

JasperGold® Apps

Command Reference Manual

Product Version 2015.09
September 2015

© 2003-2015 Cadence Design Systems, Inc. All rights reserved.

Printed in the United States of America.

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Product JasperGold Apps incorporates software developed by others and redistributed according to license agreement. For further details, see `doc/third_party_readme.txt`.

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission.

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522. All other trademarks are the property of their respective holders.

Restricted Permission: This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used only in accordance with a written agreement between Cadence and its customer.
2. The publication may not be modified in any way.
3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor

Contents

1

<u>Introduction</u>	15
<u>What's New</u>	16
<u>Reference Manual Organization</u>	45
<u>Related Documentation</u>	47
<u>JasperGold Apps Tcl</u>	48
<u>UNIX Commands</u>	48
<u>Conventions Used in This Manual</u>	50

2

<u>Installation</u>	53
<u>Supported Platforms</u>	54
<u>Java Memory Limits</u>	54
<u>Supported Languages and Standards</u>	54
<u>JasperGold Apps Installation</u>	56
<u>License Installation</u>	57
<u>TIMEOUT and TIMEOUTALL</u>	59
<u>Redundant License Servers</u>	59
<u>Installing Your License</u>	59
<u>Appending Lines to a License File</u>	60
<u>Manually Starting a New Daemon</u>	61
<u>Using the Correct License Server</u>	62
<u>Using FlexNet Publisher Utilities</u>	63
<u>Optional Environment Variables</u>	65
<u>Simultaneously Running Jasper and Cadence Licenses</u>	65
<u>Practical License Setup Example</u>	65
<u>Optimizing Performance for License Checkout</u>	66

3

<u>Commands and Variables Overview</u>	67
<u>Command-Line Options</u>	68
<u>Environment Variables</u>	73
<u>Command Summary</u>	78

4

<u>General Commands</u>	143
<u>General Information</u>	144
<u>Tab Completion</u>	144
<u>Aborting a Command</u>	144
<u>Prefix Matching</u>	145
<u>Specifying the Current Task</u>	145
<u>abstract</u>	146
<u>abvip</u>	158
<u>add_tx_attribute</u>	159
<u>analyze</u>	160
<u>assert</u>	174
<u>assume</u>	187
<u>auto_setup</u>	203
<u>auto_syn</u>	205
<u>autoprove</u>	207
<u>blackbox_assistant</u>	213
<u>capture testcase</u>	217
<u>check_arch</u>	220
<u>check_assumptions</u>	224
<u>check_bps</u>	228
<u>check_code</u>	240
<u>check_conn</u>	241
<u>check_cov</u>	258
<u>check_csr</u>	291
<u>check_dld</u>	303
<u>check_loop</u>	323
<u>check_ipv</u>	326

JasperGold Apps Command Reference Manual

check_one_of	344
check_sec	345
check_sps	399
check_sps_configure	410
check_sps_waiver	414
check_spv	421
check_unr	432
check_xprop	437
clear	453
clock	455
configure	464
connect	466
cover	471
custom_gui_action	480
database	483
dglob	499
edit	500
elaborate	501
exit	531
export	533
formal Bring_up	552
fsm_checks	554
get_annotation	557
get_clock_info	558
get_decomposed_expression	560
get_design_info	564
get_design_space_coverage	612
get_fanin	623
get_fanout	630
get_filename	633
get_flop_info	634
get_latch_info	640
get_message	643
get_needed_assumptions	645
get_proj_dir	650
get_property_info	651

JasperGold Apps Command Reference Manual

get_reset_info	660
get_root_name	665
get_signal_info	666
get_signal_list	674
get_status	678
get_tool	682
get_top_module	684
get_tx_attributes	685
get_version	686
help	687
include	690
interrupt	691
ipk	693
ipxact	696
jasper_model_divider	705
jasper_model_mpram	709
jasper_model_multiplier	715
jasper_model_ram	720
jasper_scoreboard_3	727
justify	729
liberty	735
load_radix_file	736
pretty_print	737
proof_accelerator	739
prove	741
redirect	754
remove_tx_attribute	756
report	757
reset	761
restore	779
sanity_check	782
save	787
scan	793
schematic_viewer	799
scope	802
session	815

set_annotation	824
set_current_gui	826
set_message	829
show_list	835
show_schematic	836
show_source_browser	838
show_waveform	839
stopat	840
suggest	845
task	847
template_property	853
trace	856
view	863
visualize	864
waveform	938

5

<u>Configuration Commands</u>	943
analyze_libnonamehide	944
analyze_librescan	946
analyze_libunboundsearch	948
arch_prll_property_consolidation	950
auto_disable_related_covers	952
auto_syn_first_trace_attempt	954
auto_syn_structural_trace_generator	956
automatic_library_search	958
cache_proof_simplification	960
cache_reset_values	962
capture_elaborated_design	964
cell_define_debug_visibility	966
clock_auto_stabilize	968
clock_handle_self_disabling_registers	970
clock_process_reconvergent_logic	972
clock_structural_glitch_analysis	974
cmd_time_limit	976

JasperGold Apps Command Reference Manual

command_license_wait_interval	979
command_license_wait_retries	981
complexity_manager_allow_link_expansion	983
complexity_manager_compute_statistics	985
complexity_manager_design_information_mode	987
complexity_manager_show_connected_assumptions	989
complexity_manager_show_inputs	991
complexity_manager_show_link_statistics	993
counter_speculative_detection	995
design_exploration_include_clock_logic	997
design_exploration_include_connected_assumptions	999
dst_mode	1001
elab_time_limit	1003
elaborate_quiet_trace	1005
engine_job_timers	1007
engine_mode	1009
engine_solver	1020
engine_threads	1022
engineB_before_first_mode	1024
engineB_before_increment_mode	1026
engineB_first_trace_attempt	1028
engineB_single_property	1030
engineB_trace_attempt_increment	1032
engineB_trace_attempt_time_limit	1034
engineB_trace_attempt_time_limit_incr	1036
engineC_optimization	1038
engineCG_max_mem	1040
engineD_optimization	1042
engineG_optimization	1044
engineH_single_property	1046
engineJ_attempts	1048
engineJ_max_trace_length	1050
engineJ_migrate	1052
engineJ_restarts	1054
engineJ_seed	1056
engineJ_single_property	1058

JasperGold Apps Command Reference Manual

engineL_cache	1060
engineL_generate_trails	1062
engineL_ignore_trace	1064
engineL_max_segment_length	1066
engineL_random_diversification	1068
engineL_search_mode	1070
engineL_state_removal	1072
engineL_tail_length	1074
engineQ3_max_trace_length	1076
engineQ3_random_diversification	1078
engineQ3_random_diversification_factor	1080
engineQ3_restarts	1082
engineQ3_seed	1084
export_sva_bind_enable_directive	1086
export_sva_embed_lib_modules	1088
export_sva_handshake_delay_threshold	1090
export_sva_hierarchy_separator	1092
export_sva_hierarchy_separator_escape_string	1094
export_sva_reset_condition	1096
export_sva_target_environment	1098
export_sva_trace_suffix_cycles	1100
export_sva_use_escaped_identifier_with_extended_names	1102
extract_high_level_fsm	1104
first_trace_attempt	1106
function_debug_visibility	1108
log_timestamp_format	1110
log_timestamp_mode	1112
max_trace_length	1114
message_history_limit	1116
parallel_proof_mode	1118
prompt1	1120
prompt2	1121
proof_simplification	1122
proofgrid_cluster	1124
proofgrid_engineJ_max_jobs	1126
proofgrid_engineL_max_jobs	1128

JasperGold Apps Command Reference Manual

proofgrid_extra_args	1130
proofgrid_manager	1132
proofgrid_max_jobs	1134
proofgrid_max_local_jobs	1136
proofgrid_mode	1138
proofgrid_per_engine_max_jobs	1141
proofgrid_per_engine_max_local_jobs	1144
proofgrid_per_engine_privileged_jobs	1146
proofgrid_queue	1148
proofgrid_restarts	1150
proofgrid_shell	1152
proofgrid_shell_stop	1157
proofgrid_skip_abandoned	1159
proofgrid_socket_communication	1161
property_auto_update_db	1163
property_compile_time_limit	1165
property_delay_cycles	1167
prove_no_cover_traces	1169
prove_no_traces	1171
prove_per_property_max_time_limit	1173
prove_per_property_time_limit	1176
prove_per_property_time_limit_factor	1179
prove_prefer_shortest	1181
prove_report_mem	1183
prove_report_period	1185
prove_start_timer_on_engine_start	1187
prove_time_limit	1189
prove_verbosity	1192
proven_directive	1194
reset_trace_inputs	1196
reset_vcd_fsdb_force_split_scope	1198
scan_constant_relations_seeker	1200
scan_default_assume_representation_policy	1202
scan_default_property_representation	1204
scan_no_sim_max_covers	1206
scan_per_scope_time_limit	1207

JasperGold Apps Command Reference Manual

scan_seek_depth	1209
scan_seek_mode	1211
scan_trace_stuck_at_values_limit	1215
scan_trace_witness_information	1217
scan_verbosity	1219
schematic_viewer_command	1221
schematic_viewer_connection_timeout	1223
schematic_viewer_design_script	1225
scope_clock	1227
scope_clock_edge	1229
scope_default_active_phase	1231
scope_default_clock	1233
scope_default_clock_edge	1235
scope_extract_min_ports_threshold	1237
scope_signal_width_threshold	1239
sec_auto_prove_prequalification_efforts	1241
sec_autoprove	1243
sec_autoprove_cutpoints_on_internal_signals	1245
sec_autoprove_cutpoints_on_latches	1247
sec_autoprove_max_number_of_cutpoints	1249
sec_autoprove_mode	1251
sec_prove_assumption_lifting	1253
sec_prove_levelize_helpers	1255
sec_prove_order_targets	1257
sec_prove_suppress_traces	1259
sps_arithmetic_overflow_style	1261
sps_deadcode_full_case_filter	1263
sps_no_duplicates_per_check	1265
sps_signals_flops	1267
sps_signals_max_bits	1269
sps_signals_outputs	1271
sps_stuck_at_compact	1273
sps_sva_max_depth	1275
sps_sva_max_length	1276
spv_include_control_path	1277
sst_default_trace_length	1279

JasperGold Apps Command Reference Manual

sst_deterministic	1281
stop_on_cex_limit	1283
stop_on_unreachable_limit	1284
tag_irrelevant_values	1286
task_compile_time_limit	1288
tcl_no_precondition	1290
time_format	1292
trace_extension	1294
trace_optimization	1296
trace_show_reset	1299
tunnel_library_cell	1301
visualize_auto_check_props	1303
visualize_auto_load_debugging_tables	1305
waveform_import_multiple_segments	1307
waveform_import_secondary_hier_path	1309
word_level_reduction	1311
xprop_compute_highlights	1313
xprop_use_all_undriven	1315
xprop_use_bbox_outputs	1317
xprop_use_inputs	1319
xprop_use_internal_undriven	1321
xprop_use_low_power	1323
xprop_use_reset_abstraction	1325
xprop_use_reset_state	1327
xprop_use_stopats	1329
xprop_use_x_assignments	1331
 A	
PSL Support	1333
 B	
SVA Support	1339
SVA Support Status	1340
Additional Support for SystemVerilog 2009	1349

C

<u>Liveness vs. Safety</u>	1351
--	------

D

<u>Tcl Support for SVA Expressions</u>	1353
--	------

E

<u>OVL Support</u>	1359
------------------------------------	------

<u>OVL v2.8.1 Support</u>	1360
---	------

<u>Special Handling for OVL Assertion Checkers</u>	1360
--	------

<u>Basic Flow for OVL Use</u>	1361
---	------

<u>SVA Implementation</u>	1362
---	------

<u>PSL Implementation</u>	1362
---	------

<u>VHDL Implementation</u>	1363
--	------

F

<u>IP-XACT Support</u>	1365
--	------

<u>Summary of IP-XACT Standard</u>	1366
--	------

<u>Using IP-XACT for Connectivity Verification</u>	1368
--	------

<u>Using IP-XACT for CSR Verification</u>	1369
---	------

<u>Vendor Extensions</u>	1369
--	------

<u>Examples</u>	1373
---------------------------------	------

G

<u>Proof Setup</u>	1381
------------------------------------	------

H

<u>Custom Radix Files</u>	1407
---	------

<u>Radix File Format</u>	1408
--	------

<u>Definition Clauses</u>	1408
---	------

<u>Use Clauses</u>	1409
------------------------------------	------

<u>Predefined Radix Names</u>	1409
---	------

JasperGold Apps Command Reference Manual

Example Radix File	1410
Related Documentation	1411
I	
Automatically Extracted Type Properties	1413
Introduction to VHDL Type Properties	1414
Introduction to SystemVerilog Type Properties	1415
Procedural Notes	1415
Type Property Limitations	1415
J	
Vacuity Checking with Automatically Extracted Covers	1417
Precondition Reachability Checking	1418
Property Examples	1418
SVA Immediate Properties	1418
Nested Implications	1419
Refining Vacuity Checking with Additional Covers	1420
Managing Vacuity Checking	1421
Enabling and Disabling Related Covers	1421
Limitations	1422
K	
Understanding Visualize Semantics with SVA Equivalents	1423
SystemVerilog BT/RC Keywords and NOCEX Boolean	1424
SystemVerilog BT Keyword	1424
SystemVerilog RC Keyword	1424
SystemVerilog NOCEX(p) Boolean	1424
SystemVerilog Equivalent of Visualize At Least Once Options	1424
SystemVerilog Equivalent of Visualize Force Options	1426
SystemVerilog Equivalent of Visualize Not Options	1427
SystemVerilog Equivalent of Visualize for a Signal Sequence	1427

Introduction

This chapter is an introduction to the *JasperGold Apps Command Reference Manual*. It includes the following sections:

- [What's New](#) on page 16
- [Reference Manual Organization](#) on page 45
- [Related Documentation](#) on page 47
- [JasperGold Apps Tcl](#) on page 48
- [UNIX Commands](#) on page 48
- [Conventions Used in This Manual](#) on page 50

Cadence® Design Systems, Inc. prohibits the use of our software in a way that does not comply with our written guidelines and documentation.

What's New

The release of JasperGold Apps version 2015.09 includes the debut of the UNR App, a new Clock Information window with options to run sanity checks on your clock environment, debug issues, and declare clocks. In addition, you will find usability improvements for the Visualize window, phase two of the Connectivity usability enhancements, and a new CSR Definitions Viewer. Additional details are available in [Table 1-1](#) on page 17 and in the Technical Update (incremental training presentation), which is available on [Cadence Online Support](#).

Refer to [Table 1-1](#) on page 17 for details about new and enhanced features and references to supporting documentation. Refer to the release notes in your build (`doc` directory) for information about fixes for reported issues.

- [New Release: JasperGold Coverage Unreachability App](#) on page 17
- [Connectivity Verification App](#) on page 18
- [Coverage App](#) on page 19
- [CSR Verification App](#) on page 21
- [Formal Property Verification \(FPV\) App](#) on page 22
- [Low Power Verification \(LPV\) App](#) on page 22
- [Sequential Equivalence Checking App](#) on page 24
- [Structural Property Synthesis \(SPS\) App](#) on page 25
- [Security Path Verification \(SPV\) App](#) on page 27
- [ABVIP](#) on page 29
- [Proof Accelerators \(PA\)](#) on page 30
- [Abstraction](#) on page 31
- [Clocks and Reset](#) on page 32
- [Design Exploration](#) on page 35
- [Design Synthesis](#) on page 37
- [Proofs and Visualize](#) on page 40
- [ProofGrid](#) on page 42
- [Usability](#) on page 43
- [Miscellaneous](#) on page 44

JasperGold Apps Command Reference Manual

Introduction

Table 1-1 What's New in 2015.09

New Features and Enhancements for 2015.09

New Release: JasperGold Coverage Unreachability App

JasperGold Apps version 2015.09 includes the new Coverage Unreachability (UNR) App, which uses a combination of both formal and simulation techniques to determine whether the uncovered items are unreachable. It is built on assertion-based verification (ABV) but requires very little knowledge of ABV, and in turn, significantly helps in reducing the verification time and effort.

Note: With this app, it is assumed that you have already generated the simulation coverage database.

The UNR App reads uncovered expressions, blocks, or toggles from the simulation coverage database and generates automatic assertions or covers for the uncovered items. The tool adds these assertions by default in this flow. You will generate the formal coverage database using the `-covgen` switch. The tool then merges the generated formal database with the simulation database so that the merged database contains the coverage hit count from the simulation database and unreachable information from formal. You can then load the merged database into the Incisive Metrics Center (IMC) tool to view the coverage results. The Coverage Unreachability App marks unreachable cover items as UNR.

To launch the app, use the command `jaspergold -unr -gui`.

You can access documentation for this newly released app in the following places:

- From the *Help* menu:
 - *Mini Guides – Coverage Unreachability App User's Guide*
 - *User Guide*
 - *Command Reference Manual*
 - *Tcl Command Help*
- On the command line: `help check_unr -gui`

IMPORTANT: Apps are licensed individually. Therefore, not all apps or their related commands may be accessible. Consult your Cadence representative for additional information about JasperGold Apps licensing.

Related command: [“check_unr”](#) on page 432

JasperGold Apps Command Reference Manual

Introduction

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Connectivity Verification App

Usability Improvements for Connectivity – Phase II

JasperGold Apps version 2015.06p001 included several Connectivity usability improvements designed to alleviate confusion over the various verification indicators in the Connections Table. In this phase of the usability improvements, you will see the following:

- The toolbar's *Conn. Setup* button group now includes a button *Generate Toggle Cover Properties*.
- When you press the button, the tool creates toggle checks and a gray icon appears in the *Toggle* column of the *Connections Table*.
- When you press the button *Prove Connectivity Properties*, the toggle checks status indicator replaces the gray icon.

Refer to the “Overview” chapter of the user’s guide for additional information.

Enhanced Reverse Connectivity

When extracting connections involving multiple driven signals without including the actual drivers, Reverse Connectivity is now able to extract all conditions.

Enhanced Flow for Comparing a Connectivity Map to the Implemented Design

The command `check_conn -compare_to_rtl` now compares clock information. Clock declarations are considered equal if either of the following is true:

- Both parity and clock signals are the same.
- Parity is the same and clock signals have the same factor and phase relative to the fastest clock.

Related command: [“check_conn”](#) on page 241

Listing Signals in a Connectivity Candidate's Path

The command `check_conn -candidate <candidate> -list` now supports the argument `path`, which lists the signals in the candidate's path, including the source and destination.

Related command: [“check_conn”](#) on page 241

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Coverage App

Coverage App Add Waiver Dialog Now Includes a Tag Field

With version 2015.09, the Coverage Add Waiver dialog includes a *Tag* field.

Enhanced Expose Cover Items Dialog

With this release, the “Expose Cover Items” dialog provides the option to expose more than one task. Click the drop-down menu under *Task Name* and select the tasks you want to expose.

Enhanced Remaining Coverage Reports to Print Source Code Excerpt

Now, all Coverage reports print source code information when you use the `check_cov -report... -extended_format` option. Also with this release, you can use the new `-source_code` switch with `check_cov -get_cover_item_info` to return source code information.

Related command: [“check_cov”](#) on page 258

Coverage App No Longer Generates “if_stmt” Cover Items by Default

With this release, the Coverage App no longer generates `if_stmt` cover items by default. Going forward, use the `-include_if_stmt_cover_items` switch with `check_cov -init` to include these cover items.

Related command: [“check_cov”](#) on page 258

Enhanced Summary Reporting of Instance Names

Now, instance names in summary reports are complete as expected.

Enhanced Cover Item Naming Convention

With this release, the cover item name is no longer suffixed with `-COVER_ITEM_<cover_item_id>`. See the `check_cov -get_cover_id -name` command documentation for additional information on the cover item naming convention.

Related command: [“check_cov”](#) on page 258

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Enhanced Behavior of the Coverage “-summary” Switch

With this release, the `-summary` switch returns a report with the following information per instance:

- Number of cover items in the report
- Total number of cover items
- Number of excluded cover items

The `-summary` feature has the following limitations:

- It is applicable to the reachable, COI, and proof_core reports only. If you use the `-summary` switch for any other report type, it returns an error.
- It is applicable to the complete design only, that is, you cannot use switches like `-assert`, `-filter_in`, and `-filter_out` with `-summary`.

Note: You can use the `-exclude` and `-task` switches in conjunction with `-summary`.

GUI changes related to this enhancement include the following:

- The source browser contains three new columns, *Stimuli Coverage*, *COI Coverage*, and *Proof Coverage*, which correspond to the summary information for reachable, COI, and proof_core. Populate these columns from the Coverage Report Setup dialog by clicking the *Summary* check box. Once populated, the column contents remain unchanged until you generate another summary report from the GUI.
- The *Cover Items Table* includes two new columns, *COI Coverage* and *Proof Coverage*, which show `-`, `IN`, or `OUT`. This value indicates whether the cover item is in the COI/proof core on a design-level basis, that is, over the set of all assertions in the Coverage database.

Related command: [“check_cov”](#) on page 258

Coverage Now Offers an Option to Exclude Bind Statements

With this release, you can use the new switch `-exclude_bind_hierarchies` at initialization to exclude bind statement cover items. This option is also available from the *Advanced* tab of the Enable Coverage Models dialog.

Related command: [“check_cov”](#) on page 258

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Cover Items Generated from `celldefine Blocks Now Excluded by Default

With this release, cover items generated from `celldefine blocks are excluded by default. Use the new switch `-include_celldefine_cover_items` with `check_cov -init...` to include these cover items.

Note: You can also access this option for the *Advanced* tab of the Enable Coverage Models dialog.

Related command: [“check_cov”](#) on page 258

Task and Function Cover Items Extracted by Default

Now, Coverage extracts task and function cover items by default.

CSR Verification App

New CSR Verification App Definitions Viewer

The CSR Verification App console now includes the *Definitions Viewer* pane. In this pane, you will find the following components:

- *Definitions Tree*
- *Registers Table*
- *Fields Table*
- *Attributes Table*

You can filter the tables by name (first column) and use the context menu (right-click) to choose *Prove*, *Visualize*, or *Plot Debugging Signals*. Click on a node in the tree to update the three tables with context sensitive views and synchronize the *Property Table* selection.

Note: The three tables are for information display only. They do not allow selection.

Refer to the “Overview” chapter of the user’s guide for additional information.

JasperGold Apps Command Reference Manual

Introduction

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Formal Property Verification (FPV) App

Control over Enabling and Disabling Related Covers

By default, the tool automatically disables all enabled related covers if you disable their main properties. JasperGold Apps now provides the configuration command `set_auto_disable_related_covers` to change the default behavior.

Related command: [“`set_auto_disable_related_covers`” on page 952](#)

Low Power Verification (LPV) App

Improved Support for Low Power SEC Verification

The user interface for Low Power SEC verification includes the following changes:

- The `check_lpv` command supports a new subcommand `-sec`. This subcommand takes one of two options, `-spec` or `-imp`. These options specify that the GUI is populated with information for either the “spec” or “imp” side. The *Power Domain Info Browser* contents change to show the selected side (spec or imp). Also, the *Design Hierarchy* pane updates such that the power domains display for the corresponding design (spec or imp).

See command documentation for `check_sec -spec_upf_file` and `check_sec -imp_upf_file` or for `check_sec -spec_cpf_file` and `check_sec -imp_cpf_file`.

- To make it easier to distinguish the selected database (spec or imp), the *Power Domain Info Browser* now shows the name of the top level power format file in its header. It also now shows all power format files loaded for a particular database in the power intent tree.
- Power information such as power domains and corruption highlighting in the Visualize window are also based on the currently selected SEC design. This enhancement will be improved such that both the spec and imp signals show power information simultaneously.

Related commands:

- [“`check_lpv`” on page 326](#)
 - [“`check_sec`” on page 345](#)
-

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Enhanced UPF Support

JasperGold Apps now supports the following UPF commands:

- `set_isolation -exclude_elements`
- `create_power_domain -elements { . }`

Note:

- The `create_power_domain -elements { . }` construct is new to UPF 2.1.
 - For the complete list of supported commands, refer to the app note on Cadence Online Support – JasperGold Apps product page – Application Notes – JasperGold UPF Support.
-

Enhanced CPF Support

JasperGold Apps now supports the default save and restore switches for `create_power_domain`.

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Sequential Equivalence Checking App

Improved Command Consistency for the SEC Verification App

In an effort to make the SEC Verification App commands more intuitive and consistent with other JasperGold commands, you will find the following changes in this release.

- `check_sec -set_proof_settings` is deprecated. The SEC Verification App now respects the generic `set_prove_*` commands. As a result, some `-set_proof_settings` commands became obsolete and others are replaced with `check_sec -prove` switches to control a specific proof and `set_sec_prove_*` commands to configure all subsequent SEC proofs. The following is a list of new configuration commands. Each command has a complementary `get_*` command:
 - ❑ `set_sec_autoprove`
 - ❑ `set_sec_autoprove_cutpoints_on_internal_signals`
 - ❑ `set_sec_autoprove_cutpoints_on_latches`
 - ❑ `set_sec_autoprove_max_number_of_cutpoints`
 - ❑ `set_sec_autoprove_mode`
 - ❑ `set_sec_prove_assumption_lifting`
 - ❑ `set_sec_prove_levelize_helpers`
 - ❑ `set_sec_prove_order_targets`
 - ❑ `set_sec_prove_suppress_traces`
- The following `check_sec -prove` switch names have changed:
 - ❑ `-sort_helpers` -> `-order_targets`
 - ❑ `-mode` -> `-autoprove_mode`
 - ❑ `-with_latches` -> `-cutpoints_on_latches`
 - ❑ `-auto_internal_cutpoints` ->
`-cutpoints_on_internal_signals`

Related commands: “[check_sec](#)” on page 345

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Improved Performance for SEC Verification App

This release includes enhancements that provide a better convergence time for big designs.

Enhanced Report for the SEC Interface Check

The SEC interface check (`check_sec -interface`) reports on the outcome of mapping, and with this release, the report has been expanded as follows:

- For each signal type it tells whether the check passed or failed
- For failures, it reports how many signals of the specified type in the spec and imp are unmapped.

Note: This enhancement applies to `check_sec -interface` with no additional switches.

Related commands: [“`check_sec`”](#) on page 345

Changing DUT Instances after SEC Setup

It is now possible to change the DUT instances after setup with the command `check_sec -set_dut_instance -spec <spec_dut_list> -imp <imp_dut_list>`.

Related commands: [“`check_sec`”](#) on page 345

SEC Support for Specifying Multiple Instances as DUTs

The `check_sec -setup` switches `-spec_dut` and `-imp_dut` now accept a list of DUT instances.

Related commands: [“`check_sec`”](#) on page 345

Structural Property Synthesis (SPS) App

Enhanced Structural Properties Table Filtering Options

Now, you can filter the *Structural Properties* pane by source location as well as definition. Click the new *Change Filter Column* button at the top right of the *Structural Properties* pane and select either *Filter by Definition* or *Filter by Source Location*.

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Adjusting the Depth or Length of SVA Expressions for SPS

With this release, you can use the following commands to adjust the maximum depth (in terms of gate complexity) or length (in number of characters allowed) of SVA expressions:

- `set_sps_sva_max_depth <N>`
- `set_sps_sva_max_length <N>`

The default value is 15 for max depth and 1000 for max length. Use `get_sps_sva_max_depth` and `get_sps_sva_max_length` to return current values.

Related commands:

- [“check_sps_configure” on page 410](#)
 - [“get_sps_sva_max_depth” on page 1275](#)
 - [“get_sps_sva_max_length” on page 1276](#)
 - [“set_sps_sva_max_depth” on page 1275](#)
 - [“set_sps_sva_max_length” on page 1276](#)
-

File Extensions No Longer Appended to Exported SPS Files

With this release, SPS report and waiver export commands no longer automatically append the file name with .txt, .html, or .xml extensions. Instead, the exact name specified is the generated file name.

This change requires that you update your scripts to maintain compatibility. For example, if a script uses `check_sps_waiver -export -file_name filename`, replace `filename` with `filename.txt` to maintain persistency or the subsequent import command will fail.

Related commands:

- [“check_sps_configure” on page 410](#)
 - [“export” on page 533](#)
-

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Enhanced SPS to Filter Deadcode Checks when Case is Full

With this release, you can use `set_sps_deadcode_full_case_filter true` to prevent the generation of deadcode checks for case defaults when the case is already full.

Related command: “[set_sps_deadcode_full_case_filter](#)” on page 1263

Security Path Verification (SPV) App

Excluding Control Paths from the SPV Analysis

With this release, you can create SPV properties that automatically exclude control paths from the analysis. If you want to exclude control paths for all properties, set the new configuration variable `spv_include_control_path` to `off`. If you want to exclude control paths for select properties only, use the new switch

`-exclude_control_path` with `check_spv -create`. The `-exclude_control_path` switch is effective only when `set_spv_include_control_path` is `on`, which is the default.

Note:

- If you have configured the tool to exclude control paths for all properties (`set_spv_include_control_path off`), you can use `-include_control_path` with `check_spv -create` to override the configuration for a given property.
- For additional details, consult the command documentation.

Related commands:

- “[check_spv](#)” on page 421
 - “[get_spv_include_control_path](#)” on page 1277
 - “[set_spv_include_control_path](#)” on page 1277
-

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Enhanced SPV Graphs Reflect Property Type

Now, you can choose whether an SPV property analyzes all paths, including control signals, or just data paths. If you choose to include control paths, which is the default, SPV property graphs will be signal graphs, which include all nodes from source to destination. If you choose to exclude control paths, SPV property graphs will be data path graphs only.

Note:

- The SPV proof flows also generate graphs that reflect whether the property was created to analyze control paths.
- For additional information, see the documentation for the following commands:
 - `check_spv -create... [-include_control_path | -exclude_control_path]`
 - `set_spv_include_control_path`

Related commands:

- “[check_spv](#)” on page 421
 - “[set_spv_include_control_path](#)” on page 1277
-

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

ABVIP

Enhancements for Visualize Debugging and Indexed Behavior Tables

With this release, Visualize debugging tables for ABVIP are no longer loaded by default. Instead, the Visualize *Debugging Table* includes a *Load debugging tables* button so that you can load these tables on demand. You can also set the tool to automatically load debugging tables using one of the following options:

- Click the *Load automatically* option, which is to the right of the *Load debugging tables* button.
- Set the new `visualize_auto_load_debugging_tables` variable to on.

Also with this release, the Visualize *Indexed Behaviors* table *Analyze* button and *Analyze automatically* check box have been replaced with an icon, but the functionality has not changed.

Related commands:

- [set_visualize_auto_check_props](#) on page 1303
 - [set_visualize_auto_load_debugging_tables](#) on page 1305
-

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Proof Accelerators (PA)

New `jasper_scoreboard_3` PA

This release introduces the Jasper PA `jasper_scoreboard_3`, which is an encrypted verification component used to prove the integrity of data transferred from an initiator to a target through the design under verification (DUV). You can use this PA to verify functionality such as data transfer across a bus bridge, serial-to-parallel conversion, and simple point-to-point data integrity across a FIFO.

In conjunction with this new PA, you can use the `jasper_scoreboard_3` command to load basic configuration definitions to be used with the PA's instantiations.

For additional details about this new PA, review the datasheet from the GUI *Help* menu – *Proof Accelerator Datasheets – jasper_scoreboard_3* or run the command `proof_accelerator -doc jasper_scoreboard_3`.

Related commands:

- “[jasper_scoreboard_3](#)” on page 727
 - “[proof_accelerator](#)” on page 739
-

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Abstraction

New Default Behavior and Functionality for “abstract -reset_value”

IMPORTANT: CHANGE IN DEFAULT

For consistency with the other abstract commands, abstract -reset_value is now task-based by default. It applies to the current task unless you specify a different task with the -task switch. To apply the abstraction to the environment, which was the former default, use -env.

The command abstract -reset_value supports the -bound switch to provide additional control over when the abstraction is valid.

- Use -bound with a value other than the default \$ (infinity) if you want to apply the abstraction during proof until the specified cycle.
- If you want to apply the abstraction for the reset analysis alone, use -env and the bound value 0.

In addition, this command now supports the switch -silent to suppress info messages.

Related command: [“abstract” on page 146](#)

New Command for Register Value Abstraction

A new command named abstract -register_value replaces a signal's driving logic after reset simulation with a driving expression. As an option, you can specify a bound to apply the command up to N cycles after a proof starts.

Improved Performance for “abstract -cdc -find”

This release includes improved performance (in terms of run times) for abstract -cdc -find.

Related command: [“abstract” on page 146](#)

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Clocks and Reset

New Feature: Clock Information Window

This release introduces the Clock Information GUI, which provides a centralized location for clock environment information with options for efficiently exploring the environment and debugging and resolving clocking issues. Use this window to add new relevant information, see a list of related registers, properties, and clock aliases; run sanity checks; and modify the global clock environment.

Access the GUI from the Window menu or with the command `get_clock_info -gui`.

Related command: “[get_clock_info](#)” on page 558

Expanded Support for Declaring Relative Clocks

In previous versions of JasperGold Apps, if you wanted to declare a clock relative to another clock, the base clock was required to be the fastest clock. With this release, that requirement is lifted.

Related command: “[clock](#)” on page 455

Support for Multi-Bit Clock Signals

JasperGold Apps now supports multi-bit signals for clock declarations with just one command. That is, it is not necessary to declare a single bit per command.

Related command: “[clock](#)” on page 455

Support for Clocks Rated on Both Edges

Now it is possible to add a rated input sensitive to both edges of the clock. With this enhancement in support, the switch `-input_change_both_edges` has been deprecated. Instead, you can use the command `clock -rate -both_edges`.

Note: The complementary GUI option no longer appears in the Clock GUI.

Related command: “[clock](#)” on page 455

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Dumping and Loading VCD Files with Internal Signals

It is now possible to dump and load VCD files that contain internal signals. Use the switch `-include_internal` in the commands `visualize -save -init_state`, `visualize -save -reset_sequence`, `get_reset_info -save_reset_vcd`, `reset -sequence`, and `reset -init_state`.

Note: Loading files with the switch `-include_internal` carries a risk of a mismatch in internal signals that can lead to false proof results in some cases. Internal signals in reset files might not be the same as those generated by a different JasperGold release version or a different version of the design or a different design setup and configuration. (Design setup and configuration differences include the order in which files are passed to `analyze` and `elaborate` and the order in which assumptions and assertions are created.)

Related commands:

- [“get_reset_info” on page 660](#)
 - [“reset” on page 761](#)
 - [“visualize” on page 864](#)
-

Enhanced “sanity_check” Command

Now, `sanity_check -analyze simple_reset` identifies simple AND reset expressions that have not been properly defined through the `reset` command. In the list of reset conditions, look for SIMPLE_AND_EXPRESSION to identify these problematic conditions.

Related commands:

- [“reset” on page 761](#)
 - [“sanity_check” on page 782](#)
-

“reset” Support for Tcl Lists

The commands `reset -expression` and `reset` now accept a Tcl list as an argument.

Note: To accept a Tcl list, the number of arguments used in this command must be one. Otherwise, the command behaves as previously defined.

Related command: [“reset” on page 761](#)

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Sorting “get_reset_info” Returns

The command `get_reset_info` now supports the switch `-sort_by`. Use this switch to sort the Tcl return items according to the sorting criteria you specify:

- `type`: sort alphabetically by register type (for example, D flop)
- `cycle`: sort the Tcl return according to the number of reset cycles it took a register to initialize, from greatest to least
- `module`: sort alphabetically by the name of the module where the register is instantiated
- `instance`: sort alphabetically by the name of the instance where the register is instantiated

Related command: [“get_reset_info” on page 660](#)

Saving Reset Values

The command `get_reset_info -save_values` now supports the switch `-include_encrypted`, which captures reset values of encrypted flops in an encrypted file.

Note: The command `reset -save_values` is deprecated to reserve the `reset` command for its true purpose: to specify the reset condition.

Related command: [“get_reset_info” on page 660](#)

Obsolete Command: “set_reset_save_cycle”

The commands `set_reset_save_cycle` and `get_reset_save_cycle` are no longer necessary since the tool now saves reset cycles by default.

Note: Your legacy scripts are not affected at this time. If you use these commands, the tool simply prints a warning. However, at some point in the future, the command will be completely retired and using it will trigger an error.

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Design Exploration

Improved Black-Box Module and Instance Info

The `get_design_info` reports for black-boxed modules and instances has been improved for this release. Information about whether a module or instance is a black box is now readily available, and the command no longer shows unrelated information.

Related command: [“get_design_info” on page 564](#)

Listing Signals with Multiple Drivers

It is now possible to list signals with multiple drivers. Use the command `get_design_info -list` with the default scope, `-instance` scope, or `-module` scope and the argument `multiple_driven`.

