", "test/CKGT/EN", "test/CKGT/Q", "test/CKGT/SE"}
lc_shell>set aligned_pinlist [align_lib_objects  $pinlist1 $pinlist2]
_sel4 _sel5 _sel6 _sel7 _sel8 _sel
lc_shell> set i 0
0
lc_shell>foreach pair $aligned_pinlist {
set element1 [index_collection $pair 0]
set element2 [index_collection $pair 1]
set ret1 [is_null_object $element1]
set ret2 [is_null_object $element2]
if {$ret1 == "1"} {
  set name1 "null"
} else {
  set name1 [get_object_name  $element1]
}
if {$ret2 == "1"} {
  set name2 "null"
} else {
  set name2 [get_object_name  $element2]
}
echo "pair $i:($name1, $name2)"
incr i
}
pair 0:(null, EN)
pair 1:(null, CK)
pair 2:(null, SE)
pair 3:(A, null)
pair 4:(null, Q)
pair 5:(X, null)
```

SEE ALSO

```
collections(2)
```

analyze_trend

analyze the trend of a list of float.

SYNTAX

```
analyze_trend
   float_list
   [-relative_tolerance relative_tolerance]
   [-absolute_tolerance absolute_tolerance]
```

Data Types

float_list	list
relative_tolerance	float
absolute_tolerance	float

ARGUMENTS

float_list

Specifies the sorted float list for analysis.

-relative_tolerance relative_tolerance

Specifies the relative tolerance for the float equal comparing, default is 0.01.

-absolute_tolerance absolute_tolerance

Specifies the absolute tolerance for the float equal comparing, default is 1e-6.

DESCRIPTION

This command analyzes the trend of the float data list, and returns the trend symbol of the data. The return trend symbo is listed below.

```
/:  monotonously increasing
```

```
\:  monotonously decreasing
^:  non-monotonous up
V:  non-monotonous down
-:  flat
M:  multiple peaks
W:  multiple troughs
N:  >=2 peaks with 2nd not turning low
u:  >=2 troughs with 2nd not turning up (shape V\)
X:  None of the above trends
```

The tolerance (absolute/relative) is for filter out small noise. Different tolerances may result in different trend shapes.

EXAMPLES

The following example shows trend analysis for a float array

```
lc_shell>analyze_trend {0.1 0.2 0.19 0.3}
N

lc_shell>analyze_trend {0.1 0.2 0.19 0.3} -relative_tolerance 0.1
/
```

SEE ALSO

```
get_scale(2)
```

append_to_collection

Adds objects to a collection and modifies a variable.

SYNTAX

```
collection append_to_collection
   [-unique]
   var_name
   object_spec
```

Data Types

```
var_name               collection
object_spec            list
```

ARGUMENTS

-unique

Indicates that duplicate objects are to be removed from the resulting collection. By default, duplicate objects are not removed.

var_name

Specifies a variable name. The objects matching *object_spec* are added into the collection referenced by this variable.

object_spec

Specifies a list of named objects or collections to add.

DESCRIPTION

The **append_to_collection** command allows you to add elements to a collection. This command treats the variable name given by the *var_name* option as a collection, and it appends all of the elements in *object_spec* to that collection. If the variable does not exist, it is created as a collection with elements from the *object_spec* as its value. If the variable does exist and it does not contain a collection, it is an error.

The result of the command is the collection that was initially referenced by the *var_name* option, or the collection created if the variable did not exist.

The **append_to_collection** command provides the same semantics as the common use of the **add_to_collection** command; however, this command shows significant improvements in performance.

An example of replacing the **add_to_collection** command with the **append_to_collection** command is provided below. For example,

```
set var_name [add_to_collection $var_name $objs]
```

Using the **append_to_collection** command, the command becomes:

```
append_to_collection var_name $objs
```

The **append_to_collection** command can be much more efficient than the **add_to_collection** command if you are building up a collection in a loop. The arguments of the command have the same restrictions as the **add_to_collection** command. For more information about these restrictions, see the **add_to_collection** man page.

EXAMPLES

The following example from PrimeTime shows how a collection can be built up using the **append_to_collection** command:

```
pt_shell> set xports
Error: can't read "xports": no such variable
        Use error_info for more info. (CMD-013)
pt_shell> append_to_collection xports [get_ports in*]
{"in0", "in1", "in2"}
pt_shell> append_to_collection xports CLOCK
{"in0", "in1", "in2", "CLOCK"}
```

SEE ALSO

```
add_to_collection(2)
foreach_in_collection(2)
index_collection(2)
remove_from_collection(2)
sizeof_collection(2)
```

apropos

Searches the command database for a pattern.

SYNTAX

```
string apropos
   [-symbols_only]
   pattern
```

Data Types

pattern	string

ARGUMENTS

-symbols_only

Searches only command and option names.

pattern

Searches for the specified *pattern*.

DESCRIPTION

The **apropos** command searches the command and option database for all commands that contain the specified *pattern*. The *pattern* argument can include the wildcard characters asterisk (*) and question mark (?). The search is case-sensitive. For each command that matches the search criteria, the command help is printed as though **help -verbose** was used with the command.

Whereas **help** looks only at command names, **apropos** looks at command names, the command one-line description, option names, and option value-help strings. The search can be restricted to only command and option names with the **-symbols_only** option.

When searching for dash options, do not include the leading dash. Search only for the name.

EXAMPLES

In the following example, assume that the **get_cells** and **get_designs** commands have the **-exact** option. Note that without the **-symbols_only** option, the first search picks up commands which have the string "exact" in the one-line description.

```
prompt> apropos exact
 get_cells            # Create a collection of cells
   [-exact]               (Wildcards are considered as plain characters)
   patterns               (Match cell names against patterns)

 get_designs          # Create a collection of designs
   [-exact]               (Wildcards are considered as plain characters)
   patterns               (Match design names against patterns)

 real_time            # Return the exact time of day

prompt> apropos exact -symbols_only
 get_cells            # Create a collection of cells
   [-exact]               (Wildcards are considered as plain characters)
   patterns               (Match cell names against patterns)

 get_designs          # Create a collection of designs
   [-exact]               (Wildcards are considered as plain characters)
   patterns               (Match design names against patterns)
```

SEE ALSO

```
help(2)
```

check_library

Performs consistency checks cross logic libraries, logic vs. physical libraries and physical libraries.

SYNTAX

```
status check_library
   [-logic_library_name logic_library_name_list]
   [-physical_library_name phys_library_name_list]
   [-cells cell_list]
```

Data Types

```
logic_library_name_list      list
physical_library_name_list   list
cell_list                    list
```

ARGUMENTS

-logic_library_name *logic_library_name_list*

Specifies the logic libraries (.db) to be checked. This can be in regexp, e.g. "*.db" in search_path. Other types of libraries such as .ILM are not supported.

For logic vs. logic library checking, specify two or more libraries, such as libraries at different corners. If you have set the **-compare {attribute value}** option in the **set_check_library_options** command, specify only two libraries.

If you have set the **-scaling** option in the **set_check_library_options** command, specify scaling libraries by the **create_scaling_lib_group** command.

For logic vs. logic library checking, the command first groups the libraries by cell set, and within each group or family the command checks consistencies across all libraries in the same group. The grouping is by majority (> 50%) of same name cells. For example, if lib1 has 9 cells and lib2 has 6, and 5 of them are the same, then they are grouped together.

-physical_library_name *phys_library_name_list*

Specifies the physical libraries (.frame) to be checked. You can specify multiple physical libraries.

For cross checking, specify both logic and physical libraries. If libraries are not specified explicitly in check_library, the ones loaded/opened at runtime will be used. The libraries to be checked are in the following priority:

1) explicitly specified in check_library command

2) logic libraries associated with the specified physical library

3) loaded or opened logic and physical libraries at runtime

`-cells` ***`cell_list`***

Specifies a list of cell names to be checked. If not specified, all cells in the libraries are checked.

DESCRIPTION

The **check_library** command checks and reports the items described below, based on the settings of the **set_check_library_options** command.

Use the **check_library** command to qualify the libraries before reading in a design.

This command checks library qualities in following areas:

- Logic vs. logic library consistency

Logic vs. Logic Library Consistency Checking

Logic vs. logic library consistency checks include:

- General or default checking at library, cell, pin, and timing group levels for missing cells, missing and mismatched pins or pg_pins, and timing arcs.
- Special checking for timing, noise, and power scaling
- Special checking for the UPF or multivoltage flow, such as pg_pins, power management cells, and power data.
- Special checking for the multicorner-multimode flow, such as operating conditions and power-down functions
- Library characterization and validation

For logic vs. logic library checking, at the end of checking the command reports a summary for each specified option. For example, for the **-mcmm** option, it reports:

Logic library consistency check FAILED for MCMM.

Logic vs. physical Library Consistency Checking

Logic vs. physical library consistency checks include:

1) default checks: missing cells, missing or mismatched pins/pg_pins in either DB or FRAME

2) cell areas between DB and FRAME

3) cell footprint vs. PR boundary

4) bus delimiters in the libraries.

For logic vs. physical library checking, after reporting issues in tables, at the end of checking, a summary message is printed out, for example,

Logic library is inconsistent with physical library (LIBCHK-220)

EXAMPLES

In the following example, the lib1.db logic library is checked for upf:

```
prompt> set_check_library_options -upf
1

prompt> check_library -logic_library_name lib1.db

#BEGIN_XCHECK_LIBRARY

Logic Library #1:
    Library name              lib1
    File name                 /home/goose/tmp/lib1.db
    Library type              pg_pin based db
    Library Version           1.000000
    Tool Created              J-2014.09
    Data Created              Thu Mar  9 14:21:29 2006
    Time unit                 1ns
    Capacitance unit          1000ff
    Leakage power unit        1mW
    Current unit              1mA
check_library options         -upf
Version                       J-2014.09-ALPHA5
Check date and time           Thu Jul 10 03:01:46 2014


#BEGIN_LIBSCREEN_UPF

Library#1 (lib1):
Warning: Cell 'isolation_cell' is missing 'backup_power' pg_pin, so it will become a black-box cell
Warning: Cell 'ls_1', pin 'A', The pin 'A' is missing the attribute 'input_voltage_range'.   (LBDB-
Warning: Cell 'ls_1', pin 'Y', The pin 'Y' is missing the attribute 'output_voltage_range'.   (LBDB
Warning: Cell 'ls_2', pin 'A', The pin 'A' is missing the attribute 'input_voltage_range'.   (LBDB-
Warning: Cell 'ls_2', pin 'Y', The pin 'Y' is missing the attribute 'output_voltage_range'.   (LBDB
Warning: Cell 'LVLLHX2M', pin 'A', The pin 'A' is missing the attribute 'input_voltage_range'.   (L
Warning: Cell 'LVLLHX2M', pin 'Y', The pin 'Y' is missing the attribute 'output_voltage_range'.   (
Warning: Cell 'LVLLHX8M', pin 'A', The pin 'A' is missing the attribute 'input_voltage_range'.   (L
Warning: Cell 'LVLLHX8M', pin 'Y', The pin 'Y' is missing the attribute 'output_voltage_range'.   (
Warning: Cell 'LVLLHEHX2M', pin 'A', The pin 'A' is missing the attribute 'input_voltage_range'.
Warning: Cell 'LVLLHEHX2M', pin 'Y', The pin 'Y' is missing the attribute 'output_voltage_range'.
Warning: Cell 'LVLLHEHX8M', pin 'A', The pin 'A' is missing the attribute 'input_voltage_range'.
Warning: Cell 'LVLLHEHX8M', pin 'Y', The pin 'Y' is missing the attribute 'output_voltage_range'.
Power management cell checking passed.

#END_LIBSCREEN_UPF
```

```
Information: List of cell classification (LIBCHK-312)
------------------------------------------------------------------------------
Library name                lib1(lib#1)
------------------------------------------------------------------------------
Total number                7
Inverter                    0
Buffer                      0
Level shifter               6
Differential level shifter  0
Isolation cell              1
Clock Isolation cell        0
Retention cell              0
Switch cell                 0
Always on cell              0
------------------------------------------------------------------------------
#BEGIN_XCHECK_LOGICCELLS


#END_XCHECK_LOGICCELLS

#BEGIN_XCHECK_LOGICPINS



#END_XCHECK_LOGICPINS

#BEGIN_XCHECK_LOGICPGPINS

Number of cells missing pg_pins in library 1:   0 (out of 7)

#END_XCHECK_LOGICPGPINS

#BEGIN_XCHECK_ARCS


#END_XCHECK_ARCS


Logic vs. logic library check summary:
Logic library consistency check PASSED for UPF.

#END_XCHECK_LIBRARY

1
```

In the following example, the default cross checks are performed between logic libraries a1.db and a2.db and physical library a.nlib:

```
prompt> check_library -logic_library_name "a1.db a2.db" -physical_library_name a.nlib
```

SEE ALSO

```
report_check_library_options(2)
set_check_library_options(2)
```

check_license

Checks the availability of a license for a feature.

SYNTAX

```
status check_license
   feature_list
```

Data Types

```
feature_list     list
```

ARGUMENTS

feature_list

Specifies the list of features to be checked. If more than one feature is specified, they must be enclosed in curly braces ({}). By looking at your key file, you can determine all of the features licensed at your site.

DESCRIPTION

The **check_license** command checks on a license for the named features. It does not check out the license.

The **list_licenses** command provides a list of the features that you are currently using.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

This example checks on the license for the multivoltage feature:

```
prompt> check_license {Galaxy-MV}
```

SEE ALSO

```
get_license(2)
license_users(2)
list_licenses(2)
remove_license(2)
```

check_script

Statically check a script using the a static Tcl syntax checker based on TclPro Procheck and the Synopsys checking extensions for this tool.

SYNTAX

```
string snpsTclPro::check_script [-no_tclchecker_warning]
   [-quiet]
   [-onepass]
   [-verbose]
   [-suppress messageID]
   [-application appName]
   [-W1]
   [-W2]
   [-W3]
   [-Wall]
   filePattern

string messageID
string appName
string filePattern
```

ARGUMENTS

-no_tclchecker_warning

Don't warn that TclDevKit cannot be found

-quiet

prints minimal error information

-onepass

perform a single pass without checking proc args

-verbose

prints summary information

-suppress *messageIDs*

prevent given messageIDs from being printed

-application *appName*

Check script against application other than this one

-W1

print only error messages

-W2

print only error and usage warnings

-W3

print all errors and warnings

-Wall

print all types of messages (same as W3)

filePattern

file(s) to be checked

USING THE PACKAGE

The **check_script** command is in the snpsTclPro Tcl package, and all of the commands in this package are defined in the namespace snpsTclPro. The way to use this command is to load the package, which will import the commands it exports into the global namespace. This is shown below.

```
shell> package require snpsTclPro
1.0
```

These commands can be added to the .synopsys setup file for the system, user, or current directory, or the commands can be typed when the package is needed.

DESCRIPTION

The **check_script** command simplifies running the TclDevKit tclchecker program. This application is available for purchase from ActiveState (http://www.activestate.com). If the TclDevKit is not available then there is a limited Synopsys Tcl syntax checker that is used by this command. The Synopsys version of this code does not understand Tcl8.4 constructs, and some other checking is limited.

These checkers will check builtin Tcl commands, as well as the Synopsys extension commands which are found in the script. **check_script** will locate the Synopsys checking extensions for the application and will then invoke the checker. The limited checker does not require any external license. It is part of the Synopsys installation.

The **check_script** command will raise a Tcl error if the checker detects errors in the scripts being checked.

The **check_script** command is provided as a convenience to streamline the use of the syntax checker within Synopsys applications. The checker can also be called directly on scripts, without requiring a license for the Synopsys application whose script is being checked. This checker script is available in the auxx/tcllib/bin/check_script in the installation directory of your application. When run standalone the -application option must be specified.

Synopsys Checker Extensions

The Synopsys extensions to the TclPro checker define a number of new warnings and errors. These messages are listed below, along with a short explanation of them.

Error **MisVal**
An option to a command which requires a value is missing its value.

Error **MisReq**
A required option or argument was not specified.

Error **Excl**
Two options which are mutually exclusive were both specified.

Error **ExtraPos**
An extra positional argument was specified to a command.

Error **BadRange**
The value specified for a given argument was outside of the legal range.

Error **UnkCmd**
The command specified is not known by the application.

Error **UnkOpt**
The specified option is not legal for the command.

Error **AmbOpt**
The option specified is not complete and matches 2 or more options for the command.

Warning **NotLit**
This warning indicates that the argument specified to an option was not a literal and therefore could not be checked. This message is suppressed by default. It can be enabled by setting the environment variable SNPS_TCL_NOLITERALWAN to true.

Warning **DupIgn**
An option was specified multiple times to the command, and the first value specified will be the one used. The other values will be ignored.

Warning **DupOver**
An option was specified multiple times to the command, and the last value specified will be the one used. The other values will be ignored.

EXAMPLES

```
shell> check_script myscript.tcl

Loading snps_tcl.pcx...
Loading coreConsultant.pcx...
scanning: /u/user1/myscript.tcl
checking: /u/user1/myscript.tcl
test.tcl:3 (warnVarRef) variable reference used where variable name expected
set $a $b
    ^
test.tcl:5 (SnpsE-MisReq) Missing required positional options for foreach_in_collection: body
foreach_in_collection x {
}
^
test.tcl:9 (SnpsE-BadRange) Value -1 for 'index_collection index' must be >= 0
index_collection $a -1
child process exited abnormally
Error: Errors found in script.
        Use error_info for more info. (CMD-013)

shell> check_script -Wall anotherScript.tcl

Loading snps_tcl.pcx...
scanning: /u/user1/anotherScript.tcl
checking: /u/user1/anotherScript.tcl
```

CAVEATS

The Synopsys extension messages cannot be suppressed.

Aliases and abbreviated commands will be flagged as undefined procedures.

SEE ALSO

debug_script(2)
package(2)
namespace(2)

compare_collections

Compares the contents of two collections. If the same objects are in both collections, the result is "0" (like string compare). If they are different, the result is nonzero. The order of the objects can optionally be considered.

SYNTAX

```
int compare_collections
   [-order_dependent]
   collection1
   collection2
```

Data Types

```
   collection1            collection
   collection2            collection
```

ARGUMENTS

-order_dependent

Indicates that the order of the objects is to be considered; that is, the collections are considered to be different if the objects are ordered differently.

collection1

Specifies the base collection for the comparison. The empty string (the empty collection) is a legal value for the *collection1* argument.

collection2

Specifies the collection with which to compare to *collection1*. The empty string (the empty collection) is a legal value for the *collection2* argument.

DESCRIPTION

The **compare_collections** command is used to compare the contents of two collections. By default, the order of the objects does not matter, so that a collection of cells u1 and u2 is the same as a collection of the cells u2 and u1. By using the **-order_dependent** option, the order of the objects is considered.

Either or both of the collections can be the empty string (the empty collection). If two empty collections are compared, the comparison succeeds (that is, **compare_collections** considers them identical), and the result is "0".

EXAMPLES

The following example from PrimeTime shows a variety of comparisons. Note that a result of "0" from **compare_collections** indicates success. Any other result indicates failure.

```
pt_shell> compare_collections [get_cells *] [get_cells *]
0
pt_shell> set c1 [get_cells {u1 u2}]
{"u1", "u2"}
pt_shell> set c2 [get_cells {u2 u1}]
{"u2", "u1"}
pt_shell> set c3 [get_cells {u2 u4 u6}]
{"u2", "u4", "u6"}
pt_shell> compare_collections $c1 $c2
0
pt_shell> compare_collections $c1 $c2 -order_dependent
-1
pt_shell> compare_collections $c1 $c3
-1
```

The following example builds on the previous example by showing how empty collections are compared.

```
pt_shell> set c4 ""
pt_shell> compare_collections $c1 $c4
-1
pt_shell> compare_collections $c4 $c4
0
```

SEE ALSO

```
collections(2)
```

copy_collection

Duplicates the contents of a collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

```
collection copy_collection
   collection1
```

Data Types

collection1	collection

ARGUMENTS

collection1

Specifies the collection to be copied. If an empty string is used for the *collection1* argument, the command returns the empty string (a copy of the empty collection is an empty collection).

DESCRIPTION

The **copy_collection** command is an efficient mechanism for creating a duplicate of an existing collection. It is more efficient and almost always sufficient to simply have more than one variable referencing the same collection. For example, if you create a collection of ports in PrimeTime and save a reference to it in a **c1** variable, assigning the value of the **c1** variable to another **c2** variable creates a second reference to the same collection:

```
pt_shell> set c1 [get_cells "U1*"]
{"U1", "U10", "U11", "U12"}
pt_shell> set c2 $c1
{"U1", "U10", "U11", "U12"}
```

This has not copied the collection. There are now two references to the same collection. If you change the *c1* variable, the *c2* variable continues to reference the same collection:

```
    pt_shell> set c1 [get_cells "block1"]
    {"block1"}
    pt_shell> query_objects $c2
    {"U1", "U10", "U11", "U12"}
```

There might be instances when you really do need a copy. In those cases, the **copy_collection** command is used to create a new collection that is a duplicate of the original.

EXAMPLES

The following example from PrimeTime shows the result of copying a collection. Functionally, it is not much different that having multiple references to the same collection.

```
    pt_shell> set c1 [get_cells "U1*"]
    {"U1", "U10", "U11", "U12"}
    pt_shell> set c2 [copy_collection $c1]
    {"U1", "U10", "U11", "U12"}
    pt_shell> unset c1
    pt_shell> query_objects $c2
    {"U1", "U10", "U11", "U12"}
```

SEE ALSO

```
collections(2)
```

cputime

Reports the CPU time in seconds.

SYNTAX

```
status cputime
   [-all]
   [-verbose]
```

ARGUMENTS

-all

Reports the total CPU time of the main process and its child processes. This is the default; this option is kept for backward compatibility.

-verbose

Reports in detail the CPU time and elapsed (wall-clock) time of the main process and each child process, including placement, extraction, and routing.

DESCRIPTION

The **cputime** command reports the CPU time in seconds. If you specify **-verbose**, it reports the CPU time and elapsed (wall-clock) time for the main process and each child process. The runtime numbers are collected from the operating system.

When you use the **set_host_options** command, the tool runs multiple threads. Currently the operating system measures the CPU time by adding the usage for all threads. It does not take into account parallelization of the threads. This means that the report might show a larger CPU time than elapsed time when you use **set_host_options**. The elapsed time (wall-clock time) is more appropriate for measuring multicore runtime.

The elapsed time depends on many external factors and can vary for every session. It depends on factors such as the I/O traffic, the network traffic, RAM and swap usage, and the other processes running on the same machine. When the elapsed time is much longer than the CPU time, it might point out an inefficiency of the computing environment, such as insufficient memory, too many processes running on the same machine, or a slow network. The only way to make the elapsed time close to the CPU time is to run only one session on a machine with enough RAM and using the local disk.

EXAMPLES

The following example shows the default command output:

```
prompt> cputime
1202
```

The following example shows the output when using the **-verbose** option:

```
prompt> cputime -verbose
Main process  CPU: 1202 s (0.33 hr) Elapse: 1800 s (0.50 hr)
Child "placement" called 3 times, total CPU: 200 s (0.06 hr)
  Elapse: 250 s (0.07 hr)
Child "extraction" called 2 times, total CPU: 100 s (0.03 hr)
  Elapse: 150 s (0.04 hr)
Total CPU:  1502 s ( 0.42 hr) Elapse: 1800 s ( 0.50 hr)

1523
```

SEE ALSO

```
mem(2)
set_host_options(2)
monitor_cpu_memory(3)
.prod psyn icc
PSYN-508(n)
.prod all
```

create_command_group

Creates a new command group.

SYNTAX

```
string create_command_group [-info info_text]
group_name
```

ARGUMENTS

-info *info_text*

Help string for the group

group_name

Specifies the name of the new group.

DESCRIPTION

The **create_command_group** command is used to create a new command group, which you can use to separate related user-defined procedures into functional units for the online help facility. When a procedure is created, it is placed in the "Procedures" command group. With the **define_proc_attributes** command, you can move the procedure into the group you created.

The *group_name* can contain any characters, including spaces, as long as it is appropriately quoted. If *group_name* already exists, **create_command_group** quietly ignores the command. The result of **create_command_group** is always an empty string.

EXAMPLES

The following example demonstrates the use of the **create_command_group** command:

```
prompt> create_command_group {My Procedures} -info "Useful utilities"

prompt> proc plus {a b} { return [expr $a + $b] }

prompt> define_proc_attributes plus -command_group "My Procedures"

prompt> help
My Procedures:
  plus

 ...
```

SEE ALSO

```
define_proc_attributes(2)
help(2)
proc(2)
```

create_scaling_lib_group

Specifies a scaling library group to qualify by check_library.

SYNTAX

```
status create_scaling_lib_group
   library_list
   [-name group_name]
   [-exact_match_only]
   [-exclude_voltage_names voltage_name_list]
```

Data Types

```
library_list      list
```

ARGUMENTS

library_list

Specifies a list of library file names in the scaling library group by cell set. These libraries in the group should have the same set of cells.

-name *group_name*

Specifies a group name to associate with the library group. This is used especially when multiple groups are defined.

-exact_match_only

Specifies the library_list to be used for exact match only without scaling. If not specified, the group is for scaling.

-exclude_voltage_names *voltage_name_list*

Excludes these voltage_names in scaling. For example, -exclude_voltage_names {VBP} excludes bias pin voltage_name VBP to reduce scaling dimension. Normally, a voltage_name that has dependency on other rails can be ignored. This may include, but not limited to, bias pins. By default, all the rail voltage_names that are in use are taken as independent dimension for scaling.

DESCRIPTION

The **create_scaling_lib_group** command specifies a scaling library group to qualify by check_library. The scaling library groups after qualified can be used by downstream tools such as PT for library scaling.

To remove a defined scaling library group, specify an empty library list with the group name -name to be removed:

create_scaling_lib_group -name grp2 ""

To remove all the defined scaling library groups, specify an empty library list without -name:

create_scaling_lib_group ""

EXAMPLES

In the following example, a library group consisting of 3 libraries is specified in the group name of grp1:

```
prompt> create_scaling_lib_group -name grp1 "1.db 2.db 3.db"
```

SEE ALSO

```
check_library(2)
write_scaling_lib_group(2)
```

date

Returns a string containing the current date and time.

SYNTAX

```
string date
```

DESCRIPTION

The **date** command generates a string containing the current date and time, and returns that string as the result of the command. The format is fixed as follows:

```
ddd mmm nn hh:mm:ss yyyy
```

Where:

```
ddd is an abbreviation for the day
mmm is an abbreviation for the month
nn is the day number
hh is the hour number (24 hour system)
mm is the minute number
ss is the second number
yyyy is the year
```

The **date** command is useful because it is native. It does not fork a process. On some operating systems, when the process becomes large, no further processes can be forked from it. With it, there is no need to call the operating system with **exec** to ask for the date and time.

EXAMPLES

The following command prints the date.

```
prompt> echo "Date and time: [date]"
Date and time: Thu Dec  9 17:29:51 1999
```

SEE ALSO

exec(2)

debug_script

Debug a script using the TclDevKit Tcl debugger.

SYNTAX

```
string debug_script script [port] [hostname]
```

ARGUMENTS

script

This specifies the Tcl script(s) to be debugged. This script will be sourced and instrumented for debugging.

port

Specifies the port number that the debugger is listening on for a remote debugging application. By default the debugger will be launched on an available port.

hostname

Specifies the host where Prodebug is already running. This is used to connect to an existing remote Prodebug session.

::env(SNPS_TCLPRO_HOME)

This Tcl variable is used to specify the root of the TclPro installation. It is initialized from the environment variable SNPS_TCLPRO_HOME in the user's environment.

USING THE PACKAGE

The **debug_script** command is in the snpsTclPro Tcl package, and all of the commands in this package are defined in the namespace snpsTclPro. The way to use this command is to load the package, which will import the commands it exports into the global namespace. This is shown below.

