PrimeRail
Variables and Attributes

Version M-2017.06-SP2, September 2017

SYNOPSYS®

Contents

enable_advanced_power_debug

Enables or disables either PrimeTime PX or pr_shell commands.

TYPE

Boolean

DEFAULT

false

GROUP

power_variables

DESCRIPTION

By default this variable is set to *false*, which disables all PrimeTime PX commands and enable all pr_shell commands. Setting this variable to *true* enables all PrimeTime PX commands and disables all pr_shell commands.

You can set this variable only once in a session. The variable becomes read-only when it is used or when the **update_currents** command run is successfully complete.

extract_cut_via_array

Enables or disables to cut a large via array into individual cuts.

TYPE

boolean

DEFAULT

false

DESCRIPTION

This variable controls whether a large via array is cut into individual cuts during extraction. If it is set to *true*, the extractor cuts the big via array into individual cuts. By default, the extractor does not cut a large via array.

SEE ALSO

```
set_extract_supply_parasitics_options(2)
extract_supply_parasitics(2)
extract_via_cut_row_column(3)
extract_via_square_cutting(3)
```

extract_via_cut_row_column

Sets the size of the row and the column when cutting a via array.

TYPE

integer

DEFAULT

0

DESCRIPTION

This variable specifies the size of the row and the column of the sub-array when cutting a via array during extraction. When setting the size to 0 (default), the extractor does not cut a via array. With a positive integer number *n*, the extractor cuts a via array into sub-arrays of size *n* * *n*.

SEE ALSO

```
set_extract_supply_parasitics_options(2)
extract_supply_parasitics(2)
extract_cut_via_array(3)
extract_via_square_cutting(3)
```

extract_via_square_cutting

Enables or disables to cut a via array into several square sub-arrays.

TYPE

boolean

DEFAULT

false

DESCRIPTION

This variable controls whether a via array is cut into several square (or near-square) sub-arrays during extraction. When set to *true*, the extractor cuts a via array into several square sub-arrays. For example, The via array of size 10x36 will be cut into three 10x10 sub-arrays and one 10x6 sub-array. The via array of size 10x52 will be cut into four 10x10 sub-arrays and one 10x12 sub-array.

By default, the extractor does not cut a via array into square sub-arrays.

SEE ALSO

```
set_extract_supply_parasitics_options(2)
extract_supply_parasitics(2)
extract_via_cut_row_column(3)
extract_cut_via_array(3)
```

power_rail_output_file

TYPE

string

DEFAULT

"" empty

DESCRIPTION

If the file name is provided with this variable, during power calculation relevant power information is written into the file provided. Note that information is written in a binary format. This file is then read by PrimeRail.Starting with the B-2008.12 release, the **update_power** command can per- form both average and peak power analysis. The power_analysis_mode variable can be used to specify the power analysis mode to perform. PrimeTime PX can perform different types of power analysis based on the mode setting. The default power analysis mode is averaged. For more information, see the power_analysis_mode man page. For the **update_current** command, only the averaged mode is supported.The set_power_analysis_options command can be used to specify the options for power analysis. It must be set before the **update_power** or **update_current** com- mand to take effect in power analysis. Issuing the set_power_analy- sis_options command with no option can reset all the power analysis options to default. Use the report_power_analysis_options command to get the current analysis option settings. See the set_power_analy- sis_options man page for more information.PrimeTime PX can consume either time-based (for example, a VCD file) or statistical switching activity information (for example, a SAIF file) for power calculation. For a particular analysis mode, appropriate activity information must be provided. For example, in time-based power analysis mode, a VCD file must be provided. In averaged power analysis mode, any form of switching activity information is accepted. You can specify a VCD file by using the read_vcd command. The statistical switching activity can be specified by using the read_saif or set_switching_activity commands.For the current value of this variable, type the following command: printvar power_rail_output_file.SH "SEE ALSO"

```
printvar(2)
update_power(2)
update_current(2)
power_analysis_mode(3)
set_power_analysis_options(2)
report_power_analysis_options(2)
```

power_sequential_default_static_probability

Specifies the default sequential static probability value.