Example:

```
get_design_info -list multiple_driven
```

Related command: [“get_design_info” on page 564](#)

Listing VHDL Constants and Enums with “get_design_info”

The command `get_design_info` now supports arguments to list VHDL constants and enums in the top instance, specified instance, and specified module scopes:

- `get_design_info [-instance | -module] -list constant`
- `get_design_info [-instance | -module] -list enum`

Related command: [“get_design_info” on page 564](#)

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Avoiding Specified Pins When Getting Fanin

The command `get_fanin` now supports the switch `-avoid_pin`. Use this switch with one or more of the following options to prevent `get_fanin` from going through the specified pins:

- `flop_async_reset`
- `latch_enable`
- `mux_enable`

Related command: [“`get_fanin`”](#) on page 623

Specifying Preferences for the Why Feature

The Visualize GUI now supports the following configuration options for the Why feature:

- Ignore connected assumptions avoids setting a signal to a certain value at a certain cycle, and as a result, improves performance.
- Simple buffer transitive goes through simple buffers and stops only in combinational logic.
- Stop at hierarchy change stops Why when it reaches an instance barrier so you can step through. (Without this option, Why goes right through the buffer.)

To set these preferences, use the *Edit* menu's *Preferences* option and go to the *Visualize* tab in the Preferences dialog or click on the *Why Settings Control Box* toolbar button located adjacent to the *Trace Explanation* combo box.

In addition, `visualize -why` now accepts the following switches:

- `-simple_buffer_transitive` enables transitivity through simple assignments.
Note: Concatenations, bitwise assignments, and logical inverters are not considered simple assignments.
- `-stop_at_hierarchy_change` disables “default” transitivity through the “first” hierarchy change (instance port crossing).

Related command: [“`visualize`”](#) on page 864

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Design Synthesis

Elaborating Designs with Unbuffered Concatenation

The `elaborate` command now supports the switch `-unbuffered_concatenation`, which avoids buffering concatenation elements in Verilog. You can use this switch **with caution** if Why buffers are breaking constant propagation and preventing the tool from elaborating the design.

CAUTION: If you use this switch, Why results will not highlight the values inside concatenates.

Related command: [“elaborate” on page 501](#)

Synthesis of prev/stable PSL Operators

The synthesis of prev/stable PSL operators now uses wide DFFs in JasperGold Apps.

Improved Usability for the Property Table “Type” Filter

The Property Table *Type* column filter panel now contains additional group filter check boxes for *Assert(all)*, *Cover(all)*, and *Assume(all)*. In addition, you can click the button *Show Details* to specify a finer-grained filtering scheme.

Language Mode for Tcl Expressions in the CPI Flow

In the CPI flow, the tool now uses the HDL language of the top module to determine the syntax mode for Tcl expressions. In addition, `elaborate -load_cpi` now supports the switch `-mode` to change the default syntax mode for Tcl expressions.

Related command: [“elaborate” on page 501](#)

Controlling Black Boxing for Arithmetic Operators in the CPI Flow

The CPI flow now supports control over black boxing for arithmetic operators (multiplier, divisor, modulo, and power). Operators are black-boxed if their size (sum of output bits) exceeds a threshold, which is 1 by default. However, you can change the default with the `elaborate -load_cpi` switches `-bbox_mul`, `-bbox_div`, `-bbox_mod`, and `-bbox_pow`.

In addition, `elaborate -load_cpi` now accepts `-disable_auto_bbox` to disables automatic black boxing for arithmetic operators.

Related command: [“elaborate” on page 501](#)

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Syntax Violation Warning is Now an Error

The frontend warning VERI-1691 (IEEE 1800 syntax violation) is now issued as an error. To revert to the old behavior use the following command:

```
set_message -warning VERI-1691
```

Related command: [“set_message”](#) on page 829

Downgrading Message Severity for Frontend Warnings

It is now possible to downgrade the following warnings:

- VERI-2138
- VERI-2139
- VERI-2140

Example: `set_message -info VERI-2139`

Related command: [“set_message”](#) on page 829

New Downgradable Error for setting “default_netttype” to “none”

JasperGold Apps now issues the following error if `default_netttype` is set to none for an implicit net type of input ports.

```
[ERROR (VERI-1906)] test.v(3): net type must be explicitly specified for 'in'  
when default_netttype is none (VERI-1906)
```

It is possible to downgrade this message to a warning using the following command:

```
set_message -warning VERI-1906
```

Related command: [“set_message”](#) on page 829

Enabling Bit Width Mismatch Warnings

JasperGold Apps now issues a warning for bit width mismatches on comparisons. This warning is disabled by default. To enable it, use:

```
set_message -warning VERI-9004
```

Related command: [“set_message”](#) on page 829

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Frontend Message Renumbering for VERI-9006 and VERI-9007

Verific now uses messages VERI-9XXX for Cadence-specific messages. To avoid clashes, we have changed two JasperGold internally created 9000 messages to 8000 messages. If you currently have scripts that refer to messages VERI-9006 and VERI-9007, you will need to adjust your scripts accordingly:

- VERI-9006 is now VERI-8000
 - VERI-9007 is now VERI-8001
-

Adding Verilog and VHDL Packages

In addition to supporting Verilog packages, JasperGold Apps now supports VHDL packages. Use the command `elaborate` with `-add_package top` to specify that the packages are children of the top module. This argument is the default if you specify `-add_package` without additional arguments.

Use `-add_package root` to add the `$root` instance as the root of the instance tree and make the packages children of the `$root` instance.

With this expanded support for packages, the switch `-add_verilog_packages` is being deprecated and will be retired at some point in the future.

Note:

- `elaborate -add_package` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.
- Using `-add_package` in `-f` files is prohibited. It triggers an error.

Related command: “[elaborate](#)” on page 501

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Proofs and Visualize

Stopping a Proof after N Unreachable Covers

The new configuration variable `stop_on_unreachable_limit` directs the `prove` command to stop once it has changed the status of N properties from undetermined to unreachable.

When the value of this variable is 0, which is the default, there is no restriction on the number of unreachable statuses the `prove` command is allowed to produce.

Related commands:

- [get_stop_on_unreachable_limit](#) on page 1284
 - [set_stop_on_unreachable_limit](#) on page 1284
-

Enhanced Capability for “`get_decomposed_expression`”

The command `get_decomposed_expression` was released with version 2015.06, and with this release, the command supports Verilog conditional operator (?) together with associated comparison operators (==, !=, ===, !==, <, >=, >, <=). In addition, this command now supports hierarchical references in properties. This expanded support resolves the Java error that these cases triggered prior to this release.

Related command: [“`get_decomposed_expression`”](#) on page 560

Dumping Traces

Version 2015.06 added switches to the `prove` command to dump traces to the disk as soon as the engines find them. This release includes support for the SHM format, which is now the default.

In addition, this release includes `-dump_trace` for the `autoprove` command.

Refer to the command documentation for additional options to specify file format and dump directory.

Related commands:

- [“`autoprove`”](#) on page 207
 - [“`prove`”](#) on page 741
-

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Saving Traces in SHM Format from the Visualize Window

JasperGold Apps 2015.06p001 added support for saving Visualize traces in SHM (SST2) format with the command `visualize -save -shm....`. This release adds the GUI counterpart (*File – Save VCD/SHM/FSDB*).

Note:

- When saving the SHM, you must specify an existing directory or create a new one.
- If the directory contains other files, a confirmation dialog opens asking if you want to overwrite the existing directory.

Related commands: “[visualize](#)” on page 864

Improved Visibility for Source Navigation Buttons

To enhance usability, the color of source browser navigation icons (*Back, Forward, Previous, Next*) has changed to a brighter and deeper blue.

Refer to the “Visualize” chapter in the *JasperGold Apps User’s Guide*.

Visualizing Differences between Traces in a Stack

Use the new command `visualize -highlight -diff with -prev_plot` or `-next_plot` to compare contiguous traces in a stack and add yellow highlights to show the cycles that differ. And if the current trace has more cycles than the one you are comparing it to, the tool applies orange highlights to the additional cycles.

In the GUI, use the *Go Forward* and *Go Backward* buttons. By default, the tool highlights the differences between the two plots. Use *Edit – Preferences – Visualize* to change the default. Or to change the default on the fly, click the down arrow located between the *Go Backward* and *Go Forward* buttons to reveal the check box control. Refer to the “Visualize” chapter in the *JasperGold Apps User’s Guide*.

Related command: “[visualize](#)” on page 864

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Improved Performance for Highlight Differences

This release includes optimizations to improve the performance (in terms of run times) of the Highlight Differences feature. In addition, it is now possible to interrupt the command.

Refer to the “Visualize” chapter in the *JasperGold Apps User’s Guide*.

Related command: [“visualize”](#) on page 864

Enhanced Behavior for “visualize -save”

The command `visualize -save` no longer dumps bit-blasted signals by default. Use `-bit_blast` to get the former default behavior.

Related command: [“visualize”](#) on page 864

Improved Visualize Support for FSDB

Visualize is now able to load FSDB traces from pure combinational circuits.

Visualize “-load” and “-confirm” with “-batch”

The commands `visualize -load` and `visualize -confirm` now support the switch `-batch`. Use this switch in scripting to improve performance by preventing the tool from displaying the trace it generates.

Related command: [“visualize”](#) on page 864

ProofGrid

Out-of-the-Box Productivity Boost for ProofGrid

The `per_engine_max_jobs` default has been increased from 1 to 2. This means, for example, if you are using the default `engine_mode {Hp Ht N B}`, the tool uses a total of 6 jobs instead of 4.

Related command: [“set_proofgrid_per_engine_max_jobs”](#) on page 1141

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Plots for Engine Job Restarts

For version 2015.06 we reported that in the ProofGrid Manager's "Tool Time" mode, when you restart an engine job, the plot continues onward from the "restart" point with a marker close to the x-axis that indicates that point. However, the implementation of the feature required further refinement to accurately represent the actual restart point for engines, and that refinement is included in this release.

Refer to the "ProofGrid Manager" chapter in the *JasperGold Apps User's Guide*.

Usability

Customizing GUI Buttons

JasperGold Apps now supports customer-created GUI buttons. Use the new command `custom_gui_action` to specify the following:

- Target window
- Button icon
- Script the button will run
- Tooltip the tool displays during a mouse-over

You can create buttons for the main GUI console and Visualize windows. Refer to the command documentation for additional information.

Note: This feature is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

Related command: ["custom_gui_action"](#) on page 480

BPS, SPS, and Coverage Mini Guides Now Available from Tool

With this release, you can access the BPS App user's guide, the SPS App user's guide, and the Coverage App user's guide from the menu *Help – Mini Guides*.

JasperGold Apps Command Reference Manual

Introduction

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Using “help” to Display a Command’s Help in the Browser

The JasperGold Tcl command `help` now supports launching the JasperGold Help Browser with the focus on a specified command:

```
help -gui <command_name>
```

Note: If you enter an invalid command, the tool issues an error.

Related command: [“help”](#) on page 687

Hyperlinks between Messages and Source Code

Frontend INFO, WARNING, and ERROR messages (related to `analyze` and `elaborate` commands) are now hyperlinked to the file name’s line number. The relevant message string appears blue in color, and a mouse-over changes the cursor to a “clickable” icon. Click on the hyperlink to open the Source Browser window with its focus on the line number and file name specified in the hyperlink.

In addition, you can set your preferences to open the file (with the specific line selected) in the external editor instead of the Source Browser window (*Edit – Preferences – Console – Open message hyperlink in external editor*).

Refer to the “Overview” chapter and “GUI Settings” appendix in the *JasperGold Apps User’s Guide*.

Miscellaneous

Expanded SystemVerilog Support

JasperGold Apps now supports generic interface references in module instantiation.

Expanded Support for Source Nodes

Both `cover -path` and SPV properties now accept internal signals as source nodes.

Related commands:

- [“check_spv”](#) on page 421
 - [“cover”](#) on page 471
-

Table 1-1 What's New in 2015.09, continued

New Features and Enhancements for 2015.09

Upgraded FlexNet License Server Components

JasperGold Apps now uses version 11.13.0.3 of the FlexNet Publisher licensing from Flexera Software, which is included in the product package. If you use a FlexNet Publisher license server other than the one that ships with the tool, it must be version 11.13.0.3 or newer. And if you have specified a path to the vendor daemon on a VENDOR line in the license file, you must change it so it points to a supported version of the daemon.

Note: This upgrade includes FlexNet license server components (lmgrd and cdslmrd) and lmutil.

Refer to [“Using the Correct License Server”](#) on page 62.

Limiting “Initial Release” Warnings in the Console

The command `set_message` now supports the following two new switches:

- Use `set_message -limit_initial_release_warning` if you want the tool to print a specific `initial_release` message only once in the console.
- Use `set_message -unlimit_initial_release_warning` to disable the previous limitation.

Related command: [“set_message”](#) on page 829

Reference Manual Organization

This command reference manual is organized as follows:

- [Chapter 1, “Introduction”](#)
Lists new and enhanced features in the current release, introduces knowledgebase resources, provides a basic overview of JasperGold Apps Tcl, and defines the conventions used in JasperGold Apps documentation.
- [Chapter 2, “Installation”](#)
Lists platform and language support and describes product and license installation procedures.
- [Chapter 3, “Commands and Variables Overview”](#)
Lists JasperGold Apps environment variables and provides an overview of JasperGold Apps commands with graphical user interface (GUI) equivalents.

JasperGold Apps Command Reference Manual

Introduction

- [Chapter 4, “General Commands”](#)

Documents JasperGold Apps commands, including command options, definitions, and examples. This chapter does not include the `get_*` and `set_*` commands.

- [Chapter 5, “Configuration Commands”](#)

Documents JasperGold Apps `get_*` and `set_*` commands, including command options, definitions, and examples.

- [Appendix A, “PSL Support”](#)

Lists JasperGold Apps support for PSL.

- [Appendix B, “SVA Support”](#)

Lists JasperGold Apps support for SVA.

- [Appendix C, “Liveness vs. Safety”](#)

Details the difference between liveness properties and safety properties and provides an example.

- [Appendix D, “Tcl Support for SVA Expressions”](#)

Lists JasperGold Apps support for SVA expressions in Tcl commands.

- [Appendix E, “OVL Support”](#)

Lists JasperGold Apps support for OVL.

- [Appendix F, “IP-XACT Support”](#)

Lists JasperGold Apps support for IP-XACT.

- [Appendix G, “Proof Setup”](#)

Lists proof and Visualize™ setup commands and indicates whether they apply to the global or task environment.

- [Appendix H, “Custom Radix Files”](#)

Provides guidance for formatting custom radix files.

- [Appendix I, “Automatically Extracted Type Properties”](#)

Describes JasperGold Apps support for VHDL and SystemVerilog type properties and includes procedural notes and limitations.

- [Appendix J, “Vacuity Checking with Automatically Extracted Covers”](#)

Introduces proof vacuity checking features and includes examples and guidance for managing vacuity checking.

- [Appendix K, “Understanding Visualize Semantics with SVA Equivalents”](#)

Provides SVA examples intended to explain the meaning of various Visualize commands.

Related Documentation

This book has a companion book, the *JasperGold® Apps User’s Guide*, which introduces the apps and related windows. You can access this book and the companion user’s guide from the tool’s *Help* menu. Other resources available from the JasperGold Apps *Help* menu include the following:

- Stand-alone PDF mini guides that describe the Visualize GUI, guide engine selection, and so forth.
- Proof Accelerator data sheets
- HTML guides for the various apps
- Searchable HTML Tcl command help

Note: You can also type `help` on the command line for a list of all commands or use `help <command_name>` for documentation on a specific command.

In addition, Cadence offers Cadence Online Support, a searchable knowledgebase that includes release version information, quick tip videos, answers to frequently asked questions, application notes, manuals, tutorials, and other resources. A sample of related documents available from the “Manuals” page follows:

- *JasperGold Visualize GUI Features* – provides information about color codes, highlighting and annotation features, and procedures for manipulating traces
- *Structural Property Synthesis App User’s Guide* – provides an overview of the console, instructions for setting up the database and loading the design, and a step-by-step exploration of the basic flow

The “Application Notes” page of the knowledgebase includes introductions to PSL and SVA, design-specific methodology tips, formal verification strategies, example source code for high-level requirements modeling, example discussions on various common functional components, and so forth. Reviewing these application notes can help you learn various techniques and strategies for handling verification complexity that you can generalize and apply to your specific design. A sample of available application notes follows:

- *X-Propagation Verification with JasperGold Apps* – provides setup advice, including a script you can run on the design example that is packaged with JasperGold Apps
- *SystemVerilog ‘09 Construct Support* – lists supported constructs from the SystemVerilog, IEEE® Std 1800™-2009

New and updated resources are continually added to the site. This site requires a one-time registration. To register, go to Cadence Online Support (support.cadence.com) and follow the new-user registration instructions.

JasperGold Apps Tcl

Tcl is a simple programming language. Regardless of whether you have experience programming in Tcl, you can learn enough to write interesting Tcl scripts within a few hours. Refer to the application note “Scripting with Tcl in JasperGold Apps” for new-user tips and techniques to handle challenges, such as referencing objects in an HDL. This application note is available from Cadence Online Support (support.cadence.com) and from the menu option *Help – Mini Guides – Scripting with Tcl in JasperGold Apps*.

Note: A single # in any Tcl expression is treated as a “starting comment” tag.

UNIX Commands

During a JasperGold Apps session, you can run some UNIX® commands in one of the following ways:

```
% exec {<command>}  
% exec "<command>"
```

You can also run a process in the background with &; however, this command returns a pid with a dummy value, so you cannot use it to identify the process.

Example:

```
% exec "pwd"  
% exec xterm &
```

For commands with multiple arguments, specify each argument separately with any combination of pairs of curly braces or quotes, for example,

```
% exec {ls} "-l"  
% exec {ls} {-l}
```

Specify sub-commands inside square brackets, for example,

```
% exec {ls} "-l" [exec pwd]
```

Note:

JasperGold Apps Command Reference Manual

Introduction

- Commands that access the tty do not work. And in some cases, they can cause the tool to hang. For example, `top` must access the tty and, since `top` cannot open the tty, the tool generates an error message.
- If you have elaborated a large design, you might see degraded performance when running the `exec` command.
- You can start a process in the background from the Tcl prompt. Example:
`exec xterm &`

Conventions Used in This Manual

The following tables list conventions used in syntax and text.

Table 1-2 JasperGold Apps Syntax Conventions

Convention	Definition
Courier font	Indicates text you will type on the command line.
<code>-underscore_separation</code>	Indicates a command switch. Switches are not case-sensitive.
<code>[]</code>	Indicates optional arguments. Do not type the square brackets.
<code> </code>	Indicates a choice (a logical OR) among alternatives. Do not type the vertical bar.
<code>\</code>	The backslash character (\) at the end of a line indicates that the command you are entering continues on the next line.
<code>*</code>	Indicates the preceding argument appears zero or more times per command.
<code>+</code>	Indicates the preceding argument appears one or more times per command.
<code>''</code>	Indicates the enclosed character(s) should be explicitly included on the command line. Do not type the single quotation marks.
<code><Italics></code>	A command option that you will replace with a valid value. Do not type the angle brackets.
<code>()</code>	Used as a grouping convention. Do not type the parentheses. For example, parentheses in the following syntax indicate that if you use the <code>-bbox</code> option, you will follow it with either the <code>0</code> or <code>1</code> . <code>[-bbox (0 1)]</code>

JasperGold Apps Command Reference Manual

Introduction

Table 1-3 JasperGold Apps Text Conventions

Convention	Definition
Courier font	This style indicates: <ul style="list-style-type: none">■ Text you will type in GUI fields■ Commands and options■ File names and paths■ Code samples
<i>Italics</i>	User interface items such as button and field names.
<i>Menu – Option</i>	GUI command sequence. Example: <i>Help – Command Reference Manual</i> Meaning: Click on the <i>Help</i> menu and choose the option <i>Command Reference Manual</i> .
<u>Blue text</u>	Hyperlinked cross-reference. When you view the PDF version of JasperGold Apps manuals from a computer screen, click on the blue text to view related information.
→	Single-step procedure.

JasperGold Apps Command Reference Manual

Introduction

Installation

This chapter provides instructions for installing JasperGold Apps and licenses. It includes the following sections:

- [Supported Platforms](#) on page 54
- [Supported Languages and Standards](#) on page 54
- [JasperGold Apps Installation](#) on page 56
- [License Installation](#) on page 57
- [Optional Environment Variables](#) on page 65
- [Simultaneously Running Jasper and Cadence Licenses](#) on page 65

Supported Platforms

JasperGold Apps supports the following operating systems:

- Red Hat® Enterprise Linux® 5.x (32-bit and 64-bit)
- Red Hat Enterprise Linux 6.x (32-bit and 64-bit)
- SUSE® Linux Enterprise Server 11 (32-bit and 64-bit)
- CentOS 5.x (32-bit and 64-bit)
- CentOS 6.x (32-bit and 64-bit)
- Ubuntu® 14.04 LTS (32-bit and 64-bit)

Note: Builds are superimposable. That is, it is possible to have more than one build in the same installation, for example, 32- and 64-bit builds. Furthermore, by superimposing builds, you can take advantage of the ProofGrid™ feature with your multi-platform farm.

Java Memory Limits

JasperGold Apps attempts to adjust the memory heap size (which governs how much memory Java® can use) according to your system configuration. In certain extreme circumstances, it might run out of memory. To address that issue, JasperGold Apps automatically does one of the following at startup:

- If the machine is running a CentOS, RHEL, or SUSE 64-bit OS, it sets the heap size to 512MB, and the stack size remains at the default setting.
- If the machine is running a CentOS, RHEL, or SUSE 32-bit OS, it sets the heap size to 256MB, and the stack size remains at the default setting.

Supported Languages and Standards

JasperGold Apps supports the following languages and standards:

- Verilog® 95 (IEEE® Std 1364™-1995) and 2K (IEEE Std 1364-2001)
- VHDL (IEEE Std 1076™-1993), 2K (IEEE Std 1076-2002), and VHDL '08 (IEEE Std 1076-2008)
- Mixed-language (Verilog and VHDL)
- IEEE Std 1850™-2005 Standard for PSL: Property Specification Language:

JasperGold Apps Command Reference Manual

Installation

- PSL-Verilog
- PSL-VHDL
- SystemVerilog (IEEE Std 1800TM-2005, IEEE Std 1800-2009, and IEEE Std 1800-2012) and embedded SVA in SystemVerilog designs
 - Note:** Support for SystemVerilog 2012 is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.
- JTcl 2.60 (<https://jtcl.kenai.com/docs/jtclifferences.html>)
- IEEE 1685TM IP-XACT
- UPF 1.0 and UPF 2.0 (IEEE Std 1801TM-2009) and UPF 2.1 (IEEE Std 1801-2013)
- CPF 2.0 (Si2)
- LibertyTM library modeling standard (version 2014.09)

Note:

For the Behavioral Property Synthesis App and Structural Property Synthesis App, support for VHDL (and mixed-language) has the following known limitations:

- VHDL variables/signals declared inside unnamed processes are not supported. If present in the design, they may appear in POIs, but the tool will not generate properties for such variables/signals. The limitation is for declaration only; that is, variables/signals declared at a higher scope and used inside unnamed processes are supported.

Example:

```
...
process(rst, clk)
    variable data: unsigned(3 downto 0);
...

```

- User-defined integer types are not supported (other user-defined types, such as enum, are supported).

Example of user-defined integer type:

```
type smallint is range 0 to 31;
```

- VCD is a standard trace format for Verilog designs. However, some simulators, like VCSMX, support generating VCD from VHDL or mixed-language simulations. When using a VCD file from VHDL source, you must specify the optional switch `-vcd_lang vhdl` with the commands `scan -trace` and `waveform -import`. (See “[scan](#)” on page 793 and “[waveform](#)” on page 938 for command documentation.) In addition, the VCD format does not have width information for integers, although a VHDL design may have width for integers. VHDL integers/enums with width other than 32 bits will not generate any properties, and you may see “width mismatch” warnings.

JasperGold Apps Installation

This installation guide assumes you are installing JasperGold Apps under the `/usr/local` directory. If you choose to install the tool in any other directory, adjust the instructions by replacing `/usr/local` with your actual installation directory.

JasperGold Apps requires you to set one or more environment variables. How you set variables depends on the shell you use. In the following installation example, we assume the shell is `csh`. Adjust the installation steps based on your unique shell. Also, we recommend you set environment variables in the shell startup file (for example, `.cshrc` for `csh`, `.profile` for `sh`).

1. Download the JasperGold Apps tar kit from the Cadence software download site ([downloads.cadence.com](#)). Use your Cadence Online Support login. If you are not a registered user, sign up at Cadence Online Support ([support.cadence.com](#)). If you cannot log in, contact Customer Support.
2. Uncompress and untar the kit using the `tar` command. This creates a directory named `<installation>_<version>`. Example:

```
% tar -zxvf <installation>_<version>-Linux.t.gz
```

3. Start the license manager (`lmgrd`) or reread the license file as appropriate. See “[License Installation](#)” on page 57.
4. Set the `LM_LICENSE_FILE` variable. The appropriate setting depends on the existing licensing environment. Replace `host_name` with the license server host name and replace `port` with the TCP/IP port number that the license manager daemon was started on.

```
% setenv LM_LICENSE_FILE port@host_name
```

Note: Refer to “[Optional Environment Variables](#)” on page 65 to learn how you can resolve launch delays when running the tool from remote locations.

5. Modify the `PATH` variable to add `<installation>/bin` to the executable path.

```
% setenv PATH ${PATH}:/usr/local/<installation>/bin
```

6. Verify that the installation is complete by typing `jaspergold` on your command line.

Note: The `<installation>/doc/example_jaspergold_apps` directory includes scripts that illustrate sessions for VHDL PSL, VHDL SVA, and Verilog SVA. All the files you need to run these scripts are also located in the example directory.

License Installation

Licensing is usually site specific. The information in this section is an overview. For complete details, consult your FlexNet Publisher® documentation.

This section includes the following topics:

- [TIMEOUT and TIMEOUTALL](#) on page 59
- [Redundant License Servers](#) on page 59
- [Installing Your License](#) on page 59
- [Appending Lines to a License File](#) on page 60
- [Manually Starting a New Daemon](#) on page 61
- [Using the Correct License Server](#) on page 62
- [Using FlexNet Publisher Utilities](#) on page 63



License daemon changed from Jasper (`tempusd`) to Cadence (`cdslmd`)

In JasperGold release 2015.06, we moved to the Cadence license daemon (`cdslmd`). Please note that there is a long-standing Cadence advisory note not to use the `lmremove` feature in FlexNet with `cdslmd` (see [License Alert – LMREREAD and LMREMOVE](#) at Cadence Online Support for more details). Using the `lmremove` feature in FlexNet with `cdslmd` will disable certain capabilities in JasperGold, such as the `-nice` license feature. If you currently make effective use of `-nice` and do not want to lose this capability, you can do either of the following:

- Ensure `lmremove` is enabled.
- When requesting your keys, provide a different license server host ID (not the one used to run other Cadence licenses), and enable `lmremove` for `cdslmd` only on that server.

In summary:

- To launch JasperGold 2015.06 or newer, you must use new licenses specially made for the 2015.06 release.
- Licenses you have for a version prior to 2015.06 will not run JasperGold 2015.06 and beyond.
- The new licenses are generated according to the Cadence License Standards and are compatible only with a new `cdslmd` daemon. In preparation for the new licenses, beginning with JasperGold release 2015.06, the new `cdslmd` daemon is distributed with the JasperGold product. Earlier releases of JasperGold contain the `tempusd` daemon.
- When you receive one of these new license files, you will be required to use the new Cadence `cdslmd` daemon. This Cadence format `cdslmd` license file will **not** support older releases of JasperGold, but if you need to run an older version, you can continue running it with the `tempusd` daemon and allow both licenses to run simultaneously. However, you must start the daemons with separate license files using different `lmgrd` commands.
- The license file includes instructions. These instructions are also available in this section of the chapter and in the JasperGold installation build at `/doc/INSTALL.pdf`.
- The location of the license file is defined by searching the following in order:

1. The setting of the `CDS_LIC_FILE` environmental variable.

For additional information, refer to [“Optional Environment Variables”](#) on page 65.

2. The setting of the `LM_LICENSE_FILE` environmental variable.

If your `LM_LICENSE_FILE` variable is set to multiple paths that are not all serving the expected set of JasperGold licenses, you might experience a delay when launching the tool from remote locations. To resolve a potential delay, use the environment variable `CDS_LIC_FILE` to specify JasperGold license files. If you want to avoid searching the license file locations specified in `LM_LICENSE_FILE` completely, you can define the environment variable `CDS_LIC_ONLY` (along with `CDS_LIC_FILE`), which tells the tool to ignore `LM_LICENSE_FILE`.

There are two kinds of delays you might encounter:

- ❑ The license exists, but JasperGold has to search in many license file locations before it finds the license.
- ❑ The license does not exist, and JasperGold has to search many license locations before it finds out that it does not exist.

By using `CDS_LIC_FILE`, you can reduce the first type of delay, and by using `CDS_LIC_ONLY` together with `CDS_LIC_FILE`, you can eliminate the second type of delay.

TIMEOUT and TIMEOUTALL

The FlexNet variables `TIMEOUT` and `TIMEOUTALL` specify the idle time-out for one or all features, returning them to the free pool for use by another user.

If the license server administrator specifies the `TIMEOUT` or `TIMEOUTALL` settings in the options file for the vendor daemon, the licenses checked out by an idle JasperGold Apps session are automatically checked in after the specified time. Consult the FlexNet documentation for additional information.

Note:

- The minimum allowed value of `TIMEOUT` and `TIMEOUTALL` is 900 seconds (15 minutes).
- JasperGold Apps informs the license server that it is idle if no activity has occurred for eight minutes. Idle state includes no running Tcl command in an analysis session, no running background proof, and no keystrokes or mouse clicks.

When the tool goes from being idle to non-idle (for example, when a Tcl command runs), the tool attempts to check out all licenses that were previously checked in. If the check-out fails, the tool exits.

Redundant License Servers

FlexNet Publisher supports three-server redundancy for license servers. With this arrangement, two of the three license servers must be up and running for proper functioning. Three-server redundancy is designed to provide hardware failover protection. One of the three servers is the “master,” the only server capable of issuing licenses, and all clients must contact the master.

Installing Your License

The following is an illustrative example only. It assumes that the license daemon will be run on the same machine that JasperGold Apps will run on. If this assumption is not valid, contact your system administrator.

Note: Use `lmutil lmdown` instead of `lmutil lmrreread` when you get a new JasperGold license.

JasperGold Apps Command Reference Manual

Installation

1. Install the license file in the customer-specific location. For example:

```
% cp license.dat /usr/local/<installation>/etc/license
```

2. Edit the license file as follows:

- ❑ Edit the path to the vendor daemon (that is, the path following VENDOR cds1md in the license file) so that it points to the path /usr/local/<installation>/<PLATFORM>/bin/cds1md.
- ❑ Replace <installation> with your installation name.
- ❑ Replace <PLATFORM> with your platform name: Linux or Linux64.

Note: The path must be a fully expanded path. The FlexNet Publisher license manager will not expand an environment variable.

3. Start the license manager.

```
% /usr/local/<installation>/bin/lmgrd -c <PATH_TO_LICENSE_FILE> -l  
LM_LOG_FILE
```

Note: The -l LM_LOG_FILE is optional but highly recommended. You can use it for debugging.

Licensing should work correctly at this stage. If you encounter license issues with a JasperGold Apps software installation, check the environment variables LM_LICENSE_FILE and CDS_LIC_FILE to ensure that they are set to the correct paths. If you continue to have problems, look for a hidden .flexlmrc file in your home directory and remove the line that includes CDS_LIC_FILE.

Appending Lines to a License File

License files begin with a SERVER line. In the case of three-server redundancy, there are three SERVER lines. SERVER lines are followed by one or more VENDOR lines, which are followed by FEATURE lines. You can modify the following elements in the license file and use the line-continuation character (\) to break long lines.

■ SERVER lines – Host names and optional TCP/IP port numbers

Note: Cadence ships license files with Cadence_SERVER as hostname in the SERVER line. Edit this field to enter the actual hostname prior to starting the license server. If you do not change the port number, end users must update LM_LICENSE_FILE or CDS_LIC_FILE. (See “[Optional Environment Variables](#)” on page 65.)

Example (bash):

JasperGold Apps Command Reference Manual

Installation

```
export LM_LICENSE_FILE=5280@server
```

- **VENDOR lines** – paths, options file path, optional TCP/IP port numbers (for firewall support)
- **FEATURE lines** – values in keyword=value pairs

Note: In general, a FEATURE line describes the license required to use a product. An INCREMENT line incrementally adds licenses to a prior FEATURE or INCREMENT line.

The following is a sample of the license keys section of a license file.

```
SERVER qa2 0026b92a4b75 5280
DAEMON cds1md ./cds1md
USE_SERVER
# DO NOT REMOVE THE USE_SERVER LINE
FEATURE jasper_fpv cds1md 2015.06 22-dec-2015 5 4EC2EF327ED1B81G75E5 \
    VENDOR_STRING=J:DEMO DUP_GROUP=NONE vendor_info=25-jun-2015 \
    ISSUED=25-jun-2015 SN=2015-06-24Q11:06:22:637 SIGN2="1961 CE67 \
    2A5C 5CBF 65FE 0C37 A52B 3FCB 0A3F 2967 746D BC94 1B16 6ABD \
    6A14 1031 B158 B973 641F 57B1 2380 C699 651A 260D 5D26 351D \
    F192 2674 54F5 5B4C" V7.1_LK=6E66EB3997CDAD2A8CCC
FEATURE jasper_interactive cds1md 2015.06 22-dec-2015 5 \
    9E06ACA968L410130B9E VENDOR_STRING=J:DEMO DUP_GROUP=NONE \
    vendor_info=25-jun-2015 ISSUED=25-jun-2015 \
    SN=2015-06-25T21:06:21:637 SIGN2="1445 62G7 9686 BAB3 FC24 \
    1154 B74C CDD3 64EA FC8A 9FFA 725D A6C4 9477 80EB 2B77 2A10 \
    DD52 C7D6 C0A7 F299 FBC2 9045 3909 218C F6B1 0AAB 1310 A500 \
    4A30" V7.1_LK=B6A62BA93DCE41750BAD
# ... (snip)
#
# These portions of a sample of the product license keys section are provided as
# an example of what a license file looks like. Your actual license will include
# multiple FEATURE lines specific to your case.
```

Manually Starting a New Daemon

The license server system has two components, `lmgrd` and the vendor daemon. The job of `lmgrd` is to start and maintain the vendor daemons listed in the VENDOR lines of your license file and refer the JasperGold Apps features checkouts to the correct daemon.

Note:

JasperGold Apps Command Reference Manual

Installation

- `jaspergold -nice` will not function in either of the following cases:
 - If you start `lmgrd` with `-2 -p` or `-x lmremove`.
 - If you manage licenses with `lmadmin` with the default setting (`adminOnly`).Refer to “[-nice](#)” on page 72 for additional information.
- If you are running a three-server redundant license server system, keep a separate copy of the license file and the `lmgrd` and the vendor daemon binaries on each of the three servers.

To manually start `lmgrd` from the command line, use the following syntax:

```
lmutil lmgrd -c <license_file_list> -l [+]<debug_log_path>
```

- **license_file_list** – This variable points to one or more files. You can include one or more paths to single license files or directories that contain license files. In the case of the latter, `lmgrd` uses all the license files in the directory.
- **debug_log_path** – This variable is the full path to the debugging log file. Use the `+` character to append logging entries.

Note: Use `lmutil lmgrd -help` for additional information.

Using the Correct License Server

JasperGold Apps uses FlexNet Publisher licensing from Flexera Software, version 11.13.0.3, which is included in the product package. If you use a FlexNet Publisher license server other than the one that ships with the tool, it must be version 11.13.0.3 or newer. And if you have specified a path to the vendor daemon on a VENDOR line in the license file, you must change it so it points to a supported version of the daemon.

When you upgrade your license server, use the following procedure.

1. Stop the license server.
2. Copy the new `cdslmd` over the old `cdslmd`.
3. Start the license server.
4. Verify that the update was successful by running `lmstat`.

When you run `lmstat`, it will tell you the version of the vendor daemon that is being used. For example, the following return is for a license server named `cyklop`:

```
Vendor daemon status (on cyklop) :  
cdslmd: UP v11.13.0.3
```

If you do not see v11.13.0.3 in the return, the update was not successful. In that case, carefully redo the steps listed above.

Using FlexNet Publisher Utilities

The FlexNet Publisher utilities are packaged as a single executable called `lmutil`. You can install `lmutil` as individual commands by creating links to the individual command names or making copies of `lmutil` as the individual command names. Or you can install it as a wrapper that runs the individual command as the `lmutil` command (for example, `lmstat` or `lmdown`).