```
   shell> package require snpsTclPro
   1.0
```

These commands can be added to the .synopsys setup file for the system, user, or current directory, or the commands can be typed when the pacakge is needed.

DESCRIPTION

The **debug_script** command simplifies the use of the TclDevKit Tcl debugger available from ActiveState (http://www.activestate.com). First it ensures a connection to the Prodebug debugger either by connecting to a running debug session (given a *hostname* and *port*), or by launching the debugger on *localhost* using an available port, and then connecting to it. If there is already an active connection to the debugger then that debugger will simply be used.

Once the connection to the debugger is set up, the specified script will be sourced and debugged. The default behavior for the debugger is to stop at the first line of the source'd script.

The debugger must be run using a "remote" debugging project if you want to connect the Synopsys tools to a running prodebug session. The port number in use by the debugger can be viewed via the Application tab of the "File->Project" Settings dialog.

EXAMPLES

```
# Launch the debugger and then debug the script myscript.tcl.
shell> debug_script myscript.tcl
Selected port 2576 on host localhost
launching prodebug

# now debug another script using the same session
shell> debug_script anotherScript.tcl

# debug myscript.tcl, connecting to the specified debugger session or
# creating one if the connection fails.
shell> debug_script myscript.tcl 2576 localhost
```

CAVEATS

The debugger currently kills the process that started the debugger when you exit, despite the preferences being set differently in the debugger.

When the Synopsys tool has spawned a debugger, that debugger must be exited before the Synopsys tool can exit. If this is not the case then the Synopsys tool will hang on exit, until the debugger is exited.

The **debug_script** command waits for the debug process to start up, before it tries to connect to the debugger. If **debug_script** times out before the debugger is started due to machine or network loading, simply close the debugger that was launched, and then increase the timeout setting with the command **set_debugger_start_timeout**, and then re-run the debugger. The **set_debugger_start_timeout** command takes a single argument which is the value of the timeout in miliseconds, and the default value for the timeout is 10000.

SEE ALSO

check_script(2)
package(2)
namespace(2)

define_proc_attributes

Defines attributes of a Tcl procedure, including an information string for help, a command group, a set of argument descriptions for help, and so on. The command returns the empty string.

SYNTAX

```
string define_proc_attributes
   proc_name
   [-info info_text]
   [-define_args arg_defs]
   [-define_arg_groups group_defs]
   [-command_group group_name]
   [-return type_name]
   [-hide_body]
   [-hidden]
   [-dont_abbrev]
   [-permanent]
```

Data Types

```
   proc_name      string
   info_text      string
   arg_defs       list
   group_defs     list
   group_name     string
   type_name      string
```

ARGUMENTS

proc_name

Specifies the name of the existing procedure.

-info *info_text*

Provides a help string for the procedure. This is printed by the **help** command when you request help for the procedure. If you do not specify *info_text*, the default is "Procedure".

-define_args *arg_defs*

Defines each possible procedure argument for use with **help -verbose**. This is a list of lists where each list element defines one argument.

-define_arg_groups ***group_defs***

Defines argument checking groups. These groups are checked for you in parse_proc_arguments. each list element defines one group

-command_group ***group_name***

Defines the command group for the procedure. By default, procedures are placed in the "Procedures" command group.

-return ***type_name***

Specifies the type of value returned by this proc. Any value may be specified. Some applications use this information for automatically generated dialogs etc. Please see the type_name option attribute described below.

-hide_body

Hides the body of the procedure from **info body**.

-hidden

Hides the procedure from **help** and **info proc**.

-dont_abbrev

Specifies that the procedure can never be abbreviated. By default, procedures can be abbreviated, subject to the value of the **sh_command_abbrev_mode** variable.

-permanent

Defines the procedure as permanent. You cannot modify permanent procedures in any way, so use this option carefully.

-deprecated

Defines the procedure as deprecated i.e. it should no longer be called but is still available.

-obsolete

Defines the procedure as obsolete i.e. it used to exist but can no longer be called.

DESCRIPTION

The **define_proc_attributes** command associates attributes with a Tcl procedure. These attributes are used to define help for the procedure, locate it in a particular command group, and protect it.

When a procedure is created with the **proc** command, it is placed in the Procedures command group. There is no help text for its arguments. You can view the body of the procedure with **info body**, and you can modify the procedure and its attributes. The **define_proc_attributes** command allows you to change these aspects of a procedure.

Note that the arguments to Tcl procedures are all named, positional arguments. They can be programmed with default values, and there can be optional arguments by using the special argument name *args*. The

define_proc_attributes command does not relate the information that you enter for argument definitions with **-define_args** to the actual argument names. If you are describing anything other than positional arguments, it is expected that you are also using **parse_proc_arguments** to validate and extract your arguments.

The *info_text* is displayed when you use the **help** command on the procedure.

Use **-define_args** to define help text and constraints for individual arguments. This makes the help text for the procedure look like the help for an application command. The value for **-define_args** is a list of lists. Each element has the following format:

```
arg_name option_help value_help data_type attributes
```

The elements specify the following information:

- *arg_name* is the name of the argument.
- *option_help* is a short description of the argument.
- *value_help* is the argument name for positional arguments, or a one word description for dash options. It has no meaning for a Boolean option.
- *data_type* is optional and is used for option validation. The *data_type* can be any of: string, list, boolean, int, float, or one_of_string. The default is string.
- *attributes* is optional and is used for option validation. The *attributes* is a list that can have any of the following entries:
    - "required" - This argument must be specified. This attribute is mutually exclusive with optional.
    - "optional" - Specifying this argument is optional. This attribute is mutually exclusive with "required."
    - "value_help" - Indicates that the valid values for a one_of_string argument should be listed whenever argument help is shown.
    - "values {<list of allowable values>}" - If the argument type is one_of_string, you must specify the "values" attribute.
    - "type_name <name>" - Give a descriptive name to the type that this argument supports. Some applications may use this information to provide features for automatically generated dialogs, etc. Please see product documentation for details. This attribute is not supported on boolean options.
    - "merge_duplicates" - When this option appears more than once in a command, its values are concatenated into a list of values. The default behavior is that the right-most value for the option specified is used.
    - "remainder" - Specifies that any additional positional arguments should be returned in this option. This option is only valid for string option types, and by default the option is optional. You can require at least one item to be specified by also including the required option.
    - "deprecated" - Specifying this option is deprecated i.e. it should no longer be used but is still available. A warning will be output if this option is specified. This attribute cannot be combined with obsolete or required.
    - "obsolete" - Specifying this option is obsolete i.e. it used to exist but can no longer be used. A

warning will be output if this option is specified. This attribute cannot be combined with deprecated or required.

- "min_value" <value> - Specify the minimum value for this option. This attribute is only valid for integer and float types.
- "max_value" <value> - Specify the maximum value for this option. This attribute is only valid for integer and float types.
- "default" <value> - Specify the default value for this option. This attribute is only valid for string, integer and float option types. If the user does not specify this option when invoking the command this default value will be automatically passed to the associated tcl procedure.

The default for *attributes* is "required."

Use the **-define_arg_groups** to define argument checking groups. The format of this option is a list where each element in the list defines an option group. Each element has the following format:

{<type> {<opt1> <opt2> ...} [<attributes>]}

The types of groups are

- *"exclusive"* Only one option in an exclusive group is allowed. All the options in the group must have the same required/optional status. This group can contain any number of options.
- *"together"* If the first option in the group is specified then the second argument is also required. This type of group can contain at most two options.
- \fi"related" These options are related to each other. An automatic dialog builder for this command may try to group these options together.

The supported attributes are:

- *{"label" <text>}* An optional label text to identify this group. The label may be used by an application to automatically build a grouping in a generated dialog.
- *"bidirectional"* This is only valid for a together group. It means that both options must be specified together.

Change the command group of the procedure using the **-command_group** command. Protect the contents of the procedure from being viewed by using **-hide_body**. Prevent further modifications to the procedure by using **-permanent**. Prevent abbreviation of the procedure by using **-dont_abbrev**.

EXAMPLES

The following procedure adds two numbers together and returns the sum. For demonstration purposes, unused arguments are defined.

```
prompt> proc plus {a b} { return [expr $a + $b]}

prompt> define_proc_attributes plus -info "Add two numbers" \
? -define_args {
  {a "first addend" a string required}
  {b "second addend" b string required}
  {"-verbose" "issue a message" "" boolean optional}}

prompt> help -verbose plus
Usage: plus    # Add two numbers
    [-verbose]         (issue a message)
    a                  (first addend)
    b                  (second addend)

prompt> plus 5 6
11
```

In the following example, the argHandler procedure accepts an optional argument of each type supported by **define_proc_attributes**, then displays the options and values received. Note the only one of -Int, -Float, or -Bool may be specified to the command:

```
proc argHandler {args} {
  parse_proc_arguments -args $args results
  foreach argname [array names results] {
    echo $argname = $results($argname)
  }
}

define_proc_attributes argHandler \
  -info "Arguments processor" \
  -define_args {
     {-Oos "oos help"       AnOos   one_of_string
        {required value_help {values {a b}}}}
     {-Int "int help"       AnInt   int     optional}
     {-Float "float help"   AFloat  float   optional}
     {-Bool "bool help"     ""      boolean optional}
     {-String "string help" AString string  optional}
     {-List "list help"     AList   list    optional}
     {-IDup "int dup help"  AIDup   int     {optional merge_duplicates}}
   } \
  -define_arg_groups {
     {exclusive {-Int -Float -Bool}}
   }
```

SEE ALSO

```
help(2)
info(2)
parse_proc_arguments(2)
proc(2)
sh_command_abbrev_mode(3)
```

echo

Echos arguments to standard output.

SYNTAX

```
string echo
   [-n]
   [arguments]
```

Data Types

arguments	string

ARGUMENTS

-n

Suppresses the new line. By default, **echo** adds a new line.

arguments

Specifies the arguments to be printed.

DESCRIPTION

The **echo** command prints out the value of the given arguments. Each of the arguments are separated by a space. The line is normally terminated with a new line, but if the **-n** option is specified, multiple **echo** command outputs are printed onto the same output line.

The **echo** command is used to print out the value of variables and expressions in addition to text strings. The output from **echo** can be redirected using the **>** and **>>** operators.

EXAMPLES

The following are examples of using the **echo** command:

```
prompt> echo
"Running version" $sh_product_version
Running version v3.0a

prompt> echo -n "Printing to" [expr (3 - 2)] > foo
prompt> echo " line." >> foo
prompt> sh cat foo
Printing to 1 line.
```

SEE ALSO

sh(2)

error_info

Prints extended information on errors from the last command.

SYNTAX

```
string error_info
```

ARGUMENTS

This **error_info** command has no arguments.

DESCRIPTION

The **error_info** command is used to display information after an error has occurred. Tcl collects information showing the call stack of commands and procedures. When an error occurs, the **error_info** command can help you to focus on the exact line in a block that caused the error.

EXAMPLES

This example shows how **error_info** can be used to trace an error. The error is that the iterator variable "s" is not dereferenced in the 'if' statement. It should be '$s == "a" '.

```
prompt> foreach s $my_list {
?         if { s == "a" } {
?           echo "Found 'a'!"
?         }
?       }
Error: syntax error in expression " s == "a" "
       Use error_info for more info. (CMD-013)
shell> error_info
```

```
Extended error info:
syntax error in expression " s == "a" "
    while executing
"if { s == a } {
     echo "Found 'a'"
}"
    ("foreach" body line 2)
    invoked from within
"foreach s [list a b c] {
   if { s == a } {
     echo "Found 'a'"
}
}"
 -- End Extended Error Info
```

exit

Terminates the application.

SYNTAX

```
integer exit
   [exit_code]
```

Data Types

exit_code	integer

ARGUMENTS

exit_code

Specifies the return code to the operating system. The default value is 0.

DESCRIPTION

This command exits from the application. You have the option to specify a code to return to the operating system.

EXAMPLES

The following example exits the current session and returns the code 5 to the operating system. At a UNIX operating system prompt, verify (**echo**) the return code as shown.

```
prompt> exit 5
```

```
    % echo $status
    5
```

SEE ALSO

```
quit(2)
```

filter_collection

Filters an existing collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

```
collection filter_collection
   [-regexp]
   [-nocase]
   collection1
   expression
```

Data Types

collection1	collection
expression	string

ARGUMENTS

-regexp

Specifies that the =~ and !~ filter operators will use real regular expressions. By default, the =~ and !~ filter operators use simple wildcard pattern matching with the * and ? wildcards.

-nocase

Makes the pattern match case-insensitive.

collection1

Specifies the base collection to be filtered. This collection is copied to the result collection. Objects are removed from the result collection if they are evaluated as **false** by the conditional *expression* value. Substitute the collection you want for *collection1*.

expression

Specifies an expression with which to filter *collection1*. Substitute the string you want for *expression*.

DESCRIPTION

Filters an existing collection, resulting in a new collection. The base collection remains unchanged. In many cases, application commands that create collections support a *-filter* option that filters as part of the collection process, rather than after the collection has been made.

This type of filtering is almost always more efficient than using the **filter_collection** command after a collection has been formed. The **filter_collection** command is most useful if you plan to filter the same large collection many times using different criteria.

The **filter_collection** command results in either a new collection or an empty string. A resulting new collection contains the subset of the objects in the input *collection1*. A resulting empty string (the empty collection) indicates that the *expression* filtered out all elements of the input *collection1*.

The basic form of the conditional expression is a series of relations joined together with AND and OR operators. Parentheses () are also supported. The basic relation contrasts an attribute name with a value through a relational operator. For example,

```
always_on == true and cell_leakage_power <= 1.6
```

The value side of a relation can be a simple string, quoted string, or an attribute name prefixed with "@". For example:

```
input_delay<=@output_delay
input_delay<=0.24
name=="@literal_name"
```

The relational operators are

```
==   Equal
!=   Not equal
>    Greater than
<    Less than
>=   Greater than or equal to
<=   Less than or equal to
=~   Matches pattern
!~   Does not match pattern
```

The basic relational rules are

- String attributes can be compared with any operator.
- Numeric attributes cannot be compared with pattern match operators.
- Boolean attributes can be compared only with == and !=. The value can be only either *true* or *false*.

Existence relations determine if an attribute is defined or not defined for the object. For example,

```
(sense == setup_clk_rise) and defined(sdf_cond)
```

The existence operators are

```
defined
undefined
!defined
```

These operators apply to any attribute as long as it is valid for the object class.

Collection attributes support a special sizeof(attrName) operator that returns the number of objects in the attribute. If the attribute is not set then 0 is returned.

The **filter_collection** command has a *-regexp* option that uses regular expressions when matching for string attributes. Regular expression matching is done in the same way as in the Tcl **regexp** command. When using the *-regexp* option, take care in the way you quote the filter *expression*. Using rigid quoting with curly braces around regular expressions is recommended.

Regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search simply by adding ".*" to the beginning or end of the expressions as needed. You can make the regular expression search case-insensitive by using the *-nocase* option.

EXAMPLES

The following example creates a collection of only always on cells.

```
lc_shell> set a [filter_collection [get_lib_cells */*] \
"always_on == true"]
{"cell1", "cell2"}
```

SEE ALSO

```
collections(2)
regexp(2)
```

foreach_in_collection

Iterates over the elements of a collection.

SYNTAX

```
string foreach_in_collection
   itr_var
   collections
   body
```

Data Types

```
   itr_var            string
   collections        list
   body               string
```

ARGUMENTS

itr_var

Specifies the name of the iterator variable.

collections

Specifies a list of collections over which to iterate.

body

Specifies a script to execute per iteration.

DESCRIPTION

The **foreach_in_collection** command is used to iterate over each element in a collection. You cannot use the Tcl-supplied **foreach** command to iterate over collections because the **foreach** command requires a list, and a collection is not a list. Also, using the **foreach** command on a collection causes the collection to be deleted.

The arguments for the **foreach_in_collection** command parallel those of the **foreach** command: an iterator variable, the collections over which to iterate, and the script to apply at each iteration. All arguments are required.

Note: The **foreach_in_collection** command does not allow a list of iterator variables.

During each iteration, the *itr_var* option is set to a collection of exactly one object. Any command that accepts *collections* as an argument also accepts *itr_var* because they are of the same data type (collection).

You can nest the **foreach_in_collection** command within other control structures, including another **foreach_in_collection** command.

Note that if the body of the iteration is modifying the netlist, it is possible that all or part of the collection involved in the iteration will be deleted. The **foreach_in_collection** command is safe for such operations. If a command in the body causes the collection to be removed, at the next iteration, the iteration ends with a message indicating that the iteration ended prematurely.

An alternative to collection iteration is to use complex filtering to create a collection that includes only the desired elements, then apply one or more commands to that collection. If the order of operations does not matter, the following are equivalent. The first is an example without iterators.

```
set s [get_cells {U1/*}]
command1 $s
command2 $s
unset s
```

The following is a similar approach using the **foreach_in_collection** command:

```
foreach_in_collection itr [get_cells {U1/*}] {
  command1 $itr
  command2 $itr
}
```

For collections with large numbers of objects, the non-iterator version is more efficient, though both produce the same results if the commands are order-independent.

EXAMPLES

The following example from PrimeTime removes the wire load model from all hierarchical cells in the current instance.

```
pt_shell> foreach_in_collection itr [get_cells *] {
?             if {[get_attribute $itr is_hierarchical] == "true"} {
?               remove_wire_load_model $itr
?             }
?          }
Removing wire load model from cell 'i1'.
Removing wire load model from cell 'i2'.
```

SEE ALSO

```
collections(2)
foreach(2)
```

get_app_var

Gets the value of an application variable.

SYNTAX

```
string get_app_var
   [-default | -details | -list]
   [-only_changed_vars]
   var
```

Data Types

```
   var     string
```

ARGUMENTS

-default

Gets the default value.

-details

Gets additional variable information.

-list

Returns a list of variables matching the pattern. When this option is used, then the *var* argument is interpreted as a pattern instead of a variable name.

-only_changed_vars

Returns only the variables matching the pattern that are not set to their default values, when specified with **-list**.

var

Specifies the application variable to get.

DESCRIPTION

The **get_app_var** command returns the value of an application variable.

There are four legal forms for this command:

- `get_app_var <var>`
  `Returns the current value of the variable.`

- `get_app_var <var> -default`
  `Returns the default value of the variable.`

- `get_app_var <var> -details`

Returns more detailed information about the variable. See below for details.

- get_app_var -list [-only_changed_vars] <*pattern*>

Returns a list of variables matching the pattern. If *-only_changed_vars* is specified, then only variables that are changed from their default values are returned.

In all cases, if the specified variable is not an application variable, then a Tcl error is returned, unless the application variable **sh_allow_tcl_with_set_app_var** is set to true. See the **sh_allow_tcl_with_set_app_var** man page for details.

When **-details** is specified, the return value is a Tcl list that is suitable as input to the Tcl **array set** command. The returned value is a list with an even number of arguments. Each odd-numbered element in the list is a key, and each even-numbered element in the list is the value of the previous key.

The supported keys are as follows:

name

This key contains the name of the variable. This key is always present.

value

This key contains the current value of the variable. This key is always present.

default

This key contains the default value of the variable. This key is always present.

help

This key contains the help string for the variable. This key is always present, but sometimes the value is empty.

type

This key contains the type of the application variable. Legal values of for this key are: string, bool, int, real. This key is always present.

constraint

This key describes additional constraints placed on this variable. Legal values for this key are: none, list, range. This key is always present.

min

This key contains the min value of the application variable. This key is present if the constraint is range. The value of this key may be the empty string, in which case the variable only has a max value constraint.

max

This key contains the max value of the application variable. This key is present if the constraint is "range". The value of this key may be the empty string, in which case the variable only has a min value constraint.

list

This key contains the list of legal values for the application variable. This key is present if the constraint is "list".

EXAMPLES

The following are examples of the **get_app_var** command:

```
prompt> get_app_var sh_enable_page_mode
1

prompt> get_app_var sh_enable_page_mode -default
false

foreach {key val} [get_app_var sh_enable_page_mode -details] {   echo "$key: $val"
}
=>  name: sh_enable_page_mode
    value: 1
    default: false
    help: Displays long reports one page at a time
    type: bool
    constraint: none

prompt> get_app_var -list sh_*message
sh_new_variable_message
```

SEE ALSO

```
report_app_var(2)
set_app_var(2)
write_app_var(2)
```

get_command_option_values

Queries current or default option values.

SYNTAX

```
get_command_option_values
   [-default | -current]


   -command command_name
```

Data Types

```
   command_name      string
```

ARGUMENTS

-default

Gets the default option values, if available.

-current

Gets the current option values, if available.

-command *command_name*

Gets the option values for this command.

DESCRIPTION

This command attempts to query a default or current value for each option (of the command) that has default and/or current-value-tracking enabled. Details of how the option value is queried depend on whether one of the **-current** or **-default** options is specified (see below).

A "Tcl array set compatible" (possibly empty) list of option names and values is returned as the Tcl result. The even-numbered entries in the list are the names of options that were enabled for default-value-tracking or current-value-tracking and had at least one of these values set to a not-undefined value). Each odd-numbered entry in the list is the default or current value of the option name preceding it in the list.

Any options that were not enabled for either default-value-tracking nor current-value-tracking are omitted from the output list. Similarly, options that were enabled for default-value-tracking or current-value-tracking, but for which no (not-undefined) default or current value is set, are omitted from the result list.

If neither **-current** nor **-default** is specified, then for each command option that has either default-value-tracking or current-value-tracking (or both) enabled, the value returned is as follows:

- The current value is returned if current-value-tracking is enabled and a (not-undefined) current value has been set;
- Otherwise the default value is returned if default-value-tracking is enabled and a (not-undefined) default value has been set;
- Otherwise the name and value pair for the option is not included in the result list.

If **-current** is specified, the value returned for an option is the current value if current-value-tracking is enabled, and a (not-undefined) current value has been set; otherwise the name and value pair for the option is omitted from the result list.

If **-default** is specified, the value returned for an option is the default value if default-value-tracking is enabled, and a (not-undefined) default value has been set; otherwise the name and value pair for the option is omitted from the result list.

The result list from **get_command_option_values** includes option values of both dash options and positional options (assuming that both kinds of options of a command have been enabled for value-tracking).

The command issues a Tcl error in a variety of situations, such as if an invalid command name was passed in with **-command**.

EXAMPLES

The following example shows the use of **get_command_option_values**:

```
prompt> test -opt1 10 -opt2 20
1

prompt> get_command_option_values -command test
-bar1 10 -bar2 20
```

SEE ALSO

```
    preview(2)
    set_command_option_value(2)
```

get_defined_commands

Get information on defined commands and groups.

SYNTAX

```
string get_defined_commands [-details]
   [-groups]
   [pattern]

string pattern
```

ARGUMENTS

-details

Get detailed information on specific command or group.

-groups

Search groups rather than commands

pattern

Return commands or groups matching pattern. The default value of this argument is "*".

DESCRIPTION

The **get_defined_commands** gets information about defined commands and command groups. By default the command returns a list of commands that match the specified pattern.

When **-details** is specified, the return value is a list that is suitable as input to the **array set** command. The returned value is a list with an even number of arguments. Each odd-numbered element in the list is a key name, and each even-numbered element in the list os the value of the previous key. The **-details** option is only legal if the pattern matches exactly one command or group.

When **-group** is specified with **-details**, the supported keys are as follows:

name

This key contains the name of the group.

info

This key contains the short help for the group.

commands

This key contains the commands in the group

When **-details** is used with a command, the supported keys are as follows:

name

This key contains the name of the command.

info

This key contains the short help for the command.

groups

This key contains the group names that this command belongs to.

options

This key contains the options defined for the command. The value is a list.

return

This key contains the return type for the command.

Each element in the *options* list also follows the key value pattern. The set of available keys for options are as follows:

name

This key contains the name of the option.

info

This key contains the short help for the option.

value_info

This key contains the short help string for the value

type

The type of the option.

required

The value will be 0 or 1 depending if the option is optional or required.

is_list

Will be 1 if the option requires a list.

list_length

If a list contains the list length constraint. One of any, even, odd, non_empty or a number of elements.

allowed_values

The allowed values if the option has specified allowed values.

min_value

The minimum allowed value if the option has specified one.

max_value

The maximum allowed value if the option has specified one.

EXAMPLES

```
prompt> get_defined_commands *collection
add_to_collection append_to_collection copy_collection filter_collection
foreach_in_collection index_collection sort_collection
prompt> get_defined_commands -details sort_collection
name sort_collection info {Create a sorted copy of the collection}
groups {} options {{name -descending info {Sort in descending order}
value_info {} type boolean required 1 is_list 0} {name -dictionary info
{Sort strings dictionary order.} value_info {} type boolean required 1
is_list 0} {name collection info {Collection to sort} value_info
collection type string required 0 is_list 0} {name criteria info {Sort
criteria - list of attributes} value_info criteria type list required 0
is_list 1 list_length non_empty}}
```

SEE ALSO

```
help(2)
man(2)
```

get_lib_attribute

Returns the value of an attribute on a list of design or library objects.

SYNTAX

```
list get_lib_attribute
   object_list
   attribute_name
```

Data Types

```
   object_list         list
   attribute_name      string
   [-line_number]
```

ARGUMENTS

object_list

Specifies a list of the design or library objects for which the attribute value is returned.

attribute_name

Specifies the name of the attribute whose value is returned.

-line_number

If this option is specified, will return the line number of the specified attribute under the specified object. If the attribute is like voltage_map which can be defined multiple times, it returns a line number list. It only works on those objects which are compiled by read_lib. If the object is loaded by read_db, -line_number always returns 0.

DESCRIPTION

This command searches the *object_list* for the specified attribute and returns a list of attribute values. By default, if a specified object cannot be found or the attribute does not exist on the object, that object is ignored.

Multicorner-Multimode Support

This command uses information from the current scenario only.

The following example gets the value of the **load** attribute for all the pins under lib/IN1:

```
lc_shell> get_lib_attribute [get_lib_pins lib/IN1/*] load
1.0 2.0 2.5 1.0
```

The following example gets the value and line number of the **voltage_map** attribute for the library:

```
lc_shell> get_lib_attribute [get_libs my_lib] voltage_map
{{COREVDD1 0.990000} {COREGND1 0.000000}}

lc_shell> get_lib_attribute [get_libs my_lib] voltage_map -line_number
100 101
```

SEE ALSO

```
collections(2)
foreach_in_collection(2)
set_attribute(2)
```

get_lib_cells

Creates a collection of library cells from the libraries loaded into memory.

SYNTAX

```
collection get_lib_cells
   [-quiet]
   [-regexp]
   [-nocase]
   [-filter expression]
   patterns
```

Data Types

```
expression      string
patterns        list
```

ARGUMENTS

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter *expression*

Filters the collection with the specified expression. For each library cell in the collection, the expression is evaluated based on the library cell's attributes. If the expression evaluates to true, the library cell is included in the result.

To see the list of library cell attributes that you can use in the expression, use the **list_attributes -application -class lib_cell** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

patterns

Creates a collection of library cells whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page.

DESCRIPTION

This command creates a collection of library cells from the libraries currently loaded into memory that match the specified criteria.

The command returns a collection if any library cells match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command. In addition, you can assign the result to a variable.

By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

By default, this command uses information from all active scenarios.

EXAMPLES

The following example queries all library cells that are in the misc_cmos library whose names begin with AN2. Although the output looks like a list, it is just a display.

```
prompt> get_lib_cells misc_cmos/AN2*
{misc_cmos/AN2 misc_cmos/AN2P}
```

SEE ALSO

```
collections(2)
filter_collection(2)
collection_result_display_limit(3)
wildcards(3)
```

get_lib_grid_points

Get the qualified grid points in library based on query parameter.

SYNTAX

```
lookup_table
   input_csv_file
   output_csv_file
   [matching_point]
```

Data Types

```
   input_csv_file     string
   output_csv_file    string
   matching_point     int
```

ARGUMENTS

input_csv_file

Specifies the input file name contains query parameter.

output_csv_file

Specifies the output file name for query result.

matching_point

Query method:

```
'1' - get 1-closest grid point (by default)
'4' = get 4=closest bucket grid points.
```

DESCRIPTION

This command will query the qualified charaterization data point based on the query parameters in the input csv file, and write out the query result to the output csv file. This commands depend on other tcl commands: get_lib_cells/get_timing_arcs/get_lookup_table/lookup_table/get_objects

The input csv file contains query paremters in a formal format. The first line should be the index line, all the allowed index names are listed below. These index sequence can be changed, but index names are pre-defined and can't be changed.

```
instance
db_filename
library
cell
pin
group
related_pin
timing_type
timing_sense
when_cond
related_pg_pin
input_slew
output_load
```

The index names should be separated by comma in the csv file, and each sub record should have same number of fields as the index (specify NULL if any field is empty). User can choose to use part of these indexes or all of them for query, if any index is not needed, it can be missed in the file.

The output csv file contains all the input query-parameter file info, plus the qualified data points info. The query result indexes (listed below) are added after the query parameter indexes in the first line, and the result data for each record is added similarly.

For 1-closest grid point matching:

```
idx1
idx2
value
```

For 4-closest bucket points matching:

```
idx1_a
idx2_a
value_a
idx1_b
idx2_b
value_b
idx1_c
idx2_c
value_c
idx1_d
idx2_d
value_d
```

Not all the fields are filled after query. It depends on table dimension. For 1-closest grid point matching:

```
- 2-D table: 1 point value with 2 indexes.
- 1-D table: 1 point value with only 1 index.
- Scalar table: 1 point value with no index.
```

For 4-closest bucket points matching:

```
- 2-D table: 4 point values with 2 indexes.
- 1-D table: 2 points values with only 1 index.
- Scalar table: 1 point value with no index.
```

The index and point value is formatted with 7 significant digit float number (or scientific notation).

EXAMPLES

The following example shows query 1-closest-grid point

```
Input csv file: /slow/test/input.csv

db_filename,instance,library,cell,pin,related_pin,timing_type,timing_sense,when_cond,group,input_sl
/slow/lib/tsmc.db, INT1, tsmc16nm, AND2, Z, A, combinational, positive_unate, “B”, cell_rise, 0.002
/slow/lib/tsmc.db, INT2, tsmc16nm, AND2, Z, B, combinational, positive_unate, “A”, cell_rise, 0.003
/slow/lib/tsmc.db, INT3, tsmc16nm, OR2, Z, A, combinational, positive_unate, “!B”, cell_rise, 0.004

lc_shell>get_lib_grid_points /slow/test/input.csv /slow/test/output.csv 1
1

Output csv file: /slow/test/output.csv

db_filename,instance,library,cell,pin,related_pin,timing_type,timing_sense,when_cond,group,input_sl
/slow/lib/tsmc.db, INT1, tsmc16nm, AND2, Z, A, combinational, positive_unate, “B”, cell_rise, 0.002
/slow/lib/tsmc.db, INT2, tsmc16nm, AND2, Z, B, combinational, positive_unate, “A”, cell_rise, 0.003
/slow/lib/tsmc.db, INT3, tsmc16nm, OR2, Z, A, combinational, positive_unate, “!B”, cell_rise, 0.004
```

SEE ALSO

```
lookup_table(2)
get_lookup_table(2)
get_objects(2)
get_lib_cells(2)
get_objects(2)
```

get_lib_objects

Returns a collection of objects.

SYNTAX

```
collection get_lib_objects
   [-class_type class_type]
   [-quiet]
   [-regexp]
   [-nocase]
   [-parent]
   [-line_number]
   [-user_defined_data]
   [-filter expression]
   [-traits expression]
   [patterns | -of_objects objects]
```

Data Types

```
expression      string
expression      string
patterns        string
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed.

-class_type

Specifies the liberty object class of the returned object collection. The valid values are : library, cell, pin,pg_pin, timing, ccsn_stage,pg_pin,internal_power, leakage_power and all other legal liberty syntax groups.

-nocase

Makes matches case-insensitive, for the *patterns* argument.

-parent

Specifies this option to return the parent objects of the *of_objects* argument.

-line_number

If this option is specified, it will return a list of line number of the objects specified in of_objects option. It only works on those objects which are compiled by read_lib. If the object is loaded by read_db, -line_number always returns 0.

-user_defined_data

If this option is specified, it can return the object which is user defined with the object class specified by -class_type.

-filter *expression*

Filters the collection with the specified expression. For each library in the collection, the expression is evaluated based on the library's attributes. If the expression evaluates to true, the library is included in the result.