TYPE

float

DEFAULT

0.05

DESCRIPTION

The **power_sequential_default_static_probability** and **power_sequential_default_toggle_rate** variable are used to determines the switching activity of sequential outputs that are not annotated.

If this variable is specified , then propagation is stopped for all the un-annotated sequential cell ouputs and instead assigned with this value. This mechanism can be used to speedup the runtime for activity propagation .

The value of **power_sequential_default_static_probability** should be between 0.0 and 1.0.

SEE ALSO

```
power_sequential_default_toggle_rate(3)
power_default_toggle_rate(3)
power_default_toggle_rate_reference_clock(3)
set_switching_activity(2)
```

power_sequential_default_toggle_rate

Specifies the default sequential output toggle rate value.

TYPE

float

DEFAULT

-1.0

DESCRIPTION

The **power_sequential_default_toggle_rate**, and **power_sequential_default_static_probability** variables are used to determine the switching activity of sequential outputs that are not annotated.

If this variable is specified , then propagation is stopped for all the un-annotated sequential cell ouputs and instead assigned with this value. This mechanism can be used to speedup the runtime for activity propagation .

The value of **power_sequential_default_toggle_rate** should be between 0.0 and 1.0.

SEE ALSO

```
set_switching_activity(2)
power_default_static_probability(3)
power_default_toggle_rate(3)
power_sequential_default_static_probability(3)
```

pr_checking_dir

Indicates the name of the validity checking directory being used by this session.

TYPE

string

DEFAULT

PR_CHECKING_DIR

DESCRIPTION

This variable contains the name of the session validity checking directory. This variable is read-only.

The session checking directory is used to store validity checking results used by the application while it is running. The directory is created within your current Unix working directory when the application is invoked. It uses the same numbering scheme as the session working directory pointed by variable **pr_work_dir**, and the default session checking directory name is PR_CHECKING_DIR. This means that if the working directory is PR_WORK_DIR or PR_WORK_DIR_N, the checking directory is PR_CHECKING_DIR OR PR_CHECKING_DIR_N respectively. This allows multiple simultaneous runs of the application from within the same Unix working directory, with each run using its own session checking directory.

If the checking directory already exists and the user does not have write-access permission, we will rename the existing old directory to PR_CHECKING_DIR.M or PR_CHECKING_DIR_N.M where M is 1,2,3...

When the application session is ended by **quit** or **exit**, the session validity checking directory and all its contents are not deleted. This is different from the session work directory.

SEE ALSO

```
pr_work_dir(2)
exit(2)
quit(2)
```

pr_work_dir

Indicates the name of the working directory being used by this session.

TYPE

string

DEFAULT

PR_WORK_DIR

DESCRIPTION

This variable contains the full path name of the session working directory being used by PrimeRail. This variable is read-only.

The session working directory is used to store temporary files used by the application while it is running. The directory is created within your current Unix working directory when the application is invoked. When the application session is ended by **quit** or **exit**, the session working directory and all its contents are automatically deleted.

The temporary files created within the session working directory are for use by the application only. For example, PrimeRail uses this directory to store in-session rail results (see **current_rail_result**), to offload memory that is not currently being used to disk caches, etc.

The default session working directory name is PR_WORK_DIR. If this directory name already exists when the application is invoked, then a directory of the name PR_WORK_DIR_N is created, where N is an integer starting at 1 and is the lowest value needed to create a unique (unused) directory name. This allows multiple simultaneous runs of the application from within the same Unix working directory, with each run using its own session working directory.

SEE ALSO

```
current_rail_result(2)
exit(2)
quit(2)
```

rail_autoload_mw_sidefile

Controls the automatic loading of PrimeRail library characterization sidefiles that have been stored in Milkyway.

TYPE

boolean

DEFAULT

true

DESCRIPTION

The **rail_autoload_mw_sidefile** variable controls whether pr_shell automatically loads the legacy PrimeRail library characterization sidefiles that have been saved in Milkyway. By default, these sidefiles are automatically loaded.

Before PrimeRail version F-2012.06, users had the option of saving the library characterization sidefiles within Milkyway, so the files were automatically reused whenever the design was opened. Starting with version F-2012.06, this ability is obsolete and has been replaced by the **rail_sidefile_path** methodology. The **rail_autoload_mw_sidefile** variable is provided to enable backward compatibility with Milkyway databases that was created by those previous versions.