The following helpful utilities are described in this section:

- [lmdown](#) on page 63
- [lmhostid](#) on page 64
- [lmstat](#) on page 64

lmdown

`lmdown` shuts down selected license daemons (both `lmgrd` and selected vendor daemons).

Note: Use `lmutil lmdown` instead of `lmutil lmrreread` when you get a new JasperGold license.

<code>lmutil lmdown</code>	
<code>-c <license_file_list> [-vendor <vendor_daemon>] [-q] [-all] [-force]</code>	
<code>-c license_file_list</code>	Specifies the license files you want to shut down.
<code>-vendor</code>	Shuts down the specified vendor daemon, but <code>lmgrd</code> continues to run.
<code>-q</code>	Keeps the utility from prompting or printing a header. If you don't use this option, <code>lmdown</code> prompts with, "Are you sure? [y/n] :."
<code>-all</code>	Automatically shuts down all the specified servers.
<code>-force</code>	Use this option when you have borrowed licenses. It shuts down the license daemons for the machine where the license server system is running.

JasperGold Apps Command Reference Manual

Installation

lmhostid

`lmhostid` returns the FlexNet Publisher licensing hostid of the current platform. If you run this command without any arguments, it displays the default hostid type for the current platform.

```
lmutil lmhostid  
[-n] [-utf8]
```

<code>-n</code>	Returns the hostid as a string.
<code>-utf8</code>	Displays the hostid as a UTF-8 encoded string rather than an ASCII string. Use this option if your hostid contains other than alpha-numeric characters.

Note: Consult the FlexNet documentation for additional information.

lmstat

`lmstat` returns the status of all network licensing activities (for example, running daemons and individual feature users).

```
lmutil lmstat  
[-a] [-c <license_file_list>] [-f [<feature>]]  
[-i [<feature>] [-s [<server>] [-S [<vendor>]] [-t <timeout_value>]
```

<code>-a</code>	Displays all information.
<code>-c license_file_list</code>	Reports on the specified license file(s).
<code>-f feature</code>	Displays a specified feature's users. If you do not specify a feature, it displays information for all features.
<code>-i feature</code>	Displays information from the FEATURE line for the specified feature. If you do not specify a feature, it displays information for all features.
<code>-s server</code>	Displays the status of all the license files that are listed in \$VENDOR_LICENSE_FILE or \$LM_LICENSE_FILE on the specified server. If you do not specify a server, it displays information for all servers.
<code>-S vendor</code>	Lists all users of the specified vendor's features.

JasperGold Apps Command Reference Manual

Installation

<code>-t <i>timeout_value</i></code>	Sets the connection time-out (time spent attempting to connect) to the value of <i>timeout_value</i> .
--------------------------------------	--

Optional Environment Variables

This section includes information on the `CDS_LIC_FILE` and `CDS_LIC_ONLY` optional environment variables. Refer to “[Environment Variables](#)” on page 73 for additional information about required and optional environment variables.

The location of the license file is defined by searching the following in order:

1. The setting of the `CDS_LIC_FILE` environmental variable.
2. The setting of the `LM_LICENSE_FILE` environmental variable.

Note: If your `LM_LICENSE_FILE` variable is set to multiple paths that are not all serving the expected set of JasperGold licenses, you might experience a delay when launching the tool from remote locations. To resolve a potential delay, use the environment variable `CDS_LIC_FILE` to specify JasperGold license files. If you want to avoid searching the license file locations specified in `LM_LICENSE_FILE` completely, you can define the environment variable `CDS_LIC_ONLY`, which tells the tool to ignore `LM_LICENSE_FILE`.

Simultaneously Running Jasper and Cadence Licenses

You can run the Cadence `cdslmd` daemon with the legacy Jasper `tempusd` daemon simultaneously; however, you must start the daemons with separate license files using different `lmgrd` commands.

Practical License Setup Example

For this example, we start two instances of `lmgrd` running on the same machine, which is named `my_license_server_machine`. Port 27000 serves old `tempusd` licenses and port 5280 serves `cdslmd` licenses. This information is configured on the `SERVER` line in the license file.

Note: The examples in this section use bash. If you are using a different shell, adjust the commands accordingly.

```
lmgrd -c <PATH_TO_LICENSE_FILE>/cdslmd_lic.txt -l cdslmd_lic.log  
lmgrd -c <PATH_TO_LICENSE_FILE>/tempusd_lic.txt -l tempusd_lic.log
```

Summary:

- Current Cadence `cdslmd` licenses served on `5280@my_license_server_machine`
- Old Jasper `tempusd` licenses served on `27000@my_license_server_machine`

We recommend putting the `cdslmd` license server ahead of the old license server to improve performance. To have both old and new versions working, define the variable `LM_LICENSE_FILE` as follows:

```
export  
LM_LICENSE_FILE=5280@my_license_server_machine:27000@my_license_server_machine
```

Optimizing Performance for License Checkout

There are two kinds of delays you might encounter when checking out licenses:

- The license exists, but JasperGold Apps must search in many license file locations before it finds the license.
- The license does not exist or is not available, and JasperGold Apps must search many license locations before finding out.

Following the example, for the best performance, define the environment variables below:

```
export TEMPUSD_LICENSE_FILE=27000@my_license_server_machine  
export CDS_LIC_FILE=5280@my_license_server_machine  
export CDS_LIC_ONLY=1
```

With these definitions, you can expect the following optimizations:

- Defining the `TEMPUSD_LICENSE_FILE` makes older versions ignore `LM_LICENSE_FILE` and only look into servers defined in the `TEMPUSD_LICENSE_FILE`, which minimizes both types of delays for older versions.
- Defining `CDS_LIC_FILE` makes the newer version of JasperGold Apps (2015.06 and newer) look first into it for licenses and then at `LM_LICENSE_FILE`, if it is defined, which minimizes the first type of delay.
- Defining `CDS_LIC_ONLY` makes the newer version of JasperGold Apps (2015.06 and newer) ignore `LM_LICENSE_FILE` and only query the servers from the `CDS_LIC_FILE`, which minimizes the second type of delay.

Commands and Variables Overview

This chapter provides a quick look at JasperGold Apps commands. It includes tables for the following:

- [Command-Line Options](#) on page 68
- [Environment Variables](#) on page 73
- [Command Summary](#) on page 78

Command-Line Options

By default, JasperGold Apps runs in GUI mode and opens the main JasperGold Apps window when you launch the tool. To launch JasperGold Apps with the default Formal Property Verification App, type either of the following at the command prompt:

```
% jaspergold  
% jg
```

To launch a different app, use one of the app switches listed in [Table 3-1](#) on page 68.

Syntax

```
jaspergold  
    [options]  
  
jaspergold  
    -batch [options]
```

[Table 3-1](#) on page 68 lists and briefly describes the JasperGold Apps command-line options.

Note: Command-line options are case insensitive. For example, the tool recognizes all of the following:

- jaspergold -COMMAND
- jaspergold -command
- jaspergold -ComAND

Table 3-1 JasperGold Apps Command-Line Options

Command-Line Options	Description
<file_name>.tcl	Run one or more Tcl scripts with the required .tcl extension in the order they appear on the command line as if they were entered interactively. Example: // Run script1.tcl and then run script2.tcl. % jaspergold script1.tcl script2.tcl
-help	Display command-line help information (the contents of this table).
-version	Display the version string.

JasperGold Apps Command Reference Manual

Commands and Variables Overview

Table 3-1 JasperGold Apps Command-Line Options, *continued*

Command-Line Options	Description
-arch	Launch the Architectural Modeling App.
-bps	Launch the Behavioral Property Synthesis App.
-conn	Launch the Connectivity Verification App.
-cov	Launch the Coverage App.
-csr	Launch the CSR Verification App.
-dld	Launch the Deadlock Detection App.
-fpv	Launch the Formal Property Verification App.
-lpv	Launch the Low Power Verification App.
-rtlD	Launch the RTL Development App.
-sec	Launch the Sequential Equivalence Checking App.
-spec	Launch the Executable Specification App.
-sps	Launch the Structural Property Synthesis App.
-spv	Launch the Security Path Verification App.
-unr	Launch the Coverage Unreachability App.
-xprop	Launch the X-Propagation Verification App.
-batch	Launch the tool in batch mode. Neither the Executable Specification App (-spec) nor the RTL Development App (-rtlD) support this mode.
-no_gui	Start the tool without launching the GUI, allowing Tcl command entry from the terminal. Neither the Executable Specification App (-spec) nor the RTL Development App (-rtlD) support this mode.
-proj <dir_name>	Use the specified directory as the project directory. If you do not specify a directory, the tool uses . / jgproject. The project directory stores various files, including log files, debugging files, and settings related to the current project.
-acquire_proj	Forcibly acquire ownership of the project directory if it is already locked.

JasperGold Apps Command Reference Manual

Commands and Variables Overview

Table 3-1 JasperGold Apps Command-Line Options, *continued*

Command-Line Options	Description
<code>-tcl <tcl></code>	Execute the specified Tcl script in <code>session_0</code> .
<code>-command <cmd></code>	Run the specified Tcl command after running all specified Tcl scripts and before receiving interactive commands. Add multiple commands by using the <code>-command</code> switch for each command.
<code>-db <file_name></code>	Load the specified database. Use this command to continue work on a saved database.
<code>-template</code> Empty English Japanese <code>file_name.xml</code>	Load a generic or specified template for the Executable Specification App. <ul style="list-style-type: none"> ■ By default, the tool loads an empty template. ■ Use English or Japanese to build a standard database with the specified language. ■ Specify an XML file name to build a database with a predefined template, for example, your company's approved template.
<code>-define</code> <code><tcl_variable></code> <code><tcl_value></code>	Define a Tcl variable and its value for <code>session_0</code> . Implementation guidance: <ul style="list-style-type: none"> ■ This switch is especially useful for launching two analysis sessions, each with a different version of the RTL, so you can bring up the corresponding waveforms for comparison. ■ Define multiple variables by using one <code>-define</code> switch for each definition.

JasperGold Apps Command Reference Manual

Commands and Variables Overview

Table 3-1 JasperGold Apps Command-Line Options, *continued*

Command-Line Options	Description
-remote_mode <mode> -remote_console	<p>Use the specified mode to define the configuration for launching a session on a remote machine.</p> <ul style="list-style-type: none">■ local – Launch a remote analysis session on your local machine.■ lsf – Launch a remote analysis session on the Platform LSF cluster that is expected to be available using the LSF_BINDIR environment variable. See “LSF_BINDIR” on page 76.■ oge – Launch a remote analysis session on the Oracle Grid Engine that is expected to be available using the SGE_ROOT environment variable. See “SGE_ROOT” on page 78. <p>Any other value is regarded as a custom shell script for the mode <code>shell</code>.</p> <p>Use <code>-remote_console</code> to launch the default <code>session_0</code> on the remote machine.</p> <p>Also see “session” on page 815 to learn how to launch remote sessions from the console.</p>
-no_wait	<p>Do not wait for a JasperGold Apps license to become available.</p> <p>By default, if a JasperGold Apps license is unavailable, the tool waits for one to become available. With this switch, the tool errors out instead.</p>
-wait_interval N	<p>In the event a license is not available, wait the specified number of seconds and then retry license checkout.</p> <p>By default, JasperGold Apps tries a license checkout every 60 seconds for non-nice sessions and every 30 seconds for nice sessions.</p>
-wait_retries N	<p>In the event a license is not available, try to check out a license the specified number of times.</p> <p>By default, the retry limit is five for non-nice sessions and two for nice sessions.</p>

JasperGold Apps Command Reference Manual

Commands and Variables Overview

Table 3-1 JasperGold Apps Command-Line Options, *continued*

Command-Line Options	Description
<code>-nice</code>	<p>Flag checked-out licenses to make them available for surrender to other sessions.</p> <p>When the tool loses a license, a dialog appears. It offers the following three options:</p> <ul style="list-style-type: none">■ Exit the application■ Save the database■ Wait <p>If you choose to wait, the tool attempts to reconnect to the license server and reclaim the license. When this is successful, the dialog automatically closes and the Tcl interpreter accepts Tcl commands once again.</p> <p>Note: <code>-nice</code> will not function in either of the following cases:</p> <ul style="list-style-type: none">■ If you start <code>lmgrd</code> with <code>-2 -p</code> or <code>-x lmremove</code>.■ If you manage licenses with <code>lmadmin</code> with the default setting (<code>adminOnly</code>). <p>For additional information, refer to the “Jasper ProofGrid, Per-Engine Job Management, and ProofGrid Manager” tutorial available from Cadence Online Support (support.cadence.com/jasper-raks).</p>
<code>-enable_license_option</code>	Enable license checking for IFV/IEV upgrade products.
<code>-title <title></code>	Prepend all main window titles with the specified string. Note: If your string includes spaces, you must enclose it in quotation marks.
<code>-display <display></code>	Configure the display into which the remote session should be launched.
<code>-pk <proof_kit.jdb></code>	Launch the tool with tabs for the Executable Specification App and RTL Development App and load the specified Intelligent Proof Kit database.

Environment Variables

The following table lists and briefly describes JasperGold Apps environment variables.

Table 3-2 Environment Variables

Environment Variable	Description
Required Variable	
LM_LICENSE_FILE	This variable applies to all JasperGold Apps products. It points to a valid license file. Also see “Optional Environment Variables” on page 65.
Optional Variable	
CDS_LIC_FILE	This variable applies to all JasperGold Apps products. It points to JasperGold-specific license files. Also see “Optional Environment Variables” on page 65.
CDS_LIC_ONLY	If you have defined CDS_LIC_ONLY, JasperGold Apps uses it along with CDS_LIC_FILE to find the license file and ignores LM_LICENSE_FILE.
DISPLAY	Points to a valid display device for launching an X window application. Note: If the DISPLAY environment variable does not point to a valid display, JasperGold Apps prints an error message and exits immediately.
EDITOR	Points to the application launched by the edit command. The default is vi.

JasperGold Apps Command Reference Manual

Commands and Variables Overview

Table 3-2 Environment Variables, *continued*

Environment Variable	Description
JASPER_COMM_DEBUG	<p>If you encounter a problem with a session connecting to the console, define this variable. We recommend setting it to 1, and when the tool finishes, inspect the log files.</p> <p>Note:</p> <ul style="list-style-type: none">■ You can define any value, including an empty string, but if you set it to a non-empty string, the log files for the analysis sessions contain messages for the analysis session connecting to the console (IPL001) and disconnecting from the console (IPL002).■ See also “JASPER_COMM_LOCAL_SERVERNAME” on page 74 and “JASPER_COMM_PORTS” on page 75.
JASPER_COMM_LOCAL_SERVERNAME	<p>If you encounter a problem starting the server in the main process, use the environment variable JASPER_COMM_LOCAL_SERVERNAME to specify the name of the host the analysis session will connect to. The tool only considers this value if you start the analysis session in local mode. The suggested values for this variable are localhost and 127.0.0.1.</p> <p>Note:</p> <ul style="list-style-type: none">■ This environment variable might be warranted if you see an error message such as the following: <pre>ERROR: Session labeled "session_0" has terminated due to socket error: 2 "Host not found" (server1:0) (server1:34260)</pre>■ See also “JASPER_COMM_DEBUG” on page 74 and “JASPER_COMM_PORTS” on page 75.

JasperGold Apps Command Reference Manual

Commands and Variables Overview

Table 3-2 Environment Variables, *continued*

Environment Variable	Description
JASPER_COMM_PORTS	<p>By default, the operating system assigns a port for the tool. If you encounter problems, use the environment variable JASPER_COMM_PORTS with a list of port ranges. When starting the server, the tool will attempt to use one of the ports in the specified ranges.</p> <p>Example:</p> <p>1234,12345-23456,34567</p> <p>Note:</p> <ul style="list-style-type: none">■ This environment variable might be warranted if you see an error message such as the following: ERROR: Session labeled "session_0" has terminated due to socket error: 2 "Host not found" (server1:0) (server1:34260)■ Use a comma-separated list. Each element of the list must be either an integer or a pair of integers separated by a dash. The tool attempts to use the ports in the specified order. In the example above, it first attempts port 1234, then 12345 up to 23456, and then finally 34567.■ See also ""JASPER_COMM_DEBUG" on page 74 and ""JASPER_COMM_LOCAL_SERVERNAME" on page 74.

JasperGold Apps Command Reference Manual

Commands and Variables Overview

Table 3-2 Environment Variables, continued

Environment Variable	Description
JASPER_FORCED_EXIT_TIMEOUT	<p>Specifies a time-out in seconds. When the tool has been inactive for the specified amount of time, it autosaves the database and exits.</p> <p>Note: There is no default time limit at which the tool automatically exits if inactive.</p> <p>The minimum allowed value is 3600 seconds. If you use a lower value, the tool defaults to 3600 seconds.</p> <p>After three fourths of the specified time passes, a dialog appears. It offers two options:</p> <ul style="list-style-type: none">■ Click the <i>Exit Application</i> button to autosave the database and exit the tool.■ Click the <i>Continue</i> button to close the dialog and continue working with the tool. <p>The tool is considered inactive when all analysis sessions are inactive. An inactive analysis session neither runs a Tcl command nor runs a job in the background (for example, prove or Visualize). The timer starts as soon as the tool becomes inactive and stops as soon as it becomes active.</p>
JGRC	<p>Points to an optional default Tcl script that loads at startup.</p> <p>Note: You can put initialization commands in this script. The default value of \$JGRC is \$<HOME>/ .jgrc.</p>
LSF_BINDIR	<p>This variable is necessary when you configure your server farm using proofgrid_mode lsf or the -remote_mode lsf command-line switch.</p> <p>When you use proofgrid_mode lsf, JasperGold Apps executes \$LSF_BINDIR/bsub. Consult the Platform LSF documentation to learn how to configure \$LSF_BINDIR.</p>

JasperGold Apps Command Reference Manual

Commands and Variables Overview

Table 3-2 Environment Variables, *continued*

Environment Variable	Description
PDFVIEWER	<p>Points to the PDF viewer application for viewing documents.</p> <p>Note:</p> <ul style="list-style-type: none">■ Use this variable only if there are no settings for PDF files in the <code>~/.config</code> directory settings files <code>~/.config/jasper.conf</code> or <code>~/.config/jasper/jaspergold.conf</code>. You can specify the PDF viewer directly in these files or with the Preferences dialog (<i>Edit – Preferences</i>).■ This variable is helpful if you encounter problems launching a PDF document from a GUI.■ If <code>acroread</code> is not available on your system, try setting the variable to <code>xpdf</code> or <code>evince</code>. However, for optimal print quality and application functionality, we recommend you use the current version of <code>acroread</code>.
SCHEMATICVIEWER	<p>Points to the Synopsys path for the Verdi schematic viewer and nWave viewer.</p> <p>Important things to know:</p> <ul style="list-style-type: none">■ Use the <code>set_schematic_viewer_command</code> to override this variable. (See “set_schematic_viewer_command” on page 1221.)■ Prior to engaging in a run that uses schematics or a standalone waveform viewer, add the viewer to your PATH.■ JasperGold Apps only supports Verdi standalone waveform and schematic viewers.■ To learn more about using a schematic viewer, refer to Appendix I, “Troubleshooting for External Applications” in the <i>JasperGold Apps User’s Guide</i>.

JasperGold Apps Command Reference Manual

Commands and Variables Overview

Table 3-2 Environment Variables, *continued*

Environment Variable	Description
SGE_ROOT	This variable is necessary when you configure your server farm using <code>proofgrid_mode oge</code> or the <code>-remote_mode oge</code> command-line switch. You will not set this variable if you use <code>proofgrid_mode shell</code> . When you use <code>proofgrid_mode oge</code> , JasperGold Apps executes '\$SGE_ROOT/util/arch'/qrsh or '\$SGE_ROOT/util/arch'/qsub. Consult the Oracle Grid Engine documentation to learn how to configure \$SGE_ROOT.
TMPDIR	Points to the directory where temporary files are created. If you do not define this variable, the tool uses /tmp.
VOVDIR	This variable is necessary when you configure your server farm using <code>proofgrid_mode nc</code> . You will not set this variable if you use <code>proofgrid_mode shell</code> . When you use <code>proofgrid_mode nc</code> , JasperGold Apps executes \$VOVDIR/scripts/nc. Consult the Runtime Design Automation NetworkComputer documentation to learn how to configure \$VOVDIR.

To learn about optional variables for synchronizing the BPS App with a simulator, see “Generating Configuration Files” in the *Behavioral Property Synthesis App User’s Guide*.

Command Summary

The following tables list and briefly describe JasperGold Apps commands:

- [General Command Summary](#) on page 79
- [Configuration Command Summary](#) on page 97

For the complete command reference, refer to [Chapter 4, “General Commands”](#) and [Chapter 5, “Configuration Commands.”](#)

JasperGold Apps Command Reference Manual

Commands and Variables Overview

Note: The summaries in this section include corresponding JasperGold Apps GUI actions where applicable.

Table 3-3 General Command Summary

Command	Description	Equivalent GUI Action
abstract	Create a counter, initial-value, register-value, reset-value, or CDC abstraction.	Menu: <ul style="list-style-type: none">■ <i>Application – Task</i> <p><i>Add Counter Abstraction</i> or <i>Add Initial Value Abstraction</i></p> <p><i>Task Table –</i></p> <ul style="list-style-type: none">■ Right-click and choose <i>Task Context</i> <p><i>Design Information</i> window</p> <ul style="list-style-type: none">■ Right-click on a counter name and choose <i>Add Counter Abstraction</i> or <i>Remove Counter Abstraction</i> <p><i>Visualize</i> window</p> <ul style="list-style-type: none">■ <i>View</i> menu – <i>Task Context</i>■ Click the <i>Add to Task Context</i> button¹ and then the <i>Counter Abstractions</i> tab.
abvip	Print the requested ABVIP information or set the ABVIP installation directory.	None
add_tx_attribute	Associate a signal with a behavior as an attribute.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-3 General Command Summary, *continued*

Command	Description	Equivalent GUI Action
analyze	Analyze the contents of an HDL or property file.	<ul style="list-style-type: none"> ■ Menu: <i>Design – Analyze RTL</i> ■ Click the <i>Analyze RTL</i> button.¹
assert	Add an assertion about the signals in the design to the current task. Also, remove, modify, convert, list, and so forth.	<ul style="list-style-type: none"> ■ Menu: <i>Application – Task – Add Property</i> ■ Click the <i>Capture Behavior from Waveform Selection</i> button.¹ ■ Click the <i>Add Property</i> button.¹
assume	Add an assumption about the signals in the design to the current task or all tasks. Also, remove, modify, convert, list, and so forth.	<ul style="list-style-type: none"> ■ Menu: <i>Application – Task – Add Property</i> ■ Click the <i>Capture Behavior from Waveform Selection</i> button.¹ ■ Click the <i>Add Property</i> button.¹
auto_setup	Attempt to automatically generate a setup script for loading the specified RTL.	None
auto_syn	Use property synthesis to generate helper assertions to assist in the proof process.	None
autoprove	Orchestrates a dynamic approach to verification based on the <i>prove</i> command.	None
blackbox_assistant	Bring up a design hierarchy tree without full elaboration.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-3 General Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>capture testcase</u>	Save a tarred file that contains all sourced scripts and design files.	None
<u>check_arch</u>	Print requested Architectural Modeling information or execute the requested action.	Various <ul style="list-style-type: none"> ■ Menu: <i>Application...</i> ■ Wizard buttons: Reveal a button's name by using the cursor to hover over it.
<u>check_assumptions</u>	Check whether assumptions have overconstrained the design.	<i>Task Table</i> – Right-click and choose <i>Check Assumptions</i>
<u>check_bps</u>	Manipulate property candidates obtained from the property extraction process.	Various <ul style="list-style-type: none"> ■ Menu: <i>Application...</i> ■ Wizard buttons: Reveal a button's name by using the cursor to hover over it.
<u>check_code</u>	Check whether a Tcl command returns a specific code.	None
<u>check_conn</u>	Print requested Connectivity information or execute requested action.	Various <ul style="list-style-type: none"> ■ Menu: <i>Application...</i> ■ Wizard buttons: Reveal a button's name by using the cursor to hover over it.

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-3 General Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>check_cov</u>	Compute, manage, and report proof and stimuli coverage.	Various <ul style="list-style-type: none"> ■ Menu: <i>Application...</i> ■ Wizard buttons: Reveal a button's name by using the cursor to hover over it.
<u>check_csr</u>	Print out requested CSR information or execute the requested action.	Various <ul style="list-style-type: none"> ■ Menu: <i>Application...</i> ■ Wizard buttons: Reveal a button's name by using the cursor to hover over it.
<u>check_dld</u>	Detect system-level deadlock situations.	Various <ul style="list-style-type: none"> ■ Menu: <i>Application...</i> ■ Wizard buttons: Reveal a button's name by using the cursor to hover over it.
<u>check_loop</u>	Check for signals that contribute to a combinational loop.	None
<u>check_lpv</u>	Load, elaborate, and check power-aware designs.	Various <ul style="list-style-type: none"> ■ Menu: <i>Application...</i> ■ Wizard buttons: Reveal a button's name by using the cursor to hover over it.

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-3 General Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>check_one_of</u>	Check whether an actual value matches the expectations.	None
<u>check_sec</u>	Compare two RTL designs, the specification and the implementation.	Various <ul style="list-style-type: none"> ■ Menu: <i>Application...</i> ■ Wizard buttons: Reveal a button's name by using the cursor to hover over it.
<u>check_sps</u>	Manipulate structural properties obtained from the property extraction process.	Various <ul style="list-style-type: none"> ■ Menu: <i>Application...</i> ■ Wizard buttons: Reveal a button's name by using the cursor to hover over it.
<u>check_sps_configure</u>	Configure structural property synthesis features.	<ul style="list-style-type: none"> ■ Menu: <i>Application – Configure SPS Checks</i> ■ Click the <i>Configure SPS Checks</i> button.¹
<u>check_sps_waiver</u>	Manage waivers in the SPS App.	<ul style="list-style-type: none"> ■ Menu: <i>Application – Add Waiver</i> ■ <i>Structural Properties</i> pane: Right-click and choose <i>Set as Waived/Remove Waiver</i> ■ <i>Waiver List</i> tab: Right-click and choose <i>Edit Waiver/Delete Waiver(s)</i>

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-3 General Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>check_spv</u>	Check for functional paths between sets of signals.	Various ■ Menu: <i>Application...</i> ■ Wizard buttons: Reveal a button's name by using the cursor to hover over it.
<u>check_unr</u>	Analyze unreachable branch, expression, and toggle signals in a design.	Various ■ Menu: <i>Application...</i> ■ Wizard buttons: Reveal a button's name by using the cursor to hover over it.
<u>check_xprop</u>	Perform X-propagation analysis on one or more signals.	Various ■ Menu: <i>Application...</i> ■ Wizard buttons: Reveal a button's name by using the cursor to hover over it.
<u>clear</u>	Clear proof information.	<i>Task Table</i> – Right-click and choose <i>Clear Task</i>
<u>clock</u>	Specify the global clock-related configuration, list all clock configurations, and analyze the clock tree.	■ Menu: <i>Design – Clock</i> ■ Click the <i>Clock</i> button. ¹

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-3 General Command Summary, continued

Command	Description	Equivalent GUI Action
configure	List or unlimit the current configuration or change it to default values.	<ul style="list-style-type: none"> ■ Menu: <i>Application – Proof – Proof Settings</i> ■ Click the <i>Proof Settings</i> button.¹
connect	Instantiate and connect a requirement module or entity.	<ul style="list-style-type: none"> ■ Menu: <i>Design – Connect</i> ■ Click the <i>Connect</i> button¹.
cover	Add a cover directive for signals in the design to the current task. Also, remove, modify, convert, list, and so forth.	<ul style="list-style-type: none"> ■ Menu: <i>Application – Task – Add Property</i> ■ Click the <i>Capture Behavior from Waveform Selection</i> button.¹ ■ Click the <i>Add Property</i> button.¹
custom_gui_action	Add, remove, or list custom button(s) to or from GUI windows.	
database	Access and manipulate the Executable Specification database	Refer to “Executable Specification App” in the <i>JasperGold Apps User’s Guide</i>
dglob	Sort and list files.	None
edit	Edit a specified file.	None
elaborate	Synthesize and read the netlist.	<ul style="list-style-type: none"> ■ Menu: <i>Design – Elaborate RTL</i> ■ Click the <i>Elaborate RTL</i> button.¹

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-3 General Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>exit</u>	Exit the current session.	<ul style="list-style-type: none"> ■ Menu: <i>File – Exit</i> ■ Click the <i>Close (x)</i> button in the top right corner of the console title bar.
<u>export</u>	Export Tcl properties as SVA or PSL requirements files.	Menu: <i>File – Export</i>
<u>formal_bring_up</u>	Use the BPS App to create an initial formal environment.	None
<u>fsm_checks</u>	Generate properties to check the reachability of state machine transitions/crosses.	None
<u>get_annotation</u>	Print the user-defined annotation for a property or signal.	<ul style="list-style-type: none"> ■ Click the <i>Executable Specification</i> tab and view the document pane ■ Visualize window – Use the mouse to hover over an entry in the <i>Waveforms</i> pane to view the annotation in a tooltip.
<u>get_clock_info</u>	Provide detailed information about the clock configuration.	None
<u>get_decomposed_expression</u>	Print a list of expressions for the sub-expressions of the provided argument.	None

JasperGold Apps Command Reference Manual

Commands and Variables Overview

Table 3-3 General Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_design_info</u>	Report design information.	<p>You can access the Design Information window from the context menu (right-click) from any one of multiple points:</p> <ul style="list-style-type: none"> ■ JasperGold Apps console, for example, the <i>Property Table</i> ■ Visualize window, for example, from the <i>Design</i> toolbar ■ Source Browser, for example, with the <i>Source</i> pane context menu (right-click) ■ Complexity Manager window, for example, from the <i>Action</i> menu ■ ProofGrid Manager window, for example, the <i>Properties</i> pane context menu.
<u>get_design_space_coverage</u>	Report design space coverage information.	<p>You can access the Design Space Coverage window from the toolbar, menu, or context menu (right-click) from any one of multiple points.</p>
<u>get_fanin</u>	Report the signals in the fanin of a signal or property.	None
<u>get_fanout</u>	Report the signals in the immediate fanout of the specified signal.	None

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-3 General Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_filename</u>	Report the file name for a specified module or instance name.	None
<u>get_flop_info</u>	Report information about signals related to the input flop.	None
<u>get_latch_info</u>	Report information about a latch, excluding reset information.	None
<u>get_message</u>	Return a list of user messages or the number of occurrences.	None
<u>get_needed_assumptions</u>	Find the minimal set of assumptions to achieve a target status.	None
<u>get_proj_dir</u>	Report the location of the project directory.	None
<u>get_property_info</u>	Report a list of key/value pairs with information about the property.	None
<u>get_reset_info</u>	Get a reset VCD or customized report.	None
<u>get_root_name</u>	Report the name of the root of the instance tree.	None
<u>get_signal_info</u>	Report information about a specified signal.	None
<u>get_signal_list</u>	Report a list of unique signals.	None
<u>get_status</u>	Report the aggregated proof status of a set of properties.	<ul style="list-style-type: none"> ■ Click the <i>RTL Modules</i> or <i>Task Table</i> tab. ■ Visualize window <i>Visualize Configuration</i> pane options

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-3 General Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_tool</u>	Report the name of the tool running your script.	None
<u>get_top_module</u>	Report the name of the top module that will be formally analyzed.	None
<u>get_tx_attributes</u>	Get the signal attributes of a cover directive that represents a behavior.	None
<u>get_version</u>	Report the version number and date of the product installation you are using.	Menu: <i>Help – About JasperGold</i>
<u>help</u>	<p>Return information about all commands or a specific command.</p> <p>Return information about all Error, Warning, and Info messages or a specific message.</p>	<p>Menu: <i>Help –</i></p> <ul style="list-style-type: none"> ■ <i>Command Reference Manual</i> ■ <i>Tcl Command Help</i>
<u>include</u>	Read and execute a script file of Tcl/JasperGold Apps commands.	<ul style="list-style-type: none"> ■ Menu: <i>File – Tcl Scripts</i> ■ Click the <i>Source</i> button.¹ ■ Click the <i>Source Recent/Database Script</i> button¹ to run the last script. ■ Click the arrow adjacent to the <i>Source Recent/Database Script</i> button¹ and choose a script from the list.
<u>interrupt</u>	Make a script interruptible.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-3 General Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>ipk</u>	Analyze an IPK and extract or load proof kit files.	None
<u>ipxact</u>	Print the requested IP-XACT information or execute the requested action.	None
<u>jasper_model_divider</u>	Manage the jasper_model_divider PA connection and configuration.	None
<u>jasper_model_mpram</u>	Manage the jasper_model_mpram PA connection and configuration.	None
<u>jasper_model_multiplier</u>	Manage the jasper_model_multiplier PA connection and configuration.	None
<u>jasper_model_ram</u>	Manage the jasper_model_ram PA connection and configuration.	None
<u>jasper_scoreboard_3</u>	Load definitions to configure jasper_scoreboard_3 PA.	None

JasperGold Apps Command Reference Manual

Commands and Variables Overview

Table 3-3 General Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>justify</u>	Set up the proof to justify the values in the specified signals.	<p><i>Justify</i></p> <p>Visualize Window –</p> <ul style="list-style-type: none">■ Click the <i>Add to Task Context</i> button¹ and then the <i>Justifies</i> tab.■ <i>Waveforms</i> signal pane: Right-click and choose <i>Add Justify</i>.■ Drag and drop a signal into the <i>Task Context</i> pane. <p><i>Justify/Justify Back To</i></p> <ul style="list-style-type: none">■ Complexity Manager – <i>Tree</i> pane: Right-click and choose <i>Justify</i> or <i>Justify Back To</i>. <p>Note: Justifies remain in the background until you use the <i>Set Local Task Context as AR</i> feature.</p>

Continued below.

JasperGold Apps Command Reference Manual

Commands and Variables Overview

Table 3-3 General Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>justify</u> (Continued)	Set up the proof to justify the values in the specified signals.	<p><i>Justify Conflict</i></p> <ul style="list-style-type: none"> ■ Visualize window – <i>Waveforms</i> cycle pane: Right-click and choose <i>Add Justify Conflict</i> <p>Note: Justifies remain in the background until you use the <i>Set Local Task Context as AR</i> feature.</p>
<u>liberty</u>	Load the specified Liberty file.	None
<u>load_radix_file</u>	Load a radix file.	Visualize window <i>View</i> menu – <i>File – Load Radix File</i>
<u>pretty_print</u>	Display any Tcl list in a more easy-to-read format.	None
<u>proof_accelerator</u>	Get information about Proof Accelerators (including Formal Scoreboards).	Menu: <i>Help – Proof Accelerator Datasheets</i>
<u>prove</u>	Prove properties or control a <code>prove</code> command running in the background.	<ul style="list-style-type: none"> ■ Menu: <i>Application – Proof – Prove All</i> ■ Click the <i>Prove All</i> button.¹ ■ <i>Task Table:</i> Right-click and choose <i>Prove Task</i>. ■ <i>Property Table:</i> Right-click and choose <i>Prove Task</i> or <i>Prove Property</i>.
<u>redirect</u>	Redirect <code>stdout</code> to a file or Tcl variable.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-3 General Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>remove_tx_attribute</u>	Remove the specified attribute from the specified cover.	None
<u>report</u>	Report analysis results.	None
<u>reset</u>	Specify and report the global, design reset condition.	<ul style="list-style-type: none"> ■ Menu: <i>Design – Reset</i> ■ Click the Reset button.¹
<u>restore</u>	Restore a session saved by the save command.	<ul style="list-style-type: none"> ■ Menu: <i>File – Open Database</i> or <i>File – Elaborated Design – Restore Elaborated Design</i> ■ Click the <i>Open Database</i> button.¹
<u>sanity_check</u>	Facilitate the successful setup of the global environment.	None
<u>save</u>	Save the current session.	<ul style="list-style-type: none"> ■ Menu: <i>File – Save Database</i> or <i>File – Elaborated Design – Save Elaborated Design</i> ■ Click the <i>Save Database</i> button.¹
<u>scan</u>	Scan traces and extract property candidates.	Menu: <i>Application – Scan</i>
<u>schematic_viewer</u>	Draw a schematic of a portion of or an entire loaded design.	None
<u>scope</u>	Define the scope for property extraction.	Menu: <i>Application – Scope</i>
<u>session</u>	Manipulate analysis sessions.	Various
<u>set_annotation</u>	Create an annotation for a property or signal.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-3 General Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_current_gui</u>	Set the current GUI view to the specified app view.	<ul style="list-style-type: none"> ■ Menu: <i>View – Applications...</i> ■ Click the <i>Other Applications</i> button.¹
<u>set_message</u>	Manipulates warning and info messages or changes message severity.	None
<u>show_list</u>	Format list values returned by a Tcl command.	None
<u>show_schematic</u>	Show a flat or hierarchical schematic view of the design.	<p>For a <i>hierarchical</i> schematic: Go to the Browser window – <i>Design Hierarchy</i> pane, right-click, and choose <i>Show Schematic</i>.</p> <p>For a <i>flattened</i> schematic: Go to a Visualize window –</p> <ul style="list-style-type: none"> ■ <i>Tools – Schematic Viewer</i> ■ Click a schematic viewer button¹
<u>show_source_browser</u>	Show the Source Browser window.	<ul style="list-style-type: none"> ■ Menu: <i>Window – Source Browser</i> ■ <i>Property Table</i> – right-click and choose <i>View Source</i>
<u>show_waveform</u>	Show the current trace or a VCD file in Synopsys Verdi nWave viewer.	<p>Visualize window –</p> <ul style="list-style-type: none"> ■ <i>Tools – Waveform Viewer</i> ■ Click the <i>Show waveform</i> button.¹

JasperGold Apps Command Reference Manual

Commands and Variables Overview

Table 3-3 General Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>stopat</u>	Stop traversing a netlist at the specified signals or instances.	<p>Menu – <i>Application – Task – Add Stopat</i></p> <p>Task Context dialog –</p> <ul style="list-style-type: none">■ <i>Stopats</i> tab – Click the <i>New</i> button. <p>Complexity Manager –</p> <ul style="list-style-type: none">■ <i>Tree</i> pane: Right-click and choose <i>Add Hard Stopat</i> <p>Visualize window –</p> <ul style="list-style-type: none">■ Click the <i>Add to Task Context</i> button¹ and then the <i>Stopats</i> tab.■ Menu: <i>Visualize – Add to Task Context</i>■ Drag and drop a signal into the <i>Task Context</i> pane.■ Right-click on one or more signal names and choose <i>Add Stopat</i>. <p>Note: In the Complexity Manager and Visualize window, stopats remain in the background until you use the <i>Set Local Task Context as AR</i> feature.</p>

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-3 General Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>suggest</u>	Analyze the design and suggest justifications or assumptions.	Visualize window – <ul style="list-style-type: none"> ■ <i>Tools – Suggest Actions on Conflicts and Boundaries</i> ■ Click the <i>Suggest Actions on Conflicts and Boundaries</i> button.¹
<u>task</u>	Create, set, remove, link, and report on a task.	<ul style="list-style-type: none"> ■ <i>Task Table</i> – right-click and choose any of the following: <ul style="list-style-type: none"> <input type="checkbox"/> <i>Create Task</i> <input type="checkbox"/> <i>Remove Task</i> <input type="checkbox"/> <i>Clear Task</i> <input type="checkbox"/> <i>Task Context</i> <input type="checkbox"/> <i>Set as Current Task</i> ■ <i>Property Table</i> – right-click and choose <i>Create Task</i>
<u>template_property</u>	Load and manipulate property templates.	None
<u>trace</u>	Display a signal trace (waveform).	None
<u>view</u>	View the contents of a file.	Click the <i>Executable Specification</i> tab, locate a file name in the document pane, right-click and choose <i>Open</i> .