To see the list of library attributes that you can use in the expression, use the **list_attributes -application -class class_type** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-traits *expression*

Filters the collection with the specified trait. Traits is an expression which works in the same way as filter expression.It is used to filter high level property on an object. The following are supported trais for different class_type objects. cell: std, macro, memory, multibit, flip_flop, latch, nand_cell, and_cell, nor_cell, or_cell, inverter_cell, buffer_cell, is_retention, is_level_shifter, multi_voltage, icg, isolation ccsn_stage: is_first timing:input_rising, input_falling, output_rising, output_falling, related_rising, related_falling, delay, constraint, setup_hold, sequential, combinational, noise_arc.

patterns

Creates a collection of objects whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. Patterns are only useful when the returned objectc have names.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses the * (asterisk) as the default pattern.

-of_objects *objects*

Specifies the input owner collection. For each object in the of_objects collection, get_lib_objects tries to find all the objects under it. The liberty group type of the input collection can also be the same as class_type. In that case, get_lib_objects returns the objects that pass the filter and pattern matching.

If the liberty group type of the input collection is different from class_type, get_lib_objects only finds the objects directly under the input collection and returns the objects if they are qualified.

The *patterns* and **-of_objects** arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses the * (asterisk) as the default pattern.

DESCRIPTION

This command returns a collection of objects currently loaded into memory that match the specified criteria.

If no objects match the criteria, the command returns an empty string.

For information about collections and the querying of objects, see the **collections** man pages.

EXAMPLES

The following example queries cells and timings under my_lib.

```
lc_shell> get_lib_objects my_libs/.* -class_type cell -regexp
{"my_lib/my_cell"}
lc_shell> get_lib_objects -of_objects [get_libs  my_lib] -class_type cell -traits and_cell
{"my_lib/my_cell"}
lc_shell> get_lib_objects -of_objects [get_lib_cells  my_lib/my_cell] -parent
{"my_lib"}
lc_shell> get_lib_objects -of_objects [get_lib_pins my_lib/my_cell/my_pin] -class_type timing
sel_5
lc_shell> get_lib_objects -of_objects [get_lib_cells  my_lib/*] -line_number
100
lc_shell> get_lib_objects -user_defined_data -of_objects  [get_lib_cells  my_lib/*] -class_type "ud
sel_6
```

SEE ALSO

```
collections(2)
filter_collection(2)
```

get_lib_pins

Creates a collection of library cell pins from libraries loaded into memory.

SYNTAX

```
collection get_lib_pins
   [-regexp]
   [-nocase]
   [-filter expression]
   patterns
```

Data Types

```
   expression     string
   patterns       list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed.

-nocase

Makes matches case-insensitive, both for the *patterns* argument and for the ==, =~, and !~ filter operators.

-filter expression

Filters the collection with the specified expression. For each library cell pin in the collection, the expression is evaluated based on the library cell pin's attributes. If the expression evaluates to true, the library cell pin is included in the result.

To see the list of library cell pin attributes that you can use in the expression, use the **list_attributes -application -class lib_pin** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

patterns

Creates a collection of library cell pins whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page.

DESCRIPTION

This command creates a of library cell pins from the libraries currently loaded into memory that match the specified criteria.

If no libraries have been loaded into memory, the tool loads the libraries specified in the **link_library** variable into memory the first time you run the **get_libs**, **get_lib_cells**, or **get_lib_pins** command.

The command returns a collection if any library cell pins match the criteria. If no objects match the criteria, the command returns an empty string.

You can use this command at the command prompt or you can nest it as an argument to another command, such as **query_objects**. In addition, you can assign the result to a variable.

By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

This command uses information from all active scenarios.

EXAMPLES

The following example queries all pins of the AN2 library cell in the misc_cmos library. Although the output looks like a list, it is just a display.

```
prompt> get_lib_pins misc_cmos/AN2/*
{misc_cmos/AN2/A misc_cmos/AN2/B misc_cmos/AN2/Z}
```

SEE ALSO

```
collections(2)
filter_collection(2)
get_libs(2)
get_lib_cells(2)
```

get_lib_timing_arcs

Returns a timing collection.

SYNTAX

```
collection get_objects
   -of_objects  object_list
   [-to to_list]
   [-from from_list]
   [-filter expression]
   [-arc_type  delay_arc|constraint_arc]
   [-quiet]
```

Data Types

```
   to_list      list
   from_list    list
   object_list  list
   expression        string
```

ARGUMENTS

-of_objects object_list

Required. Specifies a cell collection or a timing collection.

-to to_list

Specifies the owner pin of the timings users want to get.

-from from_list

Specifies a list of related pins of the timings users want to get.

-filter *expression*

Filters the collection with the specified expression. For each library in the collection, the expression is evaluated based on the library's attributes. If the expression evaluates to true, the library is included in the result.

To see the list of library attributes that you can use in the expression, use the **list_attributes -application -class timing** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

-arc_type delay_arc|constraint_arc

Specified the timing arc type. If it is not specified it returns all the timing arcs under the collections specified in **-of_objects**.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

DESCRIPTION

This command returns a collection of timing objects currently loaded into memory that match the specified criteria.

If no objects match the criteria, the command returns an empty string.

For information about collections and the querying of objects, see the **collections** man page.

EXAMPLES

The following example queries all timing objects under my_lib/my_cell/A.

```
prompt> get_lib_timing_arcs -of_objects [get_lib_cell my_lib/my_cell] -to A
```

SEE ALSO

```
collections(2)
filter_collection(2)
```

get_libs

Creates a collection of libraries loaded into memory.

SYNTAX

```
collection get_libs
   [-quiet]
   [-regexp]
   [-nocase]
   [-filter expression]
   [patterns]
```

Data Types

```
expression      string
patterns        list
```

ARGUMENTS

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp

Views the *patterns* argument as a regular expression rather than a simple wildcard pattern.

This option also modifies the behavior of the =~ and !~ filter operators to use regular expressions rather than simple wildcard patterns.

The regular expression matching is similar to the Tcl **regexp** command. When using the **-regexp** option, be careful how you quote the *patterns* argument and filter expression. Using rigid quoting with curly braces around regular expressions is recommended. Note that regular expressions are always anchored; that is, the expression is assumed to begin matching at the beginning of an object name and end matching at the end of an object name. You can widen the search by adding ".*" to the beginning or end of the expressions, as needed.

-nocase

Makes matches case-insensitive, for the *patterns* argument.

`-filter` ***`expression`***

Filters the collection with the specified expression. For each library in the collection, the expression is evaluated based on the library's attributes. If the expression evaluates to true, the library is included in the result.

To see the list of library attributes that you can use in the expression, use the **list_attributes -application -class library** command.

For more information about how to use the **-filter** option, see the **filter_collection** man page.

`patterns`

Creates a collection of libraries whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. For more information about using and escaping wildcards, see the **wildcards** man page.

DESCRIPTION

This command creates a collection of libraries from the libraries currently loaded into memory that match the specified criteria.

If no libraries have been loaded into memory, the tool loads the libraries specified in the **link_library** variable into memory the first time you run the **get_libs**, **get_lib_cells**, or **get_lib_pins** command.

The command returns a collection if any libraries match the criteria. If no objects match the criteria, the command returns an empty string.

By default, it displays a maximum of 100 objects. You can change this maximum by using the **collection_result_display_limit** variable.

For information about collections and the querying of objects, see the **collections** man page.

Multicorner-Multimode Support

By default, this command uses information from all active scenarios. You can select different scenarios by using the **-scenarios** option.

EXAMPLES

The following example queries all loaded libraries. Use the **list_libs** command to get a complete listing of the libraries.

```
prompt> get_libs
{misc_cmos misc_cmos_io}
```

SEE ALSO

```
collections(2)
filter_collection(2)
```

get_license

Obtains a license for a feature.

SYNTAX

```
status get_license
   feature_list
   [-quantity num_licenses]
```

Data Types

```
   feature_list     list
   num_licenses     integer
```

ARGUMENTS

feature_list

Specifies a list of features to be obtained. If you specify more than one feature, the list must be enclosed in braces ({}).

By looking at your key file, you can determine all of the features licensed at your site.

-quantity *num_licenses*

Specifies the total number of licenses to be checked out for each feature after the command has completed. If some licenses have already been checked out, the command acquires only the additional licenses needed to bring the total to the specfied quantity. If this option is not specified, only one license is checked out for each feature.

DESCRIPTION

This command obtains a license for the specified features. These features are checked out by the current user until the **remove_license** command is used or until the program exits.

If multiple licenses are required for a multicore run, you can use the **-quantity** option to specify the total number of licenses needed (which might be different than the incremental number of licenses that the command checks out). You can use this option to reserve the required number of licenses early in the session, rather than risking a failure later in the session if the licenses are not available.

The **list_licenses** command provides a list of the features that you are currently using.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example obtains the multivoltage feature:

```
prompt> get_license Galaxy-MV
```

The following example obtains two copies of the licenses required to run a multicore **compile_ultra** command in Design Compiler:

```
prompt> get_license -quantity 2 {DC-Expert DC-Ultra-Opt DC-Ultra-Features}
```

SEE ALSO

```
check_license(2)
license_users(2)
list_licenses(2)
remove_license(2)
```

get_lookup_tables

Retrieve a collection of lookup table objects.

SYNTAX

```
collection get_lookup_tables
   [-of_objects]
   [-quiet]
   group_type
```

Data Types

of_objects	list
group_type	string

ARGUMENTS

-of_objects

Specifies a collection of Tcl objects, which is the parent (owner) of the returned lookup table objects.

-quiet

Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

group_type

Specifies lookup table group type. Types of regular lookup tables are valid, for example:

```
cell_rise
cell_fall
rise_transition
fall_transition
rise_constraint
fall_constraint
ocv_sigma_cell_rise
ocv_sigma_cell_fall
rise_power
fall_power
dc_current
... ...
```

Specific CCS group types are also valid, which include:

```
output_current_rise (CCS Timing object)
output_current_fall (CCS Timing object)
output_voltage_rise (CCS Noise object)
output_voltage_fall (CCS Noise object)
pg_current (CCS Power object)
```

DESCRIPTION

This command creates a collection of lookup_table objects currently loaded into memory that match the specified criteria.

If no lookup table match the criteria, the command returns empty collection.

For information about collections and the querying of objects, see the **collections** man page.

For CCS groups, this command will created CCS objects by collecting all vector sub-groups, to form a 2-D lookup table where index_1 and index_2 are the indices, index_3 and values are 2 lists storing the corresponding time-value waveform. Please refer to lookup_table command for further information.

EXAMPLES

The following example queries cell_rise and output_current_rise tables from a timing object:

```
prompt> set luts [get_lookup_tables -of_objects $timing ocv_sigma_cell_rise]
prompt> set alut [index_collection $luts 0]
prompt> get_attribute $lut sigma_type
prompt> set ocr [get_lookup_table -of_objects $timing output_current_rise]
prompt> sizeof_collection $ocr

_sel8
```

SEE ALSO

```
collections(2)
filter_collection(2)
```

get_message_ids

Get application message ids

SYNTAX

```
string get_message_ids [-type severity]
[pattern]

string severity
string pattern
```

ARGUMENTS

-type *severity*

Filter ids based on type (Values: Info, Warning, Error, Severe, Fatal)

pattern

Get IDs matching pattern (default: *)

DESCRIPTION

The **get_message_ids** command retrieves the error, warning and informational messages used by the application. The result of this command is a Tcl formatted list of all message ids. Information about the id can be queried with the **get_message_info** command.

EXAMPLES

The following code finds all error messages and makes the application stop script execution when one of these messages is encountered.

```
foreach id [get_message_ids -type Error] {
  set_message_info -stop_on -id $id
}
```

SEE ALSO

```
print_message_info(2)
set_message_info(2)
suppress_message(2)
```

get_message_info

Returns information about diagnostic messages.

SYNTAX

```
integer get_message_info
   [-error_count | -warning_count | -info_count
     | -limit l_id | -occurrences o_id | -suppressed s_id | -id i_id]
```

Data Types

```
l_id      string
o_id      string
s_id      string
i_id        string
```

ARGUMENTS

-error_count

Returns the number of error messages issued so far.

-warning_count

Returns the number of warning messages issued so far.

-info_count

Returns the number of informational messages issued so far.

-limit *l_id*

Returns the current user-specified limit for a given message ID. The limit was set with the **set_message_info** command.

-occurrences *o_id*

Returns the number of occurrences of a given message ID.

-suppressed *s_id*

Returns the number of times a message was suppressed either using **suppress_message** or due to exceeding a user-specified limit.

-id *i_id*

Returns information about the specified message. The information is returned as a Tcl list compatible with the array set command.

DESCRIPTION

The **get_message_info** command retrieves information about error, warning, and informational messages. For example, if the following message is generated, information about it is recorded:

Error: unknown command 'wrong_command' (CMD-005)

It is useful to be able to retrieve recorded information about generated diagnostic messages. For example, you can stop a script after a certain number of errors have occurred, or monitor the number of messages issued by a single command.

You can also find out how many times a specific message has occurred, or how many times it has been suppressed. Also, you can find out if a limit has been set for a particular message ID.

EXAMPLES

The following example uses the **get_message_info** command to count the number of errors that occurred during execution of a specific command, and to return from the procedure if the error count exceeds a given amount:

```
prompt> proc \
  do_command {limit} {
    set current_errors [get_message_info -error_count]
    command
    set new_errors [get_message_info -error_count]
    if {[expr $new_errors - $current_errors] > $limit} {
      return -code error "Too many errors"
    }
    ...
  }
```

The following example uses the **get_message_info** command to retrieve information on the CMD-014 message:

```
prompt> get_message_info -id CMD-014
id CMD-014 severity Error limit 0 occurrences 0 suppressed 0 message
{Invalid %s value '%s' in list.}
```

SEE ALSO

```
print_message_info(2)
set_message_info(2)
suppress_message(2)
```

get_object_name

Returns a list of names of the objects in a collection.

SYNTAX

```
list get_object_name collection
```

Data Types

```
collection     string
```

ARGUMENTS

collection

Specifies the name of the collection that contains objects whose names are requested.

-full_path

Specifies whether to display the full path name of the object. For example, "my_lib/my_cell/my_pin" is the full path name of "my_pin". If this option is not specified, it displays as "my_pin".

DESCRIPTION

This command returns a list of names of the objects in a collection.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example returns the name top as the object contained in the collection returned by the **current_design** command.

```
prompt> get_object_name [current_design]
Current design is 'top'.
top
```

SEE ALSO

```
collections(2)
```

get_scale

Get the scale value for target unit.

SYNTAX

```
collection get_scale
   type
   target_unit
   library
```

Data Types

```
   type           string
   target_unit    string
   library        list
```

ARGUMENTS

type

Specify unit type for scaling. Allowed unit type is: time| voltage| current | power| capacitance | resistance| energy

target_unit

Specify target unit for scaling.

library

Specify the source library Tcl collection.

DESCRIPTION

This command analyzes the unit of specified unit type in the source library, and returns the scale value from the library unit to the target unit. If any error happens, return 0.

The "target_unit" argument should be specified in a unified format: [float][magnitude][unit] Recommended unit type are listed below: For time: 1ps| 1ns For voltage: 1mv| 1v For current: 1ua| 1ma| 1a For power: 1pw| 1nw| 1uw| 1mw For capacitance: 1ff| 1pf For resistance: 1ohm| 1kohm For energy: 1fj| 1pj| 1nj| 1uj

EXAMPLES

The following example gets the power scaling to 1nw for the source library

```
lc_shell>set lib [get_libs test]
{"test"}
lc_shell>get_scale power 1nw $lib
1000.0
```

SEE ALSO

```
analyze_trend(2)
```

get_unix_variable

This is a synonym for the **getenv** command.

SEE ALSO

gettenv(2)
printenv(2)
printvar(2)
set(2)
setenv(2)
sh(2)
unset(2)

getenv

Returns the value of a system environment variable.

SYNTAX

```
string getenv
   variable_name
```

Data Types

```
   variable_name      string
```

ARGUMENTS

variable_name

Specifies the name of the environment variable to be retrieved.

DESCRIPTION

The **getenv** command searches the system environment for the specified *variable_name* and sets the result of the command to the value of the environment variable. If the variable is not defined in the environment, the command returns a Tcl error. The command is catchable.

Environment variables are stored in the **env** Tcl array variable. The **getenv**, **setenv**, and **printenv** environment commands are convenience functions to interact with this array.

The application you are running inherited the initial values for environment variables from its parent process (that is, the shell from which you invoked the application). If you set the variable to a new value using the **setenv** command, you see the new value within the application and within any new child processes you initiate from the application using the **exec** command. However, these new values are not exported to the parent process. Further, if you set an environment variable using the appropriate system command in a shell you invoke using the **exec** command, that value is not reflected in the current application.

See the **set**, **unset**, and **printvar** commands for information about working with non-environment variables.

EXAMPLES

In the following example, **getenv** returns you to your home directory:

```
prompt> set home [getenv "HOME"]
/users/disk1/bill

prompt> cd $home

prompt> pwd
/users/disk1/bill
```

In the following example, **setenv** changes the value of an environment variable:

```
prompt> getenv PRINTER
laser1

prompt> setenv PRINTER "laser3"
laser3

prompt> getenv PRINTER
laser3
```

In the following example, the requested environment variable is not defined. The error message shows that the Tcl variable **env** was indexed with the value UNDEFINED, which resulted in an error. In the second command, **catch** is used to suppress the message.

```
prompt> getenv "UNDEFINED"
Error: can't read "env(UNDEFINED)": no such element in array
        Use error_info for more info. (CMD-013)

prompt> if {[catch {getenv "UNDEFINED"} msg]} {
         setenv UNDEFINED 1
       }
```

SEE ALSO

```
catch(2)
exec(2)
printenv(2)
printvar(2)
set(2)
setenv(2)
unsetenv(2)
unset(2)
```

help

Displays quick help for one or more commands.

SYNTAX

```
string help
   [-verbose]
   [-groups]
   [pattern]
```

Data Types

```
pattern      string
```

ARGUMENTS

-verbose

Displays the command options; for example *command_name* **-help**.

-groups

Displays a list of command groups only.

pattern

Displays commands matching the specified pattern.

DESCRIPTION

The **help** command is used to get quick help for one or more commands or procedures. This is not the same as the **man** command that displays reference manual pages for a command. There are many levels of help.

By typing the **help** command, a brief informational message is printed followed by the available command groups.

List all of the commands in a group by typing the group name as the argument to **help**. Each command is followed by a one-line description of the command.

To get a one-line help description for a single command, type **help** followed by the command name. You can specify a wildcard pattern for the name; for example, all commands containing the string "alias". Use the **-verbose** option to get syntax help for one or more commands. Use the **-groups** option to show only the command groups in the application. This option cannot be combined with any other option.

EXAMPLES

The following example lists the commands by command group:

```
prompt> help
Specify a command name or wild card pattern to get help on individual
commands. Use the '-verbose' option to get detailed option help for a
command. Commands also provide help when the '-help' option is passed to
the command.'

You can also specify a command group to get the commands available in
that group. Available groups are:

  Procedures:                Miscellaneous procedures
  Help:                      Help commands
  Builtins:                  Generic Tcl commands
 ...
```

The following example displays the list of procedures in the Procedures group:

```
prompt> help procedures
 ls                 # List files
 sh                 # Execute a shell command
```

The following example uses a wildcard character to display a one-line description of all commands beginning with *a*:

```
prompt> help a*
 alias                # Create a command which expands to words.
 append               # Builtin
 array                # Builtin
```

This example displays option information for the **source** command:

```
prompt> help -verbose source
 source               # Read a file and execute it as a script
   [-echo]                (Echo all commands)
   [-verbose]             (Display intermediate results)
   file_name              (Script file to read)
```

SEE ALSO

```
man(2)
sh_help_shows_group_overview(3)
```

history

Displays or modifies the commands recorded in the history list.

SYNTAX

```
string history
   [-h]
   [-r]
   [argument_list]
```

Data Types

argument_list	list

ARGUMENTS

-h

Displays the history list without the leading numbers. You can use this for creating scripts from existing history. You can then source the script with the **source** command. You can combine this option with only a single numeric argument. Note that this option is a nonstandard extension to Tcl.

-r

Reverses the order of output so that most recent history entries display first rather than the oldest entries first. You can combine this option with only a single numeric argument. Note that this option is a nonstandard extension to Tcl.

argument_list

Additional arguments to **history** (see DESCRIPTION).

DESCRIPTION

The **history** command performs one of several operations related to recently-executed commands recorded in a history list. Each of these recorded commands is referred to as an "event." The most commonly used forms of the command are described below. You can combine each with either the **-h** or **-r** option, but not both.

- With no arguments, the **history** command returns a formatted string (intended for you to read) giving the event number and contents for each of the events in the history list.
- If a single, integer argument *count* is specified, only the most recent *count* events are returned. Note that this option is a nonstandard extension to Tcl.
- Initially, 20 events are retained in the history list. You can change the length of the history list using the following:

  history keep *count*

Tcl supports many additional forms of the **history** command. See the "Advanced Tcl History" section below.

EXAMPLES

The following examples show the basic forms of the **history** command. The first is an example of how to limit the number of events shown using a single numeric argument:

```
prompt> history 3
     7  set base_name "my_file"
     8  set fname [format "%s.db" $base_name]
     9  history 3
```

Using the **-r** option creates the history listing in reverse order:

```
prompt> history -r 3
     9  history -r 3
     8  set fname [format "%s.db" $base_name]
     7  set base_name "my_file"
```

Using the **-h** option removes the leading numbers from each history line:

```
prompt> history -h 3
set base_name "my_file"
set fname [format "%s.db" $base_name]
history -h 3
```

Advanced Tcl History

The **history** command performs one of several operations related to recently-executed commands recorded in a history list. Each of these recorded commands is referred to as an "event." When specifying an event to the **history** command, the following forms may be used:

- A number, which if positive, refers to the event with that number (all events are numbered starting at 1). If the number is negative, it selects an event relative to the current event; for example, **-1** refers to the previous event, **-2** to the one before that, and so on. Event **0** refers to the current event.

- A string selects the most recent event that matches the string. An event is considered to match the string either if the string is the same as the first characters of the event, or if the string matches the event in the sense of the **string match** command.

The **history** command can take any of the following forms:

history

Same as **history info**, described below.

history add *command [exec]*

Adds the *command* argument to the history list as a new event. If **exec** is specified (or abbreviated), the command is also executed and its result is returned. If **exec** is not specified, an empty string is returned.

history change *newValue [event_number]*

Replaces the value recorded for an event with *newValue*. The *event_number* specifies the event to replace, and defaults to the *current* event (not event **-1**). This command is intended for use in commands that implement new forms of history substitution and want to replace the current event (that invokes the substitution) with the command created through substitution. The return value is an empty string.

history clear

Erases the history list. The current keep limit is retained. The history event numbers are reset.

history event *[event_number]*

Returns the value of the event given by *event_number*. The default value of *event_number* is **-1**.

history info *[count]*

Returns a formatted string, giving the event number and contents for each of the events in the history list except the current event. If *count* is specified, then only the most recent *count* events are returned.

history keep *[count]*

Changes the size of the history list to *count* events. Initially, 20 events are retained in the history list. If *count* is not specified, the current keep limit is returned.

history nextid

Returns the number of the next event to be recorded in the history list. Use this for printing the event number in command-line prompts.

history redo *[event_number]*

Reruns the command indicated by *event* and returns its result. The default value of *event_number* is **-1**. This command results in history revision. See the following section for details.

History Revision

Pre-8.0 Tcl had a complex history revision mechanism. The current mechanism is more limited, and the **substitute** and **words** history operations have been removed. The **clear** operation was added.

The **redo** history option results in much simpler "history revision." When this option is invoked, the most recent event is modified to eliminate the history command and replace it with the result of the history command. If you want to redo an event without modifying the history, use the **event** operation to retrieve an event, and use the **add** operation to add it to history and execute it.

index_collection

Given a collection and an index into it, if the index is in range, create a new collection containing only the single object at the index in the base collection. The base collection remains unchanged.

SYNTAX

```
collection index_collection
   collection1
   index
```

Data Types

```
collection1        collection
index              int
```

ARGUMENTS

collection1

Specifies the collection to be searched.

index

Specifies the index into the collection. Allowed values are integers from 0 to **sizeof_collection** - 1.

DESCRIPTION

You can use the **index_collection** command to extract a single object from a collection. The result is a new collection containing only that object. The index operation is done in constant time - it is independent of the number of elements in the collection, or the specific index.

The range of indices is from 0 to one less than the size of the collection. If the specified index is outside that range, an error message is generated.

Commands that create a collection of objects do not impose a specific order on the collection, but they do generate the objects in the same, predictable order each time. Applications that support the sorting of collections allow you to impose a specific order on a collection.

You can use the empty string for the *collection1* argument. However, by definition, any index into the empty collection is invalid. Therefore, using the **index_collection** command with the empty collection always generates the empty collection as a result and generates an error message.

Note that not all collections can be indexed.

EXAMPLES

The following example from PrimeTime uses the **index_collection** command to extract the first object of a collection.

```
pt_shell> set c1 [get_cells {u1 u2}]
{"u1", "u2"}
pt_shell> query_objects [index_collection $c1 0]
{"u1"}
```

SEE ALSO

```
collections(2)
query_objects(2)
sizeof_collection(2)
```

is_false

Tests the value of a specified variable, and returns 1 if the value is 0 or the case-insensitive string **false**; returns 0 if the value is 1 or the case-insensitive string **true**.

SYNTAX

```
status is_false
   value
```

Data Types

```
value      string
```

ARGUMENTS

value

Specifies the name of the variable whose value is to be tested.

DESCRIPTION

This command tests the value of a specified variable, and returns 1 if the value is either 0 or the case-insensitive string **false**. The command returns 0 if the value is either 1 or the case-insensitive string **true**. Any value other than 1, 0, **true**, or **false** generates a Tcl error message.

The **is_false** command is used in writing scripts that test Boolean variables that can be set to 1, 0, or the case-insensitive strings **true** or **false**. When such variables are set to **true** or **false**, they cannot be tested in the negative in an **if** statement by simple variable substitution, because they do not constitute a true or false condition. The following example is not legal Tcl:

```
set x FALSE
if { !$x } {
  set y TRUE
}
```

This results in a Tcl error message, indicating that you cannot use a non-numeric string as the operand of "!". So, although you can test the positive condition, **is_false** allows you to test both conditions safely.

EXAMPLES

The following example shows the use of the **is_false** command:

```
prompt> set x TRUE
TRUE

prompt> if { ![is_false $x] } {
?         set y TRUE
?       }
TRUE

prompt>
```

SEE ALSO

```
expr(2)
if(2)
is_true(2)
```

is_null_object

check whether one Tcl collection is null or not

SYNTAX

```
collection is_null_object
   collection1
```

Data Types

```
   collection1             collection
```

ARGUMENTS

collection1

Specifies one collection to be checked.

DESCRIPTION

You can use the **is_null_object** command to check whether one Tcl collection is null or not. if it is null, return true; otherwise return false. Currently only command **align_lib_objects** returns null object.

EXAMPLES

The following exmaple is to use is_null_object to check whether the object in the result from command align_lib_objects is null or not.

```
   lc_shell>set cell1 test/BUF/*
   test/BUF/*
```

```
lc_shell>set cell2 test/CKGT/*
test/CKGT/*
lc_shell>set pinlist1 [get_lib_pins $cell1]
{"test/BUF/A", "test/BUF/X"}
lc_shell>set pinlist2 [get_lib_pins $cell2]
{"test/CKGT/CK", "test/CKGT/EN", "test/CKGT/Q", "test/CKGT/SE"}
lc_shell>set aligned_pinlist [align_lib_objects  $pinlist1 $pinlist2]
_sel4 _sel5 _sel6 _sel7 _sel8 _sel
lc_shell> set i 0
0
lc_shell>foreach pair $aligned_pinlist {
set element1 [index_collection $pair 0]
set element2 [index_collection $pair 1]
set ret1 [is_null_object $element1]
set ret2 [is_null_object $element2]
if {$ret1 == "1"} {
  set name1 "null"
} else {
  set name1 [get_object_name  $element1]
}
if {$ret2 == "1"} {
  set name2 "null"
} else {
  set name2 [get_object_name  $element2]
}
echo "pair $i:($name1, $name2)"
incr i
}
pair 0:(null, EN)
pair 1:(null, CK)
pair 2:(null, SE)
pair 3:(A, null)
pair 4:(null, Q)
pair 5:(X, null)
```

SEE ALSO

collections(2)

is_true

Tests the value of a specified variable, and returns 1 if the value is 1 or the case-insensitive string **true**; returns 0 if the value is 0 or the case-insensitive string **false**.

SYNTAX

```
status is_true
   value
```

Data Types

```
   value      string
```

ARGUMENTS

value

Specifies the name of the variable whose value is to be tested.

DESCRIPTION

This command tests the value of a specified variable, and returns 1 if the value is either 1 or the case-insensitive string **true**. The command returns 0 if the value is either 0 or the case-insensitive string **false**. Any value other than 1, 0, **true**, or **false** generates a Tcl error message.

The **is_true** command is used in writing scripts that test Boolean variables that can be set to 1, 0, or the case-insensitive strings **true** or **false**. When such variables are set to **true** or **false**, they cannot be tested in the negative in an **if** statement by simple variable substitution, because they do not constitute a true or false condition. The following example is not legal Tcl:

```
set x TRUE
if { !$x } {
  set y FALSE
}
```

This results in a Tcl error message, indicating that you cannot use a non-numeric string as the operand of "!". So, although you can test the positive condition, **is_true** allows you to test both conditions safely.

EXAMPLES

The following example shows the use of the **is_true** command:

```
prompt> set x FALSE
FALSE

prompt> if { ![is_true $x] } {
?         set y FALSE
?       }

FALSE

prompt>
```

SEE ALSO

```
expr(2)
if(2)
is_false(2)
```

lib2saif

Creates a forward-annotation SAIF file for a specified technology library.

SYNTAX

```
status lib2saif
   [-output file_name]
   library
   [-lib_pathname lib_path_name]
```

Data Types

```
file_name          string
library            list
lib_path_name      list
```

ARGUMENTS

-output *file_name*

Specifies the name of the library forward-annotation SAIF file created by **lib2saif**. The default file name is the value of the **power_sdpd_saif_file** variable. By default, the value of the **power_sdpd_saif_file** variable is *power_sdpd.saif*.

library

Specifies the library name, or library file name for which the forward SAIF file is generated. If a library file name is specified then the library must be in .db format. The library must have state-dependent leakage power characterization and/or pin-level state-dependent and/or path-dependent internal power characterization for a library forward SAIF file to be generated.

-lib_pathname *lib_path_name*

Specifies the path name to the simulation library information. This is optional, and is only required if the library is not in the simulation tool's current path during simulation.

DESCRIPTION

This command creates a forward-annotation SAIF file that contains the state-dependent and path-dependent information for library cells. The library forward-annotation SAIF file is used in flows where a gate-level back-annotation SAIF file with state-dependent and/or path-dependent switching activity is generated.

For details on the SAIF format, see the SAIF specification.

For details on state-dependent and path-dependent power characterization, see the Library Compiler and Power Compiler documentation.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following examples illustrate the use of this command:

```
prompt> lib2saif -out a130.saif asic130_typical

prompt> lib2saif -out a130.saif ../libs/asic130.db
```

SEE ALSO

None

license_users

Lists the current users of the Synopsys licensed features.

SYNTAX

```
status license_users
   [feature_list]
```

Data Types

```
feature_list     list
```

ARGUMENTS

feature_list

Lists the licensed features for which to obtain the information. If more than one feature is specified, they must be enclosed in braces ({}). See the Synopsys *Installation Guide: UNIX-Based Platforms* for a list of features supported by the current release, or determine from the key file all of the features that are licensed at your site.

DESCRIPTION

This command displays information about all of the licenses, related users, and host names currently in use. If a feature list is specified, only information about those features is displayed.

The **license_users** command is valid only when Network Licensing is enabled.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In this example, all of the users of the Synopsys features are displayed:

```
prompt> license_users

krig@node1    DC-Expert, Design-Compiler, Design-Vision,
              Milkyway-Interface
doris@node2   Design-Compiler, HDL-Compiler, Power-Optimization
schen@node3   DC-Expert, DC-Ultra-Features, DC-Ultra-Opt,
              Design-Compiler,
wru@node4     DC-Expert, DC-Ultra-Features, DC-Ultra-Opt,
              Design-Compiler, DesignWare,
              Milkyway-Interface, Test-Compiler,
              VHDL-Compiler

4 users listed.
```

This example shows the users of the "HDL-Compiler" or "VHDL-Compiler" features.