This **rail_autoload_mw_sidefile** variable must be set before the design is read with **open_mw_cel**. When the variable is set to *true*, the **rail_sidefile_path** variable is ignored when reading the design.

Refer to the **rail_sidefile_path** man page for further information on the use of legacy PrimeRail library characterization side files within the rail analysis process.

SEE ALSO

```
list_sidefiles(2)
open_mw_cel(2)
rail_sidefile_path(3)
```

rail_event_shifting_type

TYPE

string

DEFAULT

"average"

DESCRIPTION

PrimeRail has the capability to shift the time of current waveform based on the timing information from static timing analysis. The feature applies to the RTL and zero-delay VCD information (by running the **read_vcd -rtl/-zero_delay** command) as well as vectorfree rail analysis (by running **update_currents -dynamic_vectorfree**). PrimeRail does not change the real delay gate-level VCD information.

This variable determines which type of timing information for event shifting. Valid values *min*, *average* and *max*, representing the minimum, average and maximum arrival time values from static timing analysis. The average arrival time (default) is the average time of the arrival window between minimum and maximum arrival time. The variable needs to be specified before the **update_currents** command is executed.

```
printvar(2)
read_vcd(2)
update_currents(2)
```

rail_force_tap_check

Forces the **create_taps** command to verify that the specified tap location touches the supply net layout geometry. By default, the check is done automatically for the first 100 **create_taps** command runs with the -point and -layer options. The checking on 101th (and beyond) **create_taps** command runs is not supported.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

Use this variable in conjunction with the -point and -layer options of the **create_taps** command. The default value is *false*. When setting to *true*, the command checks if the specified tap point location corresponds to the layout geometry connected to the target supply nets. By default, only the first 100 **create_taps** command runs with the -point and -layer options are checked. The variable has no effect if it is used with the -nocheck, -import or -of_object option of the **create_taps** command.

SEE ALSO

`create_taps(2)`

rail_hard_macro_continue_on_error

Determines whether PrimeRail error stops if detects a dirty design hard macro. The default behavior is no error stop for dirty design if the variable is not set.

TYPE

boolean

DEFAULT

true

DESCRIPTION

This variable controls whether to stop while the hard macro is a dirty design. For example, a hard macro with CONN view and CSF but the CSF is corrupt or in wrong format or a DWM-lite hard macro but part of model files are corrupted. If the variable is not set or is true, PrimeRail will ignore the hard macro data and continue to do next step. One particular behavior is that PrimeRail will try CONN view with CSF if it failed to access DWM-lite data.

SEE ALSO

`hard_macro_flow(2)`

rail_ignore_power_for_voltage_mismatch

Determines whether PrimeRail ignores instance power read from a PrimeTime-PX binary report file if the ideal voltage for the PG pin in the file differs from the ideal PG pin voltage assigned in the current session.

TYPE

boolean

DEFAULT

false

DESCRIPTION

This variable controls whether to use the instance power value of a PG pin read from a PrimeTime-PX binary report file if the defined voltage for the pin in the file differs from the PG pin voltage assigned in the current session.

By default, PrimeRail uses the instance power read from the PrimeTime-PX file regardless of whether the defined ideal PG pin voltages match. If there are instance PG pins where there is a voltage mismatch PrimeRail issues a summary report after loading the binary report file indicating how many pins failed to match, and pointing you to a file for further details.

The ideal voltage of a PG pin in PrimeRail is defined as the voltage of the supply net connected to the PG pin.

SEE ALSO

```
set_voltage(2)
```

rail_intrinsic_capacitance_threshold

DARK RED .SH NAME

rail_intrinsic_capacitance_threahold

A threshold to check against intrinsic capacitance value for a cell. If the intrinsic capacitance value is over the threshold, a RAIL-114 message will be issued for the cell.

TYPE

float

DEFAULT

1.0

DESCRIPTION

This variable defines a threshold to check against intrinsic capacitance value for a cell. If the intrinsic capacitance value is over the threshold, a RAIL-114 message will be issued for the cell and directed to the validity checking directory during **update_current.** Users can specify the validity checking directory with the **pr_checking_dir** variable.