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-3 General Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>visualize</u>	Generate a trace to Visualize a requirement with or without the design or to Visualize a design scenario.	Refer to Chapter 4, “Visualize” in the <i>JasperGold Apps User’s Guide</i> .
<u>waveform</u>	Manipulate waveforms loaded from trace files or PLI simulation runs.	None

1. To see the names of buttons, rest your cursor over the button in the active window.

Table 3-4 Configuration Command Summary

Command	Description	Equivalent GUI Action
<u>get_analyze_libnonamehide</u>	Report the current setting for analyze_libnonamehide, on, off, or default.	None
<u>get_analyze_librescan</u>	Report the current setting for analyze_librescan, on or off.	None
<u>get_analyze_libunboundsearch</u>	Report whether the analyze command will clear -L options when in multi- or single-file compilation unit (MFCU and SFCU) mode.	None
<u>get_arch_prll_property_consolidation</u>	Report the current setting for the generation of parallel properties.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get auto disable related covers</u>	Report whether enabled related covers will be disabled along with their main properties.	None
<u>get auto syn first trace attempt</u>	Report the length of the traces that auto_syn will generate.	None
<u>get auto syn structural trace generator</u>	Report whether traces will be based on the specified target property or on structural analysis.	None
<u>get automatic library search</u>	Report whether searches for modules in user-declared libraries for analyze are enabled.	None
<u>get cache proof simplification</u>	Report whether proof simplification caching is enabled.	None
<u>get cache reset values</u>	Report whether reset state caching is enabled for VCDs.	None
<u>get capture elaborated design</u>	Report whether the tool will save the netlist for the elaborated design.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_cell_define_debug_visibility</u>	Report whether design traversals consider results that are inside cell defines as valid results.	None
<u>get_clock_auto_stabilize</u>	Report whether support for RTL with potential race hazards is enabled.	None
<u>get_clock_handle_self_disabling_registers</u>	Report whether tool supports flops that gate their own clock.	None
<u>get_clock_process_reconvergent_logic</u>	Report the current setting for processing reconvergent logic.	None
<u>get_clock_structural_glitch_analysis</u>	Report whether structural glitch analysis is enabled.	None
<u>get_cmd_time_limit</u>	Report the time limit for each command.	None
<u>get_command_license_wait_interval</u>	Return the current time interval between license checkout attempts.	None
<u>get_command_license_wait_retries</u>	Return the number of checkout retries the tool will attempt before failing.	None
<u>get_complexity_manager_allow_link_expansion</u>	Report whether Complexity Manager is set to allow link (underlined) node expansion.	Complexity Manager View menu – <i>Allow Link Expansion</i>

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_complexity_manager_compute_statistics</u>	Report whether Complexity Manager is set to compute design information for all nodes as soon as they are added to the tree.	Complexity Manager View menu – <i>Compute Statistics for All</i>
<u>get_complexity_manager_design_information_mode</u>	Report the computing mode for Complexity Manager statistics.	Complexity Manager Action menu – <i>Design Information Mode</i>
<u>get_complexity_manager_show_connected_assumptions</u>	Report whether the Complexity Manager tree is set to display connected assumptions.	Complexity Manager View menu – <i>View Connected Assumptions</i>
<u>get_complexity_manager_show_inputs</u>	Report whether the Complexity Manager tree is set to display inputs.	Complexity Manager View menu – <i>View Inputs</i>
<u>get_complexity_manager_show_link_statistics</u>	Report whether Complexity Manager is set to display design information for link (underlined) nodes.	Complexity Manager View menu – <i>Show Link Statistics</i>
<u>get_counter_speculative_detection</u>	Report whether the tool has been set to use more permissive counter detection rules.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_design_exploration_include_clock_logic</u>	Report whether Design Information, Design Space Coverage, and Complexity Manager include clock logic in their computations.	Design Information, Design Space Coverage, and Complexity Manager View menus – <i>Include Clock Logic</i>
<u>get_design_exploration_include_connected_assumptions</u>	Report whether Design Information, Design Space Coverage, and Complexity Manager include connected assumptions in their computations.	Design Information, Design Space Coverage, and Complexity Manager View menus – <i>Include Connected Assumptions</i>
<u>get_dst_mode</u>	Display the Design Tunneling mode.	Visualize window <i>Visualize Configuration</i> pane options
<u>get_elab_time_limit</u>	Report the maximum time allowed for a single elaborate command.	None
<u>get_elaborate_quiet_trace</u>	Report whether elaborate will automatically generate global QuietTrace soft constraints.	None
<u>get_engine_job_timers</u>	Report the current engine_job_timers value.	None
<u>get_engine_mode</u>	Report the current engine_mode.	None
<u>get_engine_solver</u>	Report the currently specified engine_solver.	None
<u>get_engine_threads</u>	Report the current upper limit on the number of threads used by each engine.	None

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_engineB_before_first_mode</u>	Report how engine B will treat any cycles before its first trace attempt.	None
<u>get_engineB_before_increment_mode</u>	Report how engine B treats any cycles between consecutive trace attempts	None
<u>get_engineB_first_trace_attempt</u>	Report the cycle at which engine B will make its first trace attempt.	None
<u>get_engineB_single_property</u>	Report the <code>single_property</code> setting for engine B.	None
<u>get_engineB_trace_attempt_increment</u>	Report the increment engine B uses between one trace attempt and the next.	None
<u>get_engineB_trace_attempt_time_limit</u>	Report the limit on the amount of time engine B spends on each trace attempt.	None
<u>get_engineB_trace_attempt_time_limit_incr</u>	Return the trace attempt time limit increment for engine B.	None
<u>get_engineC_optimization</u>	Report the optimization level for engines C and C2.	None
<u>get_engineCG_max_mem</u>	Report the maximum memory allocation for building data structures with engines C and G.	None

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_engineD_optimization</u>	Report the optimization level for engines D and I.	None
<u>get_engineG_optimization</u>	Report the optimization level for engines G and G2.	None
<u>get_engineH_single_property</u>	Report the single_property setting for engine H.	None
<u>get_engineJ_attempts</u>	Report the setting for engine J attempts.	None
<u>get_engineJ_max_trace_length</u>	Report the maximum trace length setting for engine J proofs.	None
<u>get_engineJ_migrate</u>	Report whether the engine job will be migrated to Q3 once the engineJ_attempts limit is exceeded.	None
<u>get_engineJ_restarts</u>	Report the number of restarts engine J makes before abandoning the proof.	None
<u>get_engineJ_seed</u>	Report the hexadecimal setting for the seed for engine J.	None
<u>get_engineJ_single_property</u>	Report the single_property setting for engine J.	None
<u>get_engineL_cache</u>	Report whether cover point caching for engine L is enabled.	None
<u>get_engineL_generate_trails</u>	Report whether the generation of trails by engine L is enabled.	None

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_engineL_ignore_trace</u>	Report whether determined targets as sources of intermediate states are kept.	None
<u>get_engineL_max_segment_length</u>	Report the segment length limit.	None
<u>get_engineL_random_diversification</u>	Report the current setting for random diversification in searches between instances of engine L.	None
<u>get_engineL_search_mode</u>	Report whether properties are targeted in order or in parallel.	None
<u>get_engineL_state_removal</u>	Report whether engine L picks start states based on the last-found start states (fifo) or the start states that have been most fruitful (lru).	None
<u>get_engineL_tail_length</u>	Report the starting point for successive engine L trace searches.	None
<u>get_engineQ3_max_trace_length</u>	Report the maximum trace length for engine Q3 when diversification is on.	None
<u>get_engineQ3_random_diversification</u>	Report whether random diversification for engine Q3 is on or off.	None
<u>get_engineQ3_random_diversification_factor</u>	Report the random diversification factor for engine Q3.	None

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_engineQ3_restarts</u>	Report the number of restarts engine Q3 will make before abandoning the proof.	None
<u>get_engineQ3_seed</u>	Report the hexadecimal setting for the seed used by engine Q3.	None
<u>get_export_sva_bind_enable_directive</u>	Report the SVA export bind enable directive.	None
<u>get_export_sva_embed_lib_modules</u>	Report whether the SVA export operation embeds SVA library modules.	None
<u>get_export_sva_handshake_delay_threshold</u>	Report the threshold for converting the maximum delay in handshake properties to \$ during an SVA export.	None
<u>get_export_sva_hierarchy_separator</u>	Get the RTL hierarchy separator character.	None
<u>get_export_sva_hierarchy_separator_escape_string</u>	Get the RTL hierarchy separator replacement string used during SVA export.	None
<u>get_export_sva_reset_condition</u>	Report the reset condition for disabled SVA properties.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_export_sva_target_environment</u>	Report the target validation environment for SVA export.	None
<u>get_export_sva_trace_suffix_cycles</u>	Return the number of cycles added to the cover property.	None
<u>get_export_sva_use_escaped_identifier_with_extended_names</u>	Report whether escaped identifiers will be used with extended property names.	None
<u>get_extract_high_level_fsm</u>	Report whether high-level heuristics are used for FSM extraction	None
<u>get_first_trace_attempt</u>	Report the cycle at which engines will make their first trace attempt.	None
<u>get_function_debug_visibility</u>	Report whether debugging tools such as <i>Why</i> will have visibility into functions.	None
<u>get_log_timestamp_format</u>	Report the current format string used for timestamps in logging messages.	None
<u>get_log_timestamp_mode</u>	Determine whether logging messages will be prepended by a timestamp.	None
<u>get_max_trace_length</u>	Report the maximum length for displayed traces.	None
<u>get_message_history_limit</u>	Return the current limit on user message storage.	None
<u>get_parallel_proof_mode</u>	Report whether parallel proof mode is on or off.	None

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_prompt1</u>	Return the Tcl script used to define the Tcl prompt.	None
<u>get_prompt2</u>	Return the Tcl script used to define the Tcl prompt for incomplete commands.	None
<u>get_proof_simplification</u>	Report whether proof and Visualize simplification is on.	None
<u>get_proofgrid_cluster</u>	Return the sequence of hosts on which proof engine jobs will be spawned.	None
<u>get_proofgrid_engineJ_max_jobs</u>	Return the current setting for engine J jobs.	None
<u>get_proofgrid_engineL_max_jobs</u>	Return the current setting for engine L jobs.	None
<u>get_proofgrid_extra_args</u>	Return the current setting for proofgrid_extra_args.	None
<u>get_proofgrid_manager</u>	Report whether the ProofGrid Manager feature is turned on or off.	None
<u>get_proofgrid_max_jobs</u>	Report the limit for jobs used by ProofGrid.	None
<u>get_proofgrid_max_local_jobs</u>	Report the limit for jobs used by ProofGrid when proofgrid_mode is local.	None
<u>get_proofgrid_mode</u>	Report the value of the variable proofgrid_mode.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_proofgrid_per_engine_max_jobs</u>	Report the maximum number of engine jobs ProofGrid will start per engine.	None
<u>get_proofgrid_per_engine_max_local_jobs</u>	Report the maximum number of engine jobs ProofGrid will start per engine when in local mode.	None
<u>get_proofgrid_per_engine_privileged_jobs</u>	Report the number of privileged licenses.	None
<u>get_proofgrid_queue</u>	Report the queue name for parallel proofs.	None
<u>get_proofgrid_restarts</u>	Report the number of restarts ProofGrid allows per engine job.	None
<u>get_proofgrid_shell</u>	Report the value of the variable proofgrid_shell.	None
<u>get_proofgrid_shell_stop</u>	Report the value of the variable proofgrid_shell_stop.	None
<u>get_proofgrid_skip_abandoned</u>	Report whether a restarted job will return to its abandoned target.	None
<u>get_proofgrid_socket_communication</u>	Report whether communication happens over a TCP/IP socket or standard in/out.	None

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_property_auto_update_db</u>	Report whether the tool is set to automatically update the database when you create properties with the commands assert, assume, and cover.	None
<u>get_property_compile_time_limit</u>	Report the per-property time limit for compiling properties during elaboration.	None
<u>get_property_delay_cycles</u>	Report the current number of delay cycles.	None
<u>get_prove_no_cover_traces</u>	Report whether cover traces are generated during proofs.	None
<u>get_prove_no_traces</u>	Report whether traces are generated during proofs.	None
<u>get_prove_per_property_max_time_limit</u>	Report the single-property engine wall clock time limit for the proof of each property.	None
<u>get_prove_per_property_time_limit</u>	Report the maximum time allowed for proving each property.	None
<u>get_prove_per_property_time_limit_factor</u>	Report the factor for the geometric progression of property_time_limit.	None

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_prove_prefer_shortest</u>	Report whether <code>prove</code> will continue searching for a shorter trace after identifying a non-minimal trace.	None
<u>get_prove_report_mem</u>	Report whether automatic generation of memory consumption messages is <code>on</code> or <code>off</code> .	None
<u>get_prove_report_period</u>	Report the current value of the period used for requesting the memory reports from the proof jobs.	None
<u>get_prove_start_timer_on_engine_start</u>	Report whether the timer starts only after the first proof job starts.	None
<u>get_prove_time_limit</u>	Report the maximum time allowed for the <code>prove</code> command.	None
<u>get_prove_verbosity</u>	Get the verbosity level for the <code>prove</code> command.	None
<u>get_proven_directive</u>	Report whether proven directives are used as assumptions.	None
<u>get_reset_trace_inputs</u>	Report whether saving input values in the saved reset trace from a file is enabled.	None
<u>get_reset_vcd_fsdb_force_split_scope</u>	Report whether forced scope splitting with the dot character is enabled.	None

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_scan_constant_relations_seeker</u>	Report whether cross segment constant properties will be generated.	None
<u>get_scan_default_assume_representation_policy</u>	Report the default representation for assumptions generated for behavioral property synthesis.	None
<u>get_scan_default_property_representation</u>	Report the default property representation for behavioral property synthesis.	None
<u>get_scan_no_sim_max_covers</u>	Report the current maximum number of covers used by the BPS App in the scan -no_sim flow.	None
<u>get_scan_per_scope_time_limit</u>	Report the time limit for scanning each scope.	None
<u>get_scan_seek_depth</u>	Report the maximum temporal depth for property synthesis.	None
<u>get_scan_seek_mode</u>	Report which seekers are enabled.	None
<u>get_scan_trace_stuck_at_values_limit</u>	Return the values limit for the stuck_at properties.	None
<u>get_scan_trace_witness_information</u>	Report whether witness information collection is enabled or disabled.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_scan_verbosity</u>	Report how much information will be printed to the log and console during a scan.	None
<u>get_schematic_viewer_command</u>	Report the current value of the command used to launch the schematic viewer.	None
<u>get_schematic_viewer_connection_timeout</u>	Report the current setting for the schematic viewer connection time-out.	None
<u>get_schematic_viewer_design_script</u>	Report the path to the script used to import the design in the schematic viewer.	None
<u>get_scope_clock</u>	Get the overwrite clock configuration.	None
<u>get_scope_clock_edge</u>	Get the overwrite clock edge configuration.	None
<u>get_scope_default_active_phase</u>	Get the default active phase configuration.	None
<u>get_scope_default_clock</u>	Get the default clock configuration.	None
<u>get_scope_default_clock_edge</u>	Get the default clock edge configuration.	None
<u>get_scope_extract_min_ports_threshold</u>	Get the minimum port count threshold for module interface extraction.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_scope_signal_width_threshold</u>	Get the maximum signal width threshold for module, module interface, and user-defined POIs.	None
<u>get_sec_auto_prove_prequalification_efforts</u>	Get the effort level setting for the SEC autoprove prequalification process.	None
<u>get_sec_autoprove</u>	Report whether SEC check_sec -prove command uses autoprove or prove.	None
<u>get_sec_autoprove_cutpoints_on_internal_signals</u>	Report whether SEC check_sec -prove command adds cutpoints on internal signals.	None
<u>get_sec_autoprove_cutpoints_on_latches</u>	Report whether SEC check_sec -prove command adds cutpoints on latches.	None
<u>get_sec_autoprove_max_number_of_cutpoints</u>	Report the maximum number of cutpoints SEC adds on signals when running check_sec -prove in autoprove mode.	None

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_sec_autoprove_mode</u>	Report which mode of cutpoint refinement loop handling SEC uses with the <code>check_sec-prove</code> command.	None
<u>get_sec_prove_assumption_lifting</u>	Report whether SEC will include only a relevant subset of assumptions sufficient for the proof.	None
<u>get_sec_prove_levelize_helpers</u>	Report whether SEC will run a level-by-level proof.	None
<u>get_sec_prove_order_targets</u>	Report whether SEC will attempt to order the <code>prove</code> dispatch for the helper assertions based on internal ranking.	None
<u>get_sec_prove_suppress_traces</u>	Report whether SEC will suppress trace generation.	None
<u>get_sps_arithmetic_overflow_style</u>	Report the style of the arithmetic overflow checks (statement or expression).	None
<u>get_sps_deadcode_full_case_filter</u>	Report whether SPS generates dead code checks for defaults of case if the case is already full.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_sps_no_duplicates_per_check</u>	Report whether duplication POIs are allowed.	None
<u>get_sps_signals_flops</u>	Report whether generation of stuck-at and toggle properties is limited to flops of selected signals.	None
<u>get_sps_signals_max_bits</u>	Report the max bits on which stuck-at and toggle properties are generated when <code>sps_stuck_at_compact</code> is set to true.	None
<u>get_sps_signals_outputs</u>	Report whether generation of stuck-at and toggle properties is limited to outputs of selected instances.	None
<u>get_sps_stuck_at_compact</u>	Report whether bit blasting of stuck-at properties is allowed.	None
<u>get_sps_sva_max_depth</u>	Report the current maximum depth for SVA expressions.	None
<u>get_sps_sva_max_length</u>	Report the current maximum length for SVA expressions	None
<u>get_spv_include_control_path</u>	Report whether control paths will be included in the data propagation analysis.	None
<u>get_sst_default_trace_length</u>	Get the current state-space tunneling (SST) default trace length.	None

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_sst_undetermined_helpers</u>	Get the current SST mode for undetermined helper assertions.	None
<u>get_stop_on_cex_limit</u>	Report the value of the stop_on_cex_limit variable.	None
<u>get_stop_on_unreachable_limit</u>	Report the stop on unreachable limit.	None
<u>get_tag_irrelevant_values</u>	Report the current time setting for whether irrelevant values are tagged as X in the Visualize window.	None
<u>get_task_compile_time_limit</u>	Report the per-task time limit for compiling properties during elaboration.	None
<u>get_tcl_no_precondition</u>	Report whether the tool will automatically generate precondition covers for liveness assertions created with the assert command.	None
<u>get_time_format</u>	Report the current time format setting.	None
<u>get_trace_extension</u>	Report the value of the trace_extension variable.	None
<u>get_trace_optimization</u>	Report whether trace optimization is enabled.	None
<u>get_trace_show_reset</u>	Report whether showing the reset sequence as a prefix to traces is enabled.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_tunnel_library_cell</u>	Report whether Design Tunneling is enabled for designs with library cells.	None
<u>get_visualize_auto_check_props</u>	Report whether Visualize is set to automatically check property status for indexed behaviors.	None
<u>get_visualize_auto_load_debugging_tables</u>	Report whether Visualize is set to automatically load debugging tables.	None
<u>get_waveform_import_multiple_segments</u>	Report whether the tool extracts properties over multiple non-reset segments of the same trace.	None
<u>get_waveform_import_secondary_hier_path</u>	Determine the path to the signals not specified in the waveform or scan commands.	None
<u>get_word_level_reduction</u>	Determine if the proof flow will be run with word-level transformations or reductions.	None
<u>get_xprop_compute_highlights</u>	Determine whether X-Propagation highlights are enabled or disabled	None
<u>get_xprop_use_all_undriven</u>	Determine whether the tool treats all undriven nets as sources of X during X-Prop checks.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>get_xprop_use_bbox_outputs</u>		
	Determine whether the tool treats black-box outputs as sources of Xs during X-Prop checks.	None
<u>get_xprop_use_inputs</u>	Determine whether the tool treats primary inputs as sources of X during X-Prop checks.	None
<u>get_xprop_use_internal_undriven</u>		
	Determine whether the tool treats internal undriven nets as sources of X during X-Prop checks.	None
<u>get_xprop_use_low_power</u>	Determine whether the tool will treat flop reset abstractions as source of X.	None
<u>get_xprop_use_reset_abstraction</u>		
	Determine whether the tool considers Xs generated by low-power configurations as sources of X	None
<u>get_xprop_use_reset_state</u>	Determine whether the tool uses uninitialized registers as sources of X during X-Prop checks.	None
<u>get_xprop_use_stopats</u>	Determine whether the tool treats stopat nets as sources of X during X-Prop checks.	None
<u>get_xprop_use_x_assignments</u>		
	Determine whether the tool treats Xs as sources of X during X-Prop checks.	None

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_analyze_libnonamehide</u>	Enable or disable storing read modules in a temporary directory during compilation.	None
<u>set_analyze_librescan</u>	Enable or disable directory re-scanning to find Verilog modules.	None
<u>set_analyze_libunboundsearch</u>	Enable cross-file compilation unit -L option.	None
<u>set_arch_prll_property_consolidation</u>	Generate one parallel property instead of n choose 2 parallel properties.	None
<u>set_auto_disable_related_covers</u>	Specify whether enabled related covers will be disabled along with their main properties.	None
<u>set_auto_syn_first_trace_attempt</u>	Specify the length of the traces that auto_syn will generate for the specified target property.	None
<u>set_auto_syn_structural_trace_generator</u>	Enable trace generation based on structural analysis when you specify a target property with auto_syn.	None
<u>set_automatic_library_search</u>	Enable or disable searches for modules in user-declared libraries for analyze.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_cache_proof_simplification</u>	Enable or disable proof simplification caching.	Proof Settings dialog – <i>General</i> tab – <i>Cache proof simplification results</i>
<u>set_cache_reset_values</u>	Enable or disable reset value caching for VCDs.	None
<u>set_capture_elaborated_design</u>	Enable capturing the elaborated netlist to speed up subsequent design reuse.	<ul style="list-style-type: none"> ■ Menu: <i>File – Elaborated Design – Capture Elaborated Design</i> ■ In the session control panel toolbar, click the down arrow adjacent to the <i>Launch New Session</i> button and choose <i>Capture Elaborated Design</i>
<u>set_cell_define_debug_visibility</u>	Consider cell define blocks in design traversals.	None
<u>set_clock_auto_stabilize</u>	Enable support for RTL with potential race hazards due to looping clocking signals.	None
<u>set_clock_handle_self_disabling_registers</u>	Disable support for flops that gate their own clock.	None
<u>set_clock_process_reconvergent_logic</u>	Enable processing and removal of reconvergent logic.	None

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_clock_structural_glitch_analysis</u>	Enable or disable structural glitch analysis.	None
<u>set_cmd_time_limit</u>	Sets the time limit for most Tcl commands.	None
<u>set_command_license_wait_interval</u>	Set the time interval between license checkout attempts.	None
<u>set_command_license_wait_retries</u>	Set the number of checkout retries the tool should attempt before failing.	None
<u>set_complexity_manager_allow_link_expansion</u>	Disable link (underlined) node expansion. Default is on.	Complexity Manager View menu – <i>View – Allow Link Expansion</i>
<u>set_complexity_manager_compute_statistics</u>	Compute design statistics for all nodes. Default is off.	Complexity Manager View menu – <i>View – Compute Statistics for All</i>
<u>set_complexity_manager_design_information_mode</u>	Disable design information mode. Default is on.	Complexity Manager Action menu – <i>Action – Design Information Mode</i>

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_complexity_manager_show_connected_assumptions</u>	Hide connected assumptions in Complexity Manager tree. Default is on.	Complexity Manager View menu – <i>View – View Connected Assumptions</i>
<u>set_complexity_manager_show_inputs</u>	Hide inputs in Complexity Manager tree. Default is on.	Complexity Manager View menu – <i>View – View Inputs</i>
<u>set_complexity_manager_show_link_statistics</u>	Hide design information for link (underlined) nodes. Default is on.	Complexity Manager View menu – <i>View – Show Link Statistics</i>
<u>set_counter_speculative_detection</u>	Enable (or disable) the speculative counter detection algorithm.	None
<u>set_design_exploration_include_clock_logic</u>	Specify whether Design Information, Design Space Coverage, and Complexity Manager include clock logic in their computations.	Design Information, Design Space Coverage, and Complexity Manager window menus – <i>View – Include Clock Logic</i>

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_design_exploration_include_connected_assumptions</u>	Specify whether Design Information, Design Space Coverage, and Complexity Manager include connected assumptions in their computations.	Design Information, Design Space Coverage, and Complexity Manager window menus – <i>View – Include Connected Assumptions</i>
<u>set_dst_mode</u>	Enable or disable Design Tunneling.	Visualize dialog – Click <i>DST</i>
<u>set_elab_time_limit</u>	Specify the maximum time to spend on a single elaborate command.	None
<u>set_elaborate_quiet_trace</u>	Specify whether elaborate will automatically generate global QuietTrace soft constraints.	None
<u>set_engine_job_timers</u>	Specify the timer type used for engine jobs running proofs.	None
<u>set_engine_mode</u>	Optimize the proof process for different types of designs and requirements.	<ul style="list-style-type: none"> ■ Menu: <i>Application – Proof – Proof Settings</i> ■ Click the <i>Proof Settings</i> button.¹
<u>set_engine_solver</u>	Specify the solver you want the engines to use in the decision procedure for the Boolean satisfiability problem.	None
<u>set_engine_threads</u>	Specify the upper limit on the number of threads used by each engine.	Advanced Engine Settings dialog – <i>Parallelism</i> section

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_engineB_before_first_mode</u>	Specify how engine B will treat any cycles before its first trace attempt.	Advanced Engine Settings dialog – <i>Engine B</i> section
<u>set_engineB_before_increment_mode</u>	Specify how engine B treats any cycles between consecutive trace attempts.	Advanced Engine Settings dialog – <i>Engine B</i> section
<u>set_engineB_first_trace_attempt</u>	Specify the cycle at which engine B will make its first trace attempt.	Advanced Engine Settings dialog – <i>Engine B</i> section
<u>set_engineB_single_property</u>	Specify multi- or single-property mode for engine B.	Advanced Engine Settings dialog – <i>Engine B</i> section
<u>set_engineB_trace_attempt_increment</u>	Specify the increment engine B uses between one trace attempt and the next.	Advanced Engine Settings dialog – <i>Engine B</i> section
<u>get_engineB_trace_attempt_time_limit</u>	Specify the limit on the amount of time engine B spends on each trace attempt.	None
<u>set_engineB_trace_attempt_time_limit_incr</u>	Specify the time limit increment for engine B when the specified limit for trace attempts expires	None
<u>set_engineC_optimization</u>	Specify the optimization mode for engines C and C2.	Advanced Engine Settings dialog – <i>Engines C, C2, G and G2</i> section

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_engineCG_max_mem</u>	Specify the maximum memory allocation for engine C and G data structures.	Advanced Engine Settings dialog – <i>Engines C, C2, G and G2</i> section
<u>set_engineD_optimization</u>	Specify the optimization mode for engines D and I.	Advanced Engine Settings dialog – <i>Engine D</i> section
<u>set_engineG_optimization</u>	Specify the optimization mode for engines G and G2.	Advanced Engine Settings dialog – <i>Engines C, C2, G and G2</i> section
<u>set_engineH_single_property</u>	Specify multi- or single-property mode for engine H.	Advanced Engine Settings dialog – <i>Engines Hp, Ht, and H</i> section
<u>set_engineJ_attempts</u>	Specify the maximum number of engine J attempts to solve all constraints at every clock cycle before restarting.	Advanced Engine Settings dialog – <i>Engine J</i> section
<u>set_engineJ_max_trace_length</u>	Specify an upper limit on the length of traces searched for by engine J.	Advanced Engine Settings dialog – <i>Engine J</i> section
<u>set_engineJ_migrate</u>	Enable (or disable) engine job handover to Q3 if the engineJ_attempts limit is exceeded.	None
<u>set_engineJ_restarts</u>	Specify the number of restarts engine J makes before abandoning the proof.	Advanced Engine Settings dialog – <i>Engine J</i> section
<u>set_engineJ_seed</u>	Specify a hexadecimal setting for the seed used by engine J.	Advanced Engine Settings dialog – <i>Engine J</i> section
<u>set_engineJ_single_property</u>	Specify multi- or single-property mode for engine J.	Advanced Engine Settings dialog – <i>Engine J</i> section

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_engineL_cache</u>	Specify whether to cache cover points for engine L.	Advanced Engine Settings dialog – <i>Engine L</i> section
<u>set_engineL_generate_trails</u>	Enable (or disable) trail generation by engine L.	None
<u>set_engineL_ignore_trace</u>	Specify whether to keep determined targets as sources of intermediate states.	Advanced Engine Settings dialog – <i>Engine L</i> section
<u>set_engineL_max_segment_length</u>	Specify the segment length limit.	Advanced Engine Settings dialog – <i>Engine L</i> section
<u>set_engineL_random_diversification</u>	Specify whether engine L will find different traces for the same property to diversify start states for other properties	Advanced Engine Settings dialog – <i>Engine L</i> section
<u>set_engineL_search_mode</u>	Specify whether engine L will target properties in order or in parallel.	Advanced Engine Settings dialog – <i>Engine L</i> section
<u>set_engineL_state_removal</u>	Specify whether engine L should pick start states based on the last-found start states (<code>fifo</code>) or the start states that have been most fruitful (<code>lru</code>).	Advanced Engine Settings dialog – <i>Engine L</i> section
<u>set_engineL_tail_length</u>	Specify a starting point for a new search that is <i>N</i> cycles before the end of found traces.	Advanced Engine Settings dialog – <i>Engine L</i> section
<u>set_engineQ3_max_trace_length</u>	Specify the trace length upper limit for searches by Q3 when diversification is on.	Advanced Engine Settings dialog – <i>Engine Q3</i> section

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_engineQ3_random_diversification</u>	Enable random diversification for engine Q3.	Advanced Engine Settings dialog – <i>Engine Q3</i> section
<u>set_engineQ3_random_diversification_factor</u>	Specify the random diversification factor used by engine Q3.	Advanced Engine Settings dialog – <i>Engine Q3</i> section
<u>set_engineQ3_restarts</u>	Limit engine Q3 restarts to the specified non-negative integer attempts.	Advanced Engine Settings dialog – <i>Engine Q3</i> section
<u>set_engineQ3_seed</u>	Specify a hexadecimal setting as the seed for engine Q3.	Advanced Engine Settings dialog – <i>Engine Q3</i> section
<u>set_export_sva_bind_enable_directive</u>	Specify a conditional compilation directive for the SVA bind statement.	None
<u>set_export_sva_embed_lib_modules</u>	Specify whether the SVA export operation will embed SVA library modules.	None
<u>set_export_sva_handshake_delay_threshold</u>	Specify the threshold for converting the maximum delay in handshake properties to \$ during an SVA export.	None
<u>set_export_sva_hierarchy_separator</u>	Specify the hierarchy separator when exporting SVA.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_export_sva_hierarchy_separator_escape_string</u>	Specify the string used to replace the RTL hierarchy separator character.	None
<u>set_export_sva_reset_condition</u>	Specify a reset condition on which to disable exported SVA properties.	None
<u>set_export_sva_target_environment</u>	Specify the target validation environment for SVA export.	None
<u>set_export_sva_trace_suffix_cycles</u>	Specify the number of cycles that will be added to the cover property.	None
<u>set_export_sva_use_escaped_identifier_with_extended_names</u>	Specify whether to use escaped identifiers with extended property names.	Menu: <i>Application – Import/Export – Export SVA</i> – Click on <i>Use Extended Property Naming Convention</i>
<u>set_extract_high_level_fsm</u>	Disable or re-enable high-level heuristics for FSM extraction.	None
<u>set_first_trace_attempt</u>	Specify the cycle at which engines will make their first trace attempt.	Proof Settings dialog – <i>General</i> tab
<u>set_function_debug_visibility</u>	Enable (or disable) debugging for functions.	None
<u>set_log_timestamp_format</u>	Specify the format for the timestamp of logging messages.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_log_timestamp_mode</u>	Specify whether logging messages will be prepended by a timestamp.	None
<u>set_max_trace_length</u>	Specify the limit for the length of traces to search for.	Proof Settings dialog – <i>General</i> tab – <i>Limits</i> section
<u>set_message_history_limit</u>	Set the limit of stored user messages issued in the current session.	None
<u>set_parallel_proof_mode</u>	Enable parallel proof mode.	None
<u>set_prompt1</u>	Set the Tcl script used to define the Tcl prompt.	None
<u>set_prompt2</u>	Set the Tcl script used to define the Tcl prompt for incomplete commands.	None
<u>set_proof_simplification</u>	Turn off (or on) proof simplifications.	Proof Settings dialog – <i>General</i> tab
<u>set_proofgrid_cluster</u>	Specify the sequence of hosts on which to spawn proof engine jobs.	Proof Settings dialog – <i>ProofGrid</i> tab
<u>set_proofgrid_engineJ_max_jobs</u>	Control the number of engine J jobs.	None
<u>set_proofgrid_engineL_max_jobs</u>	Control the number of engine L jobs.	None
<u>set_proofgrid_extra_args</u>	Set extra arguments that will be supplied to the active proofgrid_mode.	Proof Settings dialog – <i>ProofGrid</i> tab