```
prompt> license_users {HDL-Compiler VHDL-Compiler}

doris@node2   Design-Compiler, HDL-Compiler, Power-Optimization
wru@node4     DC-Expert, DC-Ultra-Features, DC-Ultra-Opt,
              Design-Compiler, DesignWare,
              Milkyway-Interface, Test-Compiler,
              VHDL-Compiler

2 users listed.
```

SEE ALSO

```
get_license(2)
remove_license(2)
```

list_attributes

Lists the currently-defined attributes.

SYNTAX

```
.prod syn
status list_attributes
.prod icc
string list_attributes
.prod all
   [-application]
   [-class class_name]
   [-nosplit]
```

Data Types

```
   class_name     string
```

ARGUMENTS

-application

Lists application attributes attributes.

-class *class_name*

Limits the listing to attributes of a single class. The valid values are those groups defined by Liberty syntax like library, cell, pin...

-nosplit

Prevents lines from being split when column fields overflow. Most of the attribute information is listed in fixed-width columns. If the information for a given field exceeds the column width, the next field begins on a new line in the correct column.

DESCRIPTION

The **list_attributes** command displays an alphabetically-sorted list of application attributes.

Using the **-application** option adds all application attributes to the listing. Since there are many application attributes, you can limit the listing to a specific object class by using the **-class** option.

EXAMPLES

The following example adds application-defined attributes, but limits the listing to library attributes only:

```
prompt> list_attributes -application -class library

****************************************
Report : List of Attribute Definitions
Version: K-2015.06-SP1-VAL-150625
Date   : Thu Jun 25 19:31:47 2015
****************************************

Properties:
    A - Application-defined
    U - User-defined

Attribute Name            Object     Type       Properties  Constraints
---------------------------------------------------------------------------------
nom_process               library    float      A
nom_temperature           library    float      A
nom_voltage               library    float      A
```

SEE ALSO

```
get_attribute(2)
set_attribute(2)
```

list_libs

List libraries that are currently available in memory.

SYNTAX

```
Status list_libs
   [lib_list]
```

Data Types

```
   lib_list      list
```

ARGUMENTS

lib_list

Specifies the libraries to list. You can use the * (asterisk) and ? (question mark) wildcard characters when specifying the libraries to list. For more information about using and escaping wildcards, see the **wildcards** man page.By default, the tool lists all libraries.

DESCRIPTION

This comman displays all libraries that are currently available in memory.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example list all loaded libraries.

```
prompt> list_libs
----------------------------------------------------------------------------
Library                           File                                  Path
-------                           ----                                  ----
z0                                z0.db                                 /remote/dtdata1/testdata/libra
SHKA90_16384X8X4CM16_FF1P32VM40C SHKA90_16384X8X4CM16_FF1P32VM40C.db /remote/dtdata1/testdata/libra
```

SEE ALSO

```
get_libs(2)
```

lminus

Removes one or more named elements from a list and returns a new list.

SYNTAX

```
list lminus
   [-exact]
   original_list
   elements
```

Data Types

```
original_list     list
elements          list
```

ARGUMENTS

-exact

Specifies that the exact pattern is to be matched. By default, **lminus** uses the default match mode of **lsearch**, the **-glob** mode.

original_list

Specifies the list to copy and modify.

elements

Specifies a list of elements to remove from *original_list*.

DESCRIPTION

The **lminus** command removes elements from a list by using the element itself, rather than the index of the element in the list (as in **lreplace**). The **lminus** command uses the **lsearch** and **lreplace** commands to find the elements and replace them with nothing.

If none of the elements are found, a copy of *original_list* is returned.

The **lminus** command is often used in the translation of Design Compiler scripts that use the subtraction operator (-) to remove elements from a list.

EXAMPLES

The following example shows the use of the **lminus** command. Notice that no error message is issued if a specified element is not in the list.

```
prompt> set l1 {a b c}
a b c

prompt> set l2 [lminus $l1 {a b d}]
c

prompt> set l3 [lminus $l1 d]
a b c
```

The following example illustrates the use of **lminus** with the **-exact** option:

```
prompt> set l1 {a a[1] a* b[1] b c}
a a[1] a* b[1] b c

prompt> set l2 [lminus $l1 a*]
{b[1]} b c

prompt> set l3 [lminus -exact $l1 a*]
a {a[1]} {b[1]} b c

prompt> set l4 [lminus -exact $l1 {a[1] b[1]} ]
a a* b c
```

SEE ALSO

```
lreplace(2)
lsearch(2)
```

lookup_table

Provides lookup_table manipulation commands to perform several operations on a variable of single lookup table object. The lookup table object can be operated solely by lookup table manipulation commands.

SYNTAX

```
lookup_table
   [option]
   lookup_table object
   [arg arg ...]
```

Data Types

```
option        string
lookup_table object        Tcl variable
```

ARGUMENTS

option

Specifies a predefined operation on lookup_table object. Following are the valid predefined option values.

```
template_name
variables
sizes
dimension
index
slice
bucket
fit
interpolate
index_output
```

lookup_table object

Specifies a single lookup_table object.

DESCRIPTION

This command performs one of several operations on a single lookup table object variable given by $lookupTableVariable. The option argument determines what action is carried out by the command. The legal options are:

```
lookup_table template_name $lookupTableVariable
 Returns the name of the template used in the lookup table.

lookup_table variables $lookupTableVariable
 Returns a list of variables used by the lookup table.
 The order of the list is the original order of variables.

lookup_table sizes $lookupTableVariable
 Returns a list of sizes of the indices used in the lookup table.
 The order of the list is the original order of variables.
 The size of the returning list is the dimension of the table.

lookup_table dimension $lookupTableVariable
 Returns the dimension of the table. Returns 0 for scalar lookup table.

lookup_table index $lookupTableVariable
 Returns a list of lists where each element is a list storing the indices
 of one variable. The order of the outer list is the original order of variable.

lookup_table slice $lookupTableVariable [-all] [-index {variable_name value}]
lookup_table slice $lookupTableVariable [-data time] [-index {variable_name value}]
lookup_table slice $lookupTableVariable [-data values] [-index {variable_name value}]
lookup_table slice $lookupTableVariable [-data ref] [-index {variable_name value}]
 Returns a list containing all or part of values of the lookup table.

 The option -all indicates all values are returned. The option -all supports
  tables with arbitrary dimension including scalar/1D/2D/3D.

        The option -index indicates the returned list of values are fixed at the
        specified value of the variable name. The option -index supports scalar, 1D
 and 2D tables (CCS object is actually 2D table).

        For CCS objects, i.e., for output_current_rise|fall and pg_current tables,
        the table values are time-current waveforms, for output_voltage_rise|fall
 tables, the table values are time-voltage waveforms:
        1. the 3rd index index_3 stores a list of time points, the [-data time] should
           be used to retrieve index_3(time);
 2. the values stores the corresponding current values, the [-data value] should
 3. the reference_time attribute for specific index point can be retrieved by
           the [-data ref] option;

        The [-all] option does NOT support CCS object.

lookup_table bucket $lookupTableVariable [-index
  {[variable_name1 value1] [variable_name2 value2] ... [variable_nameN valueN]}]
        Support tables of arbitary dimensions.

 For 3-dimensional lookup table, the command returns collection of 8 triplets
 {index1, index2, index3, value} on the indices grid of the lookup table, where the given
 index point specified by option -index should fall on the region enclosed by the
 8 points. The order of the index in the triplets follows the order defined
 in the library by table template.
   If the given point is on the grid point, returns 8 identical points.
   If the given point is on the grid but not the grid point, returns 4 identical points on each end.
 For 2-dimensional lookup table, the command returns collection of 4 triplets
 {index1, index2, value} on the indices grid of the lookup table, where the given
```

```
     index point specified by option -index should fall on the region enclosed by the
     4 points. The order of the index in the triplets follows the order defined
     in the library by table template.
       If the given point is on the grid point, returns 4 identical points.
       If the given point is on the grid but not the grid point, returns 2 identical points on each end.
      For 1-dimensional lookup table, the command returns collection of 2 pairs
     {index, value} on the indices grid of the lookup table, where the given index
     point specified by option -index should fall on the interval enclosed by the 2 points.
       If the given point is on the grid point, returns 2 identical points.
     For scalar lookup table, the command always returns the single table value.

    lookup_table fit $lookupTableVariable [-index
      {[variable_name1 value1] [variable_name2 value2] [variable_name3 value3]}]
            Support tables of arbitary dimensions.

     Among the grid points returned from lookuptable bucket command, returns the
     one grid point that has the closest index to the given index.
     For scalar table, always return the single table value.

    lookup_table interpolate $lookupTableVariable [-index
      {[variable_name1 value1] [variable_name2 value2] [variable_name3 value3]}]
            Support scalar/1D/2D/3D tables.

     Returns value linear-interpolated from the lookup table based on given points.
     The calculation is done on top of lookup_table bucket.
     For scalar table, always returns the single table value.

    -index option

    Additional information about the -index option for lookup_table bucket/fit/interpolate commands:
        1. All index defined in lookup table must be provided by -index option.
           For example, for 2D cell_rise table, the input_net_transition and
           total_output_net_capacitance values must be defined using index option.
        2. If there are extra index defined using index option, where the extra
           index is NOT defined in lookup table, the extra index is NOT to be considered
           in the manipulation commands. For example, if both input_net_transition and
           total_output_net_capacitance values were provided using index option, for
           1-D cell_rise table that is using input_net_transition as index, the
           total_output_net_capacitance is not be considered.

    lookup_table index_output $lookupTableVariable
     Returns the index_output value of a pg_current CCS object.
```

EXAMPLES

The following example shows a list of lookup_table operations:

```
lu_table_template ("tt_vio") {
 variable_1 : "related_pin_transition";
 index_1("1.000,17.000,27.000,97.000,217.000");
 variable_2 : "constrained_pin_transition";
 index_2("1.000,17.000,27.000,97.000,217.000");
}

timing () {
 timing_type : "setup_rising";
 related_pin : "CP";
 rise_constraint ("tt_vio") {
  index_1("1, 17, 27, 97, 217");
  index_2("1, 17, 27, 97, 217");
  values("51.01, 82.01, 99.01, 94.01, 68.01", \
   "47.01, 54.01, 79.01, 67.01, 34.01", \
   "22.01, 57.01, 82.01, 64.01, 17.01", \
   "40.01, 61.01, 99.01, 81.01, 38.01", \
   "94.01, 129.01, 125.01, 121.01, 95.01");
 }
 fall_constraint ("tt_vio") {
  index_1("1, 17, 27, 97, 217");
  index_2("1, 17, 27, 97, 217");
  values("108.01, 171.01, 222.01, 353.01, 503.01", \
   "102.01, 136.01, 194.01, 319.01, 488.01", \
   "63.01, 112.01, 191.01, 295.01, 485.01", \
   "81.01, 116.01, 208.01, 300.01, 505.01", \
   "94.01, 129.01, 222.01, 313.01, 519.01");
 }
}

Assuming $a_timing is the Tcl variable pointing to the timing group above.

lc_shell>set setup_table [get_lookup_table of_objects $a_timing rise_constraint]
_sel8

lc_shell>lookup_table template_name $setup_table
tt_vio

lc_shell>lookup_table variables $setup_table
related_pin_transition constrained_pin_transition

lc_shell>set table_size [lookup_table sizes $setup_table]
5 5
lc_shell> llength $table_size
2

lc_shell>lookup_table dimension $setup_table
2

lc_shell>lookup_table index $setup_table
{1 17 27 97 217} {1 17 27 97 217}

lc_shell>lookup_table slice $setup_table -all
51.01 82.01 99.01 94.01 68.01 47.01 54.01, 79.01 67.01 34.01 22.01 57.01 82.01 64.01 17.01 40.01 61

lc_shell>lookup_table slice $setup_table -index {constrained_pin_transition 175}
171.01 136.01 112.01 116.01 129.01

lc_shell>lookup_table bucket $setup_table -index {related_pin_transition 12 constrained_pin_transit
{1 1 51.01} {1 17 82.01 } {17 1 47.01 54.01} {17 17 54.01}

lc_shell>lookup_table fit $setup_table index {related_pin_transition 12 constrained_pin_transition
{1 1 51.01}

lc_shell>lookup_table interpolate $setup_table -index {related_pin_transition 12 constrained_pin_tr
63.01
```

SEE ALSO

```
collections(2)
filter_collection(2)
```

ls

Lists the contents of a directory.

SYNTAX

```
string ls [filename ...]

string filename
```

ARGUMENTS

filename

Provides the name of a directory or filename, or a pattern which matches files or directories.

DESCRIPTION

If no argument is specified, the contents of the current directory are listed. For each *filename* matching a directory, **ls** lists the contents of that directory. If *filename* matches a file name, the file name is listed.

EXAMPLES

```
shell> ls *.db *.pt

test1.pt          c1.db          c3.db          c5.db

test2.pt          c2.db          c4.db          c6.db
```

SEE ALSO

cd(2)
pwd(2)

man

Displays reference manual pages.

SYNTAX

```
string man
   topic
```

Data Types

```
   topic      string
```

ARGUMENTS

topic

Specifies the subject to display. Available topics include commands, variables, and error messages

DESCRIPTION

The **man** command displays the online manual page for a command, variable, or error message. You can write man pages for your own Tcl procedures and access them with the **man** command by setting the **sh_user_man_path** variable to an appropriate value. See the man page for the **sh_user_man_path** variable for details.

EXAMPLES

The following command displays the man page for the **echo** command:

```
   prompt> man echo
```

The following command displays the man page for the error message CMD-025:

```
prompt> man CMD-025
```

SEE ALSO

```
help(2)
sh_user_man_path(3)
```

mem

Reports memory usage information.

SYNTAX

```
status mem
   [-all]
   [-verbose]
```

ARGUMENTS

-all

Reports the maximum of the peak memory usage among the main process and its child processes.

-verbose

Reports in detail the peak memory usage of the main process and each child process, including placement, extraction, and routing.

DESCRIPTION

This command reports the peak memory usage of processes. By default, it reports the peak memory usage of the main process in kilobytes. If you specify the **-all** option, it reports the maximum of the peak memory usage among the main process and its child processes. If you specify both the **-all** and **-verbose** options, it also prints out the peak memory usage in megabytes for the main process and each child process.

Memory peak is defined as the memory high-water mark of processes. When the tool reports a memory peak, it means the maximum memory allocated from the operating system. When child processes exist, the tool reports the maximum number among the main process and its child processes.

Note that the **mem** command reports the peak memory usage, not the swap space usage of the process. It actually reports how much memory was allocated from the operating system at the peak point. Some of the memory can be swapped out by the operating system based on the system status.

EXAMPLES

The following example shows the default command output:

```
prompt> mem
102400
```

The following example shows the output when using the **-all** and **-verbose** options:

```
prompt> mem -all -verbose
Main process mem-peak: 100 Mb
Child "placement" called 3 times, mem-peak:  60 Mb
Child "extraction" called 3 times, mem-peak: 50 Mb

102400
```

SEE ALSO

cputime(2)

merge_lib

Merges the base library and model file.

SYNTAX

```
status set_lib_group
   [-lib_group group_name]
   [-base_lib base_db_filename]
   [-model_lib model_db_filename]
   [-merged_library_name merged_library_ename]
   [-resolve_conflict { use_base | use_model | error_out }]
   [-output merged_filename]
```

Data Types

```
group_name        string
base_db_filename      string
model_db_filename     string
merged_library_name   string
merged_filename       string
use_base use_model error_out string
```

ARGUMENTS

group_name

Specifies a library group name that is defined in set_lib_group command. The merge_lib command will merge each base library with its associated model files in this group.

base_db_filename

Specifies one base library .db file name. This option and -model_file should be specified together.

model_db_filename

Specifies one model .db file name. This option and -base_lib should be specified together.

merged_library_name

Specifies a library name of the final merged library in a fixed string or a suffix starting with wild card(*) in the form of *_suffix. Each merged library will use original base library_name appended by the suffix.

This is for multi sub-group merge in one merge_lib command. If not specified, the library name of the base library will be used.

`resolve_conflict`

Specifies one: use_base to use data from base lib, use_model to use data from model file or error_out to quit merge. If not specified, the value in the base library will be used.

`merged_filename`

Specifies an output .db file name or a suffix starting with wild card (*) in the form of *_suffix. Each merged library will use original base library_name appended by the suffix.db. This is for multi sub-group merge in one merge_lib command. Alternatively, write_lib can be used to write out a DB file. If not specified, it will use merged_library_name appended by extension .db similar to read_lib.

DESCRIPTION

The **merge_lib** command merges the base library and its associated model files. The base library and its associated model files are input by one of the followings listed in precedence:

1) merge_lib -base_lib and -model_lib options. Both should be specified together.

2) merge_lib -lib_group option.

3) set_lib_group command which is defined before merge_lib command.

EXAMPLES

In the following example, the base library pvt1.db and its model file upf.db are merged into a new library with library_name merged_pvt1.

```
prompt> merge_lib -base_lib pvt1.db -model_lib upf.db -merged_library_name merged_pvt1
```

SEE ALSO

```
check_library(2)
set_lib_group(2)
```

parse_proc_arguments

Parses the arguments passed into a Tcl procedure.

SYNTAX

```
string parse_proc_arguments -args arg_list
result_array

list arg_list
string result_array
```

ARGUMENTS

-args** **arg_list

Specified the list of arguments passed in to the Tcl procedure.

result_array

Specifies the name of the array into which the parsed arguments should be stored.

DESCRIPTION

The **parse_proc_arguments** command is used within a Tcl procedure to enable use of the -help option, and to support argument validation. It should typically be the first command called within a procedure. Procedures that use **parse_proc_arguments** will validate the semantics of the procedure arguments and generate the same syntax and semantic error messages as any application command (see the examples that follow).

When a procedure that uses **parse_proc_arguments** is invoked with the -help option, **parse_proc_arguments** will print help information (in the same style as using **help -verbose**) and will then cause the calling procedure to return. Similarly, if there was any type of error with the arguments (missing required arguments, invalid value, and so on), **parse_proc_arguments** will return a Tcl error and the calling procedure will terminate and return.

If you didn't specify -help, and the specified arguments were valid, the array variable *result_array* will contain each of the argument values, subscripted with the argument name. Note that the argument name here is NOT the names of the arguments in the procedure definition, but rather the names of the arguments as defined using the **define_proc_attributes** command.

The **parse_proc_arguments** command cannot be used outside of a procedure.

EXAMPLES

The following procedure shows how **parse_proc_arguments** is typically used. The argHandler procedure parses the arguments it receives. If the parse is successful, argHandler prints the options or values actually received.

```
proc argHandler args {
  parse_proc_arguments -args $args results
  foreach argname [array names results] {
    echo "  $argname = $results($argname)"
  }
}
define_proc_attributes argHandler -info "argument processor" \
  -define_args \
  {{-Oos "oos help" AnOos one_of_string {required value_help {values {a b}}}}
   {-Int "int help" AnInt int optional}
   {-Float "float help" AFloat float optional}
   {-Bool "bool help" "" boolean optional}
   {-String "string help" AString string optional}
   {-List "list help" AList list optional}}
  {-IDup int dup AIDup int {optional merge_duplicates}}}
```

Invoking argHandler with the -help option generates the following:

```
prompt> argHandler -help
Usage: argHandler    # argument processor
        -Oos AnOos             (oos help:
                                Values: a, b)
        [-Int AnInt]           (int help)
        [-Float AFloat]        (float help)
        [-Bool]                (bool help)
        [-String AString]      (string help)
        [-List AList]          (list help)
```

Invoking argHandler with an invalid option causes the following output (and a Tcl error):

```
prompt> argHandler -Int z
Error: value 'z' for option '-Int' not of type 'integer' (CMD-009)
Error: Required argument '-Oos' was not found (CMD-007)
```

Invoking argHandler with valid arguments generates the following:

```
prompt> argHandler -Int 6 -Oos a
  -Oos = a
  -Int = 6
```

SEE ALSO

define_proc_attributes(2)
help(2)
proc(2)

preview

Evaluates its arg list without actually executing it, but option values are "stored through" if option value-tracking is enabled.

SYNTAX

```
preview arg ...
```

ARGUMENTS

arg

The arg list will be evaluated without being executed.

DESCRIPTION

The preview command (like the built-in Tcl command eval) takes one or more arguments, which together comprise a Tcl script containing one or more commands. The preview command concatenates all its arguments together (like eval), and passes the concatenated string to the CCI/Tcl interpreter recursively.

The preview command passes its concatenated command string to the CCI/Tcl command evaluator as usual, but the special feature of the preview command is that the command will not actually be executed by preview. Instead only the early stages of CCI argument/option checking will happen for the command string. Specifically, standard CCI checking of option arguments is performed, as is "store through" of new current option values for options with current-value-tracking enabled (if option checking succeeded). The preview command thus provides a way to update current values of a command's options (which have current-value-tracking enabled) without actually executing the command.

Conceptually, the CCI preview command is in the same "family" as the Tcl built-in command eval - the difference is that eval executes its concatenated command string, while preview does not, but preview does implement CCI option checking and "store through" for current values of options (which have current-value-tracking enabled).

The advantage of the preview command (compared with a command for turning on and off preview mode) is that a preview command is guaranteed to finish/exit (thus guaranteeing that the preview behavior in the CCI command evaluator will get turned off as soon as the preview command exits). If instead it was necessary to have matching commands for turning on and off "preview mode", bugs in Tcl script code might cause the "turn off preview mode" call to be skipped - then the interpreter would be stuck in "preview mode" indefinitely (and we do not want to take the risk of such a serious bug happening). By having preview as a command, we greatly reduce the risk that the interpreter could get "stuck in preview mode".

Note that the preview command itself does not implement any additional echoing to output of its concatenated command.

About preview and mutually exclusive options: preview performs all option checking for the previewed command as if it was to be executed (even though the command will not be executed). Thus the simultaneous presence of multiple mutually exclusive options will generate an error under preview (and "store through" of current option values will be aborted). This can be regarded as both a feature (it's consistent with normal command execution) and a limitation (it makes it impossible to set current values for mutually exclusive options of a command from a single run of preview). To workaround this limitation, we also provide the set_command_option_value command (which does not do checks such as the mutual exclusion check) - see the man page for set_command_option_value.

A possible power user application: power users could put preview commands in their home directory Tcl startup files for their favorite dialogs/commands to "seed" initial current values for these dialogs. This may lower the need for automatically saving replay scripts (of preview commands for each command/dialog) on exit from the application. Note: the set_command_option_value command may be easier to use than preview for this purpose.

EXAMPLES

Consider a command named foo with two integer options, -bar1 and -bar2, that have current-value-tracking enabled. In the following example, the "current values" being tracked for the -bar1 and -bar2 options are updated to 10 and 20 respectively:

```
prompt> preview foo  -bar1 10  -bar2 20
prompt> get_command_option_values -current -command foo
-bar1 10  -bar2 20

Note that foo command is not actually executed in this example.
```

SEE ALSO

```
get_command_option_values(2)
set_command_option_value(2)
```

print_message_info

Prints information about diagnostic messages that have occurred or have been limited.

SYNTAX

```
string print_message_info
   [-ids id_list]
   [-summary]
```

Data Types

```
   id_list      list
```

ARGUMENTS

-ids *id_list*

Specifies a list of message identifiers to report. Each entry can be a specific message or a glob-style pattern that matches one or more messages. If this option is omitted and no other options are given, then all messages that have occurred or have been limited are reported.

-summary

Generates a summary of error, warning, and informational messages that have occurred so far.

DESCRIPTION

The **print_message_info** command enables you to print summary information about error, warning, and informational messages that have occurred or have been limited with the **set_message_info** command. For example, if the following message is generated, information about it is recorded:

Error: unknown command 'wrong_command' (CMD-005)

It is useful to be able to summarize all recorded information about generated diagnostic messages. Much of this can be done using the **get_message_info** command, but you need to know a specific message ID. By default, **print_message_info** summarizes all of the information. It provides a single line for each message that has occurred or has been limited, and one summary line that shows the total number of errors, warnings, and informational messages that have occurred so far. If an *id_list* is given, then only messages matching those patterns are displayed. If **-summary** is given, then a summary is displayed.

Using a pattern in the *id_list* is intended to show a specific message prefix, for example, "CMD*". Note that this does not show all messages with that prefix. It shows only the messages that have occurred or have been limited.

EXAMPLES

The following example uses **print_message_info** to show a few specific messages:

```
prompt> print_message_info -ids [list "CMD*" APP-99]

Id                Limit    Occurrences   Suppressed
-------------------------------------------------------
CMD-005               0              7            2
APP-99                1              0            0
```

At the end of the session, you might want to generate some information about a set of interesting messages, such as how many times each occurred (which includes suppressions), how many times each was suppressed, and whether a limit was set for any of them. The following example uses **print_message_info** to get this information:

```
prompt> print_message_info

Id                Limit    Occurrences   Suppressed
-------------------------------------------------------
CMD-005               0             12            0
APP-027             100            150           50
APP-99                0              1            0

Diagnostics summary: 12 errors, 150 warnings, 1 informational
```

Note that the suppressed count is not necessarily the difference between the limit and the occurrences, since the limit can be dynamically changed with the **set_message_info** command.

SEE ALSO

```
get_message_info(2)
set_message_info(2)
suppress_message(2)
```

print_proc_new_vars

Checks for new variables created within a Tcl procedure.

SYNTAX

```
string print_proc_new_vars
```

ARGUMENTS

The **print_proc_new_vars** command has no arguments.

DESCRIPTION

The **print_proc_new_vars** command is used in a Tcl procedure to show new variables created up to that point in the procedure. If the **sh_new_variable_message_in_proc** and **sh_new_variable_message** variables are both set to **true**, this command is enabled. If either variable is **false**, the command does nothing. The result of the command is always an empty string.

When enabled, the command issues a CMD-041 message for each variable created in the scope of the procedure so far. Arguments to the procedure are excluded.

For procedures that are in a namespace (that is, not in the global scope), **print_proc_new_vars** also requires that you have defined some aspect of the procedure using the **define_proc_attributes** command.

Important: This feature has a significant negative impact on CPU performance. This should be used only when developing scripts or in interactive use. When you turn on the feature, the application issues a message (CMD-042) to warn you about the usage of this feature.

EXAMPLES

Assuming that the appropriate control variables are set to **true**, the following commands show new variables created within procedures:

```
prompt> proc new_proc {a} {
             set x $a
             set y $x
             unset x
             print_proc_new_vars
           }

prompt> new_proc 67
=New variable report for new_proc:
Information: Defining new variable 'y'. (CMD-041)
=END=

prompt>

prompt> proc ns::p2 {a} {
             set x $a
             print_proc_new_vars
           }

prompt> define_proc_attributes ns::p2

prompt> ns::p2 67
=New variable report for ns::p2:
Information: Defining new variable 'x'. (CMD-041)
=END=
```

SEE ALSO

```
define_proc_attributes(2)
sh_new_variable_message(3)
sh_new_variable_message_in_proc(3)
```

print_suppressed_messages

Displays an alphabetical list of message IDs that are currently suppressed.

SYNTAX

```
string print_suppressed_messages
```

ARGUMENTS

The **print_suppressed_messages** command has no arguments.

DESCRIPTION

The **print_suppressed_messages** command displays all messages that you suppressed using the **suppress_message** command. The messages are listed in alphabetical order. You only can suppress informational and warning messages. The result of **print_suppressed_messages** is always the empty string.

EXAMPLES

The following example shows the output from the **print_suppressed_messages** command:

```
prompt> print_suppressed_messages
No messages are suppressed

prompt> suppress_message {XYZ-001 CMD-029 UI-1}

prompt> print_suppressed_messages
The following 3 messages are suppressed:
```

```
          CMD-029, UI-1, XYZ-001
```

SEE ALSO

```
suppress_message(2)
unsuppress_message(2)
```

printenv

Prints the value of environment variables.

SYNTAX

```
string printenv
   [variable_name]
```

Data Types

```
   variable_name      string
```

ARGUMENTS

variable_name

Specifies the name of a single environment variable to print.

DESCRIPTION

The **printenv** command prints the values of the environment variables inherited from the parent process or set from within the application with the **setenv** command. When no arguments are specified, all environment variables are listed. When a single *variable_name* is specified, if that variable exists, its value is printed. There is no useful return value from **printenv**.

To retrieve the value of a single environment variable, use the **getenv** command.

EXAMPLES

The following examples show the output from the **printenv** command:

```
prompt> printenv SHELL
/bin/csh

prompt> printenv EDITOR
emacs
```

SEE ALSO

```
getenv(2)
setenv(2)
```

printvar

Prints the values of one or more variables.

SYNTAX

```
string printvar
   [pattern]
   [-user_defined | -application]
```

Data Types

```
pattern      string
```

ARGUMENTS

pattern

Prints variable names that match *pattern*. The optional *pattern* argument can include the wildcard characters * (asterisk) and ? (question mark). If not specified, all variables are printed.

-user_defined

Shows only user-defined variables. This option is mutually exclusive with the **-application** option.

-application

Shows only application variables. This option is mutually exclusive with the **-user_defined** option.

DESCRIPTION

The **printvar** command prints the values of one or more variables. If the *pattern* argument is not specified, the command prints out the values of application and user-defined variables. The **-user_defined** option limits the variables to those that you have defined. The **-application** option limits the variables to those created by the application. The two options are mutually exclusive and cannot be used together.

Some variables are suppressed from the output of **printvar**. For example, the Tcl environment variable array *env* is not shown. To see the environment variables, use the **printenv** command. Also, the Tcl variable **errorInfo** is not shown. To see the error stack, use the **error_info** command.

EXAMPLES

The following command prints the values of all of the variables:

```
prompt> printvar
```

The following command prints the values of all application variables that match the pattern *sh*a**:

```
prompt> printvar -application sh*a*
sh_arch                  = "sparcOS5"
sh_command_abbrev_mode   = "Anywhere"
sh_enable_page_mode      = "false"
sh_new_variable_message = "false"
sh_new_variable_message_in_proc = "false"
sh_source_uses_search_path = "false"
```

The following command prints the value of the **search_path** variable:

```
prompt> printvar search_path
search_path              = ".  /designs/newcpu/v1.6  /lib/cmos"
```

The following command prints the values of the user-defined variables:

```
prompt> printvar -user_defined
a                      = "6"
designDir              = "/u/designs"
```

SEE ALSO

```
error_info(2)
printenv(2)
setenv(2)
```

proc_args

Displays the formal parameters of a procedure.

SYNTAX

```
string proc_args
   proc_name
```

Data Types

```
   proc_name      string
```

ARGUMENTS

proc_name

Specifies the name of the procedure.

DESCRIPTION

The **proc_args** command is used to display the names of the formal parameters of a user-defined procedure. This command is essentially a synonym for the Tcl builtin command **info** with the *args* argument.

EXAMPLES

This example shows the output of **proc_args** for a simple procedure:

```
   prompt> proc plus {a b} { return [expr $a + $b] }
```

```
prompt> proc_args plus
a b

prompt> info args plus
a b

prompt>
```

SEE ALSO

```
info(2)
proc(2)
proc_body(2)
```

proc_body

Displays the body of a procedure.

SYNTAX

```
string proc_body
   proc_name
```

Data Types

```
   proc_name      string
```

ARGUMENTS

proc_name

Specifies the name of the procedure.

DESCRIPTION

The **proc_body** command is used to display the body (contents) of a user-defined procedure. This command is essentially a synonym for the Tcl builtin command **info** with the *body* argument.

EXAMPLES

This example shows the output of **proc_body** for a simple procedure:

```
prompt> proc plus {a b} { return [expr $a + $b] }

prompt> proc_body plus
```

```
  return [expr $a  + $ b]

prompt> info body plus
  return [expr $a  + $ b]

prompt>
```

SEE ALSO

```
info(2)
proc(2)
proc_args(2)
```

query_objects

Searches for and displays objects in the database.

SYNTAX

```
string query_objects
   [-verbose]
   [-truncate elem_count]
   [-class class_name]
   object_spec
```

Data Types

```
class_name          string
elem_count          int
object_spec         list
```

ARGUMENTS

-verbose

Displays the class of each object found. By default, only the name of each object is listed. With this option, each object name is preceded by its class (see the EXAMPLES).

-truncate *elem_count*

Truncates display to *elem_count* elements. By default, up to 100 elements display. To see more or less elements, use this option. To see all elements, set the *elem_count* value to 0.

-class *class_name*

For elements in the *object_spec* that are not collections, this is the class used when searching the database for objects which match the element. Valid classes are application-specific.

object_spec

Provides a list of objects to find and display. Each element in the list is either a collection or a pattern which will match some objects in the database. Patterns are explicitly searched for in the database with class *class_name*.

DESCRIPTION

The **query_objects** command finds and displays objects in the application's runtime database. The command does not have a meaningful return value; it displays the objects found and returns an empty string.

The *object_spec* is a list containing collections and/or object names. For elements of the *object_spec* that are collections, **query_objects** simply displays the contents of the collection.

For elements of the *object_spec* that are names (or contain wildcard patterns), the **query_objects** command searches the database for objects of the class specified by the *class_name* option. Note: The **query_objects** command does not have a predefined implicit order of classes for which searches are initiated. If you do not specify the *class_name* option, only those elements that are collections are displayed. Messages are displayed for the other elements (see EXAMPLES).

To control the number of elements displayed, use the -*truncate* option. If the display is truncated, you see the ellipsis (...) as the last element. Note that if the default truncation occurs, a message displays showing the total number of elements that would have displayed (see EXAMPLES).

The **query_objects** command displays the output in either a Legacy format or in a Tcl compatible format. The default output format varies by tool but you can control the format yourself by setting the application variable query_objects_format. For example to force Tcl compatible output use this command:

```
pt_shell> set_app_var query_objects_format Tcl
Tcl
```

See below the differences between Legacy and Tcl formats.

NOTE: The output from the **query_objects** command looks similar to the output from any command that creates a collection; however, the result of the **query_objects** command is always an empty string.

EXAMPLES

These following examples show the basic usage of the **query_objects** command.

```
pt_shell> query_objects [get_cells o*]
{"or1", "or2", "or3"}
pt_shell> query_objects -class cell U*
{"U1", "U2"}
pt_shell> query_objects -verbose -class cell \
?           \[list U* [get_nets n1]]
{"cell:U1", "cell:U2", "net:n1"}
```

If the output format is Tcl, then the output looks like this:

```
pt_shell> query_objects [get_cells o*]
{or1 or2 or3}
pt_shell> query_objects -class cell U*
{U1 U2}
```

```
pt_shell> query_objects -verbose -class cell \
?           \[list U* [get_nets n1]]
{{cell U1} {cell U2} {net n1}}
```

When you omit the **-class** option, only those elements of the *object_spec* that are collections generate output. The other elements generate error messages.

```
pt_shell> query_objects \[list U* [get_nets n1] n*]
Error: No such collection 'U*' (SEL-001)
Error: No such collection 'n*' (SEL-001)
{"n1"}
```

When the output is truncated, you get the ellipsis at the end of the display. For the following example, assume the default truncation is 5 (it is actually 100).

```
pt_shell> query_objects [get_cells o*] -truncate 2
{"or1", "or2", ...}
pt_shell> query_objects [get_cells *]
{"or1", "or2", "or3", "U1", "U2", ...}
Output truncated (total objects 126)
```

SEE ALSO

```
collections(2)
get_cells(2)
get_clocks(2)
get_designs(2)
get_generated_clocks(2)
get_lib_cells(2)
get_lib_pins(2)
get_libs(2)
get_nets(2)
get_path_groups(2)
get_pins(2)
get_ports(2)
get_qtm_ports(2)
get_timing_paths(2)
```

quit

Exits the shell.

SYNTAX

```
string quit
```

ARGUMENTS

The **quit** command has no arguments.

DESCRIPTION

The **quit** command exits from the application. It is basically a synonym for the **exit** command with no arguments.

EXAMPLES

The following example exits the current session:

```
prompt> quit
```

SEE ALSO

```
exit(2)
```

read_db

Reads in one or more design or library files in Synopsys database (.db) format.

SYNTAX

```
string read_db
[-return_lib_collection]
   file_names
```

Data Types

```
   file_names      list
```

ARGUMENTS

file_names

Specifies names of one or more files to be read.

-return_lib_collection

Specifies whether read_db returns a library collection. If the db library is failed to be read, read_db returns an empty collection.

DESCRIPTION

The **read_db** command reads design information from Synopsys database (.db) files into lc_shell.

This command is derived from the **read_file** command with the **-format db** option. For more information, see the man page for the **read_file** command.

SEE ALSO

```
read_lib(2)
```

read_lib

Reads a technology library into the shell.

SYNTAX

```
status read_lib
   file_name
   [-test_model CTL_file_list]
   [-signoff_screening]
   [-model_type  model_file_type]
   [-partial_model_check]
   [-html]
   [-return_lib_collection]
```

Data Types

```
   file_name                             string
   CTL_file_list                         list
   model_file_type                       string
```

ARGUMENTS

file_name

Specifies the name of the library file to be read. A technology must be in Synopsys Library Compiler format.

-test_model *CTL_file_list*

Specifies a set of one or more CTL files as test model of cells in the library. The format of the CTL_file_list is <*cell_name*:>*file_name* where *cell_name* is optional and is to specify the particular cell a CTL file models. When the *cell_name* is missing, the environment name in the CTL file is used as the cell name it is modeling. When specifying more than one CTL file, enclose them in braces ({}).

-signoff_screening

Specifies that library Compiler is screening for sign-off tools.

-model_type *model_file_type*

Specifies that Library Compiler accepts one of these model types: [lvf|ccs_power|power_aware].

-partial_model_check

Specifies that Library Compiler run minimum set of screener checks on model file.

-html

Specifies that library screener results are reported in HTML.

-return_lib_collection

Specifies whether read_lib returns a library collection. If the library file is failed to be compiled, read_lib returns an empty collection.

DESCRIPTION

The **read_lib** command reads a library file into the shell (or GUI). The library file is automatically compiled by the Synopsys Library Compiler. The library specified in *file_name* can have either a simple or complex file name. A simple file name has no directory specification, which means (in UNIX) that it does not contain a / (slash). Simple file names such as test.lib or library.db must be found in a directory listed in the **search_path** variable. A complex file name has a directory specification in it, which means that it does contain a slash. Complex file names such as ./test.lib or ~synopsys/dc/test.lib are not searched for in **search_path**.

You can read any number of libraries into the shell. Two libraries can have the same name in the system as long as they are read from different sources. The **list_libs** command shows all libraries present in the system.

The **read_lib** command will check the input model file library syntax against model syntax specified by **-model_type** option. If any content of the input model file library does not belong to the specified model, syntax error will be reported.

For details on library file syntax, see the Library Compiler documentation. Function statements and state information are ignored when reading technology library files if you do not have a Library Compiler license.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, the library file *tech_lib.lib*, is read. This file contains technology information in Synopsys technology library entry format,

```
prompt> read_lib ~synopsys/libraries/tech_lib.lib
```

The following example reads a library file as well as a CTL file when the **-test_model** option is used:

```
prompt> read_lib bar.lib -test_model {usr1.ctl usr2:usr2.ctl}
```

The following examples read in a lvf, ccs_power or power_aware model file respectively when the **-model_type** option is used:

```
prompt> read_lib aocv.lib -model_type lvf
prompt> read_lib aocv.lib -model_type ccs_power
prompt> read_lib aocv.lib -model_type power_aware
```

The following example reads a library file and report screener messages in HTML:

```
prompt> read_lib bar.lib -html
```

SEE ALSO

```
write_lib(2)
search_path(3)
```

redirect

Redirects the output of a command to a file.

SYNTAX

```
string **redirect**
[-append] [-tee] [-file | -variable | -channel] [-compress]
target
{command_string}

string target
string command_string
```

ARGUMENTS

-append

Appends the output to *target*.

-tee

Like the unix command of the same name, sends output to the current output channel as well as to the *target*.

-file

Indicates that *target* is a file name, and redirection is to that file. This is the default. It is exclusive of *-variable* and *-channel*.

-variable

Indicates that *target* is a variable name, and redirection is to that Tcl variable. It is exclusive of *-file* and *-channel*.

-channel

Indicates tha *target* is a Tcl channel, and redirection is to that channel. It is exclusive of *-variable* and *-file*.

-compress

Compress when writing to file. If redirecting to a file, then this option specifies that the output should be compressed as it is written. The output file is in a format recognizable by "gzip -d". You cannot specify this option with *-append*.

target

Indicates the target of the output redirection. This is either a filename, Tcl variable, or Tcl channel depending on the command arguments.

command_string

The command to execute. Intermediate output from this command, as well as the result of the command, will be redirected to *target*. The *command_string* should be rigidly quoted with curly braces.

DESCRIPTION

The **redirect** command performs the same function as the traditional unix-style redirection operators > and >>. The *command_string* must be rigidly quoted (that is, enclosed in curly braces) in order for the operation to succeed.

Output is redirected to a file by default. Output can be redirected to a Tcl variable by using the *-variable* option, or to a Tcl channel by using the *-channel* option.

Output can be sent to the current output device as well as the redirect target by using the *-tee* option. See the examples section for an example.

The result of a **redirect** command which does not generate a Tcl error is the empty string. Screen output occurs only if errors occurred during execution of the *command_string* (other than opening the redirect file). When errors occur, a summary message is printed. See the examples.

Although the result of a successful **redirect** command is the empty string, it is still possible to get and use the result of the command that you redirected. Construct a **set** command in which you set a variable to the result of your command. Then, redirect the **set** command. The variable holds the result of your command. See the examples.

The **redirect** command is much more flexible that traditional unix redirection operators. With **redirect**, you can redirect multiple commands or an entire script. See the examples for an example of how to construct such a command.

Note that the builtin Tcl command **puts** does not respond to output redirection of any kind. Use the builtin **echo** command instead.

EXAMPLES

In the following example, the output of the plus procedure is redirected. The echoed string and the result of the plus operation is in the output file. Notice that the result was not echoed to the screen.

```
prompt> proc plus {a b} {echo "In plus" ; return [expr $a + $b]}
prompt> redirect p.out {plus 12 13}
prompt> exec cat p.out
In plus
25
```

In this example, a typo in the command created an error condition. The error message indicates that you can use **error_info** to trace the error, but you should first check the output file.

```
prompt> redirect p.out {plus2 12 13}
Error: Errors detected during redirect
        Use error_info for more info. (CMD-013)
prompt> exec cat p.out
Error: unknown command 'plus2' (CMD-005)
```

In this example, we explore the usage of results from redirected commands. Since the result of **redirect** for a command which does not generate a Tcl error is the empty string, use the **set** command to trap the result of the command. For example, assume that there is a command to read a file which has a result of "1" if it succeeds, and "0" if it fails. If you redirect only the command, there is no way to know if it succeeded.

```
redirect p.out { read_a_file "a.txt" }
# Now what?  How can I redirect and use the result?
```

But if you set a variable to the result, then it is possible to use that result in a conditional expression, etc.

```
redirect p.out { set rres [read_a_file "a.txt"] }
if { $rres == 1 } {
  echo "Read ok!"
}
```

The **redirect** command is not limited to redirection of a single command. You can redirect entire blocks of a script with a single **redirect** command. This simple example with **echo** demonstrates this feature:

```
prompt> redirect p.out {
?         echo -n "Hello "
?         echo "world"
?       }
prompt> exec cat p.out
Hello world
prompt>
```

The **redirect** command allows you to tee output to the previous output device and also to redirect output to a variable. This simple example with **echo** demonstrates these features:

```
prompt> set y "This is "
This is
prompt> redirect -tee x.out {
          echo XXX
          redirect -variable y -append {
            echo YYY
            redirect -tee -variable z {
              echo ZZZ
            }
          }
        }
XXX
prompt> exec cat x.out
XXX
prompt> echo $y
This is YYY
ZZZ

prompt> echo $z
ZZZ
```

SEE ALSO

echo(2)
error_info(2)
set(2)

remove_from_collection

Removes objects from a collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

```
collection remove_from_collection
   [-intersect]
   collection1
   object_spec
```

Data Types

```
collection1        collection
object_spec        list
```

ARGUMENTS

-intersect

Removes objects from collection1 not found in object_spec. Without this option, removes objects from collection1 that are found in object_spec.

collection1

Specifies the base collection to be copied to the result collection. Objects matching *object_spec* are removed from the result collection.

object_spec

Specifies a list of named objects or collections to remove. The object class of each element in this list must be the same as in the base collection. If the name matches an existing collection, the collection is used. Otherwise, the objects are searched for in the database using the object class of the base collection.

DESCRIPTION

The **remove_from_collection** command removes elements from a collection, creating a new collection.

If the base collection is homogeneous, any element of the *object_spec* that is not a collection is searched for in the database using the object class of the base collection. If the base collection is heterogeneous, any element of the *object_spec* that is not a collection is ignored.

If the *-intersect* option is not specified, which is the default mode, and if nothing matches the *object_spec*, the resulting collection is a copy of the base collection. If everything in the *collection1* option matches the *object_spec*, the result is the empty collection. With the *-intersect* option the results are reversed.

For background on collections and querying of objects, see the **collections** man page.

EXAMPLES

The following example from PrimeTime gets all input ports except "CLOCK".

```
pt_shell> set cPorts [remove_from_collection [all_inputs] CLOCK]
{"in1", "in2"}
```

SEE ALSO

```
add_to_collection(2)
collections(2)
```

remove_host_options

.prod lc Removes **-max_cores** specification set by the **set_host_options** command.

SYNTAX

```
status remove_host_options
```

ARGUMENTS

The **remove_host_options** command has no options.

DESCRIPTION

The **remove_host_options** command unsets the -max_cores option set by the **set_host_options** command. It is equivalent to

```
set_host_options -max_cores 1
```

Running the **remove_host_options** command disables parallel execution of the read_lib command.

EXAMPLE

```
lc_shell> remove_host_options
```

SEE ALSO

```
set_host_options(2)
report_host_options(2)
```

remove_lib

Removes a list of libraries from lc_shell memory.

SYNTAX

```
int
remove_lib [lib_list |  -all]

list lib_list
```

ARGUMENTS

lib_list

A list of libraries to remove.

-all

Causes the removal of all libraries in lc_shell.

DESCRIPTION

Removes a list of libraries from lc_shell. If you don't specify any arguments, nothing is removed. When a library is removed, the memory it occupies is freed, so the object is no longer in lc_shell.

Note:

Memory is freed to be reused by this same process. However, memory is not returned to the operating system until you exit the Library_compiler or lc_shell.

EXAMPLES

This example removes my_lib:

```
lc_shell> remove_lib my_lib
Removing library 'my_lib'
```

This example removes all the library objects in lc_shell.

```
lc_shell> remove_lib -all
Removing library 'test_libA'
Removing library 'test_libB'
Removing library 'test_libC'
```

SEE ALSO

None.

remove_license

Removes a licensed feature.

SYNTAX

```
status **remove_license**
   feature_list
   [-keep num_licenses]
```

Data Types

```
   feature_list     list
   num_licenses     integer
```

ARGUMENTS

feature_list

Specifies the list of features to remove. If you specify more than one feature, the list must be enclosed in braces ({}).

By looking at your key file, you can determine all of the features licensed at your site.

-keep *num_licenses*

Specifies the number of licenses to be retained for each feature after the command has completed. If this option is not specified, all of the features' licenses are checked in.

DESCRIPTION

This command removes the specified licensed features from the features you are currently using.

For multicore runs, where multiple licenses might be required for certain features, you can use the **-keep** option to restrict the number of licenses that are checked in. This is analogous to the **get_license -quantity** option.

The **list_licenses** command provides a list of the features that you are currently using.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example removes the multivoltage license:

```
prompt> remove_license Galaxy-MV
```

The following example removes some, but not all, of the licenses required for a multicore **compile_ultra** run in Design Compiler:

```
prompt> list_licenses

Licenses in use:
        DC-Expert (4)
        DC-Ultra-Features (4)
        DC-Ultra-Opt (4)
        Design-Compiler
1
prompt> remove_license -keep 2 {DC-Expert DC-Ultra-Opt DC-Ultra-Features}
1
prompt> list_licenses

Licenses in use:
        DC-Expert (2)
        DC-Ultra-Features (2)
        DC-Ultra-Opt (2)
        Design-Compiler
1
```

SEE ALSO

```
check_license(2)
license_users(2)
list_licenses(2)
get_license(2)
```

rename

Renames or deletes a command.

SYNTAX

```
string rename
   old_name
   new_name
```

ARGUMENTS

old_name

Specifies the current name of the command.

new_name

Specifies the new name of the command.

DESCRIPTION

Renames the *old_name* command so that it is now called *new_name*. If *new_name* is an empty string, then *old_name* is deleted. The *old_name* and *new_name* arguments may include namespace qualifiers (names of containing namespaces). If a command is renamed into a different namespace, future invocations of it will execute in the new namespace. The **rename** command returns an empty string as result.

Note that the **rename** command cannot be used on permanent procedures. Depending on the application, it can be used on all basic builtin commands. In some cases, the application will allow all commands to be renamed.

WARNING: **rename** can have serious consequences if not used correctly. When using **rename** on anything other than a user-defined Tcl procedure, you will be warned. The **rename** command is intended as a means to wrap other commands: that is, the command is replaced by a Tcl procedure which calls the original. Parts of the application are written as Tcl procedures, and these procedures can use any command. Commands like **puts**, **echo**, **open**, **close**, **source** and many others are often used within the application. Use **rename** with extreme care and at your own risk. Consider using **alias**, Tcl procedures, or a private namespace before using **rename**.

EXAMPLES

This example renames my_proc to my_proc2:

```
prompt> proc my_proc {} {echo "Hello"}
prompt> rename my_proc my_proc2
prompt> my_proc2
Hello
prompt> my_proc
Error: unknown command 'my_proc' (CMD-005).
```

SEE ALSO

define_proc_attributes(2)

report_app_var

Shows the application variables.

SYNTAX

```
string report_app_var
   [-verbose]
   [-only_changed_vars]
   [pattern]
```

Data Types

```
pattern      string
```

ARGUMENTS

-verbose

Shows detailed information.

-only_changed_vars

Reports only changed variables.

pattern

Reports on variables matching the pattern. The default is "*".

DESCRIPTION

The **report_app_var** command prints information about application variables matching the supplied pattern. By default, all descriptive information for the variable is printed, except for the help text.

If no variables match the pattern, an error is returned. Otherwise, this command returns the empty string.

If the **-verbose** option is used, then the command also prints the help text for the variable. This text is printed after the variable name and all lines of the help text are prefixed with " #".

The Constraints column can take the following forms:

{val1 ...}

The valid values must belong to the displayed list.

val \<= a

The value must be less than or equal to "a".

val \>= b

The value must be greater than or equal to "b".

b \<= val \<= a

The value must be greater than or equal to "b", and less than or equal to "a".

EXAMPLES

The following are examples of the **report_app_var** command:

```
prompt> report_app_var sh*
Variable                 Value     Type    Default   Constraints
------------------------ --------- ------- --------- ---------------------
sh_continue_on_error     false     bool    false
sh_script_stop_severity  none      string  none      {none W E}
 \.\.\.


prompt> report_app_var sh* -verbose
Variable                 Value     Type    Default    Constraints
------------------------ --------- ------- --------- ---------------------
sh_continue_on_error     false     bool    false
   # Allows source to continue after an error
sh_script_stop_severity  none      string  none      {none W E}
   # Indicates the error message severity level which would cause
   # a script to stop executing before it completes
 \.\.\.
```

SEE ALSO

```
get_app_var(2)
set_app_var(2)
write_app_var(2)
```

report_check_library_options

Reports the values or the status of the options set by the **set_check_library_options** command.

SYNTAX

```
status report_check_library_options
   [-logic]
   [-physical]
   [-logic_vs_physical]
   [-default]
```

ARGUMENTS

-logic

Reports the values or the status of the options set by the **set_check_library_options** command to check logical libraries.

See the **-logic** option in the **set_check_library_options** man page for more information.

-physical

Reports the values or the status of the options set by the **set_check_library_options** command to check physical libraries.

-logic_vs_physical

Reports the values or the status of the options set by the **set_check_library_options** command to check cross logical and physical libraries.

See the **-logic_vs_physical** option in the **set_check_library_options** man page for more information.

-default

Reports the default values of all library-checking options. If this option is not specified, the current values of the options are reported.

DESCRIPTION

The **report_check_library_options** command reports the values or the status of the options set by the **set_check_library_options** command. The **report_check_library_options** command provides information about options used for checking between logical libraries,

Use the **report_check_library_options** command after running **set_check_library_options**.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, **report_check_library_options** checks all data for the specified *test.db* library, and then reports the option values:

```
prompt> set_check_library_options -upf

prompt> report_check_library_options -logic
1
Report options for check_library
********************************************
  Check MCMM                           : false
  Check UPF                            : true
  Validate timing models               : false
  Validate noise models                : false
  Validate power models                : false
  Check for power optimization         : false

1
```

SEE ALSO

```
check_library(2)
set_check_library_options(2)
```

report_host_options

.prod lc Prints a report of multi-CPU processing options as defined by the **set_host_options** command.

SYNTAX

```
status report_host_options
```

ARGUMENTS

The **report_host_options** command has no arguments.

DESCRIPTION

By default, the **report_host_options** command prints a report of the multi-CPU processing options defined by the **set_host_options** command used by IC Compiler.

EXAMPLES

The following is an example of the **report_host_options** command:

```
lc_shell> report_host_options
```

SEE ALSO

```
    remove_host_options(2)
    set_host_options(2)
```

report_lib

Displays information about the specified logic library.

SYNTAX

```
status report_lib
   [-all]
   [-em]
   [-fpga]
   [-k_factors]
   [-power]
   [-power_label]
   [-table]
   [-full_table]
   [-timing]
   [-timing_arcs]
   [-timing_label]
   [-noise]
   [-noise_arc]
   [-vhdl_name]
   [-yield]
   [-switch]
   [-pg_pin]
   [-char]
   [-operating_condition]
   [-op_cond_name op_cond_name]
   [-routing_rule]
   [-rwm]
   [-user_defined_data]
   library_name
   [cell_list]
```

Data Types

```
op_cond_name      string
library_name      string
cell_list         string
```

ARGUMENTS

-all

Lists all timing-related, power-related, electromigration-related, and FPGA-related information.

This option applies only to logic libraries.

-em

Lists all detailed electromigration-related information.

This option applies only to logic libraries.

-fpga

Lists all detailed FPGA-related information, such as parts and I/O cell attributes.

This option applies only to FPGA libraries.

-full_table

Lists each cell's state-table description in tabular, expanded form.

This option applies only to logic libraries.

-k_factors

Lists scaling information for library cells.

This option applies only to logic libraries.

-power

Lists all detailed power-related information.

This option applies only to logic libraries.

-power_label

Lists the power labels for all power arcs, if specified, in a tabular form.

This option applies only to logic libraries.

-routing_rule

Lists nondefault routing rule information from physical libraries.

This option applies only to physical libraries.

-rwm

Lists routing wire model information from physical libraries.

In a routing wire model report, the original routing wire model is printed out, which includes overlap wire ratio, adjacent wire ratio, wire ratio and wire length of both directions. Also reported is capacitance (in database units) per micron for each layer and for each direction (based on each layer's wire ratio), based on the default operating condition. The report shows resistance per square (in database units) for each direction based on the default operating condition. These derived numbers provide useful information before applying the routing wire-model using the **set_routing_wire_model** command.

This option applies only to physical libraries.

-table

Lists each cell's state-table description in compact form.

This option applies only to logic libraries.

-timing

Lists all detailed timing-related information.

This option applies only to logic libraries.

-timing_arcs

Lists all cell timing arcs.

This option applies only to logic libraries.

-timing_label

Lists the timing labels for all timing arcs, if specified, in a tabular form.

This option applies only to logic libraries.

-user_defined_data

Lists detailed information about the user-defined groups and attributes.

This option applies to both logic libraries and physical libraries.

-vhdl_name

Lists the cells and ports whose database (.db) names are different from their VHDL names.

This option applies only to logic libraries.

-yield

Lists failure rate information for library cells.

This option applies only to logic libraries.

-switch

Lists pin switch function information and cell-level steady state current information.

Use the **-pg_pin** option to display the switch function of a power or ground pin.

This option applies only to logic libraries.

-pg_pin

Lists power and ground pin information, such as the PG pin definition and the voltage name to which a signal pin has been linked and the PG type.

This option applies only to logic libraries.

-char

Lists cell characterization information, such as sensitization and pre-driver model.

This option applies only to logic libraries.

-operating_condition

Lists all library operating condition information.

When this is the only option selected, the **report_lib** command reports only operating condition information; no other information is reported.

-op_cond_name *op_cond_name*

Lists operating condition information by name.

When this is the only option selected, the **report_lib** command reports only operating condition information, no other information is reported.

cell_list

Specifies a list of cells about which information is to be reported. The default is to report information about all cells in the logic library or physical library.

This option applies to both logic libraries and physical libraries.

library_name

Specifies the name of the library to report.

This argument is required.

-noise

Lists all detailed noise-related information in the specified library.

This option applies only to logic libraries.

-noise_arcs

Lists all cell noise arcs.

This option applies only to logic libraries.

DESCRIPTION

The **report_lib** command displays information about the specified logic library, physical library, or symbol library.

By default, a logic library report displays a list of operating conditions, timing ranges, and wire load models, as well as a listing of all the cells in the library annotated with their attributes. You can restrict the set of cells by using the *cell_list* argument.

Any library object added by using the **update_lib** command is marked with an asterisk (*) following its name, to identify those objects that have been added since the initial library was created with the **read_lib** command.

The **-timing**, **-noise**, **-table**, **-full_table**, **-timing_label**, **-power_label**, **-em**, **-power**, **-yield**, **-pg_pin**, and **-switch** options can be used only with a logic library that was read in from a library source (.lib) file. If the library is read in from a database (.db) file, an error message is issued.

A physical library report displays a list of available layers, a list of vias, a list of sites and a list of cells. If a *cell_list* is specified with **report_lib**, the report includes only the specified cells; otherwise all cells in the given physical library are listed.

A symbol library report displays a list of the names of symbol definitions contained in the library *library_name*. It also reports the route grid and meter scale for the library.

To generate a report, the specified library must be loaded into memory, unless it is found in the search path. The **list_libs** command displays the libraries that are currently loaded.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following command generates a report for the *mylib* logic library. This type of report includes names of the library cells, wire load models, timing ranges, and operating conditions, the date the library was created, the date the library was generated, and CCS timing, CCS power, CCS Noise and LVF modeling information for the library cells.