The default unit is pF.

SEE ALSO

```
update_current(2)
pr_checking_dir(3)
```

rail_load_ccs_power

Automatically loads CCS power data from Synopsys Liberty (.db) files.

TYPE

boolean

DEFAULT

true

DESCRIPTION

The **rail_load_ccs_power** variable controls whether *pr_shell* automatically loads CCS power library data from Synopsys Liberty .db files. By default CCS power library data are automatically loaded from .db files.

This variable must be set before the design is read with **open_mw_cel**.

SEE ALSO

```
open_mw_cel(2)
```

rail_peak_current_threshold

DARK RED .SH NAME

rail_peak_current_threahold

A threshold to check against peak current value for a cell such that if it is over the threshold, a RAIL-109 message will be issued for the cell.

TYPE

float

DEFAULT

1.0

DESCRIPTION

This variable defines a threshold to check against peak current value for a cell such that if it is over the threshold, a RAIL-109 message will be issued for the cell and directed to the validity checking directory during **update_current.** User can specify the validity checking directory by a variable **pr_checking_dir.**

The default unit is Amp.

SEE ALSO

```
open_mw_cel(2)
pr_checking_dir(3)
```

rail_power_threshold

DARK RED .SH NAME

rail_power_threahold

A threshold to check against total instance power value for a cell. If the total instance power value is over the threshold, a RAIL-113 message will be issued for the cell.

TYPE

float

DEFAULT

1.0

DESCRIPTION

This variable defines a threshold to check against total instance power value for a cell. The total instance power value is over the threshold, a RAIL-113 message will be issued for the cell and directed to the validity checking directory during **update_current.** Users can specify the validity checking directory with the **pr_checking_dir** variable.

The default unit is W.

SEE ALSO

```
update_current(2)
pr_checking_dir(3)
```

rail_sidefile_path

Specifies a list of legacy PrimeRail library characterization sidefiles to be used during linking.

TYPE

list

DEFAULT

None

DESCRIPTION

Specifies a list of legacy PrimeRail library characterization sidefiles to be used during linking and analysis. Designs referencing CCS Power libraries do not need to use this variable.

These legacy sidefiles have been created by previous versions of PrimeRail as part of the library characterization process, and contain characterization data that is used during various portions of the rail analysis process. The **rail_sidefile_path** must be specified before the design is read by **open_mw_cel**; changes to the variable after the design has been read are ignored.

The **rail_sidefile_path** variable can contain filenames only.

For elements in the **rail_sidefile_path** list, *pr_shell* searches for the file using the **search_path** variable.

By default the **rail_sidefile_path** variable is empty. To determine the current value of this variable, type **printvar rail_sidefile_path** or **echo $rail_sidefile_path**.

This variable is ignored if the **rail_autoload_mw_sidefile** variable is set *true*.

SEE ALSO

```
list_sidefiles(2)
open_mw_cel(2)
rail_autoload_mw_sidefile(3)
search_path(3)
```

report_macro_internal_pin

Controls the behavior of macro pin report commands.

TYPE

boolean

DEFAULT

false

DESCRIPTION

This variable controls the behavior of pg_pin report commands. If the variable is set true, these commands report the macro internal pin. The macro internal pin name has a special key word "_mm_dev_cel_inst_". Normally users are not expected to see the internal pins so the default value is false.

EXAMPLES

The following example shows how the variable changes report_rail_voltage report.

```
pr_shell> set report_macro_internal_pin false
pr_shell> report_rail_voltage

Supply Net VDD voltage report:
Supply voltage is: 1.5
There are only 4 signals, can not report Top 5
Top 4 static voltage drop values at instance ports:
1: 71.216 (mV) at inst4/VDD (sram1056x12)
2: 66.601 (mV) at inst1/VDD (sram1056x12)
3: 50.07 (mV) at inst2/VDD (sram1056x12)
4: 48.87 (mV) at inst3/VDD (sram1056x12)

pr_shell> set report_macro_internal_pin true
pr_shell> report_rail_voltage

Supply Net VDD voltage report:
Supply voltage is: 1.5
There are only 4 signals, can not report Top 5
Top 4 static voltage drop values at instance ports:
    1: 75.704 (mV) at inst4/_mm_dev_cel_inst_VDD/15303 (sram1056x12)
    2: 73.257 (mV) at inst4/_mm_dev_cel_inst_VDD/14804 (sram1056x12)
    3: 73.257 (mV) at inst4/_mm_dev_cel_inst_VDD/14805 (sram1056x12)
    4: 72.666 (mV) at inst4/_mm_dev_cel_inst_VDD/14260 (sram1056x12)
    5: 72.666 (mV) at inst4/_mm_dev_cel_inst_VDD/14258 (sram1056x12)
```