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_proofgrid_manager</u>	Turn on (or off) the ProofGrid Manager feature.	ProofGrid Manager – Click the <i>Enable</i> (or <i>Disable</i>) button
<u>set_proofgrid_max_jobs</u>	Specify the limit for jobs used by ProofGrid.	Proof Settings dialog – <i>ProofGrid</i> tab
<u>set_proofgrid_max_local_jobs</u>	Specify the limit for jobs used by ProofGrid when <i>proofgrid_mode</i> is local.	Proof Settings dialog – <i>ProofGrid</i> tab
<u>set_proofgrid_mode</u>	Set the grid on which to run parallel proofs.	Proof Settings dialog – <i>ProofGrid</i> tab
<u>set_proofgrid_per_engine_max_jobs</u>	Specify the limit for engine jobs started by ProofGrid per engine	Proof Settings dialog – <i>ProofGrid</i> tab
<u>set_proofgrid_per_engine_max_local_jobs</u>	Specify an additional limit, beyond per engine max jobs, for engine jobs per engine started by ProofGrid when <i>proofgrid_mode</i> is local.	Proof Settings dialog – <i>ProofGrid</i> tab
<u>set_proofgrid_per_engine_privileged_jobs</u>	Specify the number of non-nice licenses used by ProofGrid.	Proof Settings dialog – <i>ProofGrid</i> tab
<u>set_proofgrid_queue</u>	Set the queue name on which to run parallel proofs.	Proof Settings dialog – <i>ProofGrid</i> tab
<u>set_proofgrid_restarts</u>	Specify the number of restarts ProofGrid allows per engine job.	Proof Settings dialog – <i>ProofGrid</i> tab
<u>set_proofgrid_shell</u>	Set the command used to launch ProofGrid jobs.	Proof Settings dialog – <i>ProofGrid</i> tab

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_proofgrid_shell_stop</u>	Set the command used to clean up ProofGrid jobs when proofgrid_mode is shell.	Proof Settings dialog – <i>ProofGrid</i> tab
<u>set_proofgrid_skip_abandoned</u>	Specify whether a restarted job will return to its abandoned target.	None
<u>set_proofgrid_socket_communication</u>	Specify whether communication happens over a TCP/IP socket or standard in/out.	Proof Settings dialog – <i>ProofGrid</i> tab
<u>set_property_auto_update_db</u>	Automatically update the database with properties as they are created.	None
<u>set_property_compile_time_limit</u>	Set the per-property time limit for compiling properties during elaboration.	None
<u>set_property_delay_cycles</u>	Specify the number of cycles you would like to delay properties.	None
<u>set_prove_no_cover_traces</u>	Specify whether to generate cover traces during proofs.	None
<u>set_prove_no_traces</u>	Specify whether to generate traces during proofs.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_prove_per_property_max_time_limit</u>	Specify the single-property engine wall clock time limit for the proof of each property.	Proof Settings dialog – <i>General</i> tab – <i>Limits</i> section
<u>set_prove_per_property_time_limit</u>	Specify the wall clock time limit for proving each property.	Proof Settings dialog – <i>General</i> tab – <i>Limits</i> section
<u>set_prove_per_property_time_limit_factor</u>	Control iterative geometric progression of the time limit imposed per property.	Proof Settings dialog – <i>General</i> tab – <i>Limits</i> section
<u>set_prove_prefer_shortest</u>	Control whether <code>prove</code> will continue searching for a shorter trace after identifying a non-minimal trace.	None
<u>set_prove_report_mem</u>	Control the generation of frequent engine job memory consumption messages.	None
<u>set_prove_report_period</u>	Set the sampling period for requesting memory reports from proof processes.	None
<u>get_prove_start_timer_on_engine_start</u>	Start prove command timer only after the first proof job starts.	None
<u>set_prove_time_limit</u>	Specify the maximum time allowed for a <code>prove</code> or <code>visualize</code> command.	Proof Settings dialog – <i>General</i> tab – <i>Limits</i> section
<u>set_prove_verbosity</u>	Specify the verbosity level for the <code>prove</code> command.	None
<u>set_proven_directive</u>	Specify whether to use proven directives as assumptions.	Proof Settings dialog – <i>General</i> tab

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_reset_trace_inputs</u>	Enable or disable saving input values from the saved reset trace that comes from a trace file.	None
<u>set_reset_vcd_fsdb_force_split_scope</u>	Enable or disable forced scope splitting with the dot character.	None
<u>set_scan_constant_relations_seeker</u>	Enable or disable the generation of cross segment constant properties.	None
<u>set_scan_default_assume_representation_policy</u>	Specify the default representation for assumptions generated for behavioral property synthesis.	None
<u>set_scan_default_property_representation</u>	Specify the default property representation for behavioral property synthesis.	None
<u>set_scan_no_sim_max_covers</u>	Specify the maximum number of covers the BPS App will try to use during the <code>scan -no_sim</code> flow.	None
<u>set_scan_per_scope_time_limit</u>	Specify the time limit for scanning each scope.	Configure Scan Settings dialog – <i>Time per scope</i> field
<u>set_scan_seek_depth</u>	Set the maximum temporal depth for property synthesis (applicable to specific seekers only).	Configure Scan Settings dialog – <i>Seek depth</i> field

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_scan_seek_mode</u>	Optimize property synthesis for different types of designs and requirements.	Configure Scan Settings dialog – <i>Seek mode</i> section
<u>set_scan_trace_stuck_at_values_limit</u>	Specify the values limit for the stuck_at properties.	Configure Scan Settings dialog – <i>Trace stuck at values limit</i> field
<u>set_scan_trace_witness_information</u>	Enable or disable witness information collection during scan.	None
<u>set_scan_verbosity</u>	Specify the verbosity level for the scan -trace command.	None
<u>set_schematic_viewer_command</u>	Specify the command to launch the schematic viewer.	None
<u>set_schematic_viewer_connection_timeout</u>	Specify the time to wait for a connection to the schematic viewer.	None
<u>set_schematic_viewer_design_script</u>	Specify the script to import the design in the schematic viewer.	None
<u>set_scope_clock</u>	Set the overwrite clock for all POIs extracted after issue.	None
<u>set_scope_clock_edge</u>	Set the overwrite clock edge for all POIs extracted after issue.	None
<u>set_scope_default_active_phase</u>	Set the default active phase for POI signals.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_scope_default_clock</u>	Set the default clock.	None
<u>set_scope_default_clock_edge</u>	Set the default clock edge.	None
<u>set_scope_extract_min_ports_threshold</u>	Set the minimum port count threshold for module interface extraction.	None
<u>set_scope_signal_width_threshold</u>	Set the maximum signal width threshold for module, module interface, and user-defined POIs.	None
<u>set_sec_auto_prove_prelqualification_efforts</u>	Configure the SEC autoprove prequalification efforts level: normal, high, or medium.	None
<u>set_sec_autoprove</u>	Specify whether SEC check_sec -prove command uses autoprove or prove.	None
<u>set_sec_autoprove_cutpoints_on_internal_signals</u>	Specify whether SEC check_sec -prove command adds cutpoints on internal signals.	None
<u>set_sec_autoprove_cutpoints_on_latches</u>	Specify whether SEC check_sec -prove command adds cutpoints on latches.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_sec_autoprove_max_number_of_cutpoints</u>	Specify the maximum number of cutpoints SEC adds on signals when running <code>check_sec -prove</code> in autoprove mode.	None
<u>set_sec_autoprove_mode</u>	Specify which mode of cutpoint refinement loop handling SEC uses with the <code>check_sec -prove</code> command.	None
<u>set_sec_prove_assumption_lifting</u>	Specify whether SEC will include only a relevant subset of assumptions sufficient for the proof.	None
<u>set_sec_prove_levelize_helpers</u>	Specify whether SEC will run a level-by-level proof.	None
<u>set_sec_prove_order_targets</u>	Specify whether SEC will attempt to order the <code>prove</code> dispatch for the helper assertions based on internal ranking.	None
<u>set_sec_prove_suppress_traces</u>	Specify whether SEC will suppress trace generation.	None
<u>set_sps_arithmetic_overflow_style</u>	Control the style of the arithmetic overflow checks (statement or expression).	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_sps_deadcode_full_case_filter</u>	Specify whether SPS generates dead code checks for defaults of case if the case is already full.	None
<u>set_sps_no_duplicates_per_check</u>	Prevent duplication POIs that would generate duplicate properties within checks.	None
<u>set_sps_signals_flops</u>	Limit the generation of stuck-at and toggle properties to flops of the selected signals.	None
<u>set_sps_signals_max_bits</u>	Limit the max bits on which stuck-at and toggle properties will be generated when <code>sps_stuck_at_compact</code> is set to true.	None
<u>set_sps_signals_outputs</u>	Limit the generation of stuck-at and toggle properties to outputs of the selected instances.	None
<u>set_sps_stuck_at_compact</u>	Control the bit blasting of stuck-at properties.	None
<u>set_sps_sva_max_depth</u>	Specify the current maximum depth (in terms of gate complexity) for SVA expressions.	None
<u>set_sps_sva_max_length</u>	Specify the current maximum length (in number of characters) for SVA expressions.	None
<u>set_spv_include_control_path</u>	Specify whether control paths will be included in the data propagation analysis.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_sst_default_trace_length</u>	Specify the SST default trace length.	None
<u>set_sst_undetermined_helpers</u>	Specify how the tool should handle undetermined helper assertions.	None
<u>set_stop_on_cex_limit</u>	Set the value of the <code>stop_on_cex_limit</code> variable.	None
<u>set_stop_on_unreachable_limit</u>	Set the stop on unreachable limit.	None
<u>set_tag_irrelevant_values</u>	Specify whether irrelevant values are tagged as X in the Visualize window.	Visualize window – ■ Menu – <i>View – Tag Irrelevant Values</i>
<u>set_task_compile_time_limit</u>	Set the per-task time limit for compiling properties during elaboration.	None
<u>set_tcl_no_precondition</u>	Disable the automatic generation of precondition covers for liveness assertions created with the <code>assert</code> command.	None
<u>set_time_format</u>	Specify the time format that appears in messages.	None
<u>set_trace_extension</u>	Extend a trace N extra cycles beyond the cycle that triggers a property.	None
<u>set_trace_optimization</u>	Specify the default trace optimizations or reduce the number of optimizations.	None

JasperGold Apps Command Reference Manual
Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_trace_show_reset</u>	Disable or re-enable the feature to prefix traces with the reset sequence.	None Note: By default, the Visualize window does not display the reset sequence. To display the prefix in the <i>Waveforms</i> pane with negative cycle numbers, use the menu <i>View – Show Reset Cycles</i> .
<u>set_tunnel_library_cell</u>	Enable manual Design Tunneling for designs with library cells.	None
<u>set_visualize_auto_check_props</u>	Compute exercised covers and violated assertions.	Visualize window – <i>Indexed Behaviors</i> tab – Click the <i>Analyze automatically</i> button
<u>set_visualize_auto_load_debugging_tables</u>	Automatically load debugging tables.	Visualize window – <i>Debugging Table</i> – Click the <i>Load automatically</i> button
<u>set_waveform_import_multiple_segments</u>	Enable or disable property extraction over multiple non-reset segments of a trace.	Configure Scan Settings dialog – <i>Waveform import multiple segments</i> check box
<u>set_waveform_import_secondary_hier_path</u>	Set a secondary hierarchical path to signals not specified in the <code>waveform</code> or <code>scan</code> commands.	None

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_word_level_reduction</u>	Enable (or disable) word-level transformations or reductions during the proof flow.	None
<u>set_xprop_compute_highlights</u>	Disable or re-enable X-Propagation Verification App highlights.	None
<u>set_xprop_use_all_undriven</u>	Enable or disable treatment of all undriven nets as sources of X during X-Prop checks.	<ul style="list-style-type: none"> ■ Menu: <i>Application – Configuration – X-Propagation Setup Wizard</i> ■ Wizard buttons: <i>X-Propagation Setup Wizard</i> button
<u>set_xprop_use_bbox_outputs</u>	Disable or re-enable treatment of black-box outputs as sources of X during X-Prop checks.	<ul style="list-style-type: none"> ■ Menu: <i>Application – Configuration – X-Propagation Setup Wizard</i> ■ Wizard buttons: <i>X-Propagation Setup Wizard</i> button
<u>set_xprop_use_inputs</u>	Enable or disable treatment of primary inputs as sources of X during X-Prop checks.	<ul style="list-style-type: none"> ■ Menu: <i>Application – Configuration – X-Propagation Setup Wizard</i> ■ Wizard buttons: <i>X-Propagation Setup Wizard</i> button

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_xprop_use_internal_undriven</u>	Enable or disable treatment of internal undriven nets as sources of X during X-Prop checks.	<ul style="list-style-type: none"> ■ Menu: <i>Application – Configuration – X-Propagation Setup Wizard</i> ■ Wizard buttons: <i>X-Propagation Setup Wizard</i> button
<u>set_xprop_use_low_power</u>	Specify whether the tool considers Xs generated by low-power configurations as sources of X	<ul style="list-style-type: none"> ■ Menu: <i>Application – Configuration – X-Propagation Setup Wizard</i> ■ Wizard buttons: <i>X-Propagation Setup Wizard</i> button
<u>set_xprop_use_reset_abstraction</u>	Enable flop reset abstractions as sources of X or reinstate the default.	<ul style="list-style-type: none"> ■ Menu: <i>Application – Configuration – X-Propagation Setup Wizard</i> ■ Wizard buttons: <i>X-Propagation Setup Wizard</i> button

JasperGold Apps Command Reference Manual
 Commands and Variables Overview

Table 3-4 Configuration Command Summary, *continued*

Command	Description	Equivalent GUI Action
<u>set_xprop_use_reset_state</u>	Disable or re-enable treatment of uninitialized registers as sources of X during X-Prop checks	<ul style="list-style-type: none"> ■ Menu: <i>Application – Configuration – X-Propagation Setup Wizard</i> ■ Wizard buttons: <i>X-Propagation Setup Wizard</i> button
<u>set_xprop_use_stopats</u>	Enable or disable treatment of stopat nets as sources of X during X-Prop checks.	<ul style="list-style-type: none"> ■ Menu: <i>Application – Configuration – X-Propagation Setup Wizard</i> ■ Wizard buttons: <i>X-Propagation Setup Wizard</i> button
<u>set_xprop_use_x_assignments</u>	Disable or re-enable treatment of all Xs in the design as sources of X during X-Prop checks.	<ul style="list-style-type: none"> ■ Menu: <i>Application – Configuration – X-Propagation Setup Wizard</i> ■ Wizard buttons: <i>X-Propagation Setup Wizard</i> button

General Commands

This chapter provides descriptions of JasperGold Apps commands with the exception of the configuration commands. Use this chapter as a guide to command definitions, syntax and usage, and related commands.

This chapter includes:

- [General Information](#) on page 144
- One section for each command

Note:

- Refer to [Chapter 5, “Configuration Commands”](#) for descriptions of configuration commands.
- The syntax for some of the longer commands is organized by category. Click on the category heading to go directly to the relevant descriptions.

General Information

This section provides general information on the efficient entry of commands and switches. It includes the following topics:

- [Tab Completion](#) on page 144
- [Aborting a Command](#) on page 144
- [Prefix Matching](#) on page 145
- [Specifying the Current Task](#) on page 145

Tab Completion

JasperGold Apps supports tab completion for commands and for command Help. Tab completion behaves similar to a standard bash shell. That is, the first time you press the `Tab` key, nothing happens unless there is a unique completion for the command.

Examples:

- If we have the commands `cmd1` and `cmd2` and you type `c` and press `Tab`, the text will be expanded to `cmd`. The second time you press `Tab`, the tool lists (in the *Log* pane) all available commands that start with the characters to the left of the cursor. To distinguish this text from other output in this pane, the text is blue. In this example, the tool lists `cmd1` and `cmd2`. (You might need to scroll to the bottom of the *Log* pane to see the return.)
- In the case of the `help` command, if we have the commands `cmd1` and `cmd2` and you type `help c` and press `Tab`, the text will be expanded to `help cmd`. The second time you press `Tab`, the tool lists (in the *log* pane) all available commands that start with the characters to the left of the cursor.

Note: Tab completion for Tcl commands is only supported when the cursor is at the start of a line, after a semi-colon (`;`), and after a left-square-bracket (`[`), but not if there is a right-square-bracket (`]`) between the `[` and the cursor.

Aborting a Command

JasperGold Apps supports the `Ctrl+C` abort command. This command is useful for aborting the currently running command (for example, halting a long-running `prove` command to identify its complexity). When you press `Ctrl+C`, the tool returns to the state that existed prior to running the current command.

Prefix Matching

JasperGold Apps supports prefix matching for command switches. You can abbreviate any switch by typing the first one or more letters to form an unambiguous abbreviation. For example, with the `assert` command, you can abbreviate the `-rename` switch with `-rena` or `-ren`. However, you cannot abbreviate `-rename` with `-re` because that causes ambiguity with the `-remove` switch.

Note:

- If you attempt to use a prefix that is not unique, the tool does not accept the command. Instead, it prints a message advising that you have used an ambiguous switch.
- Prefix matching is not available for switches inside `-f` files. You must use full switch names.

Specifying the Current Task

In commands that include the `-task` option, replace `task_name` with a period (.) to specify the current task.

abstract

Description

Use the `abstract` command to create an abstraction for a counter, the initial value of a signal, a reset value, or a register value. You can also use this command to find candidates for counter abstraction or to identify clock domain crossing (CDC) signal pairs.

If the analysis region includes a counter that has been abstracted, the tool incorporates the abstraction into its analysis. Abstracted counters in the analysis region appear orange in the Visualize™ window.

If the tool abstracts the initial value of a signal, its reset value is not included in the analysis.

This command also provides utility switches to get information about abstractions and request actions related to specified abstractions.

Syntax

[Abstracting Counters](#)

```
abstract -counter [-task <task_name> |-env]
    (<signal_name> [-values <critical_value>*] )+
abstract -counter [-task <task_name> |-env]
( -find [-silent]
| -list [-include_values] [-silent]
| -remove <signal_name>
| -clear
| -disable_condition <disable_condition>
)
```

[Abstracting Initial Values](#)

```
abstract -init_value [-task <task_name>] <signal_name>+
abstract -init_value [-task <task_name>]
( -list [-silent]
| -remove <signal_name>
| -clear
)
```

JasperGold Apps Command Reference Manual

General Commands

Abstracting Reset Values

```
abstract -reset_value
( <flop_name_tcl_list> [-expression <expression> | -constant]
[-env |-task <task_name>]
[-bound N]
[-silent]

abstract -reset_value
[-env |-task <task_name>]
( -list [-silent]
|-remove <flop_name_tcl_list> [-silent]
|-clear [-silent]
)
```

Abstracting Register Values

```
abstract -register_value
( <flop_name_tcl_list> [-expression <expression>]
[-env |-task <task_name>]
[-bound N]

abstract -register_value
[-env |-task <task_name>]
( -list [-silent]
|-remove <flop_name_tcl_list> [-silent]
|-clear [-silent]
)
```

Abstracting CDC Flops

```
abstract -cdc [-task <task_name>]
( -inject <flop_name> [-no_prop])
[ -remove <signal>+
|-clear
|-list [-silent] ]
|-find [-coi] [-silent]
)
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
Abstracting Counters <pre>-counter [-task task_name -env] signal_name [-values critical_value...]</pre>	<p>Create a counter abstraction for the specified signal.</p> <ul style="list-style-type: none">■ Use <code>-task</code> to specify a task other than the current task.■ Use <code>-env</code> to specify the global environment instead of the current task.■ Use <code>-values</code> to specify one or more values that are critical to the counter, that is, values that must not be abstracted out. <p>Note:</p> <ul style="list-style-type: none">□ This command accepts HDL constants for values; however, the feature is expression-mode sensitive. That is, if you use <code>elaborate -mode verilog</code>, the tool only supports Verilog constants. And if you use <code>elaborate -mode vhdl</code>, the tool only supports VHDL constants.□ If the signal does not appear to be a counter, the tool does not create an abstraction.□ If any other abstraction is already applied to the counter, the tool issues an error message.□ A new implementation of <code>abstract -counter</code> became available beginning with version 2015.06. It is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.□ We provide the command <code>abstract -old_counter</code> to specify the old implementation.

JasperGold Apps Command Reference Manual

General Commands

```
-counter [-task task_name | -env]
  (-find [-silent]
    |-list [-include_values] [-silent]
    |-remove signal_name
    |-clear
    |-disable_condition disable_condition)
```

Perform the specified action on counters or counter abstractions or get the specified information.

- Use `-task` to specify a task other than the current task.
- Use `-env` to specify the global environment instead of the current task.
- Use `-find` to return a list of counter candidates in the cone of influence of the properties in the current or specified task. And use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

Note:

- Counters returned with `-find` are possible candidates for abstraction. In cases where the tool cannot abstract a candidate, it issues a warning.
 - The tool prints the widths of the counters to help you prioritize.
- Use `-list` to show counter abstractions in the task.
 - Use `-include_values` to specify the following Tcl return format:
`[<counter_name> <initial_value> <critical_values>]*`
- Note:** The format is a list of Tcl values. If there is more than one critical value, they are printed inside braces.

 - Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.
- Use `-remove` to remove the counter abstraction on the specified signal in the current or specified task.
- Use `-clear` to remove all counter abstractions in the current or specified task.
- Use `-disable_condition` to specify the condition under which the counter's original value should be used instead of the abstraction-computed one. For instance, synchronous reset conditions.

Abstracting Initial Values

```
-init_value {-task task_name] signal_name...
```

Create initial-value abstractions in the current or specified task for one or more specified signals.

This command converts a known initial value on the output of a register into an undefined X value, which permits all possible initialization values for the specified signals to be explored. The undefined value will be present for the prove and visualize commands.

- Specify a signal name or a wildcard string representing one or more signals to create initial-value abstractions in the current or specified task.
- Use `-task` to specify a task other than the current task.

Note:

- This command supports wildcards (*) and (?) in the `signal_name` argument.
- As this command ignores concrete values coming from reset analysis, if the flop is also reset during proof, this command has no effect. Therefore, if you want to also ignore the flop reset value during proof, see `abstract -reset_value`.

```
-init_value {-task task_name]  
( -list [-silent]  
| -remove signal_name  
| -clear  
)
```

Perform the specified action on one or more specified signals or get the specified information.

- Use `-task` to specify a task other than the current task.
- Use `-list` to show initial-value abstractions in the current or specified task. And use `-silent` to turn off the output to the screen and return signal names with a Tcl return value, which is helpful in Tcl scripts.
- Use `-remove` to remove the abstraction on the specified signal from the current or specified task.
- Use `-clear` to remove all initial value abstractions in the current or specified task.

Abstracting Reset Values

`-reset_value`

flop_name_tcl_list [-expression *expression*] [-constant]
[-env] [-task *task_name*] [-bound *N*] [-silent]

Create reset-value abstractions for one or more specified flops in the global environment.

By default, this command applies to the current task. Use `-env` or `-task` to apply the abstraction to the global environment or a specified task.

This command essentially disconnects the reset value pin of a flop from its driving logic, thus preventing the specified flops from receiving a concrete value when being reset (that is, during reset analysis or proof).

- Use `-expression` to define an expression that determines a new reset value for the abstracted flops.
- Use `-constant` to specify that the flop has an unknown but constant reset value.
- Use `-env` to enter the abstraction into the global environment instead of the current task.

Note: Using `-env` also affects reset analysis; thus, all loads will be affected during reset analysis as well.

- Use `-task` to apply the command to the specified task instead of the current task.
- Use `-bound` to apply the specified abstraction for *N* number of cycles after reset analysis. By default, this abstraction has bound \$, that is, it applies to all proof cycles.
- Use `-silent` to turn off the output to the screen.

Note:

- If you apply the abstraction to a wide signal containing flops and elements other than flops, the tool only applies the abstraction to the signal flops.
- This abstraction will only affect reset analysis when using `-env`.
- If you use `-bound` with a value other than the default \$ (infinity), the abstraction affects the flop reset pin during proof until the specified cycle. If you want to apply this abstraction for the reset analysis alone, you must use the bound value 0 combined with the `-env` option.

JasperGold Apps Command Reference Manual

General Commands

```
-reset_value
[-env |-task task_name]
( -list [-silent]
  |-remove flop_name_tcl_list [-silent]
  |-clear [-silent]
)
```

Perform the specified action on one or more specified flops or reset-value abstractions or list reset-value abstractions.

- Use `-env` to apply this command to the global environment instead of the current task.
- Use `-task` to apply the command to the specified task instead of the current task.
- Use `-list` to show reset-value abstractions. And use `-silent` to turn off the output to the screen and return flop names with a Tcl return value, which is helpful in Tcl scripts.
- Use `-remove` to remove the abstraction on the specified flop. And use `-silent` to turn off the output to the screen.

Note: If the abstraction you intend to remove was applied to a wide signal containing flops and elements other than flops, you can use a list of signals as the argument.

- Use `-clear` to remove all reset-value abstractions. And use `-silent` to turn off the output to the screen.

Abstracting Register Values

-register_value

flop_name_tcl_list [-expression *expression*]
[-env | -task *task_name*] [-bound *N*]

Create register-value abstractions for one or more specified flops.

By default, this command applies to the current task. Use `-env` or `-task` to apply the abstraction to the global environment or a specified task.

This command essentially replaces a register value with an expression during proof for an infinite number of cycles or up to a specified number of cycles after reset analysis for one or more specified flops.

- Use `-expression` to define an expression that determines a new value for the abstracted flops.
- Use `-env` to enter the abstraction into the global environment instead of the current task.
- Use `-task` to apply the command to the specified task instead of the current task.
- Use `-bound` to apply the specified abstraction for the specified number of cycles after reset. By default, this abstraction only applies to all proof cycles.

Note:

- This command supports Tcl lists.
- If you apply the abstraction to a wide signal containing flops and elements other than flops, the tool only applies the abstraction to the signal flops.
- Although this command replaces a register value with an expression, this case is different from an assumption because it does not constrain the underlying logic. Therefore, if you want to constrain the register and its underlying logic to an expression, see “[assume](#)” on page 187.

JasperGold Apps Command Reference Manual

General Commands

```
-register_value [-env | -task task_name]
  (-list [-silent]
   |-remove flop_name_tcl_list [-silent]
   |-clear [-silent]
  )
```

Perform the specified action on one or more specified flops or register-value abstractions or list register-value abstractions.

- Use `-env` to apply this command to the global environment instead of the current task.
- Use `-task` to apply the command to the specified task instead of the current task.
- Use `-list` to show register-value abstractions. And use `-silent` to turn off the output to the screen and return flop names with a Tcl return value, which is helpful in Tcl scripts.
- Use `-remove` to remove the abstraction on the specified flop. And use `-silent` to turn off the output to the screen.

Note: If the abstraction you intend to remove was applied to a wide signal containing flops and elements other than flops, you can use a list of signals as the argument.

- Use `-clear` to remove all register-value abstractions. And use `-silent` to turn off the output to the screen.
-

JasperGold Apps Command Reference Manual

General Commands

Abstracting CDC Flops

```
-cdc [-task task_name]
  ( -inject flop_name [-no_prop]
    [ -remove signal_list
      | -clear
      | -list [-silent] ]
    | -find [-coi] [-silent]
```

Automatically inject a CDC abstraction in the crossing domain path between the specified target flop name (destination) and each flop of all sources that are in a different clock domain.

- Use `-task` to specify a task other than the current task.
- Use `-inject` to specify the target flop.

By default, this command creates the following built-in properties:

- `sig_gray_encoding` – Checks that the signal on which asynchronicity is being modeled is Gray coded.
- `nondet_event` – Checks that it is possible to inject CDC jitter.

Use `-no_prop` to exclude the properties.

- Use `-remove` to remove the CDC abstraction from the specified target signal list in the specified task.
- Use `-clear` to remove all CDC abstractions from the specified task.
- Use `-list` to show all CDC abstractions in the current or specified task. And use `-silent` to turn off output to the screen.

Note:

- Property names are in the form `cdc_N:sig_gray_encoding_SRC` and `cdc_N:nondet_event_SRC` to distinguish properties when you have more than one CDC abstraction injected due to different destinations(N) or sources(SRC).
- Run the command `clock -domain -list` to list the current clock domain definition, or run `clock -domain -signal_list` to manually redefine the clock domain.

-cdc (Continued)

- Use -find to print all clock domain crossing boundary pairs for the current or specified task. The pairs are listed with the following format: dst (dstClk) src (srcClk).
 - Use -coi to scan only flops in the task's COI.
 - Use -silent to turn off output to the screen and return CDC pairs with a Tcl return value, which is helpful in Tcl scripts.

Note:

- -find uses the clock domain definition derived from `clock -analyze`.
- When listing the clock domain crossing paths, abstract -cdc -find also considers clock domains you defined with the command `clock -domain -signal_list`.

Return

No value returned, except as noted for -find, -list, and -info.

Examples

```
% abstract -reset_value [list a b c]
% abstract -reset_value {a b c}
% abstract -reset_value -reset [list a b c]
% abstract -reset_value -reset {a b c}

% abstract -counter my_counter

// List counters and show the counter width in parentheses
% abstract -counter -find
cnt1 (3)
cnt2 (5)
big_cnt (64)

% abstract -init_value a
% abstract -reset_value q
% abstract -reset_value q -expression {~a}
% abstract -reset_value -reset q
```

JasperGold Apps Command Reference Manual

General Commands

```
% abstract -reset_value q -expression {~b} -bound 4
% abstract -register_value q -expression 1'b0
% abstract -register_value q -expression {b | c} -bound 2
% abstract -cdc -find

// Connects all ports automatically.
% abstract -cdc -inject {bridge.xb.ofifo[0].ififo[0].ififo_i.pop_wr_ptr_sync.ff_model.ff}
```

See Also

[clock](#)

abvip

Description

Use the `abvip` command to set the ABVIP installation directory and access the documentation files.

Syntax

```
abvip -set_location <abvip_directory>
abvip -list
abvip -clear
```

Argument	Definition
<code>-set_location <abvip_directory></code>	<p>Set the ABVIP installation to the specified directory. You will find the ABVIP documentation in <i>Help – ABVIP User Guides</i>.</p>
<code>-list</code>	<p>List the ABVIP checkers available in the installation. Note: Only checkers for which documentation is available will be listed.</p>
<code>-clear</code>	<p>Clear ABVIP settings. ABVIP checkers will no longer be listed and their documentation will no longer be accessible.</p>

Return

The `-list` subcommand returns a list of ABVIP checkers.

See Also

No related commands

add_tx_attribute

Description

Use the `add_tx_attribute` command to associate a signal with a cover directive that is used to capture a behavior. With this command, the signal becomes an element attribute. In the Visualize window, the specified signal is added to the group of signals that display the behavior specified by the cover directive.

Note: This command supports regular expressions in cover names.

Syntax

```
add_tx_attribute (<cover_name> [-regexp]) <signal_name>
```

Argument	Definition
<code>cover_name</code> [-regexp]	Add the attribute to the specified <code>cover_name</code> .
<code>signal_name</code>	Add the specified <code>signal_name</code> as an attribute to <code>cover_name</code> .

Default

No default values

Return

No return value.

Examples

```
% add_tx_attribute send_packet data  
% add_tx_attribute send_packet address
```

See Also

[get_tx_attributes](#)
[remove_tx_attribute](#)

analyze

Description

Use this command to analyze the contents of HDL or property files. The tool does not synthesize and read in the netlist until you use the `elaborate` command.

Note: JasperGold Apps supports SystemVerilog configurations (for example, as stated in Section 33 of the SystemVerilog '09 LRM). This feature helps the designer configure the implementation that will be used for each instance during the synthesis process.

A configuration is simply an explicit set of rules that specify the exact source description used to represent each instance in a design.

- The configuration process can be divided into two steps: library mapping and configuring the instances.
 - Library mapping places the module implementations in different libraries.
 - Configuring is the operation that selects a source representation for an instance.

In JasperGold Apps, library mapping can be done in two ways:

- Use the switch `-lib` with the command `analyze`.
- Use library map files written according to the LRM (for example, in Section 33.2.1 in the SystemVerilog '09 LRM) and specify them first with the `analyze` command.

To configure the design, you must do the following:

- Use the notation described in the LRM (for example, Section 33.4 of the SystemVerilog '09 LRM).
- Analyze the file.
- Elaborate the configuration cell as the top module. For example, if the configuration is called `cfg1`, use the following `elaborate` command:

```
elaborate -top cfg1
```

Note: It is not currently possible to configure interface instances or to use localparams in configurations with SystemVerilog.

Syntax

```
analyze ( -vhdl |-vhdl93 |-vhdl2k |-vhdl08
          [-alias <alias_name> <target_library>]
          |-verilog |-v2k |-v95
          |-sv |-sv05 |-sv09 |-sv12 )
          [-req |-psl ]
          [-ignore_translate_off]
          [-ignore_embedded_psl]
          [-y <dir_name>]*
          [-v <file_name>]*
          [+define(+<name>[=<value>])]+
          [-f <file_name>]*
          [+libext+<ext>[+<ext>]*]
          [+incdir+<dir_name>[+<dir_name>]*]
          [-lib <default_lib_name>]
          [( -L <package_name>)+ <package_declaration.sv>
           |-L <lib_name>]
          [<lib_name>::]<file_name>+
          [-sort]
          [+liborder] [+librescan] [+libnonamehide]
          [+libunboundsearch]
          [-bbox_m <string>]
          [-no_bbox_m <string>]
          [-enable_dpram]
          [-enable_unnamed_generate_naming]
          [-keep_logfile]
          [-sfcu |-mfcu]

analyze -register
-vhdl |-vhdl93 |-vhdl2k |-vhdl08
[-alias <alias_name> <target_library>]
[-f <file_name>]*
[-lib <default_lib_name>]
[-L <library_name>]
<[lib_name::]<file_name>*
[-keep_logfile]

analyze -register -clear

analyze -clear [-req]
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<code>(-vhdl -vhdl93 -vhdl2k -vhdl08 [-alias alias_name target_library]</code>	Analyze the specified VHDL files. <ul style="list-style-type: none">■ Use <code>-vhdl</code> for files that follow IEEE® Std 1076™-1993.■ Use <code>-vhdl93</code> for files that follow IEEE Std 1076-1993.■ Use <code>-vhdl2k</code> for files that follow IEEE Std 1076-2002.■ Use <code>-vhdl08</code> for files that follow IEEE Std 1076-2008.■ Use the optional <code>-alias</code> switch to specify an alias for a library.
 <i>Important</i>	With regard to the <code>-alias</code> switch... <ul style="list-style-type: none">□ When creating <code>alias_name</code>, there must be no previously created library alias with the same name.□ Each <code>analyze</code> command can have more than one <code>-alias</code> switch.□ Aliases remain in effect in subsequent <code>analyze</code> commands.□ You can create an alias for an alias; for example, <code>-alias A B -alias C A</code>.□ Only use <code>-alias</code> with a <code>-vhdl*</code> switch; otherwise, the tool generates an error message.
<code>-verilog -v2k -v95</code>	Analyze the specified Verilog files. <ul style="list-style-type: none">■ Use <code>-verilog</code> for files that follow IEEE Std 1364™-1995.■ Use <code>-v2k</code> for files that follow IEEE Std 1364-2001.■ Use <code>-v95</code> for files that follow IEEE Std 1364-1995. This switch is the same as <code>-verilog</code>.

JasperGold Apps Command Reference Manual

General Commands

-sv | -sv05 | -sv09 | -sv12

Analyze the specified SystemVerilog files.

- Use `-sv` for files that follow IEEE Std 1800TM-2005.
- Use `-sv05` for files that follow IEEE Std 1800-2005. This switch is the same as `-sv`.
- Use `-sv09` for files that follow IEEE Std 1800-2009.
- Use `-sv12` for files that follow IEEE Std 1800-2012.

Note: `-sv12` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

Note: Refer to [“SVA Support”](#) on page 1339 for additional information.

-req

The specified files contain high-level requirements.

Note: Using `-req` in `-f` files is prohibited. It triggers an error.

-psl

The specified files contain PSL.

Note:

- Use this switch if you are analyzing PSL that is not embedded in the RTL, for example, PSL in vunit files.
- Refer to [“PSL Support”](#) on page 1333 for additional information.

-ignore_translate_off

Disable the `translate_off` pragma during analysis.

-ignore_embedded_psl

Ignore embedded PSL.

- By default, JasperGold Apps analyzes embedded PSL.
- In some cases, the tool is not able to parse certain types of PSL comments. Use this command for cases when you decide the tool does not need to include the embedded PSL.

JasperGold Apps Command Reference Manual

General Commands

`-y dir_name`

When the source code includes module instances without a module definition, go to the specified directory of Verilog files to find the definitions.

The default extension for the files in this directory is `.v`, but you can specify a different extension with the `+libext` argument.

The tool treats the modules from this directory as regular Verilog modules rather than library modules. Also see the `+libext` definition below.

Note: Also refer to the `-f` definition for cases when you specify more than one `dir_name`.

`-v file_name...`

When the source code includes module instances without a module definition, go to the specified Verilog library file to find the definition.

Note:

- Instances that are from the Verilog library module or analyzed using the `-v` switch will be considered as library cells, which are treated in the same way as cell define modules. All design traversal features (for example, `get_fanin`) skip cell defines by default. For additional information see [“set cell define debug visibility”](#) on page 966.
- JasperGold Apps does not recognize PSL properties in library files specified with `-v`. As a workaround, you can analyze these files directly, that is, as part of the design.
- The `analyze` command automatically loads files with the `.v` extension if you do not specify `+libext`. However, if you do use `+libext`, it only loads what you have specified with that command.