```
prompt> report_lib mylib

****************************************
Report : library
Library: mylib
Version: C-2009.06
****************************************

Library Type            : Technology
Tool Created            : 2003.06
Date Created            : 21-Jun-101 (INF CREATED ON 1-JUL-2001)
Library Version         : 1.000000
Time Unit               : 1ns

Capacitive Load Unit    : 1.000000pf
Pulling Resistance Unit : 1kilo-ohm
Voltage Unit            : 1V
Current Unit            : 1uA
Dynamic Energy Unit     : 1.000000pJ (derived from V,C units)
Leakage Power Unit      : 1pW
Bus Naming Style        : %s[%d] (default)

Operating Conditions:

    Operating Condition Name : mylib
    Library : mylib
    Process :   1.20
    Temperature : 125.00
    Voltage :   1.08
    Interconnect Model : worst_case_tree

Input Voltages:

    No input_voltage groups specified.
```

```
Output Voltages:

    No output_voltage groups specified.

default_wire_load_capacitance:  0.000096
default_wire_load_resistance:   0.000380
default_wire_load_area: 0.010000

Wire Loading Model:

Name           :   05x05
Location       :   mylib
Resistance     :   0.00038
Capacitance    :   9.6e-05
Area           :   0.01
Slope          :   15.9634
Fanout   Length   Points Average Cap Std Deviation
--------------------------------------------------------------
     1     9.80
     2    22.01
     3    35.15
     4    49.29
     5    64.54
     6    80.97
     7    98.68
     8   117.77
     9   138.32
    10   160.41
    11   184.15
    12   209.61
    13   236.90
    14   266.09
    15   297.29
    16   330.57
    17   366.04
    18   403.77
    19   443.86
    20   486.41

 ...

Wire Loading Model Selection Group:

   Name           : a

        Selection             Wire load name
    min area   max area
   -------------------------------------------
        0.00       5.00          05x05
        5.00      10.00          10x10
       10.00      20.00          20x20
       20.00      30.00          30x30

Wire Loading Model Mode: enclosed.

Wire Loading Model Selection Group: a.

Porosity information:

    No porosity information specified.

Routing Layers:
```

```
           "metal1"   "metal2"   "metal3"   "metal4"   "metal5"
              "metal6"   "metal7"   "metal8"

default_min_porosity:   0.000000

In_place optimization mode:  match_footprint

Timing Ranges:

    No timing ranges specified.

Delay Threshold Trip-Points:

        input_threshold_pct_rise:       50
        output_threshold_pct_rise:      50
        input_threshold_pct_fall:       50
        output_threshold_pct_fall:      50

Slew Threshold Trip-Points:

        slew_lower_threshold_pct_rise:  20
        slew_upper_threshold_pct_rise:  80
        slew_lower_threshold_pct_fall:  20
        slew_upper_threshold_pct_fall:  80

        slew_derate_from_library:       0.6

Components:

    Attributes:
       af - active falling
       ah - active high
       al - active low
       ar - active rising
        b - black box (function unknown)
       ce - clock enable
       cg - clock gating integrated cell
       ub - unbuffered
      pgt - pass gate
      ctl - has CTL test model
        d - dont_touch
      iso - isolation cell
       ls - level shifter
       mo - map_only
     1p0g - power only, no ground
     0p1g - ground only, no power
     0p0g - no power/ground
        p - preferred
       hp - hold_preferred
     pwrg - power gating cell
        r - removable
        s - statetable
      sa0 - dont_fault stuck-at-0
      sa1 - dont_fault stuck-at-1
     sa01 - dont_fault both stuck-at-0 and stuck-at-1
       sz - use_for_size_only
        t - test cell. t(scan_type) as
          t(mux_ff) - muxed flip-flop
          t(mux_ld) - muxed latch
          t(clk_scan) - clocked scan
          t(clk_scan_ld) - clocked scan latch
          t(lssd_dld) - double latch LSSD
          t(lssd_sld) - single latch LSSD
          t(lssd_clk) - clocked LSSD
```

```
            t(lssd_auxclk) - auxiliary clock LSSD
          u - dont_use
        udp - usable for datapath generators
        ufc - user_function_class. ufc(type) as
            ufc(u) - user-defined
            ufc(a) - automatically generated
        ccs - composite current source
      c_ccs - compact ccs timing modeling
       ccsn - ccs noise data modeling
       ccsp - ccs power data modeling
       recv - receiver model
        lvf - lvf modeling
        imc - macro cell
  switch_cg - switch cell (coarse grain)
        ret - retention cell
         ao - always on cell
       gicg - generic integrated clock gating
         va - variation-aware ccs timing modeling
        val - variation-aware ccs leakage power modeling
        fil - filler cell
        tap - tap cell
      decap - decap cell
        pll - pll cell


  Pin Attributes:
         ao - always on pin
        lse - level shifter enable pin
       isoe - isolation cell enable pin
       scmr - standard cell main rail

     Cell                Footprint    Attributes
     -----------------------------------------
     AN2                              ccs, recv
     OR2                              ccs
   ...
  1
```

The following command generates a report on library timing in the *AND2* cell of the *tech_lib* logic library:

```
  prompt> report_lib tech_lib -timing AND2
  ****************************************
  Report : library
  Library: tech_lib
  Version: C-2009.06
  ****************************************

  Library Type            : Technology
  Tool Created            : 2003.06
  Date Created            : 21-Jun-101 (INF CREATED ON 1-JUL-2001)
  Library Version         : 1.000000

  Delay Model             : table_lookup
  Time Unit               : 1ns

  Timing Attributes Defaults:

     Attribute                    Default
     -----------------------------------------
       max_transition
       max_fanout
       fanout_load                1
       max_capacitance
```

```
Lookup Table Template:

   Template_name
   --------------------------------------------------------------------
   del_0_5_7_w
       VARIABLE_1: input_net_transition
       VARIABLE_2: total_output_net_capacitance
       INDEX_1:  0.0150  0.2500  0.6500  1.4000  3.0000
       INDEX_2:  0.0000  0.0035  0.0070  0.0192  0.0402  0.0752  0.1750
   ...

CELL(AND2): 1.25;

  PIN(A): in, 0.0035, 1, 3, , ;
  END_PIN A;

  PIN(B): in, 0.00375, 1, 3, , ;
  END_PIN B;

  PIN(Z): out, 0, 18, 2, 0.175, , ;
    DELAY: A, Z, prop, pos_unate, '', ( ,  ), ( ,  ), ( ,  );
      cell_rise ( del_0_5_7_w ) :
         VALUES :    0.0710  0.1000  0.1240  0.2050  0.3440  0.5750
                     1.2330  0.1120  0.1410  0.1650  0.2470  0.3860
                     0.6170  1.2750  0.1520  0.1840  0.2100  0.2920
                     0.4320  0.6630  1.3210  0.1990  0.2360  0.2630
                     0.3470  0.4870  0.7190  1.3770  0.2630  0.3080
                     0.3400  0.4280  0.5700  0.8030  1.4630

      cell_fall ( del_0_5_7_w ) :
         VALUES :    0.0620  0.0900  0.1160  0.2020  0.3500  0.5950
                     1.2940  0.1050  0.1360  0.1620  0.2480  0.3960
                     0.6420  1.3410  0.1440  0.1770  0.2040  0.2930
                     0.4410  0.6860  1.3860  0.1870  0.2240  0.2530
                     0.3430  0.4920  0.7390  1.4380  0.2410  0.2850
                     0.3180  0.4130  0.5650  0.8140  1.5150

      rise_transition ( del_0_5_7_w ) :
         VALUES :    0.0450  0.0960  0.1480  0.3400  0.6720  1.2270
                     2.8080  0.0490  0.0980  0.1500  0.3400  0.6720
                     1.2270  2.8080  0.0580  0.1060  0.1560  0.3440
                     0.6740  1.2270  2.8080  0.0720  0.1200  0.1660
                     0.3480  0.6770  1.2300  2.8080  0.0930  0.1460
                     0.1910  0.3630  0.6860  1.2370  2.8120

      fall_transition ( del_0_5_7_w ) :
         VALUES :    0.0370  0.0900  0.1440  0.3380  0.6720  1.2300
                     2.8190  0.0430  0.0930  0.1460  0.3380  0.6720
                     1.2300  2.8200  0.0510  0.1020  0.1530  0.3420
                     0.6740  1.2300  2.8190  0.0650  0.1150  0.1640
                     0.3480  0.6780  1.2330  2.8200  0.0870  0.1400
                     0.1870  0.3660  0.6920  1.2430  2.8250

    DELAY: B, Z, prop, pos_unate, '', ( ,  ), ( ,  ), ( ,  );
       ...

  END_PIN Z;
END_CELL AND2;

Number of library cells: 1
Number of reported cells: 1
1
```

The following command generates a report on library noise in the *AND2* cell of the *tech_lib* logic library:

```
prompt> report_lib tech_lib -noise AND2
 ...
CELL(AND2): 4, ;

  PIN(A): in, 2, , , ;
  END_PIN A;

  PIN(B): in, 2, , , ;
  END_PIN B;

  PIN(Z): out, , , , , , (, ), (, );
      noise_immunity_high ( noise5x5 ) :
         INDEX_2 :   0.3620  0.7250  1.0870  1.4490  1.8120
         VALUES :    0.7330  1.0730  1.4120  1.7520  2.0920  0.8730
                     1.2140  1.5540  1.8940  2.2340  1.0950  1.4420
                     1.7870  2.1320  2.4770  1.2980  1.6480  1.9960
                     2.3430  2.6910  1.7030  2.0600  2.4140  2.7660
                     3.1190
  END_PIN Z;
END_CELL AND2;
```

For details about library timing report contents and syntax, see the *Library Compiler Modeling Timing, Signal Integrity, and Power in Technology Libraries User Guide*.

The following example generates a report on power information in the *tech_lib* logic library:

```
prompt> report_lib tech_lib -power
 ...
Power Information:

    Attributes:
       a - average power specification
       i - internal power
       l - leakage power
      rf - rise and fall power specification

                            Power

      Cell          #  Attr    Toggling pin   source of path  When
   ----------------------------------------------------------------
   AN2           0  l
                 1  i,a     Z              A
                 2  i,a     Z              B
   INV           0  l
                 1  i,a     Z              A
   NO2           0  l
                 1  i,a     Z              A
                 2  i,a     Z              B
   flop1         0  l
                 1  i,a     CP
                 2  i,a     Q              CP
                 3  i,a     Q              D
                 4  i,a     QN             CP
                 5  i,a     QN             D
   latch1        0  l
                 1  i,a     G
                 2  i,a     Q              G
                 3  i,a     Q              D
                 4  i,a     QN             G
                 5  i,a     QN             D
```

The following command generates a report on library electromigration in the *ad3* cell of the *tech_lib* logic library:

```
prompt> report_lib tech_lib -em ad3
 ...
CELL(ad3): 2;
  PIN(y): out, 0, , , 1.812, , ;
    ELECTROMIGRATION: a;
      em_max_toggle_rate ( output_by_cap_and_trans ) :
         VALUES :    2.0000  1.0000  0.5000  1.5000  0.7500  0.3300
                     1.0000  0.5000  0.1500

    ELECTROMIGRATION: b;
      em_max_toggle_rate ( output_by_cap_and_trans ) :
         VALUES :    2.0000  1.0000  0.5000  1.5000  0.7500  0.3300
                     1.0000  0.5000  0.1500

    ELECTROMIGRATION: c;
      em_max_toggle_rate ( output_by_cap_and_trans ) :
         VALUES :    2.0000  1.0000  0.5000  1.5000  0.7500  0.3300
                     1.0000  0.5000  0.1500

  END_PIN y;

  PIN(a): in, 1, , ;
  END_PIN a;

  PIN(b): in, 1, , ;
  END_PIN b;

  PIN(c): in, 1, , ;
    ELECTROMIGRATION:
      em_max_toggle_rate ( input_by_trans ) :
         VALUES :    1.5000  1.0000  0.5000

  END_PIN c;
END_CELL ad3;

Number of library cells: 1
Number of reported cells for em: 1
1
```

For details about library electromigration report contents and syntax, see the *Library Compiler Modeling Timing, Signal Integrity, and Power in Technology Libraries User Guide*.

The following command generates a report on timing arcs in the *tech_lib* logic library. The *INV_NEW* cell is identified as having been added using the **update_lib** command:

```
prompt> report_lib tech_lib -timing_arcs
 ...
                         Arc                    Arc Pins
   Cell         Attributes   #   Sense/type     From        To  When
   -----------------------------------------------------------------
   AND                       0    pos unate       A           Z
                             1    pos unate       B           Z
   INVET                     0    neg unate       A           Z
   INV_NEW                   0    neg unate       A           Z
 ...
```

If the **-timing_arcs** option is not used, the last five columns do not appear.

The following command generates a report for a symbol library. The report includes the names of all symbol definitions, the route_grid, and the meter_scale:

```
prompt> report_lib sym_lib.sdb
```

The following command generates a report that displays timing arcs for the inverter cell from the *tech_lib* logic library:

```
prompt> report_lib tech_lib -timing_arcs inverter
```

The following example generates a report that displays noise arcs for the inverter cell from the *tech_lib* logic library:

```
prompt> report_lib tech_lib -noise_arcs inverter
```

The following is an example of a compact report on state-table information in the *tech_lib* logic library:

```
prompt> report_lib tech_lib -table

State table descriptions:

CELL(D_LATCH):

  PIN(Q): CD, D, G
    TABLE: HNLNLLLL
  END_PIN Q;

  PIN(QN):
    STATE_FUNCTION: Q'
  END_PIN QN;
END_CELL D_LATCH;
```

The following is an example of a tabular report on state-table information in the *tech_lib* logic library:

```
prompt> report_lib tech_lib -full_table

State table descriptions:

CELL(D_LATCH):

  PIN(Q): CD, D, G
    TABLE:
    000    L
    001    L
    010    L
    011    L
    100    N
    101    L
    110    N
    111    H
  END_PIN Q;

  PIN(QN):
    STATE_FUNCTION: Q'
  END_PIN QN;

END_CELL D_LATCH;
```

The following is an example of a pg_pin report in the *tech_lib* logic library:

```
prompt> report_lib tech_lib -pg_pin

 ...
CELL(LVLHLEHX2):

  PG_PIN(VDDI):
    VOLTAGE_NAME: VDDH
    PG_TYPE: primary_power
  END_PIN VDDI;

  PG_PIN(VSS):
    VOLTAGE: VSS
    PG_TYPE: primary_ground
  END_PIN VSS;

  PIN(QN):
    RELATED_POWER_PIN : VDDI
    RELATED_GROUND_PIN : VSS
  END_PIN QN;

END_CELL LVLHLEHX2;
```

The following is an example of a characterization report in the *tech_lib* logic library:

```
prompt> report_lib tech_lib -char

****************************************
Report : library
Library: tech_lib
Version: A-2007.12
****************************************

Library Type             : Technology
Tool Created             : A-2007.12
Date Created             : Not Specified
Library Version          : Not Specified

Sensitization : 2IN_1OUT
  pin_names : (A, B, C)
  vector(0) : (0, 0, 0)
  vector(1) : (0, 0, 1)
  vector(2) : (0, 1, 0)
  vector(3) : (0, 1, 1)
  vector(4) : (1, 0, 0)
  vector(5) : (1, 0, 1)
  vector(6) : (1, 1, 0)
  vector(7) : (1, 1, 1)
  vector(8) : (1, 1, 1)

CELL(AN2): 2;
  SENSITIZATION_MASTER: 2IN_1OUT
  PIN_NAME_MAP : (A, B, Z)

  PIN(A): in, 1, , , , ;
  END_PIN A;

  PIN(B): in, 1, , , , ;
  END_PIN B;

  PIN(Z): out, 0, , , , , ;
    DELAY: A, Z, prop, pos_unate, '', (1, 1), (0, 0), (0.1443, 0.0523);
      sensitization_master: 2IN_1OUT
      pin_name_map : (A, B, Z)
      wave_rise : (3, 4, 5)
      wave_fall : (0, 3, 7)
      wave_rise_sampling_index: 2
      wave_fall_sampling_index: 2
      wave_rise_timing_interval : (0, 0.5)
      wave_fall_timing_interval : (0, 0.45)
    DELAY: B, Z, prop, pos_unate, '', (0.48, 0.77), (0, 0), (0.1443, 0.0523);
  END_PIN Z;
END_CELL AN2;
```

The following is an example of a switch-cell report in the *tech_lib* logic library:

```
prompt> report_lib tech_lib -switch
 ...
CELL(FD1): 2;

  DC_CURRENT:
      dc_current ( ccsn_dc_29x29 ) :
      RELATED_SWITCH_PIN : D
      RELATED_PG_PIN : PWR
      RELATED_INTERNAL_PG_PIN : VVDD
         INDEX_1 :  -1.2000 -0.6000 -0.2400 -0.1200  0.0000  0.0600
                     0.1200  0.1800  0.2400  0.3000  0.3600  0.4200
                     0.4800  0.5400  0.6000  0.6600  0.7200  0.7800
                     0.8400  0.9000  0.9600  1.0200  1.0800  1.1400
                     1.2000  1.3200  1.4400  1.8000  2.4000

  ...
END_CELL AN2;
```

SEE ALSO

```
read_lib(2)
system_variables(3)
update_lib(2)
write_lib(2)
```

set_app_var

Sets the value of an application variable.

SYNTAX

```
string set_app_var
   -default
   var
   value
```

Data Types

```
var        string
value      string
```

ARGUMENTS

-default

Resets the variable to its default value.

var

Specifies the application variable to set.

value

Specifies the value to which the variable is to be set.

DESCRIPTION

The **set_app_var** command sets the specified application variable. This command sets the variable to its default value or to a new value you specify.

This command returns the new value of the variable if setting the variable was successful. If the application variable could not be set, then an error is returned indicating the reason for the failure.

Reasons for failure include:

- The specified variable name is not an application variable, unless the application variable **sh_allow_tcl_with_set_app_var** is set to true See the **sh_allow_tcl_with_set_app_var** man page for details.
- The specified application variable is read only.
- The value specified is not a legal value for this application variable

EXAMPLES

The following example attempts to set a read-only application variable:

```
prompt> set_app_var synopsys_root /tmp
Error: can't set "synopsys_root": variable is read-only
       Use error_info for more info. (CMD-013)
```

In this example, the application variable name is entered incorrectly, which generates an error message:

```
prompt> set_app_var sh_enabel_page_mode 1
Error: "sh_enabel_page_mode" is not an application variable
        Use error_info for more info. (CMD-013)
```

This example shows the variable name entered correctly:

```
prompt> set_app_var sh_enable_page_mode 1
1
```

This example resets the variable to its default value:

```
prompt> set_app_var sh_enable_page_mode -default
0
```

SEE ALSO

```
get_app_var(2)
report_app_var(2)
write_app_var(2)
```

set_check_library_options

Sets options for the **check_library** command for various logic library and physical library checks.

SYNTAX

```
status set_check_library_options
   [-scaling {scaling_types}]
   [-mcmm]
   [-upf]
   [-single_mode]
   [-optimization {power}]
   [-compare {construct | attribute | value}]
   [-tolerance {type relative_tolerance absolute_tolerance}]
   [-validate {timing noise [value]}]
   [-analyze {nominal_vs_sigma | table_trend | table_bound | table_slope | table_index | sensitivity |
   [-criteria {criteria_spec}]
   [-group_attribute {groups_or_attributes}]
   [-report_format {format_spec}
   [-leq]
   [-by_group_order]
   [-char_integrity]
   [-significant_digits]
   [-compensation]
   [-cell_area]
   [-cell_footprint]
   [-bus_delimiter]
   [-logic]
   [-routability]
   [-min_pin_layer layer_name]
   [-antenna]
   [-signal_em]
   [-same_name_cell]
   [-tech]
   [-placement]
   [-pattern_must_join_pin]
   [-pattern_must_join_pin_exclusion_list lib_pin_list]
   [-physical]
   [-logic_vs_physical]
   [-all]
   [-reset]
   [-report_all]
```

Data Types

scaling_types	list
type	string
relative_tolerance	float
absolute_tolerance	float

```
criteria_spec              list
groups_or_attributes       list
format_spec                list
layer_name                 string
lib_pin_list               list
```

ARGUMENTS

-scaling {*scaling_types*}

Specifies a list of logic library consistency checks for various types of scaling. You can specify one or more of the following values:

- **timing** specifies logic library consistency checking for composite current source (CCS) timing scaling.
- **noise** specifies logic library consistency checking for CCS noise scaling.
- **power** specifies logic library consistency checking for CCS or nonlinear delay model (NLDM) power scaling and includes driver and receiver models, load indexes, base curve_x, waveform, noise, and power models.

-mcmm

Specifies logic library consistency checking for multicorner-multimode, and includes the power_down_function attribute and power data.

-upf

Specifies logic library consistency checking for multivoltage and IEEE 1801, also known as Unified Power Format (UPF), and includes power special cells, the pg_pin, voltage_names, and power_down_function attributes, and power data.

-single_mode

Specifies logic library consistency checking for single mode libraries such as etm. The libraries are generated at the same corner but with different modes and should be specified by set_lib_group command. If not specified so, the libraries specified in -logic_library_name option or runtime libraries will be used for the check. Those libraries should be consistent in cell/pin/pg_pin groups/attributes.

-optimization {power}

Checks the libraries for various types of optimization. The **power** option checks for power optimization related issues in the libraries, namely, missing high threshold_voltage_group cells specified by the **target_library** variable or **check_library -logic_library_name** option (the latter having higher priority). When you use the **-optimization** option, use the **-criteria** option to specify the low threshold_voltage_group names. With optimization power checking, the tool checks for the presence of high-threshold-voltage library cells having the same logic functions as the low-threshold-voltage library cells.

-validate {timing noise value}

Specifies logic library consistency checking for library characterization between different models. The

{timing} option is for validation between different timing models (NLDM versus CCS). The optional **value** is for validation of the characterization value between models of different dimensions, such as a one-dimensional NLDM and a two-dimensional CCS model. The **{noise}** option is for validation of delay/slew between CCSN and NLDM.

-compare {construct | attribute | value}

Specifies the type or types of information compared between logic libraries, which can be one or more of the keywords **construct**, **attribute**, or **value**. For **attribute** or **value** comparison, only two libraries are compared; if more than two libraries are specified, only the first two are compared.

- **-compare {construct}** checks to see if constructs or attributes are missing.
- **-compare {attribute}** checks to see if constructs and attributes are missing and if the attribute values are inconsistent. This option checks all groups, subgroups, and attribute values except characterization values. It is a superset of the **-compare {construct}** option.
- **-compare {value}** compares values of each group and attribute between the libraries. This option is a superset of the **-compare {attribute}** option. In comparing characterization values, if the source is not from the .lib library files, the values are not reported, such as vector and capacitance values.

The comparison of the characterization values is controlled by the tolerances specified by the **-tolerance** option, or by the default values if the **-tolerance** option is not used. For minimum and maximum library checking and scaling group library checking, you should use the **-compare {attribute}** option to check all the library contents except for characterization values. When you specify this option, the following default checks are also performed: missing cells, missing and mismatched pins, and timing arcs.

-tolerance {*type relative_tolerance absolute_tolerance*}

Specifies a relative tolerance and an absolute tolerance for characterization value comparison, such as delay values. The format is

```
-tolerance {type rel_tol abs_tol}
```

The valid values for the *type* argument are delay, slew, delay_ocv, slew_ocv, delay_ccsn, slew_ccsn, delay_sensitivity, slew_sensitivity, delay_interpolation, slew_interpolation, constraint_interpolation, voltage, constraint, slew_index, load_index, time, power, current, energy, energy_ccsp, and capacitance. If the type is not specified, the values are for the output load capacitance. When you specify an absolute tolerance, do not include the unit. The units are fixed and can be reported by the **set_check_library_options** command. For example, specify the tolerance values for delay and slew as shown below:

```
-tolerance {delay 0.1 0.2 slew 0.3 0.4}
```

If you do not specify tolerance values for a type, the default values are used. The default values that are set as follows, in compliance with PrimeTime and the Library Quality Assurance System:

```
Relative tolerance for load index    : 0.01
Absolute tolerance for load index    : 0.001pF
Relative tolerance for time          : 0.04
Absolute tolerance for time          : 0.015ns
Relative tolerance for power         : 0.04
Absolute tolerance for power         : 5pW
Relative tolerance for delay         : 0.02
Absolute tolerance for delay         : 0.002ns
Relative tolerance for slew          : 0.03
```

```
Absolute tolerance for slew          : 0.003ns
Relative tolerance for ocv delay     : 0.02
Absolute tolerance for ocv delay     : 0.005ns
Relative tolerance for ocv slew      : 0.03
Absolute tolerance for ocv slew      : 0.0075ns
Relative tolerance for ccsn delay    : 0.03
Absolute tolerance for ccsn delay    : 0.003ns
Relative tolerance for ccsn slew     : 0.05
Absolute tolerance for ccsn slew     : 0.005ns
Relative tolerance for sensitivity delay : 0.1
Absolute tolerance for sensitivity delay : 0.01ns
Relative tolerance for sensitivity slew  : 0.15
Absolute tolerance for sensitivity slew  : 0.015ns
Relative tolerance for constraint    : 0.04
Absolute tolerance for constraint    : 0.015ns
Relative tolerance for capacitance   : 0.01
Absolute tolerance for capacitance   : 0.001pF
Relative tolerance for voltage       : 0.05
Absolute tolerance for voltage       : 0.02V
Relative tolerance for voltage index : 0.002
Absolute tolerance for voltage index : 0.002V
Relative tolerance for energy        : 0.01
Absolute tolerance for energy        : 0.1fJ

Please note that for 10nm (set lc_enable_10nm_mode true),
the default tolerances are as follows:
Relative tolerance for constraint    : 0.01
Absolute tolerance for constraint    : 0.001ns
Relative tolerance for voltage       : 0.01
Absolute tolerance for voltage       : 0.01V
```

-analyze {nominal_vs_sigma | table_trend | table_bound | table_slope | table_index | sensitivity | voltage_range | value_range | interpolation | lvf | pg_current | ccsp_time_constant}

Specifies what is analyzed. You can specify one of the following values:

- **nominal_vs_sigma** analyzes multiple characterization tables. In the case of variation-aware models, analyze nominal, -sigma, and +sigma tables to see the trend and standard deviation for each slew and load index grid of a linear model. The linear model is modeled by the va_parameters option.

  Specify the va_parameters in the **-group_attribute** option. If you do not specify the va_parameters, the variation-aware models for all va_parameters are analyzed. Wildcards apply to va_parameters.

- **table_trend** analyzes a characterization table for trend. It analyzes the trend within the table to check if the values change monotonically in terms of slew or load indexes.

- **table_bound** analyzes a characterization table for bounds. It analyzes the values of the table to check if the values exceed the upper or lower bound.

- **table_slope** analyzes a characterization table for slope. It analyzes the slope formed by two adjacent grid points within the table to check if the slope value exceeds the maximum or minimum slope criteria. The slope calculation uses normalized values to eliminate the effect of the unit and table types.

- **table_index** analyzes table index accuracy by calculating the ratio of each pair of adjacent indexes (next index to current index) and reports the index tables if at least one ratio meets the condition specified by the **-criteria {table_type max_index_ratio=max_ratio}** option.

- **sensitivity** analyzes a cell's sensitivity. It reports that the CCSN model response to the clamped waveform has an output delay or output slew that deviates from the CCSN model response to the original or normal non-clamped version of the waveform. Sensitivity analysis is a soft rule, meaning the failure in sensitivity does not affect cell pass rate.
- **voltage_range** analyzes output waveform's voltage swing to see if it reaches 95% of VDD or within user specified tolerance.
- **value_range** analyzes the range of a scalar attribute or table and reports those that are out of the specified range. It applies to all attributes that have values of floating point numbers. The attribute name is specified as the first parameter of the triplet in -criteria as follows: **set_check_library_options** -analyze {value_range} -criteria {[group/]attribute min_value=val1 max_value=val2}
- **interpolation** analyzes the delay/slew/constraint table accuracy by fitting the rows and columns of the 2-D NLDM timing model with bicubic and bilinear models, and comparing the values at mid index points.
- **lvf** analyzes 1) Ratio of sigma to nominal values and 2) Ratio of early (or late) sigma to later (or early) sigma values, and reports missing OCV/LVF models.
- **pg_current** analyzes ratio of peak current in power/ground pg_current when output is switching, and reports those smaller than min_power_to_ground_current_ratio.
- **ccsp_time_constant** analyzes ratio of time constant value between CCSP vs. CCST/NLDM, and reports those larger than max_ccsp_to_ccst_time_const_ratio/max_ccsp_to_nldm_time_const_ratio.

-criteria ***criteria_spec***

Specifies analysis criteria for lvf, variation analysis (multiple tables), non-variation analysis (single table), or low threshold_voltage_group for power optimization. This option is used with the **-analyze**, or **-optimization** option. It is specified in the following format:

```
{std_error>m.m slope>n.n trend=trend_symbol |
 table_type upper_bound=upper_bound_value |
 lower_bound=lower_bound_val table_type[/table_sub_index] min_slope=min_value max_slope=max_valu
 table_type max_index_ratio=max_ratio clamping_ratio=ratio_val min_power_to_ground_current_ratio
```

The std_error and slope are used by the linear regression models. The **trend_symbol** includes the following nine symbols:

```
/    monotonically increasing
\    monotonically decreasing
^    non-monotonically up
V    non-monotonically down
-    flat
M    Multiple peaks
W    Multiple troughs
N    >=2 peaks with 2nd not turning low, i.e., increasing, decreasing and increasing
u    >=2 peaks with 2nd not turning high, i.e., decreasing, increasing and decreasing
```

To specify the monotonically decreasing trend, use {trend = '\'}, {trend = \\}, or "trend = \\ ". Do not use {trend = \}. To specify the inequality trend, use {trend!= }. For the std_error and slope expressions, you can also use the less-than sign (<).

The variation-aware analysis reports the results that meet any of the specified criteria.

This option is used with the **-analyze** option.

You can also set criteria for table bounds and slope analysis in this field.