SEE ALSO

```
report_rail_voltage(2)
calculate_rail_voltage(2)
```

sh_allow_tcl_with_set_app_var

Allows the **set_app_var** and **get_app_var** commands to work with application variables.

TYPE

string

DEFAULT

application specific

DESCRIPTION

Normally the **get_app_var** and **set_app_var** commands only work for variables that have been registered as application variables. Setting this variable to **true** allows these commands to set a Tcl global variable instead.

These commands issue a CMD-104 error message for the Tcl global variable, unless the variable name is included in the list specified by the **sh_allow_tcl_with_set_app_var_no_message_list** variable.

SEE ALSO

```
get_app_var(2)
set_app_var(2)
sh_allow_tcl_with_set_app_var_no_message_list(2)
```

sh_allow_tcl_with_set_app_var_no_message_list

Suppresses CMD-104 messages for variables in this list.

TYPE

string

DEFAULT

application specific

DESCRIPTION

This variable is consulted before printing the CMD-104 error message, if the **sh_allow_tcl_with_set_app_var** variable is set to **true**. All variables in this Tcl list receive no message.

SEE ALSO

```
get_app_var(2)
set_app_var(2)
sh_allow_tcl_with_set_app_var(2)
```

sh_arch

Indicates the system architecture of your machine.

TYPE

string

DEFAULT

platform-dependent

DESCRIPTION

The **sh_arch** variable is set by the application to indicate the system architecture of your machine. Examples of machines being used are sparcOS5, amd64, and so on. This variable is read-only.

sh_command_abbrev_mode

Sets the command abbreviation mode for interactive convenience.

TYPE

string

DEFAULT

application specific

DESCRIPTION

This variable sets the command abbreviation mode as an interactive convenience. Script files should not use any command or option abbreviation, because these files are then susceptible to command changes in subsequent versions of the application.

Although the default value is **Anywhere**, it is recommended that the site startup file for the application set this variable to **Command-Line-Only**. It is also possible to set the value to **None**, which disables abbreviations altogether.

To determine the current value of this variable, use the **get_app_var sh_command_abbrev_mode** command.

SEE ALSO

```
sh_command_abbrev_options(3)
get_app_var(2)
set_app_var(2)
```

sh_command_abbrev_options

Turns off abbreviation of command dash option names when false.

TYPE

boolean

DEFAULT

application specific

DESCRIPTION

When command abbreviation is currently off (see sh_command_abbrev_mode) then setting this variable to false will also not allow abbreviation of command dash options. This variable also impacts abbreviation of the values specified to command options that expect values to be one of an allowed list of values.

This variable exists to be backward compatible with previous tool releases which always allowed abbreviation of command dash options and option values regardless of the command abbreviation mode.

It is recommended to set the value of this variable to false.

To determine the current value of this variable, use the **get_app_var sh_command_abbrev_options** command.

SEE ALSO

```
sh_command_abbrev_mode(3)
get_app_var(2)
set_app_var(2)
```

sh_command_log_file

Specifies the name of the file to which the application logs the commands you executed during the session.

TYPE

string

DEFAULT

empty string

DESCRIPTION

This variable specifies the name of the file to which the application logs the commands you run during the session. By default, the variable is set to an empty string, indicating that the application's default command log file name is to be be used. If a file named by the default command log file name cannot be opened (for example, if it has been set to read only access), then no logging occurs during the session.

This variable can be set at any time. If the value for the log file name is invalid, the variable is not set, and the current log file persists.

To determine the current value of this variable, use the **get_app_var sh_command_log_file** command.