`+define +name=value...`

Specifies one or more Verilog ``define` directives or PSL `#define` macros (available in VHDL flavor) with the specified `name` and `value`.

Refer to [“Examples”](#) on page 172

JasperGold Apps Command Reference Manual

General Commands

`-f file_name...`

Specifies a file listing HDL sources, options, and other `-f` calls.

Note:

- By default, the tool starts its search for modules in the directory where it last found a module and returns to the beginning of the directory list if necessary. See also “[+librescan](#)” on page 166 and “[+libnonamehide](#)” on page 166.
- Do not include the language switch (for example, `-sv09`) in an `-f` file. It must appear on the command line.
- Only unknown and prohibited switches in `-f` files trigger messages.
- This command does not support file names that include whitespace or the character \$.
- Switches included in `-f` files are case sensitive; that is, `-l` is not interpreted as `-L`.

`+libext +ext...`

Specifies file extensions for one or more library directories.

The `analyze` command automatically loads files with the `.v` extension if you do not specify `+libext`. However, if you do use this switch, it only loads what you have specified.

`+inmdir +dir...`

Specifies an include directory. The tool uses this switch to locate files that are included using the Verilog ``include` directive or the PSL `#include` directive that is available in the VHDL flavor.

`-lib default_lib_name`

Use `default_lib_name` as the `lib_name` for file names that do not have an explicit `lib_name::` part.

Note: Using `-lib` in `-f` files is prohibited. It triggers an error.

`(-L package_name)+ package_declaration.sv |-L lib_name`

Use `-L` to specify Verilog packages or a VHDL or Verilog library for the tool to search. For additional information, see “[set automatic library search](#)” on page 958.

JasperGold Apps Command Reference Manual

General Commands

`lib_name:: file_name...`

Read `file_name` into library `lib_name`. The `lib_name::` part is only valid for VHDL, and if you do not specify one, the default library is `work`.

`-sort`

Analyze the registered library files in sorted order.

This switch takes a list of file names for the tool to register, sort, and analyze on-the-fly. Use this switch with any other of the main `analyze` arguments except `-req` and the Verilog switches.

Note:

- VHDL sort is not able to sort dependencies if they are not explicitly specified in the use clause.
- `-sort` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

`+libborder`

Search for module definitions in the library where the instance is located and then search remaining libraries in the order they were specified.

`+librescan`

Always scan all Verilog directories in the directory list when looking for a new module.

`+libnonamehide`

Keep all read modules in a temporary library during compilation.



- For the default global file compilation unit (GFCU) mode of `analyze`, this setting is the default.
- Use `set_analyze_libnonamehide off` to disable the feature.
- For multi- and single-file compilation unit (MFCU and SFCU) mode, this switch is disabled by default.

`+libunboundsearch`

Do not clear `-L` options before and after each compilation unit.

JasperGold Apps Command Reference Manual

General Commands

`-bbox_m string`

Black box the specified set of modules.

Note:

- Specify modules with a wildcard string. For example, the following argument black boxes all modules whose names match the wildcard string.
`-bbox_m FIFO_*_TWO_PORT`
- If you specify a name with an escape character, you must surround the name with curly braces, for example:
`analyze... -bbox_m {\arb$.()z}`
- The tool issues an error during elaboration if you attempt to black box the `-top` module; therefore, use care when creating your wildcard string.

`-no_bbox_m string`

Do not black box the specified set of modules.

Note:

- Specify modules with a wildcard string. For example, the following argument black boxes all modules whose names match the wildcard string.
`-bbox_m FIFO_*_TWO_PORT`
- If you specify a name with an escape character, you must surround the name with curly braces, for example:
`analyze... -no_bbox_m {\arb$.()z}`
- `-no_bbox_m` takes priority over `-bbox_m`; therefore, if you specify a module for both `-bbox_m` and `-no_bbox_m`, the tool does not black box it.
- Using `-no_bbox_m` without `-bbox_m` has no effect.

JasperGold Apps Command Reference Manual

General Commands

-enable_dpram

Enable dual port RAM detection during synthesis.



- This switch provides an alternative RAM detection mode in the event you experience undesirable tool behavior.
- This is the RAM detection algorithm that was available prior to version 2012.08.
- You must use this switch in *all* analyze commands *and* in the elaborate command. (You cannot mix RAM detection algorithms.)

-enable_unnamed_generate_naming

Automatically name un-named generate blocks following the rules defined in Section 27.6 of the SystemVerilog 2009 LRM.

Note: You must use this switch with both the analyze and elaborate commands.

-keep_logfile

Save the analyze log file.



- Only use this switch when requested by your Cadence representative. It is designed for tool debugging only.
- The tool stores the log in the following file:

<proj>/sessionLogs/<session_identifier>analyze.log*

JasperGold Apps Command Reference Manual

General Commands

-sfcu

Use the single-file compilation unit analyze mode.

In this mode, the tool does not pass any declarations or macro definitions from one analyzed *file* to another.

Note:

- The Tcl parser recognizes accumulated macros regardless of compilation unit mode.
 - The default `analyze` mode is global; that is, `analyze` accumulates declarations and macro definitions. Using `-sfcu` can resolve potential conflicts that this handling introduces.
-

-mfcu

Use the multi-file compilation unit analyze mode.

In this mode, the tool does not pass any declarations or macro definitions from one *command* to another.

Note:

- The Tcl parser recognizes accumulated macros regardless of compilation unit mode.
 - The default `analyze` mode is global; that is, `analyze` accumulates declarations and macro definitions. Using `-mfcu` can resolve potential conflicts that this handling introduces.
-

JasperGold Apps Command Reference Manual

General Commands

```
-register -vhdl | -vhdl93 | -vhdl2k | -vhdl08
[-alias alias_name target_library]
[-f file_name]*
[-lib default_lib_name]
[-L library_name]
[lib_name:::]file_name*
[-keep_logfile]
```

Register the specified library files for later sorting based on inter-file dependencies.

- Use `-vhdl` for files that follow IEEE Std 1076-1993.
- Use `-vhdl93` for files that follow IEEE Std 1076-1993.
- Use `-vhdl2k` for files that follow IEEE Std 1076-2002.
- Use `-vhdl08` for files that follow IEEE Std 1076-2008.
- Use `-alias` to specify an alias for a library.

Important things to know about the `-alias` switch:

- When creating `alias_name`, there must be no previously created library alias with the same name.
- Each `analyze` command can have more than one `-alias` switch.
- Aliases remain in effect in subsequent `analyze` commands.
- You can create an alias for an alias; for example: `-alias A B -alias C A`
- Only use `-alias` with a `-vhdl*` switch; otherwise, the tool generates an error message.
- Use `-f` to specify a file listing HDL sources, options, and other `-f` calls.

Note:

- By default, the tool starts its search for modules in the directory where it last found a module and returns to the beginning of the directory list if necessary. See also `+librescan` and `+libnonamehide`.
- Do not include the language switch (for example, `-vhdl08`) in an `-f` file. It must appear on the command line.
- Only unknown and prohibited switches in `-f` files trigger messages.

-register (Continued)

- This command does not support file names that include whitespace or the character \$.
- Switches included in -f files are case sensitive; that is, -l is not interpreted as -L.
- Use -lib to specify that you want to use *default_lib_name* as the *lib_name* for file names that do not have an explicit *lib_name*:: part.
- Use -L to specify a VHDL library for the tool to search. Also see ["set automatic library search"](#) on page 958.
- Specify *lib_name*::*file_name*... if you want the tool to read *file_name* into library *lib_name*. The *lib_name*:: part is only valid for VHDL, and if you do not specify one, the default library is work.
- Use -keep_logfile to save the analyze log file.

Important things to know about this command:

- Only use this switch when requested by your Cadence representative. It is designed for tool debugging only.
- The tool stores the log in the following file:
<project>/sessionLogs/<session_identifier>/analyze.log*

Note:

- The commands elaborate and clear -all clear the registered file list.
- -register is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

LIMITATIONS:

- -register does not support Verilog.
- You can only register library files. Directories are not supported.
- The tool cannot sort the files in the correct order if any of the files has a configuration declaration that has some declaration before the binding indication.

JasperGold Apps Command Reference Manual

General Commands

-register clear

Remove the list of previously registered files.

-clear

Clear all previously analyzed modules from memory but preserve the existing data structure to continue work in the current session.

Use this option for the following situations:

- Before analyzing a design with a different top module and running the elaborate command.
- After a successful elaboration to end the separate frontend process and free the associated memory. Subsequent elaborate commands result in an error if you end the jg_frontend process beforehand.

Use the clear -all command to clear the entire data structure and proof or Visualize information.

-clear -req

Clear all previously analyzed requirement files but preserve the design to continue work in the current session.

Return

No value returned.

Examples

```
% analyze -v2k my_hdl_file.v
% analyze -v2k -req my_hlr_file.v
% analyze -vhdl -psl my_property_file.vhdl.psl
% analyze -verilog -ignore_translate_off my_hdl_file.v
% analyze -v2k my_hdl_file.v +define+A=1+B=2

// Importing packages not specified in the current or standard library:
% analyze -sv -lib pkg p.w
% analyze -sv -lib pkg2 q.w
% analyze -sv -L pkg -L pkg2 pkg_decl.sv
% elaborate

// Analyzing libraries with some definitions defined in both. Use -mfcu to solve
// the conflicts:
```

JasperGold Apps Command Reference Manual

General Commands

```
% analyze -sv -mfcu -lib lib1 lib1.v
% analyze -sv -mfcu -lib lib2 lib2.v
% elaborate -top lib1.top

// Register and sort design files:
% analyze -register -vhdl {lib3::dirA/dirB/libfile.vhd}
% analyze -register -vhdl -lib lib2 libx_a.vhd
% analyze -register -vhdl {lib1::dirA/dirB/libfile.vhd}

% analyze -sort -vhdl entity.vhdl
```

See Also

[elaborate](#)
[set analyze libnonamehide](#)
[set analyze librescan](#)

assert

Description

Use this command to declare an interactive (specified on-the-fly) property with an assert directive. By default, this command applies to the current task and does not automatically update the database. Use `-task` to specify a different task.

Note: Using this command to add assertions to static tasks (vunit derived or embedded) does not actually capture the assertion in the vunit or embedded code itself.

Syntax

[Creating an Assertion](#)

```
assert [-task <task_name> | -name <name> | -name <task_name>::<name>]
       <expression>
       [-no_precondition]
       [-label <label>] [-annotation '{'<annotation>'}']
       [-update_db | -update_all_sessions | -suppress_db_update]
```

[Editing an Assertion](#)

```
assert -replace
       -name <name>
       |-name <task_name>::<name>
       <expression>
       [-label <label>] [-annotation '{'<annotation>'}']
       [-update_db | -update_all_sessions | -suppress_db_update]
```

[State Space Tunneling](#)

```
assert -helper
       -name <name>
       |-name <task_name>::<name>
       <expression> [-annotation '{'<annotation>'}']

assert -set_helper
       <name>
       |<task_name>::<name>

assert -clear_helper
       <name>
       |<task_name>::<name>
```

JasperGold Apps Command Reference Manual

General Commands

```
assert -mark_proven
    <name>
    |<task_name::name>
```

[Creating an X-Propagation Assertion](#)

```
assert -xprop
    [-from <input_signal_tcl_list>]
    -to <barrier_signal_tcl_list>
    [-precond <boolean_expression>]
    [-sampling_clock <clock>]
    [-delay <N>]
    [-task <task_name> |-name <property_name> |-name <task_name::name>]
    [-annotation '{ '<annotation> ' }']
```

[Creating an Assertion from an Assumption](#)

```
assert -from_assume <assume_name> [-regexp]
    [-update_db |-update_all_sessions |-suppress_db_update]
```

[Utility Switches](#)

```
assert -list [-silent] [-type (xprop | spv)] [-task <task_name>]
assert -clear [-task <task_name>]
assert -show <name>
assert -remove <name>+ [-regexp]
assert -disable <name>+ [-regexp]
assert -enable <name>+ [-regexp]
assert -rename <old_name> <new_name>
    [-update_db |-update_all_sessions |-suppress_db_update]
assert -is_liveness <property_name>
```

Argument	Definition
Creating an Assertion	
	<pre>[-task task_name -name name -name task_name::name] expression [-no_precondition] [-label label -annotation {annotation}] [-update_db -update_all_sessions -suppress_db_update]</pre> <p>Create an assertion for the specified Boolean expression that evaluates to either true or false.</p> <p>Use the <code>prove</code> command to verify this assertion.</p> <p>Note: JasperGold Apps supports SVA expressions on the command line. Refer to Appendix D, “Tcl Support for SVA Expressions” for guidance.</p> <p>Implementation guidance:</p> <ul style="list-style-type: none">■ Use <code>-task</code> to apply the command to the specified task instead of the current task.■ Use <code>-name</code> to assign a name to the property.■ Use <code>-no_precondition</code> if you are creating a liveness assertion and you do not want the tool to automatically generate precondition covers.■ Use <code>-label</code> to add an element heading. The GUIs will display this heading instead of the property name.■ Use <code>-annotation</code> to annotate the specified property with the description enclosed by the curly braces. You can view annotations in the Visualize window <i>Waveforms</i> pane by positioning the mouse to hover over the property and reveal a tooltip. And you can print annotations to the screen with the <code>get_annotation</code> command.■ Use <code>-update_db</code> to add the specified assertion to the database.■ Use <code>-update_all_sessions</code> when you are running multiple sessions and you want to add the specified assertion to all of them and the database.

JasperGold Apps Command Reference Manual

General Commands

expression (Continued)

- Use `-suppress_db_update` when you have used `set_property_auto_update_db` on to set the tool to update the database automatically but you do not want to include the specified property.

Note:

- Do not use * or ? in property names.
- Do not use the `-task` and `-name` switches in the same command. To specify both a `name` and `task_name`, use the `-name` switch and prepend the property name with `<task>::`. (Replace `<task>` with an actual task name.) Example:

```
assert -name taskA::A {~rdy => ~trx}
```

Editing an Assertion

```
-replace
  -name name
  |-name task_name::name
  expression
  [-label label -annotation {annotation}]
  [-update_db |-update_all_sessions |-suppress_db_update]
```

Replace the expression for the specified assertion.

Use this command to edit an assertion used as a Visualize target without triggering a `-clear_all`. Replot the trace after you use this command.

Implementation guidance:

- Use `-name` to specify the assertion you are editing.
- Use `-label` to add or change the property's element heading.
- Use `-annotation` to add or change the property's annotation.
- Use `-update_db` to add the specified assertion to the database.
- Use `-update_all_sessions` when you are running multiple sessions and you want to add the specified assertion to all of them and the database.
- Use `-suppress_db_update` when you have used `set_property_auto_update_db` on to set the tool to update the database automatically but you do not want to include the specified property.

State Space Tunneling

```
-helper
  -name name
  |-name task_name::name
  expression [-annotation {annotation}]
```

Create a helper assertion with the specified expression.

Use this command to create a helper assertion that is used by the engines to prove divergent properties. See [“State-Space Tunneling Visualize”](#) on page 921 for more information.

- Use `-name` to assign a name to the property.
- Use `-annotation` to annotate the specified property with the description enclosed by the curly braces. You can view annotations in the *Visualize* window *Waveforms* pane by positioning the mouse to hover over the property and reveal a tooltip. And you can print annotations to the screen with the `get_annotation` command.

```
-set_helper
  name
  |task_name::name
```

Convert the specified regular assertion to a helper assertion.

```
-clear_helper
  name
  |task_name::name
```

Convert the specified helper assertion to a regular assertion.

```
-mark_proven
  name
  |task_name::name
```

Mark the specified undetermined, unprocessed, or `ar_cex` assertion as proven regardless of its actual status.

This command is typically used with state-space tunneling (SST) to speed up proofs of complex properties.

Note: If you mark an assertion `proven`, the tool treats the specified assertion the same way it treats the assertions it proves. So if you subsequently run commands such as `prove -all`, `prove -task`, or `prove -property -with_proven`, the `marked_proven` status affects the results of other properties; that is, the tool does not consider situations where these `marked_proven` properties may not hold. To clear the `marked_proven` status, you must clear all results for the assertion's task.

Creating an X-Propagation Assertion

```
-xprop
[-from input_signal_tcl_list]
-to barrier_signal_tcl_list
[-precond boolean_expression]
[-sampling_clock clock]
[-delay N]
[-task task_name |-name property_name |-name task_name::name
[-annotation}]
```

Generate an assertion to check that X values cannot propagate between two points in the design.

The tool extracts the internal nodes that have explicit X values and starts from there.

Note: To include X-propagation checks for SVA assertions that use the \$isunknown keyword, do the following:

- ❑ Elaborate with -enable_sva_isunknown.
- ❑ Use the assert command with the following format instead of using the -xprop switch:

```
assert -name p1 {$isunknown(<signal>)}
```
- Use -from to add originating signals for the path. This switch adds X sources to the default list.
- Use -to to select the end points for the path. If you do not use -to, the tool uses the list of output signals.
- Use -precond to limit the X-propagation checking to situations where the precondition (triggering) expression is true; that is, instead of checking hasXInOutput, check preconditionExpression & hasXInOutput.

Note: The precondition must be a Boolean expression; that is, the result must be true or false. Temporal expressions are not supported.

JasperGold Apps Command Reference Manual

General Commands

-xprop (Continued)

- Use `-sampling_clock` to enable handling for sampled properties. The signal name must be a single-bit signal (that is, a simple net or a bit of an array). See examples below:
 - `assert -xprop -to a ==> assert property !isunknown(a)`
 - `assert -xprop -to a -sampling_clock clk ==> assert property @ (posedge clk) !isunknown(a)`
- Use `-delay` to delay verification for the specified number of cycles. With this switch, there can be no CEX until after the specified cycle.
- Use `-task` to assign the property to a task other than the current task.
- Use `-name` to assign a name to the property.
- Do not use * or ? in property names.
- Do not use both `-task` and `-name`. To specify both, use the `-name` switch and prepend the property name with `<task>::`.
- Use `-annotation` to annotate the specified property with the description enclosed by the curly braces. You can view annotations in the *Visualize* window *Waveforms* pane by positioning the mouse to hover over the property and reveal a tooltip. And you can print annotations to the screen with the `get_annotation` command.

JasperGold Apps Command Reference Manual

General Commands

-xprop (Continued)



- When you use `-from`, the tool does not consider explicit assignments (for example, `foo == 1'b1`) as X for verification purposes. Uninitialized register handling is controlled by a different command, `set_xprop_use_reset_state`.
- The `-from` and `-to` switches support Tcl lists and lists of signals that can be formed using wildcards, for example:

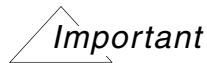
```
assert -xprop -from io_dram?_data_valid -to dram_io_cas0_1
assert -xprop -from io_dram*_data_valid -to dram_io_cas0_1
```
- Signals used with the `-from` switch must be inputs or stopats.
- JasperGold Apps does not support sampled properties with `assert -xprop` unless they are set on the global clock.
- You can use ProofGrid to prove multiple `-xprop` properties in parallel. To enforce serial processing, put properties in different tasks.
- X-Prop properties are not supported with tunneling. The tool ignores these properties when verifying tasks that contain them.
- Refer to the following documentation to change related default configuration:
 - [“set_xprop_use_all_undriven” on page 1315](#)
 - [“set_xprop_use_bbox_outputs” on page 1317](#)
 - [“set_xprop_use_inputs” on page 1319](#)
 - [“set_xprop_use_internal_undriven” on page 1321](#)
 - [“set_xprop_use_reset_state” on page 1327](#)
 - [“set_xprop_use_stopats” on page 1329](#)

Creating an Assertion from an Assumption

```
-from_assume assume_name -regexp  
[-update_db | -update_all_sessions | -suppress_db_update]
```

Create an assertion from the specified assumption and disable the original assumption.

- Use `-update_db` to add the specified assertion.
- Use `-update_all_sessions` when you are running multiple sessions and you want to add the specified assertion to all of them.
- Use `-suppress_db_update` when you have used `set_property_auto_update_db` on to set the tool to update the database automatically but you do not want to include the specified property.



- Use `assert -from_assume <assume_name>` without additional switches.
- Wildcards and regular expressions are supported; for example:
`assert -from_assume {^.*$} -regexp`
- You cannot automatically reverse this command. You must manually remove or disable the assertion and enable the assumption.

```
// Creating an assertion from an assumption  
assert -from_assume <embedded>:assume123  
// New assertion created  
// Original assumption is disabled  
  
// Reverting back to the original with two  
// commands  
assert -disable {<embedded>:assume123:assert}  
// assume123:assert is disabled  
assume -enable {<embedded>:assume123}  
// assume123 is re-enabled
```

JasperGold Apps Command Reference Manual

General Commands

Utility Switches

-list

```
[-silent]
[-type (xprop | spv)]
[-task task_name]
```

List the assertions in the current or specified task.

- Use `-silent` to turn off output to the screen and return assertions with a Tcl return value. This option is helpful in Tcl scripts.
- Use `-type` with `xprop` or `spv` to return only the list of X-Prop or SPV assertions.
- Use `-task` to apply the command to the specified task instead of the current task.

-clear

```
[-task task_name]
```

Remove all interactive assertions (that is, assertions you specified on-the-fly).

Use `-task` to apply the command to the specified task instead of the current task.

-show

name

Show the actual expression for the assertion with the specified name.

Refer to the definitions for `assert -name`.

JasperGold Apps Command Reference Manual

General Commands

-remove

 name...
 [-regexp]

Remove assertions with the specified name.



- Use this switch to specify one or more assertions.
- Wildcards and regular expressions are supported; for example:
assert -remove {^.*\$} -regexp
- You can only remove properties created with the GUI or on the command line. That is, you cannot remove properties added automatically (for example, SVA and PSL embedded properties). However, you can disable and enable these properties.

-disable

 name...
 [-regexp]

Keep the assertion in the current task, but do not include it in any analysis.



- Specify one or more assertions.
- Wildcards and regular expressions are supported; for example:
assert -disable {^.*\$} -regexp

-enable

 name...
 [-regexp]

Re-enable the specified (previously disabled) assertion.



- Specify one or more assertions.
- Wildcards and regular expressions are supported; for example:
assert -enable {^.*\$} -regexp

JasperGold Apps Command Reference Manual

General Commands

```
-rename old_name new_name
[-update_db | -update_all_sessions | -suppress_db_update]
```

Change the *old_name* assertion to *new_name*.

- Use `-update_db` to add the specified assertion to the database.
- Use `-update_all_sessions` when you are running multiple sessions and you want to add the specified assertion to all of them and the database.
- Use `-suppress_db_update` when you have used `set_property_auto_update_db` on to set the tool to update the database automatically but you do not want to include the specified property.

```
-is_liveness
property_name
```

Return true if the assertion is a liveness assertion; otherwise, return false.

Return

Returns a string (identifier name) for the assertion.

Examples

```
% assert a==b
% assert -name illegal_size {wr_size != 2'b11}
% assert {scan_enable == 1'b0}
% assert -name illegal_size {WR_SIZE /= "11"}
% assert {SCAN_ENABLE = '0'}
```

Note: JasperGold Apps supports the implication overlapping operator (`->`) on the command line to declare properties. This operator indicates that the expressions on both sides happen at the same iteration.

```
% assert {~rdy -> ~trx}
% assert {(rdy = '1') -> (trx = '1')}

# Pattern-matching with regular expressions:
# -----
# Disable assertions that match zero or more characters followed by either
# "_ended" or "_started"
```

JasperGold Apps Command Reference Manual

General Commands

```
% assert -disable {.*(_ended|_started)} -regexp  
  
# Remove assertions that match "trans" followed by 1 or more numbers and  
# followed by zero or more characters  
% assert -remove {trans([0-9]+).*} -regexp
```

See Also

[assume](#)

[cover](#)

[prove](#)

[visualize](#)

assume

Description

Use this command to add an assumption about the signals (constraint) in the DUV or requirement specification. By default, this command applies to the current task and does not automatically update the database. Use `-env` or `-task` to apply the assumption to the global environment or a specified task. The `reset` and `clock` commands only consider global assumptions.

Note: Using this command to add assumptions to static tasks (vunit derived or embedded) does not actually capture the assumption in the vunit or embedded code itself.

Syntax

[Creating an Assumption](#)

```
assume [-env
        [ -name <property_name>]
        | -task <task_name> | -name <property_name>
        | -name <task_name>::<property_name>]
        [-type (approved | temporary)]
        ([ -bound <N>] <expression>
         | -constant <signal_name>
        )
        [-label <label>] [-annotation '{'<annotation>'}']
        [-update_db | -update_all_sessions | -suppress_db_update]
        [-replace]
```

[Creating a Soft Assumption](#)

```
assume -soft
[-env
 [ -name <property_name>]
 | -task <task_name> | -name <property_name>
 | -name <task_name>::<property_name>]
[-bound <N>]
[-label <label>] [-annotation '{'<annotation>'}']
[-update_db | -update_all_sessions | -suppress_db_update]
(<expression> | -quiet | -noisy | -quiet_from | -max_from)
[-replace]
```

JasperGold Apps Command Reference Manual

General Commands

[Creating an X-Propagation Assumption](#)

```
assume -xprop
    [-bound <N>]
    -to <signal_name_tcl_list>
    [-precond <boolean_expression>]
    [-sampling_clock <clock>]
    [-task <task_name> |-name <property_name>
        |-name <task_name::property_name>]
    [-annotation '{'<annotation>'}'']
    [-soft]
```



```
assume -xprop -is_x
    -to <signal_name_tcl_list>
    [-task <task_name> |-name <property_name>
        |-name <task_name::property_name>]
    [-annotation '{'<annotation>'}'']
    [-soft]
```

[Creating an Assumption from an Assertion](#)

```
assume -from_assert <assert_name> [-regexp]
    [-update_db |-update_all_sessions |-suppress_db_update]
```

[Utility Switches](#)

```
assume -list [-silent] [-env |-task <task_name>]
assume -clear [-env |-task <task_name>]
assume -show <property_name> [-env]
assume -remove <property_name>+ [-regexp] [-env]
assume -disable |-enable <property_name>+ [-regexp] [-env]
assume -rename <old_name> <new_name> [-env]
    [-update_db |-update_all_sessions |-suppress_db_update]
assume -set_type <property_name> (approved | temporary)
assume -is_liveness <property_name>
```

[Creating an Assumption Only Valid During Reset](#)

```
assume -reset <expression> [-name <property_name>]
    [-replace |-replace_if_needed]
assume -reset ( -show <property_name>
    |-remove <property_name>
    |-list [-silent]
    |-clear)
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
Creating an Assumption	
<pre>[-env [-name <i>property_name</i>] -task <i>task_name</i> [-name <i>property_name</i>] -name <i>task_name</i>::<i>property_name</i>] [-type {approved temporary}] ([-bound <i>N</i>] <i>expression</i> -constant <i>signal_name</i>) [-label <i>label</i>] [-annotation {<i>annotation</i>}] [-update_db -update_all_sessions -suppress_db_update] [-replace]</pre>	<p>Create an assumption for the specified Boolean expression that evaluates to either true or false.</p> <p>All proofs are carried out assuming that the expression is true. Scenarios in which the expression is false are ignored in the proof.</p> <p>Note: JasperGold Apps supports SVA expressions.</p> <ul style="list-style-type: none">■ Use <code>-env</code> to enter the assumption into the global environment instead of the current task.■ Use <code>-name</code> to assign a name to the property.■ Use <code>-task</code> to apply the command to the specified task instead of the current task.■ Use <code>-type</code> to add a tag to the assumption:<ul style="list-style-type: none">□ <code>approved</code> – The user has approved the assumption as valid.□ <code>temporary</code> – The user will discard the assumption later. This is the default tag.■ Use <code>-bound</code> to apply the specified assumption for <i>N</i> number of cycles (after reset). By default an assumption is applicable for all cycles.

Creating an Assumption (Continued)

- Use `-constant` to assume that the signal is pseudo constant, that is, the signal has a fixed but unknown value throughout the analysis. The tool analyzes the design for all possible values for the specified signal; however, the value of the signal remains constant for a trace.

For example, a design register that defines an input mode may have an arbitrary value ranging from 0 to 3. The analysis must consider all possible mode values, but it can exploit the fact that the mode does not change.

Note: The command `assume -env -constant some_inputs` does not apply during the reset phase. As a workaround, you can do one of the following:

- Specify a constant value
 - Use `reset -none` and manually specify the reset scheme using assumptions.
- Use `-label` to add an element heading. The GUIs will display this heading instead of the property name.
- Use `-annotation` to annotate the specified property with the description enclosed by the curly braces. You can view annotations in the Visualize window waveforms pane by positioning the mouse to hover over the property and reveal a tooltip. And you can print annotations to the screen with the `get_annotation` command.
- Use `-update_db` to add the specified assumption to the database.
- Use `-update_all_sessions` when you are running multiple sessions and you want to add the specified assumption to all of them and the database.
- Use `-suppress_db_update` when you have used `set_property_auto_update_db on` to set the tool to update the database automatically, but you do not want to include the specified property.

Creating an Assumption (Continued)

- Use `-replace` to modify the specified assumption.



- The `-name` switch is required with `-replace`.
- Replot any traces after you use this command.
- This command does not apply to embedded assumptions.
- This command does not apply to X-Prop assumptions.

Note:

- The following command is the equivalent of using `-env: assume -name ::<property_name>`
- Do not use * or ? in property names.
- Do not use the `-task` and `-name` switches in the same command. If you need to specify both a `property_name` and `task_name`, use the `-name` switch and prepend the property name with `<task>::`. (Replace `<task>` with an actual task name.) Example:

```
assume -name taskA::A {~rdy => ~trx}
```

Creating a Soft Assumption

```
-soft [-env  
      [ -name property_name]  
      | -task task_name | -name property_name  
      | -name task_name::property_name]  
      [-bound N]  
      [-label label] [-annotation '{{'annotation'}}']  
      [-update_db | -update_all_sessions | -suppress_db_update]  
      (expression | -quiet | -noisy | -quiet_from | -max_from)  
      [-replace]
```

Create a soft assumption, which is a mere preference on delivered traces rather than a hard constraint that must unconditionally be satisfied.

- Use `-env` to enter the assumption into the global environment instead of the current task.
 - Use `-name` to assign a name to the property.
 - Use `-task` to apply the command to the specified task instead of the current task.
 - Use `-bound` to apply the specified assumption for *N* number of cycles (after reset). By default, an assumption applies to all cycles.
- Note:** The switches `-quiet_from` and `-max_from` do not yet support this switch.
- Use `-label` to add an element heading. The GUIs will display this heading instead of the property name.
 - Use `-annotation` to annotate the specified property with the description enclosed by the curly braces. You can view annotations in the Visualize window *Waveforms* pane by positioning the mouse to hover over the property and reveal a tooltip. And you can print annotations to the screen with the `get_annotation` command.
 - Use `-update_db` to add the specified assumption.
 - Use `-update_all_sessions` when you are running multiple sessions and you want to add the specified assumption to all of them and the database.
 - Use `-suppress_db_update` when you have used `set_property_auto_update_db` on to set the tool to update the database automatically but you do not want to include the specified property.

Note: This command supports wide signals.

Creating a Soft Assumption (Continued)

- Use `-quiet` to prefer all inputs to be low.
- Use `-noisy` to prefer edges on all outputs.
- Use `-quiet_from` to generate traces that rarely propagate originating signals (from signals).
- Use `-max_from` to generate traces that prefer propagated originating signals (from signals).
- Use `-replace` to modify the specified assumption.



- The `-name` switch is required with `-replace`.
- Replot any traces after you use this command.
- This command does not apply to embedded assumptions.
- This command does not apply to X-Prop assumptions.

Note:

- The following command is the equivalent of using `-env: assume -name ::<property_name>`
- Do not use * or ? in property names.
- Do not use the `-task` and `-name` switches in the same command. If you need to specify both a `property_name` and `task_name`, use the `-name` switch and prepend the property name with `<task>::`. (Replace `<task>` with an actual task name.) Example:

```
assume -name taskA::A {~rdy => ~trx}
```

Creating an X-Propagation Assumption

```
-xprop [-bound N] -to signal_name_tcl_list
[-precond boolean_expression] [-sampling_clock clock]
[-task task_name] [-name property_name] [-name task_name::name]
[-annotation {annotation}]
[-soft]
```

Create an assumption that instructs the verification to ignore traces where X propagates to the specified signals.

This command disables propagation of X from the specified signals and forbids traces where X propagates to the signals.

Note: As an alternative to the `-xprop` switch, you can specify SVA assumptions that use the `$isunknown` keyword:

- ❑ Elaborate with `-enable_sva_isunknown`.
- ❑ Use the `assume` command with the following format instead of using the `-xprop` switch:

```
assume -name p1 {$isunknown(<signal>)}
```
- Use `-bound` to apply the specified assumption for *N* number of cycles (after reset). By default, an assumption is applicable for all cycles.
- Use `-to` to select the end points for the path (required).
- Use `-precond` to limit the X-propagation assumption to situations where the precondition (triggering) expression is true; that is, instead of assuming `hasXInOutput`, assume `preconditionExpression & hasXInOutput`.

The precondition must be a Boolean expression.

- Use `-sampling_clock` to enable handling for sampled properties. The signal name must be a single-bit signal (that is, a simple net or a bit of an array).

Note: If you specify `-sampling_clock`, the tool produce no traces where the `-to` net is X at the rising edge of the sampling clock. However, X can still appear and be propagated on the `-to` net in time points where the sampling clock does not rise.

JasperGold Apps Command Reference Manual

General Commands

-xprop -bound... (Continued)

- Use `-task` to assign the property to a task other than the current task.
- Use `-name` to assign a name to the property.
- Do not use both `-task` and `-name`. If you need to specify both, use the `-name` switch and prepend the property name with `<task_name>::`
- Use `-annotation` to annotate the specified property with the description enclosed by the curly braces. You can view annotations in the Visualize window *Waveforms* pane by positioning the mouse to hover over the property and reveal a tooltip. And you can print annotations to the screen with the `get_annotation` command.
- Use `-soft` to turn the assumption into a mere preference on delivered traces rather than a hard constraint that must unconditionally be satisfied. This switch is not supported in combination with `-precond`, `-sampling_clock`, or `-bound`.



- Global X-Prop assumptions (`-env`) are not supported.
- Assumptions created with this command are relevant for verification of X-propagation (XPROP), `cover -path`, and security verification (SPV), but the tool ignores them for other kinds of properties (such as those from the Formal Property Verification App and Sequential Equivalence Checking App).
- `-xprop` properties are not supported with tunneling. The tool ignores these properties when verifying tasks.
- The `-to` switch supports Tcl lists and lists of signals that can be formed using wildcards, for example:

```
assume -xprop -to io_dram?_data_valid  
assume -xprop -to io_dram*_data_valid
```

JasperGold Apps Command Reference Manual

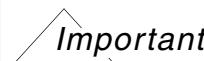
General Commands

```
-xprop -is_x -to signal_name_tcl_list
[ -task task_name | -name property_name | -name task_name::name ]
[-annotation {annotation}]
[-soft]
```

Create an assumption that instructs the verification to assume X values to one or more specified signals.

When you use this command, any specified signal behaves like 1'bX, which propagates to its fanout.

- Use `-to` to select the end points for the path (required).
- Use `-task` to assign the property to a task other than the current task.
- Use `-name` to assign a name to the property.
- Do not use both `-task` and `-name`. If you need to specify both, use the `-name` switch and prepend the property name with `<task_name>::`
- Use `-annotation` to annotate the specified property with the description enclosed by the curly braces. You can view annotations in the Visualize window *Waveforms* pane by positioning the mouse to hover over the property and reveal a tooltip. And you can print annotations to the screen with the `get_annotation` command.
- Use `-soft` to turn the assumption into a mere preference on delivered traces rather than a hard constraint that must unconditionally be satisfied. This switch is not supported in combination with `-precond`, `-sampling_clock`, or `-bound`.



- Global X-Prop assumptions are not supported.
- Assumptions created with this command are relevant only for the verification of x-propagation.
- `-xprop` properties are not supported with tunneling. The tool ignores these properties when verifying relevant tasks.

JasperGold Apps Command Reference Manual

General Commands

-xprop -is_x... (Continued)

- The **-to** switch supports Tcl lists and lists of signals that can be formed using wildcards, for example:

```
assume -xprop -to io_dram?_data_valid
assume -xprop -to io_dram*_data_valid
```
 - The **-to** switch supports inputs and stopats. You cannot specify an internal signal driven by some logic.
-

Creating an Assumption from an Assertion

```
-from_assert assert_name [-regexp]
[-update_db | -update_all_sessions | -suppress_db_update]
```

Create an assumption from the specified assertion and disable the original assertion.

- Use `-update_db` to add the specified assumption to the database.
- Use `-update_all_sessions` when you are running multiple sessions and you want to add the specified assumption to all of them and the database.
- Use `-suppress_db_update` when you have used `set_property_auto_update_db` on to set the tool to update the database automatically but you do not want to include the specified property.



- Use `assume -from_assert <assert_name>` without additional switches.
- The tool warns you when an assumption might be weaker than expected.
- Wildcards and regular expressions are supported:

```
assume -from_assert {^.*$} -regexp
```
- You cannot automatically reverse this command. You must manually remove or disable the assumption and enable the assertion. Example:

```
// Creating an assumption from an assertion
assume -from_assert <embedded>::assert123
// New assumption created
// Original assertion is disabled

// Reverting back to the original with two commands
assume -disable {<embedded>::assert123:assume}
// assert123:assume is disabled
assert -enable {<embedded>::assert123}
// assert123 is re-enabled
```

JasperGold Apps Command Reference Manual

General Commands

Utility Switches

`-list [-silent] [-env | -task task_name]`

Show the list of assumptions in the current or specified task.

- Use `-silent` to turn off output to the screen and return assumptions with a Tcl return value. This option is helpful in Tcl scripts.
- Use the `-env` to list the assumption set for the global environment.
- Use `-task` to apply the command to the specified task instead of the current task.

`-clear [-env | -task task_name]`

Remove all interactive assumptions (that is, assumptions you specified on-the-fly).

- Use the `-env` to remove the assumption set for the global environment.
- Use `-task` to apply the command to the specified task instead of the current task.

`-show property_name [-env]`

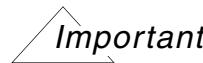
Show the actual expression for the assumption with the specified name.

Use `-env` to apply this command to the global environment.

`-remove property_name... [-regexp] [-env]`

Remove the assumption with the specified name.

Use `-env` to apply this command to the global environment.



- Use this switch to specify one or more assumptions.
- Wildcards and regular expressions are supported; for example:
`assume -remove {^.*$} -regexp`
- You can only remove properties created with the GUI or on the command line. That is, you cannot remove properties added automatically (for example, SVA and PSL embedded properties or automatically extracted type properties). However, you can disable and enable these properties.

JasperGold Apps Command Reference Manual

General Commands

```
-disable | -enable property_name... [-regexp] [-env]
```

Disable or re-enable the assumption with the specified name.

When you disable an assumption, it remains in the database, but the tool does not include it in any analysis.

Use `-env` to apply this command to the global environment.



- Specify one or more assumptions.
- Wildcards and regular expressions are supported; for example:
`assume -disable {^.*$} -regexp`

```
-rename old_name new_name [-env]  
[-update_db | -update_all_sessions | -suppress_db_update]
```

Change `old_name` assumption to `new_name`.

- Use `-env` to apply this command to the global environment.
- Use `-update_db` to add the specified assumption to the database.
- Use `-update_all_sessions` when you are running multiple sessions and you want to add the specified assumption to all of them and the database.
- Use `-suppress_db_update` when you have used `set_property_auto_update_db` on to set the tool to update the database automatically but you do not want to include the specified property.