```
{table_type lower_bound=lower_value upper_bound=upper_value |
 table_type[/table_sub_index] min_slope=min_value max_slope=max_value}
```

```
The table type, table sub_index, and associated group names are:
------------------------------------------------------------
Type Name       Index Type              Related Group Names
------------------------------------------------------------
delay           slew_index                        cell_rise
                load_index                        cell_fall
------------------------------------------------------------
transition      slew_index                  rise_transition
                load_index                  fall_transition
------------------------------------------------------------
constraint      constrained_index           rise_constraint
                related_index               fall_constraint
------------------------------------------------------------
energy          slew_index                       rise_power
                load_index                       fall_power
------------------------------------------------------------
current         time                    output_current_rise
                                        output_current_fall
------------------------------------------------------------
capacitance     slew_index        receiver_capacitance1_rise
                load_index        receiver_capacitance1_fall
                                  receiver_capacitance2_rise
                                  receiver_capacitance2_fall
------------------------------------------------------------
dc_current      input_voltage                    dc_current
                output_voltage
------------------------------------------------------------
ccsn            time                    output_voltage_rise
                                        output_voltage_fall
                                       propagated_noise_low
                                      propagated_noise_high
------------------------------------------------------------
pg_current      time                             pg_current
ccsp_time_constant  time                         pg_current
------------------------------------------------------------
```

{lvth_groups=*lvt_name_list*} Specifies a list of low threshold_voltage_group strings for power optimization, such as {lvth_groups="lvt svt"}. lvth_groups is equivalent to the **-lvth_groups** option of the **set_multi_vth_constraint** command.

{clamping_ratio=*ratio_value*} Specifies a percentage for sensitivity analysis. By default clamping_ratio=0.95. This is used with -analyze {sensitivity} option.

{max_sigma_to_nominal_ratio=*r1* max_sigma_to_sigma_ratio=*r2*} Specifies two ratios: max_sigma_to_nominal_ratio and max_sigma_to_sigma_ratio. max_sigma_to_nominal_ratio is the maximum ratio of sigma value to nominal value, and max_sigma_to_sigma_ratio is the maximum ratio of early (or late) sigma value to late (early) sigma value. The calculated ratios that exceed the specified ratios will be reported. In the case of max_sigma_to_sigma_ratio, the condition can be either early/late or late/early exceeds max_sigma_to_sigma_ratio to be reported. The user specified values should meet max_sigma_to_nominal_ratio <= 1 and max_sigma_to_sigma_ratio > 1. Otherwise, the value will be ignored. By default, max_sigma_to_nominal_ratio=0.25 and max_sigma_to_sigma_ratio=3.

In the checking for the ratios, when the nominal or sigma slew (or delay) values at a grid point is <= the absolute slew (or delay) tolerance, the check will not be performed at such grid points. However, user can set to a tighter tolerance using -tolerance option to report such points or report all points violating the max ratios using -report_all.

This criteria is used with -analyze {lvf} option.

{min_power_to_ground_current_ratio=*ratio_value*} Specifies a ratio for pg_current peak current analysis. By default min_power_to_ground_current_ratio=10. This is used with -analyze {pg_current} option.

{max_ccsp_to_ccst_time_const_ratio=*ratio_value*} Specifies a ratio for CCSP vs. CCST time constant analysis. By default max_ccsp_to_ccst_time_const_ratio=10. This is used with -analyze {ccsp_time_constant} option. If either CCSP-time-constant/CCST-time-constant or reverse is larger than the ratio, it will be reported.

{max_ccsp_to_nldm_time_const_ratio=*ratio_value*} Specifies a ratio for CCSP vs. NLDM time constant analysis. By default max_ccsp_to_nldm_time_const_ratio=10. This is used with -analyze {ccsp_time_constant} option. If either CCSP-time-constant/NLDM-time-constant or reverse is larger than the ratio, it will be reported.

{linear_resistance min_value=*min_val* max_value=*max_val*} Specifies the min/max value for linear resistance calculated from dc_current in switch cell. Recommended min_value=10 Ohm and max_value=1000 Ohm. This is used with -analyze {value_range}.

-group_attribute {*groups_or_attributes*}

Checks or compares specific groups and attributes only. These groups and attributes are either liberty syntax or user defined ones existing in the library. You can specify one or more groups or attributes. An item can be a group, an attribute, a group and an attribute together with a value, or a group and an attribute together without a value. Use a space to separate each item in the list.

Use a forward slash (/) to combine a group name and an attribute name that you are specifying together. For example, the **{pin/direction}** option combines the pin group and the direction attribute under the pin group. In this case, the command checks the direction only in the pin group.

Use curly braces ({}) or double quotation marks ("") to enclose an expression delimited by spaces. If the specified group/attribute is not in the library or not liberty syntax or user defined one, it will be ignored with warning.

You can also specify mega-attributes: timing, noise, power, ccs_timing, ccs_noise and ccs_power that include all the associated groups in that type. ccs_timing includes: output_current_rise output_current_fall compact_ccs_rise compact_ccs_fall va_compact_ccs_rise va_compact_ccs_fall base_curve curve_x curve_y ccs_noise includes: output_voltage_rise output_voltage_fall propagated_noise_low propagated_noise_high ccs_power includes: dynamic_current switching_group intrinsic_parasitic intrinsic_resistance intrinsic_capacitance leakage_current pg_current compact_ccs_power gate_leakage timing includes both NLDM timing subgroups and ccs_timing: *cell_rise, *cell_fall, *rise_transition, *fall_transition, *rise_constraint, *fall_constraint, output_current_rise, output_current_fall, compact_ccs_rise, compact_ccs_fall, *receiver_capacitance* base_curve, curve_x, and curve_y. noise includes: ccsn_first_stage, ccsn_last_stage, dc_current, output_voltage_rise, output_voltage_fall, propagated_noise_low, and propagated_noise_high. power includes: leakage_power, internal_power, fall_power, rise_power, dynamic_current, switching_group, intrinsic_parasitic, intrinsic_resistance, intrinsic_capacitance leakage_current, pg_current, compact_ccs_power and gate_leakage

To specify specific group/tables, use -group_attribute {[group[(modifier)]/][attribute]}. In the above format, group is a group name with optional modifier: group_name(modifier), modifier is a logical expression in the form of attribute_name=attribute_value that serves as a constraint condition to specify a specific group. e.g. timing(timing_type=hold_rising). Please note there is no space in specifying group_name(attribute_name=attribute_value). attribute_name is a scalar attribute name, e.g. timing_type, and attribute_value is a value of the attribute_name. In the case of string type attribute_value, regular expressions with wildcards can be used. If there is a space in the string value a double quote is required on the whole string, e.g. timing(when="A B"); otherwise, it is optional. Please note that it does not support complicated logical operations such as OR (||), AND (&&). Here is an example of specifying the timing arc with timing_type=min_pulse_width: -group_attribute {timing(timing_type=min_pulse_width)}

The following example checks only the pin group whose name is A:

-group_attribute {"pin = A"}

The following wildcard characters are available for pattern matching:

- `An asterisk (*) matches any string, including a null string.`
- `A question mark (?) matches any single character.`

For example:

- **-group_attribute {pin}** checks the pin group only.
- **-group_attribute {direction}** checks the direction attribute only.
- **-group_attribute {pin/direction}** checks the pin's direction only.
- **-group_attribute {pin/direction=input}** checks the pin direction only when it is input.
- **-group_attribute {cell_rise cell_fall output_current_*}** checks and validates the NLDM and CCS timing models only.
- **-group_attribute {va_* compact_ccs_*}** checks and validates the variation-aware timing models only.

If you do not specify the **-group_attribute** option, the **check_library** command checks all groups and attributes that apply to the specified mode.

`-report_format {`*`format_spec`*`}`

Specifies the format for the generated report. The syntax of the specification is as follows:

```
[csv[=csv_dir]] [sql=sql_filename] [nosplit] [html] [display] [sort_by_cell | sort_by_group_type
```

You must specify at least one of the following values in the specification:

- **csv**[=*csv_dir*]

Selects comma-separated values (CSV) as the report format, which is compatible with spreadsheet programs. By default, the CSV files are stored in the current working directory. Specify the directory to store the CSV files by using the *csv_dir* argument.

The file names are constructed by concatenating library names with "_vs_" appended by check type and .csv, e.g. ff_1.2v_-30c_vs_ss_1.3v_80c_timing_mismatched.csv. When the names are longer than the system limit (255 chars), only the first library name with library count is used.

If you perform multiple runs with the same settings, the directory for the previous run is saved to a directory named *csv_dir*_bak.

If you do not specify this value, the **check_library** command generates a plain ASCII report.

- **sql**=*sql_filename*
Specifies sqlite database filename, e.g. test.db3 to store the check_library results in.

- **nosplit**
Prevents line-splitting, and facilitates writing scripts to extract information from the report output. Most of the information is listed in fixed-width columns. If the information in a given field exceeds the column width, the next field begins on a new line, starting in the correct column.

- **html**
Selects HTML as the report format. Optionally specify the directory in which to store the HTML files by using the *html_dir* argument. By default, it is the current directory.

- **display**
Launches a browser to view the check results in HTML. This should be used only with the **html** report format.

- **sort_by_cell | sort_by_group_type**
Specifies the sorting method. Use **sort_by_cell** to sort the validation and analysis tables by cell names. Use **sort_by_group_type** to sort the validation and analysis tables by group types, such as delay, slew, constraints, and receiver capacitance.

By default, the validation and analysis tables are sorted by the group type.

If you do not specify this option, the **check_library** command generates an ASCII report that is sorted by the group type.

-leq

Checks the logical equivalence for *when* conditions. For example, if you set the **-leq** option, the following expressions are reported as identical:

```
when : A+B
when : B+A
```

By default, logical equivalence is disabled and string comparisons are used.

-by_group_order

Compares or checks libraries by the order of library groups for non-unique groups and groups without name, such as leakage_power. This is for concurrent checking of multiple libraries.

-logic

Checks the following options for the logic library:

```
-scaling
-mcmm
-upf
-optimization {power}
```

-logic_vs_physical

Checks the following options for the logic vs. physical libraries:

```
-cell_area
-cell_footprint
-bus_delimiter
```

-all

Checks the following options for the library:

```
-logic
-logic_vs_physical
-physical
```

-char_integrity

Checks the following options for accuracy:

```
-validate { timing noise }
-analyze { sensitivity voltage_range }
When lc_enable_10nm_mode is set to true, -analyze { interpolation }
and -analyze { table_trend } -criteria {trend!=/ trend!=\} for constraint table monotonicity
are also included.
```

-report_all

Check/reports all grid points. In the case of 7x7 tables, it will report 49 grid points even when the value difference is 0. This is equivalent to 0 tolerances.

-significant_digits *digits*

Specifies the number of significant digits displayed for floating-point numbers. Allowed values are from 0 through 13. The default value is 6.

-compensation

Waives bad CCSN issues in -validate {noise} mode that can be tolerated in PT with slight loss of accuracy.

-cell_area

Compares the cell area defined by the area attribute in the logic library with the area determined by the cell PR boundary in the physical library. In a FRAM view, the rectangular area covered by the place-and-route boundary for a standard cell, or the area enclosed by the cell boundary of a macro in a polygon, is compared with the cell area in the .db file. This comparison determines an area ratio for each cell. If the ratio is found to be the same for all the cells in the logic library, the cell areas are considered consistent. However, if the ratio for a cell deviates from the normal or average ratio for this library by a margin of 5 percent or more, the cell areas are considered inconsistent. There is no area check for the pad cells because they are special cells located at the chip boundary, and are not used as internal gates. They are considered to have zero area, and are therefore not checked for area.

-cell_footprint

Checks that the cell PR boundaries in the physical library is consistent among a set of cells when the **cell_footprint** attribute is specified in the logic library. This consistency check is done only when cell_footprint attribute is specified in the cell group. This option reports cell_footprint's and the cell names and their PR boundaries.

-bus_delimiter

Reports bus delimiters or bus naming styles in the logic and physical libraries. If the library does not define a bus delimiter, it reports it as Undefined.

`-routability`

Checks pin routability

`-min_pin_layer` ***`layer_name`***

Specifies the minimum layer name. Pins on layers lower than this layer are checked for existence. This option goes with **-routability**.

`-antenna`

Checks missing antenna property

`-signal_em`

Checks missing signal EM rules

`-same_name_cell`

Checks same name cells in the physical libraries

`-tech`

Checks technology data in the library

`-placement`

Checks cell placement constraint, layers, sites

`-pattern_must_join_pin`

Checks pattern must-join related items

`-pattern_must_join_pin_exclusion_list` ***`lib_pin_list`***

Specifies a collection of lib_pins that don't need the pattern must-join check. This option goes with **-pattern_must_join_pin**

`-physical`

Checks the following options for the physical library:

```
-routability
-antenna
-signal_em
-same_name_cell
-tech
-placement
-pattern_must_join_pin
-pattern_must_join_pin_exclusion_list
-min_pin_layer
```

`-reset`

Resets the options of the command to their default, missing cells, missing or mismatched pins are checked between logic libraries. All other options are ignored if used with the **-reset** option.

DESCRIPTION

The **set_check_library_options** command sets specific options for the **check_library** command for logic library checking.

Run the **set_check_library_options** command before you run the **check_library** command. If you do not specify any option with the **set_check_library_options** command, by default the command checks missing cells, missing and mismatched pins, including pg_pins versus power and ground pins.

A status indicating success or failure is returned.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

In the following example, the **-upf** options are set, the lib1.db logic library is checked for upf:

```
prompt> set_check_library_options -upf
1
prompt> check_library -logic_library_name lib1.db

#BEGIN_XCHECK_LIBRARY

Logic Library #1:
    Library name              lib1
    File name                 /home/goose/tmp/lib1.db
    Library type              pg_pin based db
    Library Version           1.000000
    Tool Created              J-2014.09
    Data Created              Thu Mar  9 14:21:29 2006
    Time unit                 1ns
    Capacitance unit          1000ff
    Leakage power unit        1mW
    Current unit              1mA
check_library options         -upf
Version                       J-2014.09-ALPHA5
Check date and time           Thu Jul 10 03:01:46 2014


#BEGIN_LIBSCREEN_UPF

Library#1 (lib1):
Warning: Cell 'isolation_cell' is missing 'backup_power' pg_pin, so it will become a black-box cell
Warning: Cell 'ls_1', pin 'A', The pin 'A' is missing the attribute 'input_voltage_range'.   (LBDB-
Warning: Cell 'ls_1', pin 'Y', The pin 'Y' is missing the attribute 'output_voltage_range'.   (LBDB
Warning: Cell 'ls_2', pin 'A', The pin 'A' is missing the attribute 'input_voltage_range'.   (LBDB-
Warning: Cell 'ls_2', pin 'Y', The pin 'Y' is missing the attribute 'output_voltage_range'.   (LBDB
Warning: Cell 'LVLLHX2M', pin 'A', The pin 'A' is missing the attribute 'input_voltage_range'.   (L
Warning: Cell 'LVLLHX2M', pin 'Y', The pin 'Y' is missing the attribute 'output_voltage_range'.   (
Warning: Cell 'LVLLHX8M', pin 'A', The pin 'A' is missing the attribute 'input_voltage_range'.   (L
Warning: Cell 'LVLLHX8M', pin 'Y', The pin 'Y' is missing the attribute 'output_voltage_range'.   (
Warning: Cell 'LVLLHEHX2M', pin 'A', The pin 'A' is missing the attribute 'input_voltage_range'.
Warning: Cell 'LVLLHEHX2M', pin 'Y', The pin 'Y' is missing the attribute 'output_voltage_range'.
Warning: Cell 'LVLLHEHX8M', pin 'A', The pin 'A' is missing the attribute 'input_voltage_range'.
Warning: Cell 'LVLLHEHX8M', pin 'Y', The pin 'Y' is missing the attribute 'output_voltage_range'.
Power management cell checking passed.

#END_LIBSCREEN_UPF


Information: List of cell classification (LIBCHK-312)
------------------------------------------------------------------------------
Library name                lib1(lib#1)
------------------------------------------------------------------------------
Total number                7
Inverter                    0
Buffer                      0
Level shifter               6
Differential level shifter  0
Isolation cell              1
Clock Isolation cell        0
Retention cell              0
Switch cell                 0
Always on cell              0
------------------------------------------------------------------------------
#BEGIN_XCHECK_LOGICCELLS


#END_XCHECK_LOGICCELLS

#BEGIN_XCHECK_LOGICPINS



#END_XCHECK_LOGICPINS

#BEGIN_XCHECK_LOGICPGPINS

Number of cells missing pg_pins in library 1:   0 (out of 7)

#END_XCHECK_LOGICPGPINS

#BEGIN_XCHECK_ARCS


#END_XCHECK_ARCS


Logic vs. logic library check summary:
Logic library consistency check PASSED for UPF.

#END_XCHECK_LIBRARY

1
```

In the following example, the **-logic_vs_physical** option is set, the logic library test.db vs. physical library test.ndm are checked for consistency:

```
prompt> set_check_library_options -logic_vs_physical
prompt> check_library -logic_library_name "test.db"  -physical_library_name test.ndm

#BEGIN_XCHECK_LIBRARY
```

```
check_library options -logic_vs_physical
Version              O-2018.06-DEV
Check date and time   Mon Feb  5 17:56:11 2018

#BEGIN_XCHECK_CELLS

Number of cells missing in logic library: 0 out of 747

Number of cells missing in physical library: 0 out of 747

#END_XCHECK_CELLS

#BEGIN_XCHECK_PINS

Warning: List of pins missing in logic library (LIBCHK-212)
--------------------------------------------------------------------------------
Physical library             Cell name           Pin name            direction
--------------------------------------------------------------------------------
tcbn90ghvt                   AN2HVTD0            VDD                 in
tcbn90ghvt                   AN2HVTD0            VSS                 in
--------------------------------------------------------------------------------

Number of cells missing pins in logic library: 1

Number of cells missing pins in physical library: 0

Error: List of pins mismatched in logic and physical libraries (LIBCHK-213)
--------------------------------------------------------------------------------
                                       Pin direction/type
Cell name          Pin name  tcbn90ghvtbc        tcbn90ghvt
--------------------------------------------------------------------------------
ANTENNAHVT         I         signal              diode
--------------------------------------------------------------------------------

Number of cells with mismatched pins: 1

#END_XCHECK_PINS

#BEGIN_XCHECK_FOOTPRINT

Warning: List of cells with cell_footprint attribute (LIBCHK-215)
--------------------------------------------------------------------------------------------
Logic library name      cell_footprint Cell name           PR boundary         Compatibility
--------------------------------------------------------------------------------------------
tcbn90ghvtbc            TIEHHVT        TIEHHVT             {0 0} {0.64 2.52}
tcbn90ghvtbc            TIELHVT        TIELHVT             {0 0} {0.64 2.52}
tcbn90ghvtbc            an2d1          AN2HVTD0            {0 0} {1.6 2.52}    Incompatible
tcbn90ghvtbc            an2d1          AN2HVTD1            {0 0} {1.6 2.52}    Incompatible
tcbn90ghvtbc            an2d1          AN2HVTD2            {0 0} {1.92 2.52}   Incompatible
--------------------------------------------------------------------------------------------

Number of incompatible cell_footprints in the libraries: 1 out of 3

#END_XCHECK_FOOTPRINT


#BEGIN_XCHECK_CELLAREA

Number of cells with inconsistent areas: 0
Average cell area ratio physical/logic in the libraries: 1

#END_XCHECK_CELLAREA
```

```
#BEGIN_XCHECK_BUS_DELIMITER

Information: List of bus naming styles (LIBCHK-214)
--------------------------------------------------------------------------------
Library name                                 Library type      Bus naming style
--------------------------------------------------------------------------------
tcbn90ghvtbc                                 Logic library     Undefined
tcbn90ghvt                                   Physical library  []
--------------------------------------------------------------------------------

Inconsistent bus delimiters found in the libraries.

#END_XCHECK_BUS_DELIMITER

Information: Logic library is INCONSISTENT with physical library (LIBCHK-220)

#END_XCHECK_LIBRARY
```

In the following example, the **-physical** option is set, the physical library test.ndm are checked

```
prompt> set_check_library_options -physical
prompt> check_library -physical_library_name { test.ndm }


#BEGIN_CHECK_PHYSICAL_LIBRARY

#BEGIN_CHECK_ROUTABILITY
--------------------------------------------------------------------------------
Target Library: test
--------------------------------------------------------------------------------

Report of missing pin access:
Total number of cells without optimal routability: 0 (out of 3)

Report of irregular cut shapes or routing blockages:
Total number of cells with irregular cut layer shapes or routing blockages: 0
(out of 3)

Summary of cell objects:
-----------------------------------------------------------------------------------------------
 Cell Name # Terminals # Real Shapes # Regular # Zero-spacing # Design-rule
 Routing Blockages Routing Blockages Routing Blockages
-----------------------------------------------------------------------------------------------
 AND2X1_LVT 5 0 22 0 0
 ZZY 3 0 0 0 0
 TXL 2 0 0 0 0
-----------------------------------------------------------------------------------------------

Report of terminal overlapped:
Total number of cells with terminal overlapped: 0 (out of 3)

Report of blocked terminals:
Total number of cells with blocked terminals: 0 (out of 3)

Report of improper routing blockages:
Total number of cells with improper routing blockages: 0 (out of 3)

Report of routing ports without geometry:
Total number of cells with routing ports without geometry: 0 (out of 3)

Report of ports with inconsistent routing attributes:
Total number of cells with ports with inconsistent routing attributes: 0 (out
of 3)
```

```
Report of improper internal port settings:
Total number of cells with improper internal port settings: 0 (out of 3)

Report of via regions on wire tracks:
Warning: List of pins with off-track or missing via regions. (FRAM-018)
--------------------------------------------------------------------------------
Target Library : test
Technology Library : test
Routing Direction : M1 H M2 unknown M3 unknown M4 unknown (skip upper metal
layers)
Pitch : M1 0.320 M2 0.320 M3 0.320 M4 0.320 (skip upper metal layers)
Offset : M1 0.000 M2 0.000 M3 0.000 M4 0.000 (skip upper metal layers)
--------------------------------------------------------------------------------
 Cell Name Pin Name Pin Layer Off-track
 (lower/upper metal layers)
--------------------------------------------------------------------------------
 AND2X1_LVT Q M1 Horizontal / No tracks info
 ZZY P1 M2 No tracks info &amp; No tracks info
 ZZY P2 M2 No tracks info &amp; No tracks info
 ZZY P3 M2 No tracks info &amp; No tracks info
 TXL T1 M3 No tracks info &amp; No tracks info
 TXL T2 M3 No tracks info &amp; No tracks info
--------------------------------------------------------------------------------
Warning: The routing direction of layer M2 is not defined for wire tracks
calculation. (FRAM-020)
Warning: The routing direction of layer M3 is not defined for wire tracks
calculation. (FRAM-020)
Warning: The routing direction of layer M4 is not defined for wire tracks
calculation. (FRAM-020)
Total number of cells with off-track or missing via regions: 3 (out of 3)


Detail list of wire track check for via regions:
-----------------------------------------------------------------------------------------------
 Cell Name Pin Terminal Terminal Contact Code Access Track Access Track BBox /
Points
 Name Name Layer (Lower Metal) (Upper Metal)
-----------------------------------------------------------------------------------------------
 AND2X1_LVT IN1 IN1 M1 VIA12C (15) Horizontal No tracks info (0.7199,
0.9299)(1.0201, 0.9951)
 AND2X1_LVT IN1 IN1 M1 VIA12C (15) rotated Horizontal No tracks info (0.7599,
0.8899)(0.9801, 1.0351)
-----------------------------------------------------------------------------------------------

Report of pins on color wire tracks:

--------------------------------------------------------------------------------
Target Library : test
Technology Library : test
Site : unit
Routing Direction : M1 H M2 unknown M3 unknown (skip upper metal layers)
Pitch : M1 0.320 M2 0.320 M3 0.320 (skip upper metal layers)
Offset : M1 0.000 M2 0.000 M3 0.000 (skip upper metal layers)
Track color : M1 N/A M2 N/A M3 N/A (skip upper metal layers)
-----------------------------------------------------------------------------------------------
 Cell Object Object Object Object Track Color Center line aligned Fat
 Name Type Name Layer Color Color Matched with Track Shape
-----------------------------------------------------------------------------------------------
 AND2X1_LVT Terminal IN1 M1 no_mask N/A N/A No Yes
 AND2X1_LVT Regular blockage RB_3 M1 no_mask N/A N/A No Yes
-----------------------------------------------------------------------------------------------
Warning: The routing direction of layer M2 is not defined for wire tracks
calculation. (FRAM-020)
```

```
Warning: The routing direction of layer M3 is not defined for wire tracks
calculation. (FRAM-020)
Warning: The track color information of layer M1 is not defined for color wire
tracks calculation. (FRAM-056)
Warning: The track color information of layer M2 is not defined for color wire
tracks calculation. (FRAM-056)
Warning: The track color information of layer M3 is not defined for color wire
tracks calculation. (FRAM-056)
Color track info of all 3 cells with 11 terminals are reported.

#END_CHECK_ROUTABILITY
Report placement checking for library: test

#BEGIN_CHECK_PLACEMENT
-------------------------------------------------------------------------------
Target Library: test
-------------------------------------------------------------------------------

Report of improper placement constraints:
Total number of cells without placement layers and constraints: 0 (out of 3)

Report of improper site definition:
Total number of cells with improper site definition: 0 (out of 3)

#END_CHECK_PLACEMENT
Report drc checking for library: test


Report antenna for library: test

#BEGIN_CHECK_ANTENNA

Warning: The library is missing antenna rules. (LIBCHK-116)
Total number of cells missing gate area and diode protection antenna
properties: 2 (out of 3)
------------------------------------------------------------------------------------------
Cell name Cell type Missing property Pin name(s)
------------------------------------------------------------------------------------------
TXL lib_cell gate_area T1 (Total 1)
TXL lib_cell diff_area T2 (Total 1)
ZZY lib_cell gate_area P1 P2 (Total 2)
ZZY lib_cell diff_area P3 (Total 1)

#END_CHECK_ANTENNA


#BEGIN_CHECK_SIGNALEM

Warning: The library is missing signal EM data. (LIBCHK-114)

#END_CHECK_SIGNALEM


#BEGIN_CHECK_SAMENAMECELL

#END_CHECK_SAMENAMECELL


#BEGIN_CHECK_TECH

-------------------------------------------------------------------------------
Warning: Cut layer 'CO' has a non-cross primary default ContactCode 'POLYCON'.
(line 1659) (TECH-083w)
```

```
---------------------------------------------------------------------------------
#END_CHECK_TECH

#END_CHECK_PHYSICAL_LIBRARY

Physical library checks done.
```

In the following example, the **-same_name_cell** option is set, the physical libraries phys1.ndm phys2.ndm are checked.

```
prompt> set_check_library_options -same_name_cell
prompt> check_library -physical_library_name { phys1.ndm phys2.ndm }

#BEGIN_CHECK_PHYSICAL_LIBRARY

#BEGIN_CHECK_SAMENAMECELL

Information: List of cells with same names (LIBCHK-112)
---------------------------------------------------------------------------------
Cell name Library list
---------------------------------------------------------------------------------
AND2X1_LVT phys1 phys2
---------------------------------------------------------------------------------

#END_CHECK_SAMENAMECELL

#END_CHECK_PHYSICAL_LIBRARY
```

SEE ALSO

```
check_library(2)
report_check_library_options(2)
```

set_command_option_value

Set a command option default or current value.

SYNTAX

```
string set_command_option_value
   -default | -current

   -command command_name

   -option option_name | -position positional_option_position

   -value value | -undefined
```

ARGUMENTS

-default

Set the default option value.

-current

Set the current option value.

-command *command_name*

Set the option value for an option of this command.

-option *option_name*

Set the option value for this option of the command.

-position *positional_option_position*

Set the option value for this positional option of the command.

-value *value*

Set the option value to this value.

-undefined

Set the option value to the "undefined value".

DESCRIPTION

Sets the current or default value of a command option.

Either -default must be specified (to specify setting the default value of the option), or -current must be specified (to specify setting the current value of the option). Unlike for get_command_option_values, one of -current or -default must be specified for set_command_option_value.

An option of a command must be specified (for value setting) by passing a command name via -command and either an option name (for an option of the command) via -option, or the position of a positional option via -position. The *option_name* passed via -option (for a dash option) must not have its leading dash character omitted. An option name passed via -option may be either the name of a dash option or the name of a positional option. A positional option may be specified either via -option or via -position. Positional option positions of a command are numbered 0, 1, 2, ... (N-1), where N is the number of positional options of the command.

To set the (current or default) value of the option, a string value should be passed via -value. Alternatively, you can pass -undefined to reset the option to have the "undefined value". (CAVEAT: For some "set value implementations" for numeric options, -undefined does not really "undefine" the value - instead the value is set to 0.)

Processing of the -value value does not include any CCI option checking of the value (such as checking whether the option value has correct syntax for its type).

The command "throws" a Tcl error for various conditions:
* the command does not exist
* the command exists but no such option of the command exists
* the set operation failed (could be due to a failed conversion for one of the "set value" implementations which does a conversion from string to one of integer, double, etc.)

The -undefined option is a mutually exclusive alternative to the -value option. -undefined has the effect of resetting the option value to the "undefined value".

A possible power user application: power users could put set_command_option_value commands in their home directory Tcl startup files for their favorite dialogs/commands to "seed" initial current values for these dialogs. This may lower the need for automatically saving replay scripts (of set_command_option_value commands for each command/dialog option/field) on exit from the application.

EXAMPLES

The following example sets the current value for the -bar option of the foo command to a value of abc.