SEE ALSO

```
get_app_var(2)
set_app_var(2)
```

sh_continue_on_error

Allows processing to continue when errors occur during script execution with the **source** command.

TYPE

Boolean

DEFAULT

application specific

DESCRIPTION

This variable is deprecated. It is recommended to use the **-continue_on_error** option to the **source** command instead of this variable because that option only applies to a single script, and not the entire application session.

When set to **true**, the **sh_continue_on_error** variable allows processing to continue when errors occur. Under normal circumstances, when executing a script with the **source** command, Tcl errors (syntax and semantic) cause the execution of the script to terminate.

When **sh_continue_on_error** is set to **false**, script execution can also terminate due to new error and warning messages based on the value of the **sh_script_stop_severity** variable.

To determine the current value of the **sh_continue_on_error** variable, use the **get_app_var sh_continue_on_error** command.

SEE ALSO

```
get_app_var(2)
set_app_var(2)
source(2)
sh_script_stop_severity(3)
```

sh_dev_null

Indicates the current null device.

TYPE

string

DEFAULT

platform dependent

DESCRIPTION

This variable is set by the application to indicate the current null device. For example, on UNIX machines, the variable is set to **/dev/null**. This variable is read-only.

SEE ALSO

`get_app_var(2)`

sh_enable_page_mode

Displays long reports one page at a time (similar to the UNIX **more** command.

TYPE

Boolean

DEFAULT

application specific

DESCRIPTION

This variable displays long reports one page at a time (similar to the UNIX **more** command), when set to **true**. Consult the man pages for the commands that generate reports to see if they are affected by this variable.

To determine the current value of this variable, use the **get_app_var sh_enable_page_mode** command.

SEE ALSO

```
get_app_var(2)
set_app_var(2)
```

sh_enable_stdout_redirect

Allows the redirect command to capture output to the Tcl stdout channel.

TYPE

Boolean

DEFAULT

application specific

DESCRIPTION

When set to **true**, this variable allows the redirect command to capture output sent to the Tcl stdout channel. By default, the Tcl **puts** command sends its output to the stdout channel.

SEE ALSO

```
get_app_var(2)
set_app_var(2)
```

sh_help_shows_group_overview

Changes the behavior of the "help" command.

TYPE

string

DEFAULT

application specific

DESCRIPTION

This variable changes the behavior of the **help** command when no arguments are specified to help. Normally when no arguments are specified an informational message with a list of available command groups is displayed.

When this variable is set to false the command groups and the commands in each group is printed instead. This variable exists for backward compatibility.

SEE ALSO

```
help(2)
set_app_var(2)
```

sh_new_variable_message

Controls a debugging feature for tracing the creation of new variables.

TYPE

Boolean

DEFAULT

application specific

DESCRIPTION

The **sh_new_variable_message** variable controls a debugging feature for tracing the creation of new variables. Its primary debugging purpose is to catch the misspelling of an application-owned global variable. When set to **true**, an informational message (CMD-041) is displayed when a variable is defined for the first time at the command line. When set to **false**, no message is displayed.

Note that this debugging feature is superseded by the new **set_app_var** command. This command allows setting only application-owned variables. See the **set_app_var command man page for details.**

Other variables, in combination with **sh_new_variable_message**, enable tracing of new variables in scripts and Tcl procedures.

Warning: This feature has a significant negative impact on CPU performance when used with scripts and Tcl procedures. This feature should be used only when developing scripts or in interactive use. When you turn on the feature for scripts or Tcl procedures, the application issues a message (CMD-042) to warn you about the use of this feature.

To determine the current value of this variable, use the **get_app_var sh_new_variable_message** command.

SEE ALSO

```
get_app_var(2)
set_app_var(2)
sh_new_variable_message_in_proc(3)
sh_new_variable_message_in_script(3)
```

sh_new_variable_message_in_proc

Controls a debugging feature for tracing the creation of new variables in a Tcl procedure.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

The **sh_new_variable_message_in_proc** variable controls a debugging feature for tracing the creation of new variables in a Tcl procedure. Its primary debugging purpose is to catch the misspelling of an application-owned global variable.

Note that this debugging feature is superseded by the new **set_app_var** command. This command allows setting only application-owned variables. Please see the **set_app_var** command man page for details.