```
-set_type property_name (approved | temporary)
```

Add a tag to the specified existing assumption (`property_name`):

- `approved` – The user has approved the assumption as valid.
- `temporary` – The user will discard the assumption later.

Note:

- By default, all assumptions are tagged as `temporary`.
- This switch supports wildcards (*) and (?) in property names.
- This switch does not apply to `-soft` assumptions.

JasperGold Apps Command Reference Manual

General Commands

```
-is_liveness property_name
```

Return true if the assumption is a liveness assumption; otherwise, return false.

Creating an Assumption Only Valid During Reset

```
-reset expression [-name property_name] [-replace | -replace_if_needed]
```

Create an assumption that is only valid during reset analysis.

- Use `-reset <expression>` to create a reset-specific assumption.
- Use `-name` to assign a name to the assumption. If you do not specify a name, the tool assigns one automatically in the form `reset_assumption_$id`.
- Use `-replace` to replace an existing assumption with the same name. If there is no assumption with the same name, this command fails.
- Use `-replace_if_needed` to replace an existing assumption or create a new assumption if it does not exist.

```
-reset ( -show property_name
          | -remove property_name
          | -list [-silent]
          | -clear)
```

Manage reset-specific assumptions as specified.

- Use `-show` to see the actual expression for the assumption with the specified name.
- Use `-remove` to remove the specified reset-specific assumption.
- Use `-list` to list all declared reset-specific assumptions and use `-silent` to return the list as Tcl return values without printing the report on the screen.
- Use `-clear` to remove all declared reset-specific assumptions.

Return

Returns a string (identifier name) for the assumption.

Examples

```
% assume -name illegal_size {wr_size != 2'b11}
% assume -env {scan_enable == 1'b0}
```

JasperGold Apps Command Reference Manual

General Commands

```
% assume -name illegal_size {WR_SIZE /= "11"}  
% assume -env {SCAN_ENABLE = '0'}  
Note: JasperGold Apps supports the implication overlapping operator (->) on the command  
line to declare properties. This operator indicates that the expressions on both sides happen  
at the same iteration.  
  
% assume {~rdy -> ~trx}  
% assume {(rdy = '1') -> (trx = '1')}  
  
# X-Prop examples  
# -----  
# Assume that the X won't occur for 10 cycles.  
% assume -xprop -bound 10 -to target_counter  
  
# The following command tells the tool to assume that signal WILL NOT have  
# an X value during the verification.  
% assume -xprop -to signal  
  
# The following command tells the tool to assume that signal will not have  
# an X value during the verification when p holds.  
% assume -xprop -to signal -precond p  
  
# For the assumption "assume -env {!rstN |-> din==0}", use the following  
# command to randomize inputs:  
% assume -xprop -is_x -to din -precond ~rstN  
  
# Pattern-matching with regular expressions:  
# -----  
# Disable assumptions that match zero or more characters followed by either  
# "_ended" or "_started"  
% assume -disable {.*(_ended|_started)} -regexp  
  
# Remove assumptions that match "trans" followed by 1 or more numbers and  
# followed by zero or more characters  
% assume -remove {trans([0-9]+).*} -regexp
```

See Also

[assert](#)
[cover](#)

auto_setup

Description

Use this command to generate a candidate setup script that analyzes and elaborates the RTL files and specifies clocks and resets. The generated setup is intended to be a starting point for further development, and it might need adjustments in terms of `defines, parameters, clocks, and resets.

Note: Although this command can jump-start the setup process, you must have a sign-off process for the setup script it generates because this feature can, in some cases, create unintended side-effects including the following:

- Incorrect files analyzed
- Erroneous elaboration switches

Syntax

```
auto_setup (-vhdl | -verilog | -v2k | -v95 | -sv)
           [-path <path>]
           [-suffix <suffix>]
           [-top_mod <module_name>]
           [-top_file <file_name>]
           [-save -file <file_name> [-force]]
```

Argument	Definition
-vhdl -verilog -v2k -v95 -sv	Tailor the setup script for the specified language. Note: This feature does not support mixed-language designs.
-path <i>path</i>	Use the specified path when locating files for automatic setup. If you do not specify a path, the tool uses the current path.
-suffix <i>suffix</i>	Filter the file set according to the specified suffix. If you do not specify a suffix, the tool does not filter by suffix.

JasperGold Apps Command Reference Manual

General Commands

`-top_mod module_name`

Use the specified module as the top module.

If you do not specify a module, the tool attempts to determine it automatically.

`-top_file file_name`

Locate the top module in the specified file.

If you do not specify a file, the tool attempts to determine it automatically.

`-save -file file_name [-force]`

In addition to generating the setup script and printing it to the screen, save it to the specified file.

Use `-force` if the specified setup file already exists, and you want to overwrite it.

Return

No value returned

Examples

```
% auto_setup -sv -suffix v -path ./design/rtl -top_file my_design.v -top_mod \
my_design -save -file setup.tcl -force
```

See Also

No related commands

auto_syn

Description

auto_syn is an automatic property synthesis flow that uses the BPS App to generate helper assertions to assist in the proof process, for example, to help a target property converge. If you did not configure the BPS App prior to calling the auto_syn command, then this command runs the BPS App default configuration. However, if you had previously configured the BPS App with the scope, scan, or waveform commands or by setting configuration variables, then auto_syn respects that configuration.

Note: Use the set_scan_no_sim_max_covers command to specify the maximum number of covers the BPS App will use for property synthesis when you use the -from_property switch. This setting has no effect with the -traces switch.

Syntax

```
auto_syn [-task <task_name>]
          [-target_property <property_name>]
          [-time_limit <time_limit>]
          [-from_property <property_list>
             |-traces '{'<file>+'}' -hier_path <path>]
          [-export_sva]
          [-dump_vcd <path>]
```

Argument	Definition
-target_property <i>property_name</i>	Use this property to focus the search for helper assertions.
-task <i>task_name</i>	Define the working task for the specified flow. If you do not specify a task, the flow uses the current task.
-time_limit <i>time_limit</i>	Set a time limit for auto_syn command operation. The default is two hours.
-from_property <i>property_list</i>	Create helper assertions from the specified properties.
-traces <i>file...</i>	

JasperGold Apps Command Reference Manual

General Commands

Create helper assertions from the specified trace files.

`-hier_path path`

Specify the hierachal path when using external waveforms.

`-export_sva`

Export the helper assertions as an SVA module and create a Tcl script for loading the exported SVA file in JasperGold Apps.

`-dump_vcd path`

Create VCD files for debugging and reuse, and specify the path of the directory for the generated VCD traces.

Return

Various

Examples

```
% auto_syn -task task_a  
% auto_syn -from_property {c1 c2 c3}  
% auto_syn -task task_a -traces {t1.fsdb t2.fsdb} -hier_path tb.top
```

See Also

[check_bps](#)

[scan](#)

[scope](#)

[set_scan_no_sim_max_covers](#)

[set_scan_seek_mode](#)

[task](#)

[waveform](#)

autoprove

Description

Autoprove is an automatic engine orchestration command that attempts to increase the verification results using a given set of resources, both time and licenses. It is a hybrid command in the sense that it calls the basic prove commands several times during the execution while dynamically deciding on the execution parameters, engines, and order of execution.

Syntax

```
autoprove (-all
           [-task <task_name_tcl_list>]
           [-property <property_name_tcl_list> [-regexp]
            ]
           [-bg]
           [-with_helpers] [-with_proven]
           [-time_limit <time_budget>]
           [-mem_limit <memory_budget>] [-chunks <N>]
           [-mode (default | prove | cover | hard | sps | coverage)]
           [-effort (low | medium | high | (user [-engines {<engine_list>}]))]
           ]
           [-asserts]
           [-covers]
           [-suppress_traces]
           [-keep_traces]
           [-assumption_lifting]
           [-dump_vcd]
           [-dump_trace
              [-dump_trace_type (shm | fsdb | vcd)]
              [-dump_trace_dir <dir_name>]
           ]
           [-verbosity <N>] [-silent]
```

Argument	Definition
-all	Prove all assertions and cover properties in all tasks.

JasperGold Apps Command Reference Manual

General Commands

```
-task [task_name_tcl_list]
```

Prove all assertions and cover properties in the specified tasks.

Note:

- If you do not supply a task name, the tool proves all assertions and covers in the current task.
- When specifying a list of task names, enclose the list in curly braces.
- This switch supports wildcards (*) and (?) in task names.

```
-property property_name_tcl_list [-regexp]
```

Prove the specified properties.

- Use `-with_helpers` to prove with helper assertions. You can prioritize helper assertions over regular assertions by submitting them first to the engines with `assert -helper...`
- Use `-with_proven` to use all proven assertions as assumptions regardless of the value of `proven_directive`.

Note:

- When specifying a list of properties, enclose the list in curly braces.
- To prove properties not in the current task prepend the property names with `<task>::`.
- All listed properties must belong to the same task.
- This switch supports wildcards (*) and regular expressions in property names, for example:

- autoprove -property top.ing?.i_ingress.c_read
autoprove -property top.ing\[1-3\].i_ingress.c_valid -regexp

```
-bg
```

Run the `autoprove` command in the background and continue to accept and run new commands.

```
-with_helpers
```

Prove with helper assertions.

You can prioritize helper assertions over regular assertions by submitting them first to the engines with `assert -helper...`

JasperGold Apps Command Reference Manual

General Commands

-with_proven

Use all proven assertions as assumptions regardless of the value of proven_directive.

-time_limit *time_budget*

Override the default two hours prove_time_limit for autoprove.

Specify *time_budget* values as an integer followed by s, m, or h (no spaces).

- s = seconds
- m = minutes
- h = hours

-mem_limit *memory_budget*

Override the default 2048MB engineCG_max_mem for autoprove.

Specify *memory_budget* values as an integer followed by m or g (no spaces).

- m = megabytes
- g = gigabytes

-chunks *N*

Split the given job into *N* subjobs.

This option is useful for big designs. By default, the tool issues one chunk.

Note:

- When using chunks, an unlimited prove_time_limit is not supported.
- -chunks is not supported in combination with -mode sps or -mode hard.

JasperGold Apps Command Reference Manual

General Commands

```
-mode default | prove | cover | hard | sps | coverage
```

Focus autoprove on the specified mode:

- `prove` – Assertions
- `cover` – Covers
- `hard` – Hard-to-prove/cover properties. For this argument, the tool must provide an adequate *time_budget*
- `sps` – Properties generated by the Structural Property Synthesis App
- `coverage` – Properties generated by the Coverage App.

```
-effort low | medium | high | user [-engines {engine_list}]
```

Override the number of engines used by `autoprove`, which is translated to the amount of load `autoprove` puts on the system.

Levels: The actual number of engines the following levels use is app-specific.

- `low` – This level is useful for a local run with limited resources.
- `medium` – Use the default `autoprove` engine set.
- `high` – Use all suitable engines. This level is very stressful. It consumes considerable resources.
- `user` – Override the engines used by `autoprove`. Follow `user` with `-engines` and `engine_list`, for example,
`autoprove -effort user -engines {B D}`

Note:

- If you do not use `-engines` and an engine list, `autoprove` uses the current `engine_mode` set.
- `autoprove` uses the new `engine_list` through all phases.

```
-asserts
```

Prove only assertions, not covers.

This command is an alias for operation mode `prove`.

```
-covers
```

Prove only covers, not assertions.

This command is an alias for operation mode `cover`.

JasperGold Apps Command Reference Manual

General Commands

-suppress_traces

Suppress trace generation.

-keep_traces

Disable trace suppression in all modes.

Note: Some autoprove operation modes suppress trace generation by default. Use this switch to ensure that traces are not suppressed.

-assumption_lifting

Only include a relevant subset of assumptions that is sufficient for the prove.

Note: This option might be useful in cases with too many assumptions.

-dump_vcd

Dump a VCD file for each Covered/Cex property found.

The VCD files are saved in the `traces` directory in the current JasperGold project directory.

Note:

- Any previously existing `traces` directory is renamed `traces.bak`.
- Any previously existing `traces.bak` directory is deleted.

-dump_trace

```
[ -dump_trace_type (shm | fsdb | vcd) ]  
[ -dump_trace_dir dir_name ]
```

Dump traces to the disk as soon as the engines find them. The tool uses the SHM format as the default and writes files in the project directory.

- Use `-dump_trace_type` with `shm`, `fsdb`, or `vcd` to specify the file dump type. VCD files are dumped as compressed files to save disk space.
- Use `-dump_trace_dir` to specify the dump directory where the files will be written.

-verbosity *N*

Override the default `prove_verbosity` level, which is 6.

This switch controls the amount of information printed to the log, the console, and the proof messages tab. 0 means silent. The higher the value, the more output.

JasperGold Apps Command Reference Manual

General Commands

-silent

Turn off output to the screen and return information with a Tcl return value.

This option is helpful in Tcl scripts. It is an alias for verbosity level 0.

Return

For autoprove return values, see [“Prove Command Return Values”](#) on page 751.

Examples

```
% autoprove -task taskA  
% autoprove -time_limit 1h  
% autoprove -mem_limit 4g  
% autoprove -task { taskA taskB taskC }  
% autoprove -property { propertyA propertyB propertyC }
```

See Also

[get_status](#)
[prove](#)
[set_dst_mode](#)
[set_engine_mode](#)
[set_max_trace_length](#)
[set_proof_simplification](#)
[set_proofgrid_max_local_jobs](#)
[set_proofgrid_per_engine_max_local_jobs](#)
[set_proofgrid_per_engine_privileged_jobs](#)
[set_prove_per_property_time_limit](#)
[set_prove_time_limit](#)
[set_proven_directive](#)
[visualize](#)

blackbox_assistant

Description

Use the command `blackbox_assistant` for cases when a full elaboration is not possible due to the complexity of the design. This command opens a dialog with a design hierarchy tree you can browse to determine which modules should be manually black boxed. You can then add the corresponding `-bbox_m` switch to the `elaborate` command.

Note:

- The extraction process attempts to elaborate each instantiated module once with all submodules black boxed, using many separate elaboration steps. You will typically run this command after a series of `analyze` commands. If you have done an elaboration already, this command destroys the existing elaboration unless you use the `-parallel` switch.
- `blackbox_assistant` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

Syntax

```
blackbox_assistant -extract_full_hier -top <module_name>
    [-parallel <num_sessions> <max_num_elabs>]
    [-elaborate_opts <elaborate_options_list>]

blackbox_assistant -suggest_bboxm -top <module_name>
    [-connectivity_map <csv_file_name.csv>]
    [-elaborate_opts <elaborate_options_list>]

blackbox_assistant -revise_bboxm <bbox_list>

blackbox_assistant -show_tree

blackbox_assistant -clear
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<pre>-extract_full_hier -top module_name [-parallel num_sessions max_num_elabs] [-elaborate_opts elaborate_options_list]</pre>	<p>Use the specified module as the top module to extract the hierarchy tree.</p> <ul style="list-style-type: none">■ Use <code>-parallel</code> to perform the underlying elaborations in separate analysis sessions, keeping the current elaboration intact.■ Use <code>-elaborate_opts</code> to pass the specified arguments to the <code>elaborate</code> command. <p> <i>Important</i></p> <ul style="list-style-type: none">■ More sessions should allow the extraction process to finish quicker.■ You might be able to avoid potential tool stability issues that can occur when you use a single analysis session to perform an extremely large number of elaborations.■ This command discards the old design tree and creates a new one.
<pre>-suggest_bboxm -top module_name [-connectivity_map csv_file_name.csv] [-elaborate_opts elaborate_options_list]</pre>	<p>Extract a “maximal black box list” to find a single, effective elaboration.</p> <ul style="list-style-type: none">■ Use <code>-top</code> to use the specified top module to extract the hierarchy tree.■ Use <code>-connectivity_map</code> if you have not already loaded a map.■ Use <code>-elaborate_opts</code> to pass the specified arguments to the <code>elaborate</code> command. <p>Note: This command discards the old design tree and creates a new one if the top is different or if you use <code>-elaborate_opts</code>.</p>

JasperGold Apps Command Reference Manual

General Commands

-revise_bboxm -bbox_list

Prune the suggested list of black-box candidates by removing the specified modules.



- Use this command when the list of suggested modules includes some that are necessary to prove connections but are not mentioned in the connectivity map.
- To avoid a quick growth in the elaborated design, for each connection, this command adds at most one more level of modules. Therefore, you might need to run the command multiple times to prove all connections.
- When it is not possible to further improve the `-bbox_m` list, the command issues an info message.
- This command may expand the design tree, but it never clears nor prunes it.

-show_tree

Show the design hierarchy tree in a dialog.

Note: The dialog displays information that is the result of the most recent `-suggest_bboxm` or `-extract_full_hier`. This command is useful when you want to review the design hierarchy tree without re-running `blackbox_assistant`.

-clear

Clear all configuration and information cached by `blackbox_assistant`.

Note: `blackbox_assistant` stores design information in a cache to avoid elaborating the same instance multiple times. With this command, all previous results stored by `blackbox_assistant` are lost.

Return

The design tree switches have no return values. The switches for black box suggestions return elaboration parameters in string format.

Examples

```
% blackbox_assistant -extract_full_hier -top top(rtl) -elaborate_opts {-vhdl}
```

JasperGold Apps Command Reference Manual

General Commands

See Also

[check_conn](#)

capture_TestCase

Description

Run this command at any point after you analyze and elaborate a design to save a testcase as a compressed file that contains all Tcl scripts run with the `source` and `include` commands and all referenced design and property files. If you used a reset file during setup, this command includes it in the compressed file. You will generally use this command to capture a testcase to share with the Cadence representative supporting you.

Note:

- The `captured.env` file, which is at the top of the directory, records the PWD at the time `capture_TestCase` is run. This information might be helpful for finding the directory where scripts were launched.
- This command does not add symbolic link targets to the testcase. Instead, it saves the links as files with the contents of their respective targets.
- The tool attempts to use `gzip` to create a compressed file, and if it does not find it, `bzip2`. If it cannot find either utility, the files remain uncompressed and the tool prints a warning.

Syntax

```
capture_TestCase <testcase>
  [-include_logs] [-include_files <file_name_list>]
  [-exclude_env] [-force]
```

Argument	Definition
<code>testcase</code>	Save the specified tarred file that contains all sourced scripts and design files and reset. <code>testcase</code> can be a full or relative path to the file you are saving.
<code>-include_logs</code>	Add all the files in the current project directory to the set of files captured by this command. This switch is helpful when you have used the GUI to run the tool rather than scripts alone.

JasperGold Apps Command Reference Manual

General Commands

-include_files *file_name_list*

Add the specified files to the set of files captured by this command.



- Use this switch to include signal files, screen shots, README files, and so forth.
- Surround file lists with curly braces.
- This switch accepts a list of files and directories. For any directories you specify, the tool includes all files within that directory and all of its sub-directories.
- Use the Tcl `glob` command to specify a list of files to include using wildcards (see "[Examples](#)" on page 218 section).

-exclude_env

Do not save a `captured.env` file, that is, do not capture environment variables.

Note: Using this switch can, in some cases, hinder our ability to run captured testcases.

-force

If the specified testcase name already exists, overwrite it.

Return

No value returned.

Examples

```
% capture_testcase -include_files {fileA ../../fileB ../../dirC/fileC}

// Launch the tool and use scripts to run your session.
% capture_testcase testcase1
Testcase tar file: testcase1.tgz
Files captured: 7

// Launch the tool and use the GUI or a combination of GUI and scripts to run
// your session.
```

JasperGold Apps Command Reference Manual

General Commands

```
% capture_testcase -include_logs testcase2
Testcase tar file: testcase2.tgz
Files captured: 19

// Capture all files within "../my_scripts/" and any of its sub-directories,
// and include any files matching the pattern "../.../*.txt".
% capture_testcase my_capture -include_files "../my_scripts/" [glob ../...*.txt]"
```

See Also

No related commands

check_arch

Description

Use `check_arch` to get information from an XML worksheet that defines a module in Architectural Modeling format or to request actions related to this module.

Note: This command has two scopes, the loaded worksheet and a specified table. The default is the loaded worksheet. To specify a table, use the `-table` switch.

Syntax

```
check_arch -table <table_name>
    -list (for | state | in | wire | tag)

check_arch [-table <table_name>]
    -list (property | assume | assert | cover)

check_arch -list (table | param | lparam | input | step)

check_arch ( -load (<xml_file_name.xml> | <murphi_file_name.m>)
    [-parameter <param_name> <param_value>]
    |-bind (<xml_file_name.xml> | <murphi_file_name.m>)
    [<instance_to_bind>]
)
[-analyze_opts <analyze_options_list>]
[-elaborate_opts <elaborate_options_list>]

check_arch -prove [-table <table_name>]

check_arch -clear
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<pre>-table table_name -list (for state in wire tag)</pre>	<p>List the information that matches the specified criteria.</p> <p>Note: The following switches are valid for the <code>-table</code> scope only.</p> <ul style="list-style-type: none">■ Use <code>for</code> to list the FOR cell range.■ Use <code>state</code> to list STATE signals defined in the specified table.■ Use <code>in</code> to list IN signals defined in the specified table.■ Use <code>wire</code> to list WIRE signals defined in the specified table.■ Use <code>tag</code> to list all tags specified for the specified table. Valid tags are COVER, FULL, NEVER, NEVER_INIT, PRLL.
<pre>[-table table_name] -list (property assume assert cover)</pre>	<p>List the information that matches the specified criteria generated by the loaded Architectural Model worksheets.</p> <p>Note: The following switches are valid for both the default and <code>-table</code> scopes.</p> <ul style="list-style-type: none">■ Use <code>property</code> to list all the properties generated.■ Use <code>assume</code> to list all the assumptions generated.■ Use <code>assert</code> to list all the assertions generated.■ Use <code>cover</code> to list all the covers generated.

JasperGold Apps Command Reference Manual

General Commands

```
-list (table | param | lparam | input | step)
```

List the information that matches the specified criteria.

Note: The following switches are valid for the worksheet scope. They do not support the -table switch.

- Use `table` to list all tables defined in the loaded Architectural Model worksheets.
- Use `param` to list all the loaded Architectural Model worksheet's parameters and their values.
- Use `lparam` to list all the loaded Architectural Model worksheet's local parameters and their values.
- Use `input` to list all the loaded Architectural Model worksheet's inputs.
- Use `step` to list all the loaded Architectural Model worksheet's step tags.

```
-load xml_file_name.xml | murphi_file_name.m  
[-parameter param_name param_value]
```

Load the module generated by the specified XML worksheet file or Murphi program as the top module.

Use `-parameter` to specify the parameter values for the top-level module.

Note:

- This switch also loads the GUI.
- This switch is not valid for the -table scope.

```
-bind xml_file_name.xml | murphi_file_name.m [instance_to_bind]
```

Bind the module generated by the specified XML worksheet file or Murphi program.

To specify a name for the instance you are binding, use `instance_to_bind`; otherwise, the instance name will be the same as the module name.

Note:

- This switch also loads the GUI.
- This switch is not valid for the -table scope.
- The module connection is handled as a requirement module.

JasperGold Apps Command Reference Manual

General Commands

```
[-analyze_opts analyze_options_list]
```

Pass the specified arguments to the `analyze` command.

```
[-elaborate_opts elaborate_options_list]
```

Pass the specified arguments to the `elaborate` command.

```
[-table table_name] -prove
```

Prove the properties generated by the loaded Architectural Model worksheet.

Use the `-table` scope to prove only properties related to the specified table.

Note: The `set_cmd_time_limit` command has no effect on `-prove`; instead, it honors prove time-limit commands.

```
-clear
```

Clear the Architectural Model worksheet.

Note: You must manually clear the properties that were created.

Return

A list of requested objects or the output of a requested action.

Examples

```
% check_arch -table table1 -list for  
% check_arch -list table  
% check_arch -table table1 -list assert  
% check_arch -list assert  
% check_arch -load ace.xml  
% check_arch -bind ace.xml  
% check_arch -bind ace.xml ace_2  
% check_arch -table table1 -prove  
% check_arch -prove
```

See Also

No related commands

check_assumptions

Description

Use the `check_assumptions` command to prove the `check_assumptions` assertions, which are special properties built into every task to check the effects of assumptions. The properties `:noConflict` and `:noDeadEnd` will be shown in your task when you run this command. For example, you can see them displayed in the Property Table. The property types are `Assert (noConflict)` and `Assert (noDeadEnd)`. The following list defines the properties and explains what you can learn from the proof results they produce on your design.

- `:noConflict`: This property asserts that the assumptions and proof setup should allow infinite traces. The property will have one of the following statuses:
 - `cex`: A violation trace will illustrate that at a certain cycle (the last in the trace) it is impossible to satisfy all assumptions. We call this an (unavoidable) conflict.
 - `undetermined`: A sanity trace is likely available. It will illustrate that there are no conflicts this short. A longer and longer sanity trace will become available during proof. The bounds show how many cycles are conflict free.
 - `proven`: A looping sanity trace is available. It will illustrate that there are infinite traces, that is, there is `no_conflict`.
- `:noDeadEnd`: This property asserts that every reachable state has a next cycle satisfying all assumptions. The property will have one of the following statuses:
 - `cex`: A violation trace will illustrate that at a certain cycle (the last in the trace) it is impossible to satisfy all assumptions, given the preceding state. We call this a `dead_end`.
 - `undetermined`: The bounds show how many cycles are `dead_end` free.
 - `proven`: There is `no_dead_end` at all. The bound is infinite.

This feature helps you understand the effects assumptions have on the design. When a `cex` is found, it warns you about conditions that might indicate that the assumptions are stronger than intended and provides a trace for inspection. In effect, `check_assumptions` gives you confidence that you have verified what you intended.

Note:

- Sequential assumptions may in themselves introduce benign dead ends that are hard to avoid, so a `dead_end` does not necessarily indicate a problem in itself.

- The command uses a custom engine mode, but beyond that, it respects all ProofGrid and prove settings. See “[prove](#)” on page 741 for additional information.
- The command may, as a side effect, prove other properties as well.

Implementation guidance:

- It is always recommended that you run the `check_assumptions` command at least once when the proof environment is finalized with all the constraints to detect any unexpected overconstraint problem.
- `check_assumptions` does not check liveness assumptions.

Note: Assumptions might be stronger than you intended even if they do not cause any conflicts. Assumptions still need to be validated by other means, that is, assume-guarantee.

Syntax

```
check_assumptions [-all |-task <task_name_tcl_list>]
                  [-conflict |-dead_end]
                  [-bg]
                  [-time_limit <time_limit>]
                  [-max_length <N>]
                  [-verbosity <N> |-silent]
                  [-minimize]

check_assumptions -trace
                  [-task <task_name>]
                  [-conflict |-dead_end]

check_assumptions -show [-all |-task <task_name_tcl_list>]
```

Argument	Definition
<code>[-all -task task_name_tcl_list]</code>	<p>Prove the <code>check_assumptions</code> assertions in the specified tasks.</p> <p>Note:</p> <ul style="list-style-type: none">■ If you do not use <code>-all</code> or <code>-task</code>, the command applies to the current task.■ When specifying a list of task names, enclose the list in curly braces.■ The <code>-task</code> switch supports wildcards (*) and (?).

JasperGold Apps Command Reference Manual

General Commands

`[-conflict | -dead_end]`

Use `-conflict` to only run the engines focused on conflict, and use `-dead_end` to only run the engines focused on dead end. In all cases you may get results for both properties.

`[-bg]`

Run the proof in the background and continue to accept and run new commands. See “[prove](#)” on page 741 for operations available on a proof running in the background.

`[-max_length N]`

Override the current `max_trace_length` for this command. See “[set_max_trace_length](#)” on page 1114.

`[-time_limit time_limit]`

Override the current `prove_time_limit` for this command. See “[set_prove_time_limit](#)” on page 1189.

`-verbosity N`

Override the current `prove_verbosity` for this command. See “[set_prove_verbosity](#)” on page 1192.

`-silent`

Alias for `-verbosity 0`.

`-minimize`

If a conflict or a dead end is established invoke `get_needed_assumptions` to provide a minimized debugging environment for the cex. See “[get_needed_assumptions](#)” on page 645 for additional information.

Note: `-minimize` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

`-trace`

`[-task task_name]`
`[-conflict | -dead_end]`

Display the trace for the specified property (if available).

`-show [-all | -task task_name]`

Show the `check_assumptions` properties in the current or specified task(s).

Return

When the `check_assumptions` command is not running in the background, the tool returns the values listed in [Table 4-1](#) on page 227 below.

Table 4-1 Command Return Values

Return Values	Definition
<code>conflict</code>	There is an unavoidable conflict at a given cycle. This implies that there is also a dead end.
<code>no_dead_end</code>	There are no dead end states. This implies that there also are no conflicts.
<code>dead_end</code>	There is a dead end state.
<code>no_conflict</code>	There are no unavoidable conflicts, and it is unknown whether there are <code>dead_end</code> states.
<code>sanity</code>	There are no unavoidable conflicts up to a given length.
<code>undetermined</code>	Not even a sanity trace is available.
<code>background</code>	The <code>check_assumptions</code> command is running in the background.
<code>background (thread <i>thread_id</i>)</code>	The <code>check_assumptions</code> command is running in the background and is associated with thread <i>thread_id</i> . (Applies only if the <code>parallel_proof_mode</code> variable is on.)

Examples

```
% check_assumptions -task <embedded> -conflict
% check_assumptions -trace -task <embedded> -conflict
```

See Also

[get_needed_assumptions](#)
[prove](#)
[set prove time limit](#)

check_bps

Description

Use the `check_bps` command to manipulate property candidates obtained from the Behavioral Property Synthesis (BPS) App property extraction process. Each property candidate has an observer type, a type, a classification, and a state in the progress report.

Note: In some commands, the observer type and the type are collapsed to make the command less verbose.

The following list shows the observer types, types, classifications, collapsed types, and Progress Report states.

- Observer types – hole or exercised
- Types – assert, assume, cover
- Classifications – certified, dont_care, unclassified
- Collapsed types – assert, assume, exercised_cover, coverage_hole
- Progress Report states – new, obsolete, hole, covered, covered_now_hole, assert, violated, certified_hole, certified_assert, dont_care

The tool uses intelligent heuristics to determine the desirable default type. For example, a `stuck_at` situation is by default a coverage hole.

Initially, all extracted properties are unclassified allowing you to determine the relative significance of these properties.

Furthermore, the tool ranks each property, and you can use the `-focus` switch to manipulate only the high-ranked candidates, high-and-medium-ranked candidates, or all candidates.

Syntax

```
check_bps -init  
  
check_bps <id_tcl_list>  
    -set_type ( assert | assume | exercised_cover | coverage_hole  
                | assert_assume)  
    [-silent]  
  
check_bps <id_tcl_list>  
    -set_class (certified | dont_care | unclassified)  
    [-silent]
```

JasperGold Apps Command Reference Manual

General Commands

```
check_bps <id_tcl_list>
    -set_description <description>
    [-silent]

check_bps <id_tcl_list>
    -set_rank (high | medium | low)
    [-silent]

check_bps <id_tcl_list>
    -set_active_task <task_name>
    [-silent<]

check_bps <id_tcl_list>
    -get (type | observer_type | class | description | expression | rank
        | representation | scope | status | seeker| task | fpv_property
        | progress_state | signal)
    [-baseline] [-silent]

check_bps -export
    (-type (assert | assume | exercised_cover | coverage_hole))*
    (-class (certified | dont_care | unclassified))*
    (-status (all | cex | ar_cex | proven | ar_covered | unprocessed
        | covered_reset | unreachable | undetermined | partial_cex
        | partial_ar_cex | partial_proven | partial_ar_covered
        | partial_covered_reset | partial_unreachable | partial_undetermined))*
    [-focus (high | medium | low)]
    (-progress_state (new | obsolete | hole | covered | covered_now_hole
        | assert | violated | certified_hole | certified_assert | dont_care))*
    [-module <sva_module_name>]
    [-task <task_name>]
    [-current_task]
    [-baseline]
    [-silent]

check_bps -export
    -property <id_tcl_list>
    [-module <sva_module_name>]
    [-task <task_name>]
    [-baseline]
    [-silent]

check_bps -export_fpv_setup
    [-file <setup_file_name>]
    [-task <task_name>]
    [-property <id_tcl_list>]

check_bps -rank [-mode (imp_only | spec_only)]
    [-engine_mode <engine_mode_list>]
    [-time_limit <time_limit>]
    [-iter <N>]
    [-task <task_name>]
    [-property <id_tcl_list>]
    [-batch]
    [-debug]
```

JasperGold Apps Command Reference Manual

General Commands

```
check_bps -show <id_tcl_list>
    [-baseline] [-silent]

check_bps -list
    [-type (assert | assume | exercised_cover | coverage_hole)]*
    [-class (certified | dont_care | unclassified)]*
    [-regexp <property_expression_regexp>]
    [-seeker <seeker_name>]
    [-focus (high | medium | low)]
    [-status (all | cex | ar_cex | proven | ar_covered | unprocessed
        | covered_reset | unreachable | undetermined| partial_cex
        | partial_ar_cex | partial_proven | partial_ar_covered
        | partial_covered_reset | partial_unreachable | partial_undetermined)]*
    [-progress_state (new | obsolete | hole | covered | covered_now_hole
        | assert | violated | certified_hole | certified_assert | dont_care)]*
    [-scope <poi_tcl_list>]
    [-signal (<signal>)+]
    [-baseline] [-silent]

check_bps -progress
    [-weekly |-monthly]
    [-top |-instance <instance_name>]
    [-focus (high | medium | low)]
    [-gui] [-silent]

check_bps -highlight <id>

check_bps -witness <id_tcl_list>
    [-baseline]

check_bps -clear
```

Argument	Definition
-init	Initialize the settings required by the Behavioral Property Synthesis App.

JasperGold Apps Command Reference Manual

General Commands

```
id_tcl_list -set_type  
  (assert | assume | exercised_cover | coverage_hole | assert_assume)  
  [-silent]
```

Change the observer type and the type of the specified property candidates.

This operation typically changes the expression of the property as well. Properties can be generated by different engines that determine whether they can be converted to another type. Properties may be converted to one other, two other, or no other representations/types.

Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

```
id_tcl_list -set_class (certified | dont_care | unclassified)  
  [-silent]
```

Change the class of the specified property candidates.

Use this switch to control which candidates should be exported.

Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

```
id_tcl_list -set_description description  
  [-silent]
```

Add comments to the specified property candidates.