```
prompt> set_command_option_value -current -command foo \
   -option "-bar" -value abc
```

The following example sets the current value for the first positional argument of the foo command to a value of xyz.

```
prompt> set_command_option_value -current -command foo \
   -position 1 -value xyz
```

SEE ALSO

```
get_command_option_values(2)
preview(2)
```

set_current_command_mode

SYNTAX

```
string set_current_command_mode


   -mode command_mode | -command command

string command_mode
string command
```

ARGUMENTS

-mode *command_mode*

Specifies the name of the new command mode to be made current. **-mode** and **-command** are mutually exclusive; you must specify one, but not both.

-command *command*

Specifies the name of the command whose associated command mode is to be made current. **-mode** and **-command** are mutually exclusive; you must specify one, but not both.

DESCRIPTION

The **set_current_command_mode** sets the current command mode. If there is a current mode in effect, the current mode is first canceled, causing the associated clean up to be executed. The initialization for the new command mode is then executed as it is made current.

If the name of the given command mode is empty, the current command mode is cancelled and no new command mode is made current.

A current command mode stays in effect until it is displaced by a new command mode or until it is cleared by an empty command mode.

If the command completes successfully, the new current command mode name is returned. On failure, the command returns the previous command mode name which will remain current and prints an error message unless error messages are suppressed.

EXAMPLES

The following example sets the current command mode to a mode called "mode_1"

```
shell> set_current_command_mode -mode mode_1
```

The following example clears the current command mode without setting a new mode.

```
shell> set_current_command_mode -mode ""
```

The following example sets the current command mode to the mode associated with the command "modal_command"

```
shell> set_current_command_mode -command modal_command
```

set_host_options

.prod lc Controls the maximum number of CPU cores that can be used for read_lib command.

SYNTAX

```
status set_host_options
   -max_cores number_of_cores
```

Data Types

```
   number_of_cores       integer
```

ARGUMENTS

-max_cores *number_of_cores*

Specifies the maximum number of CPU cores that are allowed for parallel execution.

DESCRIPTION

The **set_host_options** command is used to specify the maximum number of CPU cores that the read_lib command can use for parallel execution. The default is 1, which indicates that parallel execution is not enabled. The maximum number of cores that can be specified is 16.

To disable parallel execution, set the **-max_cores** option to 1.

EXAMPLES

The following example tells the tool that it can use up to eight cores for parallel execution:

```
lc_shell> set_host_options -max_cores 8
```

SEE ALSO

```
remove_host_options(2)
report_host_options(2)
```

set_lib_attribute

Sets an existing attribute to a specified value on the specified list of objects.

SYNTAX

```
collection set_lib_attribute
   objects
   attribute_name
   attribute_value
```

Data Types

```
objects          collection
attribute_name       string
attribute_value      string
```

ARGUMENTS

objects

Specifies the objects on which the attribute is to be set.

attribute_name

Specifies the name of the attribute to be set.

attribute_value

Specifies the value of the attribute. The data type must be the same as that of the attribute.

DESCRIPTION

This command sets the value of an attribute on an object. For a complete list of attributes, This command returns a collection of objects that have the specified attribute value set. If the attribute is not set on any objects, the command returns an empty string.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information if there is no scenario dependency for the attributes. However, for attributes such as **max_leakage_power**, which are scenario dependent, this command uses information from the current scenario only.

EXAMPLES

The following example sets attribute this area to 7.291200 on lib/testCell. hierarchy:

```
prompt> set_lib_attribute lib/testCell area 7.291200
```

SEE ALSO

```
collections(2)
get_attribute(2)
```

set_lib_group

Specifies a library (sub)group to qualify by check_library or merge by merge_lib. The (sub)group has base libraries and their associated model files or a list of ETM libraries of the same PVT, depending on the options.

SYNTAX

```
status set_lib_group
   group_name
   [-corner corner_name]
   [-base_lib base_library_list]
   [-model_lib model_library_list]
   [-single_mode_lib single_mode_library_list]
   [-reset]
```

Data Types

```
base_library_list        list
model_library_list       list
single_mode_library_list list
```

ARGUMENTS

group_name

Specifies a group name. This is used especially when multiple groups are defined. All the libraries in the same group name should be of the same cell set.

-corner

Specifies a corner name for the single mode etm libs. Those are typically of the same PVT(corner). This is optional.

-base_lib

Specifies one or multiple base library .db file names. It should be specified if -model_lib is specified.

-model_lib

Specifies one or multiple model .db file names in the type of lvf ccs_power or power_aware.

-single_mode_lib

Specifies a list of single mode etm db file name and associated mode pairs in {} or double quote. The mode is optional for single_mode checking, in which case the etms in this option are taken as different modes at the same corner.

-reset

Remove/resets this group group_name. Specify group name. The other options should not be specified and will be ignored if specified so.

DESCRIPTION

The **set_lib_group** command specifies

1) base library and model file association as a subgroup and different corner libraries as a group if -base_lib and -model_lib are specified.

2) a list of ETM library db files of the same PVT and associated modes if -single_mode_lib is specified. This (sub)group will be used in successing check_library or merge_lib commands. It does not load the libraries at this point.

EXAMPLES

In the following example, a library group consisting of 3 base libraries and 1 model file is specified in the group name of grp1:

```
prompt> set_lib_group grp1 -base_lib "1.db 2.db 3.db" -model_lib "upf.db"
```

In the following example, a library group consisting of 2 etm libraries with different modes is specified in the group name of grp2 at corner c1:

```
prompt> set_lib_group grp2 -corner c1 -single_mode_lib "{etm1.db m1} {etm2.db m2}"
```

SEE ALSO

```
check_library(2)
merge_lib(2)
write_lib_group(2)
```

set_lib_mismatch_fix_options

Specifies options for check_library command to generate a tcl script to be used for library mismatch fixes.

SYNTAX

```
status set_lib_mismatch_fix_options
   [-output lib_mismatch_fix_tcl_filename]
   [-resolve_by_db db_attribute_list]
   [-resolve_by_frame frame_attribute_list]
   [-remove_missing {cell pin pg_pin}]
```

Data Types

```
lib_mismatch_fix_tcl_filename string
db_attribute_list   list
frame_attribute_list    list
```

ARGUMENTS

-output

Specifies an output tcl filename. If not specified, lib_mismatch_fix.tcl will be used by default.

-resolve_by_db

Specifies to resolve conflicts to the attributes with db as golden. The attributes are bus_naming_style, antenna_diode_type, cell_type, pin_type, pg_type and direction.

-resolve_by_frame

Specifies to resolve conflicts to the attributes with frame as golden. The attributes are bus_naming_style, pg_type and pg_pin/direction for pg_pin fixes from frame.

-remove_missing {cell pin pg_pin}

Specifies to remove the cells missing in the other library by remove_lib_cell. e.g. -remove_missing {cell} removes the cells existing in either logic or physical libraries but not both when sourcing the mismatch fix script. -remove_missing {pin pg_pin} will remove the cells that are missing pin/pg_pins as there is no remove_lib_pin command available in the tools that will source the script.

All the options are optional.

DESCRIPTION

The **set_lib_mismatch_fix_options** command specifies options for check_library command to generate a tcl script to be used for library mismatch fixes. The tools that source the script should be those that accept the commands (e.g. set_attribute) and app options. This command should be issued before check_library.

EXAMPLES

In the following example, mismatches in bus_naming_style, antenna_diode_type(is_diode) and signal pin direction will be resolved by using defined DB attribute values to update FRAME attributes while the mismatches in PG types and directions will be resolved by using defined FRAME attribute values to update corresponding DB attributes. These actions are recorded in the tcl script lib_auto_fix.tcl.

```
prompt> set_lib_mismatch_fix_options -output lib_auto_fix.tcl
  -resolve_by_db {bus_naming_style antenna_diode_type direction}
  -resolve_by_frame {pg_type pg_pin/direction}
```

SEE ALSO

```
check_library(2)
```

set_message_info

Set some information about diagnostic messages.

SYNTAX

```
string set_message_info -id message_id [-limit max_limit|-stop_on|-stop_ff]

string message_id
integer max_limit
```

ARGUMENTS

-id *message_id*

Information is to be set for the given *message_id*. The message must exist. Although different constraints allow different message types, no constraint allows severe or fatal messages.

-limit *max_limit*

Set the maximum number of occurrences for *message_id*. This is an integer greater than or equal to zero. If you set it to zero, that means the number of occurrences of the message is unlimited. Messages which occur after a limit is reached are automatically suppressed.

-stop_on

Force Tcl error if message is emitted.

-stop_off

Turn off a previous -stop_on directive

DESCRIPTION

The **set_message_info** command sets constraints on diagnostic messages (typically error, warning, and informational messages).

Currently, you can set a upper limit for the number of occurrences of a message. You can set this to zero to indicate that there is no limit. You can retrieve the current limit for a message using the **get_message_info** ***command. When the limit is exceeded, all future occurrences of the message are automatically suppressed. A count of total occurrences (including those suppressed) can be retrieved using get_message_info.***

EXAMPLES

The following example uses **set_message_info** to set a limit on the number of APP-027 messages to 100. When the 101st APP-027 message is about to be issued, you will be warned that the limit has been exceeded, and that all future occurrences will be suppressed.

```
prompt> set_message_info -id APP-027 -limit 100
prompt> do_command
Warning: can't find node U27.1 (APP-027)
Warning: can't find node U27.2 (APP-027)
Warning: can't find node U27.3 (APP-027)
 ...
Warning: can't find node U27.100 (APP-027)
Note - message 'APP-027' limit (100) exceeded.  Remainder will be suppressed.
1
```

SEE ALSO

```
get_message_info(2)
print_message_info(2)
suppress_message(2)
```

set_process_label

Sets the process label for one or more libraries.

SYNTAX

```
string set_process_label
   label
   -libraries libraries
```

Data Types

```
   label        string
   libraries    list
```

ARGUMENTS

label

The process label to be used for specified libraries.

-libraries *libraries*

A list of libraries in memory to which to apply new process label.

DESCRIPTION

The **set_process_label** command is used to modify **process_label** attribute under default operating_conditions group. This command can only take effect after read_lib/read_db. Following is the corresponding liberty syntax:

```
     library (lib_name) {
    operating_conditions ( opc_name ) {
     …
     process_label : "name";
    }
```

```
      }
```

The updated "process_label" attribute can be queried by **get_lib_attribute** and **report_lib**. **write_lib** can be used to save the updated process_label in disk.

Multicorner-Multimode Support

This command should be used after read_lib or read_db

EXAMPLES

This example shows how to change/query the process label for a library.

```
prompt> set_process_label -libraries { wc_v0p99_t0 wc_v0p81_t0 } test_label
prompt> get_lib_attribute [get_lib_objects -class_type operating_conditions -of_objects [get_libs]]
prompt> report_lib -operating_condition wc_v0p99_t0
```

SEE ALSO

```
read_db(2)
read_lib(2)
report_lib(2)
check_library(2)
get_lib_attribute(2)
```

set_unix_variable

This is a synonym for the **setenv** command.

SEE ALSO

gettenv(2)
printenv(2)
printvar(2)
set(2)
setenv(2)
sh(2)
unset(2)

setenv

Sets the value of a system environment variable.

SYNTAX

```
string setenv variable_name new_value

string variable_name


string new_value
```

ARGUMENTS

variable_name

Names of the system environment variable to set.

new_value

Specifies the new value for the system environment variable.

DESCRIPTION

The **setenv** command sets the specified system environment *variable_name* to the *new_value* within the application. If the variable is not defined in the environment, the environment variable is created. The **setenv** command returns the new value of *variable_name*. To develop scripts that interact with the invoking shell, use **getenv** and **setenv**.

Environment variables are stored in the Tcl array variable **env**. The environment commands **getenv**, **setenv**, and **printenv** are convenience functions to interact with this array.

The **setenv** command sets the value of a variable only within the process of your current application. Child processes initiated from the application using the **exec** command after a usage of **setenv** inherit the new variable value. However, these new values are not exported to the parent process. Further, if you set an environment variable using the appropriate system command in a shell you invoke using the **exec** command, that value is not reflected in the current application.

EXAMPLES

The following example changes the default printer.

```
shell> getenv PRINTER
laser1
shell> setenv PRINTER "laser3"
laser3
shell> getenv PRINTER
laser3
```

SEE ALSO

exec(2)
getenv(2)
unsettenv(2)
printenv(2)
printvar(2)
set(2)
sh(2)
unset(2)

sh

Executes a command in a child process.

SYNTAX

```
string sh [args]

string args
```

ARGUMENTS

args

Command and arguments that you want to execute in the child process.

DESCRIPTION

This is very similar to the **exec** command. However, file name expansion is performed on the arguments. Remember that quoting and grouping is in terms of Tcl. Arguments which contain spaces will need to be grouped with double quotes or curly braces. Tcl special characters which are being passed to system commands will need to be quoted and/or escaped for Tcl. See the examples below.

EXAMPLES

This example shows how you can remove files with a wildcard.

```
prompt> ls aaa*
aaa1      aaa2      aaa3
prompt> sh rm aaa*
prompt> ls aaa*
```

```
Error: aaa*: No such file or directory
       Use error_info for more info. (CMD-013)
```

This example shows how to grep some files for a regular expression which contains spaces and Tcl special characters:

```
prompt> exec cat test3.out
blah blah blah
blah blah blah c blah
input [1:0] A;
output [2:0] B;
prompt>
prompt> sh egrep -v {[ ]+c[ ]+} test3.out
blah blah blah
prompt>
prompt> sh egrep {t[ ]+\[} test3.out
input [1:0] A;
output [2:0] B;
```

SEE ALSO

exec(2)

sizeof_collection

Returns the number of objects in a collection.

SYNTAX

```
int sizeof_collection
   collection1
```

Data Types

collection1	collection

ARGUMENTS

collection1

Specifies the collection for which to get the number of objects. If the empty collection (empty string) is used for the *collection1* argument, the command returns 0.

DESCRIPTION

The **sizeof_collection** command is an efficient mechanism for determining the number of objects in a collection.

EXAMPLES

The following example from PrimeTime shows a simple way to find out how many objects matched a particular pattern and filter in the **get_cells** command.

```
pt_shell> echo "Number of hierarchical cells: \
?                 [sizeof_collection \
?                   [filter_collection [get_cells * -hier \
?                     "is_hierarchical == true"]]]"
Number of hierarchical cells is: 10
```

The following example from PrimeTime shows what happens when the argument for the **sizeof_collection** command results in an empty collection.

```
pt_shell> set s1 [get_cells *]
{"u1", "u2", "u3"}
pt_shell> set ssize [filter_collection $s1 "area < 0"]
pt_shell> echo "Cells with area < 0: [sizeof_collection $ssize]"
Cells with area < 0: 0
pt_shell> unset s1
```

SEE ALSO

```
collections(2)
filter_collection(2)
```

sort_collection

Sorts a collection based on one or more attributes, resulting in a new, sorted collection. The sort is ascending by default.

SYNTAX

```
collection sort_collection
   [-descending]
   [-dictionary]
   collection
   criteria
```

Data Types

collection	collection
criteria	list

ARGUMENTS

-descending

Indicates that the collection is to be sorted in reverse order. By default, the sort proceeds in ascending order.

-dictionary

Sort strings dictionary order. For example "a30" would come after "a4".

collection

Specifies the collection to be sorted.

criteria

Specifies a list of one or more application or user-defined attributes to use as sort keys.

DESCRIPTION

You can use the **sort_collection** command to order the objects in a collection based on one or more attributes. For example, to get a collection of leaf cells increasing alphabetically, followed by hierarchical cells increasing alphabetically, sort the collection of cells using the *is_hierarchical* and *full_name* attributes as *criteria*.

The *criteria* can specify an attribute on another object. The other object must be available as an attribute on this object. For example, if you had a collection of net shapes, you can sort the objects by net using *owner_net.name* for instance.

In an ascending sort, Boolean attributes are sorted with those objects first that have the attribute set to *false*, followed by the objects that have the attribute set to *true*. In the case of a sparse attribute, objects that have the attribute come first, followed by the objects that do not have the attribute.

Sorts are ascending by default. The *-descending* option reverses the order of the objects.

EXAMPLES

The following example from PrimeTime sorts a collection of cells based on hierarchy, and adds a second key to list them alphabetically. In this example, cells i1, i2 and i10 are hierarchical, and o1 and o2 are leaf cells. Because the *is_hierarchical* attribute is a Boolean attribute, those objects with the attribute set to *false* are listed first in the sorted collection.

```
pt_shell> set zc [get_cells {o2 i2 o1 i1 i10}]
{"o1", "i2", "o1", "i1", "i10"}
pt_shell> set zsort [sort_collection $zc {is_hierarchical full_name}]
{"o1", "o2", "i1", "i10", "i2"}
pt_shell> set zsort [sort_collection -dictionary $zc {is_hierarchical full_name}]
{"o1", "o2", "i1", "i2", "i10"}
```

SEE ALSO

```
collections(2)
```

sort_lib_objects

sort objects in Tcl collections based on one or more attributes.

SYNTAX

```
collection sort_lib_objects
   [-by attribute_list]
   collection1
```

Data Types

```
collection1              collection
attribute_list          list
```

ARGUMENTS

collection1

specifies Tcl collection to be sort.

-by *attribute_list*

Specifies a list of one or more application or user-defined attributes to use as sort keys. Most of time, the option should be specified. the option can be omitted when there is name for the object in the collection and the name is used as key to sort. If the option is not specified and there is no name for the object in the collection, the command does nothing and returns empty collection.

DESCRIPTION

You can use the **sort_lib_objects** command to sort objects in collection based on one or more attributes.

EXAMPLES

The following exmaple sort a collection of pins and use pin name as key.

```
lc_shell>set cell_list [get_lib_cells test/*]
{"test/BUF", "test/CKGT", "test/FSB2"}
lc_shell>set sort_cell [sort_lib_objects $cell_list]
{"test/BUF", "test/CKGT", "test/FSB2"}
lc_shell>sizeof_collection $sort_cell
3
lc_shell>foreach_in_collection element $cell_sort {
  echo "cell: " [get_object_name $element]
}
cell:  BUF
cell:  CKGT
cell:  FSB2
```

SEE ALSO

collections(2)

source

Read a file and evaluate it as a Tcl script.

SYNTAX

```
string source [-echo] [-verbose] [-continue_on_error] file

string file
```

ARGUMENTS

-echo

Echoes each command as it is executed. Note that this option is a non-standard extension to Tcl.

-verbose

Displays the result of each command executed. Note that error messages are displayed regardless. Also note that this option is a non-standard extension to Tcl.

-continue_on_error

Don't stop script on errors. Similar to setting the shell variable sh_continue_on_error to true, but only applies to this particular script.

file

Script file to read.

DESCRIPTION

The **source** command takes the contents of the specified *file* and passes it to the command interpreter as a text script. The result of the source command is the result of the last command executed from the file. If an error occurs in evaluating the contents of the script, then the **source** command returns that error. If a return command is invoked from within the file, the remainder of the file is skipped and the source command returns normally with the result from the return command.

By default, source works quietly, like UNIX. It is possible to get various other intermediate information from the source command using the **-echo** and **-verbose** options. The **-echo** option echoes each command as it appears in the script. The **-verbose** option echoes the result of each command after execution.

NOTE: To emulate the behavior of the dc_shell **include** command, use both of these options.

The file name can be a fully expanded file name and can begin with a tilde. Under normal circumstances, the file is searched for based only on what you typed. However, if the system variable sh_source_uses_search_path is set to "true", the file is searched for based on the path established with the search_path variable.

The **source** command supports several file formats. The *file* can be a simple ascii script file, an ascii script file compressed with gzip, or a Tcl bytecode script, created by TclPro Compiler. The **-echo** and **-verbose** options are ignored for gzip formatted files.

EXAMPLES

This example reads in a script of aliases:

```
prompt> source -echo aliases.tcl
alias q quit
alias hv {help -verbose}
alias include {source -echo -verbose}
prompt>
```

SEE ALSO

search_path(3)
sh_source_uses_search_path(3)
sh_continue_on_error(3)

suppress_message

Disables printing of one or more informational or warning messages.

SYNTAX

```
string suppress_message [message_list]

list message_list
```

ARGUMENTS

message_list

A list of messages to suppress.

DESCRIPTION

The **suppress_message** command provides a mechanism to disable the printing of messages. You can suppress only informational and warning messages. The result of **suppress_message** is always the empty string.

A given message can be suppressed more than once. So, a message must be unsuppressed (using **unsuppress_message**) as many times as it was suppressed in order for it to be enabled. The **print_suppressed_messages** command displays the currently suppressed messages.

EXAMPLES

When the argument to the **unalias** command does not match any existing aliases, the CMD-029 warning message displays. This example shows how to suppress the CMD-029 message:

```
prompt> unalias q*
Warning: no aliases matched 'q*' (CMD-029)
prompt> suppress_message CMD-029
prompt> unalias q*
prompt>
```

SEE ALSO

unsuppress_message(2)
set_message_info(2)

unalias

Removes one or more aliases.

SYNTAX

```
string **unalias**
   *patterns*
```

ARGUMENTS

patterns

Specifies the patterns to be matched. This argument can contain more than one pattern. Each pattern can be the name of a specific alias to be removed or a pattern containing the wildcard characters ***** and **%**, which match one or more aliases to be removed.

DESCRIPTION

The **unalias** command removes aliases created by the **alias** command.

EXAMPLES

The following command removes all aliases.

```
prompt> unalias *
```

The following command removes all aliases beginning with f, and the alias rt100.

```
prompt> unalias f* rt100
```

SEE ALSO

```
alias(2)
```

unsetenv

Removes a system environment variable.

SYNTAX

```
string getenv
   variable_name
```

Data Types

```
variable_name      string
```

ARGUMENTS

variable_name

Specifies the name of the environment variable to be unset.

DESCRIPTION

The **unsetenv** command searches the system environment for the specified *variable_name* and removes variable from the environment. If the variable is not defined in the environment, the command returns a Tcl error. The command is catchable.

Environment variables are stored in the **env** Tcl array variable. The **unsetenv**, commands is a convenience function to interact with this array. It is equivalent to 'unset ::env(*variable_name*)'

The application you are running inherited the initial values for environment variables from its parent process (that is, the shell from which you invoked the application). If you unset the variable using the **unsetenv** command, you remove the variable value in the application and in any new child processes you initiate from the application using the **exec** command. However, the variable is still set in the parent process.

See the **set** and **unset** commands for information about working with non-environment variables.

EXAMPLES

In the following example, **unsetenv** remove the DISPLAY varible from the environment:

```
prompt> getenv DISPLAY
host:0
prompt> unsetenv DISPLAY
prompt> getenv DISPLAY
Error: can't read "::env(DISPLAY)": no such variable
        Use error_info for more info. (CMD-013)
```

SEE ALSO

```
catch(2)
exec(2)
printenv(2)
set(2)
unset(2)
setenv(2)
getenv(2)
```

unsuppress_message

Enables printing of one or more suppressed informational or suppressed warning messages.

SYNTAX

```
string unsuppress_message [messages]

list messages
```

ARGUMENTS

messages

A list of messages to enable.

DESCRIPTION

The **unsuppress_message** command provides a mechanism to re-enable the printing of messages which have been suppressed using **suppress_message**. You can suppress only informational and warning messages, so the **unsuppress_message** command is only useful for informational and warning messages. The result of **unsuppress_message** is always the empty string.

You can suppress a given message more than once. So, you must unsuppress a message as many times as it was suppressed in order to enable it. The **print_suppressed_messages** command displays currently suppressed messages.

EXAMPLES

When the argument to the **unalias** command does not match any existing aliases, the CMD-029 warning message displays. This example shows how to re-enable the suppressed CMD-029 message. Assume that there are no aliases beginning with 'q'.

```
prompt> unalias q*
prompt> unsuppress_message CMD-029
prompt> unalias q*
Warning: no aliases matched 'q*' (CMD-029)
```

SEE ALSO

print_message_info(2)
suppress_message(2)

which

Locates a file and displays its pathname.

SYNTAX

```
string which filename_list

list filename_list
```

ARGUMENTS

filename_list

List of files to locate.

DESCRIPTION

Displays the location of the specified files. This command uses the search_path to find the location of the files. This command can be a useful prelude to read_db or link_design, because it shows how these commands expand filenames. The **which** command can be used to verify that a file exists in the system.

If an absolute pathname is given, the command searches for the file in the given path and returns the full pathname of the file.

EXAMPLES

The following examples are based on the following search_path.

prompt> **set search_path "/u/foo /u/foo/test"**

The following command searches for the file name foo1 in the search_path.

```
prompt> which foo1
/u/foo/foo1
```

The following command searches for files foo2, foo3.

```
prompt> which {foo2 foo3}
/u/foo/test/foo2 /u/foo/test/foo3
```

The following command returns the full pathname.

```
prompt> which ~/test/designs/sub_design.db
/u/foo/test/designs/sub_design.db
```

SEE ALSO

link_design(2)
read_db(2)
search_path(3)

write_app_var

Writes a script to set the current variable values.

SYNTAX

```
string write_app_var
   -output file
   [-all | -only_changed_vars]
   [pattern]
```

Data Types

```
   file         string
   pattern      string
```

ARGUMENTS

-output *file*

Specifies the file to which to write the script.

-all

Writes the default values in addition to the current values of the variables.

-only_changed_vars

Writes only the changed variables. This is the default when no options are specified.

pattern

Writes the variables that match the specified *pattern*. The default is "*".

DESCRIPTION

The **write_app_var** command generates a Tcl script to set all application variables to their current values. By default, variables set to their default values are not included in the script. You can force the default values to be included by specifying the **-all** option.

EXAMPLES

The following is an example of the **write_app_var** command:

```
prompt> write_app_var -output sh_settings.tcl sh*
```

SEE ALSO

```
get_app_var(2)
report_app_var(2)
set_app_var(2)
```

write_lib

Writes a compiled library to disk in the Synopsys .db file,

SYNTAX

```
int write_lib
   library_name
   [-output file_name]
```

Data Types

```
library_name          string
file_name             string
```

ARGUMENTS

library_name

In the Synopsys .db file format, the *library_name* argument specifies the name of the technology library to be written.

-output *file_name*

Specifies an output file name or path name for the library.

-format *db*

Specifies the external format of the specified library. Only supports the .db file format.

DESCRIPTION

The **write_lib** command saves a compiled technology library to the disk.

If you specify the **-output** option, the files are written in the specified format with the appropriate file name extensions. If you use the **-output** option to specify a file name in the Synopsys .db file format, the file is saved with the specified name. If you do not use the **-output** option, the library is saved in the current directory with the name, *library_name*.db.

Note: This command overwrites files without issuing a warning.

Multicorner-Multimode Support

This command has no dependency on scenario-specific information.

EXAMPLES

The following example shows how to read and write a technology library.

```
prompt> read_lib my_lib.lib
```

prompt> **write_lib my_lib**

The following example shows how to write a technology library to a specified file.

```
prompt> write_lib my_lib -output ~synopsys/test/test.db
```

SEE ALSO

```
read_lib(2)
```

write_lib_group

Writes out a tcl script containing the library groups defined by set_lib_group command and link_library.

SYNTAX

```
status write_lib_group
   [-output filename]
```

Data Types

```
filename      string
```

ARGUMENTS

-output *filename*

Specifies a tcl file name for the output. If not specified, it will output to screen.

DESCRIPTION

The **write_lib_group** command writes out a tcl script containing the library groups defined by set_lib_group command with base library and model file associatioon. If there are multiple set_lib_group command lines, they will be sorted by lib_group name and each base library and all of its associated model files will have one command line. This command can also be used to dump out intermediate library groups for debug use. The output script will have comments if there are issues found in library group qualification with error code.

EXAMPLES

In the following example, the filename libgrps_out.tcl is written out after the library group is qualified by check_library.

```
prompt> write_lib_group -output libgrps_out.tcl

libgrps_out.tcl file has the following contents:
set link_library "liberty_FF_1p19_m40.db  liberty_FF_1p23_m40.db "
set_lib_group grp1 -base_lib " liberty_FF_1p19_m40.db liberty_FF_1p23_m40.db " -model_lib upf.db
```

SEE ALSO

```
check_library(2)
set_lib_group(2)
```

write_scaling_lib_group

Writes out a tcl script containing the scaling library groups and link_library after running check_library.

SYNTAX

```
status write_scaling_lib_group
   [-output filename]
   [-auto_adjust]
   [-no_link_library]
```

Data Types

```
   filename      string
```

ARGUMENTS

-output *filename*

Specifies a file name for the output.

-auto_adjust

Outputs a subset group that is the best possible with valid formation by removing off-grid corners. In the case of multiple subsets of groups, it outputs one of the best groups with maximum number of libraries and scaling dimensions which is the first in the group name in LIBCHK-304 table.

-no_link_library

Does not write out link_library in the output script as a valid variable setting but as comments.

DESCRIPTION

The **write_scaling_lib_group** command writes out a tcl script containing the scaling library groups that meet the library formation after qualified by check_library. This command should be issued after check_library command. This command can also be used to dump out intermediate scaling library groups for debug use when the libraries are not completely qualified. This way users can fix library issues reported by check_library and re-qualify. The output script will have comments if there are issues found in library data and/or formation with error code.

EXAMPLES

In the following example, the slg_out.tcl is written out after the library group is qualified by check_library.

```
prompt> write_scaling_lib_group -output slg_out.tcl

slg_out.tcl file has the following contents:
set link_library "liberty_FF_1p19_m40.db  liberty_FF_1p23_m40.db "
create_scaling_lib_group -name grp1 " liberty_FF_1p19_m40.db liberty_FF_1p23_m40.db "
```

SEE ALSO

```
check_library(2)
create_scaling_lib_group(2)
```