Note that the **sh_new_variable_message** variable must be set to **true** for this variable to have any effect. Both variables must be set to **true** for the feature to be enabled. Enabling the feature simply enables the **print_proc_new_vars** command. In order to trace the creation of variables in a procedure, this command must be inserted into the procedure, typically as the last statement. When all of these steps have been taken, an informational message (CMD-041) is generated for new variables defined within the procedure, up to the point that the **print_proc_new_vars** commands is executed.

Warning: 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 use of this feature.

To determine the current value of this variable, use the **get_app_var sh_new_variable_message_in_proc** command.

SEE ALSO

```
get_app_var(2)
print_proc_new_vars(2)
```

```
set_app_var(2)
sh_new_variable_message(3)
sh_new_variable_message_in_script(3)
```

sh_new_variable_message_in_script

Controls a debugging feature for tracing the creation of new variables within a sourced script.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

The **sh_new_variable_message_in_script** variable controls a debugging feature for tracing the creation of new variables within a sourced script. Its primary debugging purpose is to catch the misspelling of an application-owned global variable.

Note that this debugging feature is superseded by the new **set_app_var** command. This command allows setting only application-owned variables. See the **set_app_var** command man page for details.

Note that the **sh_new_variable_message** variable must be set to **true** for this variable to have any effect. Both variables must be set to **true** for the feature to be enabled. In that case, an informational message (CMD-041) is displayed when a variable is defined for the first time. When **sh_new_variable_message_in_script** is set to **false** (the default), no message is displayed at the time that the variable is created. When the **source** command completes, however, you see messages for any new variables that were created in the script. This is because the state of the variables is sampled before and after the **source** command. It is not because of inter-command sampling within the script. So, this is actually a more efficient method to see if new variables were created in the script.

For example, given the following script a.tcl:

```
echo "Entering script"
set a 23
echo a = $a
set b 24
echo b = $b
echo "Exiting script"
```

When **sh_new_variable_message_in_script** is **false** (the default), you see the following when you source the script:

```
prompt> source a.tcl
Entering script
```

```
  a = 23
  b = 24
  Exiting script
  Information: Defining new variable 'a'. (CMD-041)
  Information: Defining new variable 'b'. (CMD-041)
  prompt>
```

Alternatively, when **sh_new_variable_message_in_script** is **true**, at much greater cost, you see the following when you source the script:

```
  prompt> set sh_new_variable_message_in_script true
  Warning: Enabled new variable message tracing -
           Tcl scripting optimization disabled. (CMD-042)
  true
  prompt> source a.tcl
  Entering script
  Information: Defining new variable 'a'. (CMD-041)
  a = 23
  Information: Defining new variable 'b'. (CMD-041)
  b = 24
  Exiting script
  prompt>
```

Warning: 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 use of this feature.

To determine the current value of this variable, use the **get_app_var sh_new_variable_message_in_script** command.

SEE ALSO

```
get_app_var(2)
set_app_var(2)
sh_new_variable_message(3)
sh_new_variable_message_in_proc(3)
```

sh_product_version

Indicates the version of the application currently running.

TYPE

string

DESCRIPTION

This variable is set to the version of the application currently running. The variable is read only.

To determine the current value of this variable, use the **get_app_var sh_product_version** command.

SEE ALSO

`get_app_var(2)`

sh_script_stop_severity

Indicates the error message severity level that would cause a script to stop running before it completes.

TYPE

string

DEFAULT

application specific

DESCRIPTION

When a script is run with the **source** command, there are several ways to get it to stop running before it completes. One is to use the **sh_script_stop_severity** variable. This variable can be set to **none**, **W**, or **E**.

- When set to **E**, the generation of one or more error messages by a command causes a script to stop.
- When set to **W**, the generation of one or more warning or error messages causes a script to stop.
- When set to **none**, the generation messages does not cause the script to stop.

Note that **sh_script_stop_severity** is ignored if **sh_continue_on_error** is set to **true**.

To determine the current value of this variable, use the **get_app_var sh_script_stop_severity** command.

SEE ALSO

```
get_app_var(2)
set_app_var(2)
source(2)
sh_continue_on_error(3)
```

sh_source_emits_line_numbers

Indicates the error message severity level that causes an informational message to be issued, listing the script name and line number where that message occurred.

TYPE

string

DEFAULT

application specific

DESCRIPTION

When a script is executed with the **source** command, error and warning messages can be emitted from any command within the script. Using the **sh_source_emits_line_numbers** variable, you can help isolate where errors and warnings are occurring.

This variable can be set to **none**, **W**, or **E**.

- When set to **E**, the generation of one or more error messages by a command causes a CMD-082 informational message to be issued when the command completes, giving the name of the script and the line number of the command.
- When set to **W**, the generation of one or more warning or error messages causes a the CMD-082 message.

The setting of **sh_script_stop_severity** affects the output of the CMD-082 message. If the setting of **sh_script_stop_severity** causes a CMD-081 message, then it takes precedence over CMD-082.

To determine the current value of this variable, use the **get_app_var sh_source_emits_line_numbers** command.

SEE ALSO

```
get_app_var(2)
set_app_var(2)
source(2)
sh_continue_on_error(3)
sh_script_stop_severity(3)
```

```
CMD-081(n)
CMD-082(n)
```

sh_source_logging

Indicates if individual commands from a sourced script should be logged to the command log file.

TYPE

Boolean

DEFAULT

application specific

DESCRIPTION

When you source a script, the **source** command is echoed to the command log file. By default, each command in the script is logged to the command log file as a comment. You can disable this logging by setting **sh_source_logging** to **false**.

To determine the current value of this variable, use the **get_app_var sh_source_logging** command.

SEE ALSO

```
get_app_var(2)
set_app_var(2)
source(2)
```

sh_source_uses_search_path

Indicates if the **source** command uses the **search_path** variable to search for files.

TYPE

Boolean

DEFAULT

application specific

DESCRIPTION

When this variable is set to \ftrue the **source** command uses the **search_path** variable to search for files. When set to **false**, the **source** command considers its file argument literally.

To determine the current value of this variable, use the **get_app_var sh_source_uses_search_path** command.

SEE ALSO

```
get_app_var(2)
set_app_var(2)
source(2)
search_path(3)
```

sh_tcllib_app_dirname

Indicates the name of a directory where application-specific Tcl files are found.

TYPE

string

DESCRIPTION

The **sh_tcllib_app_dirname** variable is set by the application to indicate the directory where application-specific Tcl files and packages are found. This is a read-only variable.

SEE ALSO

`get_app_var(2)`

sh_user_man_path

Indicates a directory root where you can store man pages for display with the **man** command.

TYPE

list

DEFAULT

empty list

DESCRIPTION

The **sh_user_man_path** variable is used to indicate a directory root where you can store man pages for display with the **man** command. The directory structure must start with a directory named *man*. Below *man* are directories named *cat1*, *cat2*, *cat3*, and so on. The **man** command will look in these directories for files named *file.1*, *file.2*, and *file.3*, respectively. These are pre-formatted files. It is up to you to format the files. The **man** command effectively just types the file.

These man pages could be for your Tcl procedures. The combination of defining help for your Tcl procedures with the **define_proc_attributes** command, and keeping a manual page for the same procedures allows you to fully document your application extensions.

The **man** command will look in **sh_user_man_path** after first looking in application-defined paths. The user-defined paths are consulted only if no matches are found in the application-defined paths.

To determine the current value of this variable, use the **get_app_var sh_user_man_path** command.

SEE ALSO

```
define_proc_attributes(2)
get_app_var(2)
man(2)
set_app_var(2)
```

synopsys_program_name

Indicates the name of the program currently running.

TYPE

string

DESCRIPTION

This variable is read only, and is set by the application to indicate the name of the program you are running. This is useful when writing scripts that are mostly common between some applications, but contain some differences based on the application.

To determine the current value of this variable, use **get_app_var synopsys_program_name**.

SEE ALSO

get_app_var(2)

synopsys_root

Indicates the root directory from which the application was run.

TYPE

string

DESCRIPTION

This variable is read only, and is set by the application to indicate the root directory from which the application was run.

To determine the current value of this variable, use **get_app_var synopsys_root**.

SEE ALSO

get_app_var(2)