```
id_tcl_list -set_rank (high | medium | low)  
  [-silent]
```

Change the rank of the specified property candidates.

Use this switch to overwrite the default rank assigned by the tool.

Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

```
id_tcl_list -set_active_task task_name  
  [-silent]
```

Change the active export task of the specified property candidates.

Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

JasperGold Apps Command Reference Manual

General Commands

```
id_tcl_list -get
( type | observer_type | class | description | expression
| rank | representation | scope | status | seeker | task
| fpv_property | progress_state | signal)
[-baseline]
[-silent]
```

Report the specified attribute of the specified property candidates.

- Use `type` to report the property candidate type: assertion, assumption, exercised cover, or coverage_hole.
- Use `observer_type` to report whether the property candidates are exercised or holes.
- Use `class` to report the class for property candidates: certified, dont_care, or unclassified.
- Use `description` to display comments for the specified property candidates.
- Use `expression` to report the expressions for property candidates.
- Use `rank` to report the property rank: high, medium, or low.
- Use `representation` to report available representations for property candidates, for example, assert, assume, or cover.
- Use `scope` to report the scope (POI ID) for the property candidates.
- Use `status` to report the proof status for the property candidates.
- Use `seeker` to report the seeker that generated the property.
- Use `task` to report the tasks for which the property candidates were exported.
- Use `fpv_property` to report the Formal Property Verification property names in the active exported tasks for the property candidates.
- Use `progress_state` to report the latest state of a property in the progress report. Use `signal` to report all the signal names in the property candidate's expression.
- Use `-baseline` to show the baseline property candidates.
- Use `-silent` to turn off output to screen.

JasperGold Apps Command Reference Manual

General Commands

```
-export
  -type (assert | assume | exercised_cover | coverage_hole) +
  -class (certified | dont_care | unclassified) +
  -status ( all | cex | ar_cex | proven | ar_covered | unprocessed
            | covered_reset | unreachable | undetermined
            | partial_cex | partial_ar_cex | partial_proven
            | partial_ar_covered | partial_covered_reset
            | partial_unreachable | partial_undetermined) +
  [-focus (high | medium | low)]
  -progress_state ( new | obsolete | hole | covered
                     | covered_now_hole | assert | violated
                     | certified_hole | certified_assert
                     | dont_care) +
  [-module sva_module_name]
  [-task task_name]
  [-baseline]
  [-silent]
```

Export the property candidates to the Formal Property Verification App.

- Use **-type** to specify a type of property candidate to export. The default is **assert** and **cover**.
- Use **-class** to specify a class of property candidate to export. The default is **certified**.
- Use **-status** to specify a status of property candidate to export. The default is **all**.
- Use **-focus** to select how many property candidates to export.
 - **high** – Export only high-rank property candidates.
 - **medium** – Export both high- and medium-rank property candidates.
 - **low** – Export all property candidates.

The default is **high**.

JasperGold Apps Command Reference Manual

General Commands

-export (Continued)

- Use -progress_state to specify a progress report state of property candidate to export. The default is all.
- Use -module to specify a module name for SVA export. This command overwrites the default name.
- Use -task to export the property candidates to the specified task instead of creating a new task.
Note: Replace *task_name* with a period (.) to specify the current task.
- Use -baseline to use the property candidates in the baseline.
- Use -silent to suppress messages and generate only Tcl return values.

-export

```
-property id_tcl_list
[-module sva_module_name]
[-task task_name]
[-baseline]
[-silent]
```

Export the property candidates to the Formal Property Verification App.

- Use -property to specify a set of property candidate IDs to export.
- Use -module to specify a module name for SVA export. This command overwrites the default name.
- Use -task to export the property candidates to the specified task instead of creating a new task.
Note: Replace *task_name* with a period (.) to specify the current task.
- Use -baseline to use the property candidates in the baseline.
- Use -silent to suppress messages and generate only tcl return values.

JasperGold Apps Command Reference Manual

General Commands

```
-export_fpv_setup  
  [-file setup_file_name]  
  [-task task_name]  
  [-property id_tcl_list]
```

Export a Formal Property Verification App setup environment from the BPS App. This setup environment includes basic set up, clock configuration, and assumptions export.

- Use `-file` to generate a Formal Property Verification App setup file (Tcl and SVA).
 - Use `-task` to export the property candidates to the specified task instead of creating a new task.
- Note:** Replace `task_name` with a period (.) to specify the current task.
- Use `-property` to export a specified set of property candidates to a task or file.



- For clock configuration, if needed, use `-clock_edge auto_detect` in the scope `-extract` command.
- To get constraints on the outputs of a black box, use the `-bbox` switch when extracting module interfaces.
- To get constraints on stopats, use `scope -add [stopat -list]`.
- To automatically create an initial formal environment, use the command `formal Bring_up`. See "["formal Bring_up"](#)" on page 552.

JasperGold Apps Command Reference Manual

General Commands

```
-rank
[-mode {imp_only | spec_only}]
[-engine_mode engine_mode_list]
[-time_limit time_limit]
[-iter N]
[-task task_name]
[-property id_tcl_list]
[-batch]
[-debug]
```

Rank candidate properties by functional complexity.

- Use `-mode` to analyze complexity based on RTL implementation only, `imp_only`, or embedded properties only, `spec_only`.
- Use `-engine_mode` to specify one or more engines.
- Use `-time_limit` to specify a per property time limit.
- Use `-iter` to specify an iteration limit.
- Use `-task` to select a task from which to read embedded properties. The default is `<embedded>`.
- Use `-property` to select the properties to filter. The default is high-rank properties.
- Use `-batch` to run in batch mode.
- Use `-debug` to print debugging messages.

```
-show id_tcl_list
[-baseline]
[-silent]
```

Show the details of the property candidates.

- Use `-baseline` to show the baseline property candidates.
- Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

JasperGold Apps Command Reference Manual

General Commands

```
-list
[-type (assert | assume | exercised_cover | coverage_hole)+]
[-class (certified | dont_care | unclassified)+]
[-regexp property_expression_regexp]
[-seeker seeker_name]
[-focus (high | medium | low)]
[-status ( all | cex | ar_cex | proven | ar_covered
           | unprocessed | covered_reset | unreachable
           | undetermined | partial_cex | partial_ar_cex
           | partial_proven | partial_ar_covered
           | partial_covered_reset | partial_unreachable)+]
[-progress_state ( new | obsolete | hole | covered
                     | covered_now_hole | assert | violated
                     | certified_hole | certified_assert
                     | dont_care)+]
[-scope poi_tcl_list]
[-signal signal_name]
[-baseline]
[-silent]
```

List the IDs for the property candidates that match the specified criteria.

Note: If you use `check_bps -list` with no additional arguments, the tool exports high focus property candidates.

- Use `-type` to list properties of a specific type.
- Use `-class` to list properties of a specific classification.
- Use `-regexp` to filter properties by property expression using regular expressions.
- Use `-seeker` to filter properties by seeker name.
- Use `-focus` to filter out properties by rank.
- Use `-status` to list properties of a specific status. The default is `all`.
- Use `-progress_state` to list properties of a specific progress report state. The default is `all`.
- Use `-scope` to list only properties derived from the specified POIs.
- Use `-signal` to list only properties that involve at least one of the specified signals.
- Use `-baseline` to use the baseline property candidates.
- Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

JasperGold Apps Command Reference Manual

General Commands

-progress

```
[-weekly | -monthly]
[-top | -instance instance_name]
[-focus ( high | medium | low )]
[-gui] [-silent]
```

Summarize the property progress state and print it to the console.

- Use `-weekly` to summarize progress by weekly aggregation.
- Use `-monthly` to summarize progress by monthly aggregation.
- Use `-top` to summarize progress for the top instance.
- Use `-instance` to summarize progress for a specified instance.
- Use `-focus` to summarize progress for the specified focus.
- Use `-gui` to display a chart of the summarized data.
- Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

-highlight *id*

Highlight the property with the specified ID in the *Property Candidates* pane. (The tool takes no action if the ID is invalid or it is currently invisible due to filtering.)

-witness *id_tcl_list*

```
[-baseline]
```

Show trace examples for property behavior. The tool provides examples as trace name and simulation start/end time pair.

Use `-baseline` to show trace cycles for baseline property candidates.

-clear

Clear all BPS App information without clearing the results of the analysis and elaboration.

Return

Context-sensitive return values

Examples

```
% check_bps -list
% check_bps -list -regexp {.*my_signal_name.*} -focus low
```

JasperGold Apps Command Reference Manual

General Commands

```
% check_bps -rank -mode imp_only -property {1 2 3 4}  
% check_bps -export_fpv_setup -file setup_file  
% check_bps 3 -get property_status
```

See Also

[formal_bring_up](#)

[scan](#)

[scope](#)

[waveform](#)

check_code

Description

Use this command to check whether an actual value matches the expectations. If it does not, `check_code` returns with an error code and aborts command interpretation unless you also use `catch`. The tool evaluates the optional `on_error` block before returning with an error code.

Note: This command is useful when creating a regression script.

Syntax

```
check_code <actual> <expected> [ <on_error> ]
```

Argument	Definition
<code>actual</code>	
	Actual value.
<code>expected</code>	
	Expected value.
<code>[on_error]</code>	
	Evaluate when the actual and the expected values do not match.

Return

No value returned.

Examples

```
% check_code [ prove ~pci_code.error ] cex
% check_code [ prove -status -silent ] 0.H {prove -stop}
% check_code [join [get_design_info -list fsm -silent]] fsm1 fsm2
```

See Also

[check_one_of](#)

check_conn

Description

Use `check_conn` to get information from an XML worksheet that defines properties in the Connectivity format or to request actions related to this worksheet.

This command has two scopes, the loaded worksheet and a specified connection. The default is the loaded worksheet. To specify a connection, use the `-connection` switch.

Syntax

[List Switches](#)

```
check_conn [-connection <connection_name>]
    -list ( condition | condition_stable | cond_expr | latencies
            | src_signal | dest_signal | property | assert | cover)

check_conn -interface_connection <connection_name>
    -list (src_signal | dest_signal)

check_conn -candidate <candidate_id>
    -list (src | dest | condition | clock | latencies | path)

check_conn -list ( connection | interface_connection | interface
                  | interface_map | assume)

check_conn -list candidate
    [-src <signal_name>] [-dest <signal_name>] [-silent]
```

[Action Switches](#)

```
check_conn -load (<xml_file_name.xml> | <csv_file_name.csv>)
    [-filter [-regexp]]

check_conn -save_as <file_name> [-force]

check_conn -import <csv_file_name.csv>

check_conn -prove [-connection <connection_name>]
    [-bg]

check_conn -partitioned_prove [-parallel] [-partition <partition_name>]

check_conn -clear [candidate]

check_conn -generate_template <file> [-force]
    [-csv | -xml | -xlsm | -ods] [verbose | compact]
```

JasperGold Apps Command Reference Manual

General Commands

```
check_conn -expand_interface_connection
    (<interface_connection_tcl_list> | -all)
check_conn -generate_toggle_checks <connection_name_list>
```

[Reverse Connectivity](#)

```
check_conn -suggest_missing_connections
    [(-csv | -xml) <file_name>] [-append] [-force] [-silent]
check_conn -generate_candidates -complexity ( straightforward | simple
                                              | conditional | complex)
    [-src '{' (<signal_name> | <module_name> | <instance_name>)+ '}' ]
    [-dest '{' (<signal_name> | <module_name> | <instance_name>)+ '}' ]
    [-level (down | same | up)]
    [-hier <N>]
    [-max_candidates <N>]
    [-max_length <N>]
    [-max_bits_inspected <N>]
    [-silent]
check_conn -save_candidates -file <file_name> [-force] [-append]
check_conn -generate_connection_map
    (<candidate_id_list> |-file <file_name>)
    -save_as <file_name>
    [-force] [-append] [-xml]
    [-refine_candidates] [-remove_unreachable]
check_conn -compare_to_rtl
    [-no_suggestions]
    [-report_file <csv_file_name.csv>]
    [-append] [-force]
    [-max_length <N>]
    [-max_comparisons <N>]
check_conn -validate
check_conn -extract_condition
    -src <signal_name>
    -dest <signal_name>
    [-verbosity <N>]
    [-silent]
```

[IP-XACT Related Switches](#)

```
check_conn -load_ipxact <spirit:design_name>
    [-instance <instance_name>]
    [-interfaces] [-infer_interface_definitions] [-no_debug]
```

JasperGold Apps Command Reference Manual

General Commands

Report Switches

```
check_conn -get_status [<connection_name>]  
check_conn -show_trace <connection_name>  
check_conn -show_schematic -connection <connection_name>
```

Partition Switches

```
check_conn -load_partition_map  
check_conn -save_partition_map (<xml_file_name.xml>)
```

Argument	Definition
List Switches	
-connection <i>connection_name</i> -list (condition condition_stable cond_expr latencies src_signal dest_signal property assert cover)	

JasperGold Apps Command Reference Manual

General Commands

List the information that matches the specified criteria.

Note: The following switches are valid for the loaded worksheet (default scope) or a specified connection.

- Use `-connection` to specify a connection other than the loaded worksheet.
- Use `condition` to list all `CONDITION` clauses defined in the worksheet or for the specified connection.
- Use `condition_stable` to list all `CONDITION (STABLE)` clauses defined in the worksheet or for the specified connection.
- Use `cond_expr` to list all `COND_EXPR` clauses defined in the worksheet or for the specified connection.
- Use `latencies` to list all `LATENCIES` clauses defined in the worksheet or for the specified connection.
- Use `src_signal` to list source signals defined in the worksheet or for a specified connection.
- Use `dest_signal` to list destination signals defined in the worksheet or for a specified connection.
- Use `property` to list all properties generated by Connectivity from the loaded worksheet or for a specified connection.
- Use `assert` to list all assertions generated by Connectivity from the loaded worksheet or for a specified connection.
- Use `cover` to list all precondition cover directives generated by Connectivity.

```
-interface_connection connection_name
    -list (src_signal | dest_signal)
```

List the information that matches the specified criteria.

Note: The following switches are valid for the specified interface connection. The source and destination signals are listed in the same order.

- Use `src_signal` to list the signals defined for the source interface.
- Use `dest_signal` to list the signals defined for the destination interface.

```
-candidate candidate_id
    -list (src | dest | condition | clock | latencies | path)
```

JasperGold Apps Command Reference Manual

General Commands

List the information that matches the specified criteria.

- Use `src` to access the source signal or constant.
- Use `dest` to access the destination signal.
- Use `condition` to access the condition expression.
- Use `clock` to access clock information. The result is a pair: clock name and edge.
- Use latencies to access latency values. The result is a list of five natural numbers. To learn more about these, see the Connectivity App guide (*Help – Application Guides – Connectivity Verification*).
- Use `path` to list the signals in the candidate's path, from source to destination.

```
-list (connection | interface_connection | interface | interface_map | assume)
```

List the information that matches the specified criteria.

Note: The following switches are valid for the worksheet scope. They do not support the `-connection` scope.

- Use `connection` to list all CONNECTION names defined in the worksheet.
- Use `interface_connection` to list all INTERFACE_CONNECTION names defined in the worksheet.
- Use `interface` to list all INTERFACE names defined in the worksheet.
- Use `interface_map` to list all INTERFACE_MAP relationships defined in the worksheet.
- Use `assume` to list all assumptions generated by Connectivity from the loaded worksheet.

```
-list -candidate [-src signal_name] [-dest signal_name] [-silent]
```

List candidate IDs, which can be used to access candidate data.

- Use `src` to list all candidates with a given source signal.
- Use `dest` to list all candidates with a given destination signal.
- Use `-silent` to turn off printing the candidates on the screen.

JasperGold Apps Command Reference Manual

General Commands

Action Switches

```
-load (xml_file_name.xml | csv_file_name.csv) [-filter [-regexp]]
```

Load the Connectivity map present in the XML or CSV workbook file and the properties generated from its specification.

- Use `-filter` to specify that you want the tool to consider wildcards for source or destination signal inputs. And use `-regexp` to consider regular expression matching.
 - Wildcards are allowed in the following cases:
 - The source or destination signal in a CONNECTION row
 - The source or destination interface instance name in an INTERFACE_CONNECTION row
 - You can only specify wildcard and regular expression filtering for a source **or** destination, not for both sides.
 - You can specify any of the following reserved constants in the opposite side of a wildcard specification:
 - JDA:LOW – All bits of each matched signal are tied to zero
 - JDA:HIGH – All bits of each matched signal are tied to one
 - JDA:SAME – Signal names are the same in both sides
 - JDA:DEFAULT – Each matched signal is tied to the default value specified in the interface map

Note: JDA:LOW and JDA:HIGH apply to both CONNECTION and INTERFACE_CONNECTION; JDA:SAME applies to CONNECTION only; and JDA:DEFAULT applies to INTERFACE_CONNECTION only.

JasperGold Apps Command Reference Manual

General Commands

-load (Continued)



- -load also loads the GUI.
- These switches are not valid for the connection scope.
- You can load multiple Connectivity maps in the same setup in an analysis session.
- When specifying values in CSV format, do not use spaces between a comma and the double quote used to enclose a field as this causes unintended parsing.

Example:

```
# The following causes a parsing error:  
COND_EXPR, "bridge_addr.bridge_i.fsm_tx[1]"  
  
# The following produces the desired  
# results:  
COND_EXPR, "bridge_addr.bridge_i.fsm_tx[1]"
```

-save_as file_name [-force]

Save all connections loaded into the Connectivity App to the specified file.

This command saves connections in expanded format, as displayed in the Connections Table.

Use -force if the specified file already exists and you want to overwrite it

-import csv_file_name.csv

Import a proprietary connectivity format data file into JasperGold connectivity format.

Note: Not all formats are supported. Contact support@cadence.com if properties are generated incorrectly when using this option.

JasperGold Apps Command Reference Manual

General Commands

```
-prove [-connection connection_name] [-bg]
```

Prove all properties in the Connectivity task.

- Use `-connection` to limit the proof to properties generated for the specified connection.
- Use `-bg` to run the command in the background and continue to accept and run new commands.

Note:

- The session-specific commands, such as `visualize -violation`, `visualize -cover`, and `get_status` do not return the data from this analysis.
- The `set_cmd_time_limit` command has no effect on `-prove`; instead, it honors prove time-limit commands.

```
-partitioned_prove [-parallel] [-partition partition_name]
```

Prove properties of each partition in the current analysis session, serially.

- Use `-parallel` to open one helper analysis session for each partition and prove properties in these helper analysis sessions in parallel.

Helper analysis sessions send the proof results back to the primary analysis session and close themselves automatically. Each helper analysis session has a different design loaded, according to the black-box modules list generated during the partition's creation. You can add or remove modules from the black-box list through the GUI.
- Use `-partition` to run a proof of a specified partition.

If you have also specified `-parallel`, the tool proves the partition in a helper analysis session using the black-box modules list.

```
-clear [candidate]
```

Remove the Connectivity properties from the current connectivity map(s) and the Connectivity task created by the `check_conn` command.

Use the `candidate` argument to clear only candidate data.

JasperGold Apps Command Reference Manual

General Commands

```
-generate_template file [-force]
[-csv | -xml | -xlsm | -ods] [-verbose | -compact]
```

Generate a template in which to complete the Connectivity map.

- Use `-force` to overwrite the existing file.
- Use `-csv` to generate a template in comma-separated values format (default).
- Use `-xml` to generate a template in Microsoft® Excel® 2004 XML worksheet format.
- Use `-xlsm` to generate a macro-based template for Microsoft® Excel® 2013 XMLS worksheet format
- Use `-ods` to generate a macro-based template for OpenOffice format.
- Use `-verbose` to generate a verbose template, which includes column headers for the conditions. This is the default setting.
- Use `-compact` to generate a compact template, which prints the conditions directly beneath the connection.

```
-expand_interface_connection (interface_connection_tcl_list | -all)
```

Expand interface connections, creating properties for each pair of signals.

Use `-all` to expand all interface connections. Otherwise, provide a list of interface connections. This command does not accept an empty list.

```
-generate_toggle_checks connection_name_list
```

Generate toggle checks for the specified connections.

Note: Toggle checks are cover properties that ensure that each source signal bit will change value (from 0 to 1 and from 1 to 0).

Reverse Connectivity

The switches `-suggest_missing_connections`, `-refine_candidates`, and `-compare_to_rtl` are initial release features to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

```
-suggest_missing_connections  
[ (-csv | -xml) file_name] [-append] [-force] [-silent]
```

Suggest missing connections based on the structural analysis of the logic within the top module.

- Use `-csv` to create a workbook file in CSV format.
 - Use `-xml` to create a workbook file in XML format.
 - Use `-append` to add the content to an existing file.
 - Use `-force` to overwrite the existing file.
 - Use `-silent` to return the list as Tcl return values without printing the list on the screen.
-

JasperGold Apps Command Reference Manual

General Commands

```
-generate_candidates -complexity
(straightforward | simple | conditional | complex)
[-src '{{ (signal_name | module_name | instance_name)+ '}}']
[-dest '{{ (signal_name | module_name | instance_name)+ '}}']
[-level (down | same | up)]
[-hier N]
[-max_candidates <N>]
[-max_length <N>]
[-max_bits_inspected <N>]
[-silent]
```

Extract connection candidates from the design using structural analysis. The result is a list of connection candidate IDs, which you can use to list specific candidate information or generate a connectivity map.

The connection candidates are classified in four categories, depending on the gates that the connection may go through:

- straightforward – buffers
- simple – buffers, flops, and inverters
- conditional – buffers, flops, inverters, latches, muxes, AND gates, and inouts
- complex – buffers any gate

Note: If you do not supply a category, the tool uses conditional.

To guide the extraction, you must provide either a list of sources, a list of destinations, or both. Each of these lists may contain signal, instance, and module names. For instances and modules, the extraction starts with their inputs, outputs, and inouts.

- Use `-src` to provide the list of sources.
 - Use `-dest` to provide the list of destinations.
-

JasperGold Apps Command Reference Manual

General Commands

-generate_candidates (Continued)

- Use `-level` with one of the following enums to limit the extraction by specifying the direction that the algorithm should follow, starting from the specified side of the connection.
 - `same` – Both source and destination should be either signals of the same instance or signals of sibling instances.
 - `down` – Guides the extraction to inspect only subinstances.
 - `up` – Guides the extraction to inspect only instances directly above.
- Note:** `-level` is not allowed if you use both `-src` and `-dest`.
- Use `-hier` to limit the extraction by stopping the algorithm at a certain hierarchical depth difference. For example, use `-hier 1` to find connections in direct subinstances or the parent instance.
- Use `-max_candidates` to limit the number of candidates extracted.
- Use `-max_bits_inspected` to limit the number of signal bits inspected during extraction.
- Use `-max_length` to limit the length of candidates extracted.
- Use `-silent` to suppress the extraction summary.

-save_candidates -file *file_name* [-force] [-append]

Save candidate data into a file that can later be used to generate a connection map.

- Use `-force` to overwrite any existing data in the file.
- Use `-append` to append the candidate data to the specified file.

JasperGold Apps Command Reference Manual

General Commands

```
-generate_connection_map (candidate_id_list | -file file_name)
    -save_as file_name [-force] [-append] [-xml]
    [-refine_candidates] [-remove_unreachable]
```

Generate a connectivity map using the specified connection candidates.

- Use `-file` to get the candidates from a file, previously saved with `check_conn -save_candidates`. Otherwise, specify a list of candidate IDs.
- Use `-save_as` to specify the name of the file where the connectivity map should be saved.
- Use `-force` to overwrite any existing data in the file.
- Use `-append` to append the candidate data to the specified file.
- Use `-xml` to generate the connectivity map in XML format. The default format is CSV.
- Use `-refine_candidates` to run a functional analysis that inspects each connection candidate. This switch will improve the connection condition; however, since the proof engines are required, it might impact performance.
- Use `-remove_unreachable` to ignore connection candidates with unreachable conditions.

Note: This feature requires the proof engines; therefore, it might impact performance.

JasperGold Apps Command Reference Manual

General Commands

```
-compare_to_rtl [-noSuggestions]
[-report_file csv_file_name.csv]
[-append] [-force]
[-max_length N]
[-max_comparisons N]
```

Run structural analysis to compare connections specified by the user to those found in the implementation. This feature outputs a summary of specified connections that were not implemented (`missing`), differences identified between specified and implemented connections (`differences`), and connections implemented but not specified (`suggestions`).

- Use `-noSuggestions` to limit the comparisons to specified connections. If your objective is to confirm specified connections, this switch should result in better performance.
- Use `-report_file` to save the output to a file in CSV format.
- Use `-append` to add the content to an existing file.
- Use `-force` to overwrite the existing file.
- Use `-max_length` to limit the length of connections inspected by the structural analysis.
- Use `-max_comparisons` to limit the total number of connection bits compared during the analysis. For instance, for a connection of width 4, at least 4 comparisons are required.

```
-validate
```

Check, for each connection, if the source is in the cone influence of the destination and update the *Connections Table* to indicate the result.

JasperGold Apps Command Reference Manual

General Commands

```
-extract_condition -src <signal_name> -dest <signal_name>
[-verbosity N]
[-silent]
```

Extract the condition expression for the connection between the specified signals. The result is a list of pairs of signals and values.

Note: This command does not support connections with latency.

- Use `-src` to provide the source signal.
 - Use `-dest` to provide the destination signal.
 - Use `-silent` for `-verbosity 0` (see below).
 - Use `-verbosity N` to control the output of this command.
 - `-verbosity 0` results in no output.
 - `-verbosity 1` prints each part of the condition in one line
 - `-verbosity 2` and greater prints all the output of intermediary commands.
-

JasperGold Apps Command Reference Manual

General Commands

IP-XACT Related Switches

```
-load_ipxact [spirit:design_name [-instance instance_name] ]  
[-interfaces] [-infer_interface_definitions] [-no_debug]
```

Generate a Connectivity map based on connections found in *spirit:design_name* and load it into the Connectivity App. You must load the corresponding IP-XACT specifications through ipxact -load before running check_conn -load_ipxact. See “[Interface Connectivity Map](#)” on page 701.

The resulting map has interface definitions and interface connections.

- Use -instance when *spirit:design_name* is implemented by *instance_name*. Otherwise, the tool assumes that the top module implements *spirit:design_name*.
- Use -interfaces to generate a map without connections. The map contains only interface definitions and port maps.
- If you do not specify *spirit:design_name*, the tool generates connectivity information for all <spirit:component> elements loaded. Connections are only generated for <spirit:component> elements with corresponding <spirit:design> elements.
- Use -infer_interface_definitions to generate INTERFACE_MAP definitions for missing <spirit:abstractionDefinition> elements.
- Use -no_debug to load the map without any reference to the IP-XACT file names and line numbers.

Report Switches

```
-get_status connection_name
```

Return the proof status of the specified connection assertion.

If you do not specify a connection, the tool lists the status of all connections.

```
-show_trace connection_name
```

Run Visualize to get the violation trace of the specified connection assertion.

```
-show_schematic -connection connection_name
```

Run the JasperGold Apps Schematic Viewer and show the path between the source and destination signals.

Partition Switches

```
-save_partition_map xml_file_name.xml
```

Save partition information in the specified file in XML format.

The partition map XML file contains:

- Connections specification: complete information in CSV format
- Partition specification: name, connections, parent, suggested black-box list
- Connections status

```
-load_partition_map xml_file_name.xml
```

Load the specified XML partition map file that was saved previously by the tool.

Return

A list of requested objects or the output of a requested action.

Examples

```
% check_conn -connection conn1 -list condition  
% check_conn -list connection  
% check_conn -connection conn1 -list assert  
% check_conn -list assert  
% check_conn -load conn.xml  
% check_conn -connection conn1 -prove  
% check_conn -prove
```

See Also

[blackbox_assistant](#)
[ipxact](#)

check_cov

Description

Use the `check_cov` command to compute, manage, and report proof and stimuli coverage with the Coverage App.

Note: The Coverage App supports Verilog and SystemVerilog designs. VHDL support is initial release and mixed-language support is planned for a future release.

Syntax

```
check_cov -init
    [-model (statement | branch)+]
    [-type (all | stimuli | coi | proof | bound)+]
    [-enable_proof_core]
    [-enable_cover_item_traces]
    [-force_minimal_trace (on | off)]
    [-skip_ternary_branches]
    [-hierarchies {<instance_list>}]
    [-exclude_hierarchies {<instance_list>}]
    [-exclude_cover_expression_calculation]
    [-include_x_cover_items]
    [-include_empty_branch_cover_items]
    [-include_celldefine_cover_items]
    [-exclude_bind_hierarchies]

check_cov -clear

check_cov -save [file_name] [-database_id <id>] [-uncompressed]
    [-compute_coi] [-compute_proof_core]

check_cov -load [file_name] [-database_id <id> [-force]] [-uncompressed]

check_cov -merge -src_database_id <id_list> [-dest_database_id <id>]
    [-cover_items]
    [-targets]
    [-ta_prove_status]
    [-ci_prove_status [-negative]]
    [-coi]
    [-proof_core]
    [-bounded [-intersect_reports]]
    [-waivers]
    [-force]

check_cov -close [-database_id <id>]

check_cov -get_database_id
```

JasperGold Apps Command Reference Manual

General Commands

```
check_cov -setup -model ([statement] [branch])+
    [-hierarchies {<instance_list>}]
    [-exclude_hierarchies {<instance_list>}]
    [-exclude_cover_expression_calculation]
    [-include_x_cover_items]
    [-include_empty_branch_cover_items]

check_cov -prove [-no_prove]
    [-prove_opts <prove_options_tcl_string>]

check_cov -add_assert {<target_list>}
check_cov -remove_assert {<target_list>}
check_cov -add_user_cover {<cover_items>}
check_cov -remove_user_cover {<cover_items>}
check_cov -type task
    [-prove_options <prove_options_tcl_string>]
    [-autoprove] [-no_prove]
    [-include_hierarchies <included_hierarchies>]
    [-exclude_hierarchies <excluded_hierarchies>]
    [-task <task_list>]
    [ (-cover_item_id <cover_items_list>)
      | (-cover_item_name {<cover_items_list>})]

check_cov -report -type
    ( unreachable | undetermined | coi | outcoi
    | bounded | proof_core | out_proof_core | custom | all)
    [-database_id <id>]
    [-merge]
    [-gui]
    [-filter_in {expr_list} [-regexp]]
    [-filter_out {expr_list} [-regexp]]
    [-bound <override_bound>]
    [-exclude ([const] [assume] [reset] [waived] [unreachable])+]
    [-assert {<assert_name_list>}] [-detailed] ]
    [(-task <task_list>) | (-snapshot <snapshot_name>)]
    [-reference <reference_task_name>]
    [-limit N]
    [-no_return]
    [-status {<statuses_list>}]
    [-extended_format]
    [-query <query_string>]
    [-summary]
    [-report_file <file_name> [-xml] [-no_xml_stylesheet]]
    [-no_merge_tasks] [-source_merge] [-expose]
    [-model ([branch] [statement] [user]))+]

check_cov -report -type proof_core -list flop
    [-assert {assert_name_list} [-detailed] [-regexp]]
    [-task task_list]
```

JasperGold Apps Command Reference Manual

General Commands

```
check_cov -get_cover_item_info <cover_item_id> [-expression] [-database_id <id>]
check_cov -get_cover_item_info cover_item_id_list
  [-task tasks_list] [-database_id id]
  [-name] [-prove_status] [-bound] [-engine] [-task] [-time]
  [-instance] [-module] [-file] [-source_location] [-source_code] [-silent]
check_cov -get_target_info target_id_list [-database_id id]
  [-name] [-prove_status] [-bound] [-engine] [-task] [-time]
  [-coi] [-proof_core] [-silent]
check_cov -get_cover_id [-signal |-name]
  [ -source_signal {<source_signal_arg_list>} -direction (fanin | fanout)
  | {<target_id_list>}]
  [-database_id <id>]
check_cov -get_assert_id [-name] [-database_id <id>]
check_cov -snapshot ([-list] | ([-task <task_name>] [<snapshot_name>]) )
  [-database_id <id>] [-silent]
check_cov -waivers
  -import (-file_name <file_name>) [-force] [-silent] [-database_id <id>]
  |-export (-file_name <file_name>) [-force] [-silent] [-database_id <id>]
  |-add -comment <comment> [-database_id <id>]
    ([ -source_file <file_name>]
     [-start_line <start_line>] [-end_line <end_line>]
     [-start_column <start_column>] [-end_column <end_column>]
     [-instance <instance>]
     [-expression <expression_arg>]
     [-tag <tag>]
     [-type <type>]
     [-expression_only] | [-cover_item_id <cover_item_id_list>])
     [-force] [-silent])
  |-remove (-all |-id <waiver_id> |-tag <tag_name>) [-database_id <id>]
  |-edit -id <waiver_id> [-database_id <id>]
    [-source_file <file_name>]
    [-start_line <start_line>] [-end_line <end_line>]
    [-start_column <start_column>] [-end_column <end_column>]
    [-instance <instance>]
    [-comment <comment>]
    [-expression <expression>]
    [-tag <tag>]
    [-type <type>]
    [-force] [-silent]
```

JasperGold Apps Command Reference Manual

General Commands

```
| -get (-all |-id <waiver_id_list> |-tag <tag_list>) [-database_id <id>]  
| -list_matching_cover_items (-all |-id <waiver_id>) [-database_id <id>]
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
Setup Switches -init [-model (statement branch)+] [-type (all stimuli coi proof bound)+] [-enable_proof_core] [-enable_cover_item_traces] [-force_minimal_trace (off on)] [-skip_ternary_branches] [-hierarchies instance_list] [-exclude_hierarchies instance_list] [-exclude_cover_expression_calculation] [-include_x_cover_items] [-include_empty_branch_cover_items] [-include_celldefine_cover_items] [-exclude_bind_hierarchies]	<p>Initialize the elaboration flags to generate the required cover items in the netlist.</p> <ul style="list-style-type: none">■ Use the <code>-model</code> switch before elaboration to select the coverage model you want the tool to extract during elaboration.<ul style="list-style-type: none">□ Use <code>statement</code> to initialize elaboration for statement cover items.□ Use <code>branch</code> to initialize elaboration flags for branch cover items.■ Use <code>-type</code> to enable automatic Coverage setup based on the provided argument(s) as follows:<ul style="list-style-type: none">□ If the arguments list includes <code>stimuli</code> or <code>bound</code>, then automatically expose cover items after elaboration.□ If the arguments list includes <code>proof</code>, then automatically enable proof core calculation and disable proof simplification.□ If the list excludes <code>bound</code>, then allow engines to reuse traces with <code>-force_minimal_trace off</code>. <p>Note:</p> <ul style="list-style-type: none">□ Automatic setup creates a coverage model and adds all assertions to the coverage database.□ If you use the <code>-type</code> switch, Coverage setup automatically runs after elaboration. Therefore, you do not need to use <code>check_cov -setup</code> if you have included the <code>-type</code> switch at initialization.□ If you do not use the <code>-type</code> switch, automatic setup will not be run. Run <code>check_cov -setup</code> manually after elaboration.

-init (Continued)

- Use -enable_proof_core to collect proof core data from the engines.
- Note:**
- When you run `check_cov -init` with the proof core enabled, proof simplification is disabled.
 - For `-type {stimuli | coi | bound}`, the `-enable_proof_core` switch is ignored.
- Use `-enable_cover_item_traces` to enable cover item trace generation, which is disabled by default.
 - Use `-force_minimal_trace off` to allow engines to reuse traces.
 - Use `-force_minimal_trace on` to affect bounded proof due to not guaranteed minimal bounds. This is the default.
 - Use `-skip_ternary_branches` to exclude ternary cover items.
- Note:** Cover items reached in the reset phase can be excluded from reports using the `check_cov -report... -exclude reset` command. See "[Reporting Switches](#)" on page 269.
- Use `-hierarchies` to specify a list of instances you want to extract.
 - Use `-exclude_hierarchies` to specify a list of instances you want to exclude from extraction.
 - Use `-exclude_cover_expression_calculation` to prevent the tool from calculating the expression of the cover items. Expressions are calculated by default.
 - Use `-include_x_cover_items` to include cover items generated from default branches with X assignments or null statements in VHDL.
 - Use `-include_empty_branch_cover_items` to include cover items generated from missing else branches and empty default case statements.
 - Use `-include_celldefine_cover_items` to include cover items generated from `celldefine blocks.
 - Use `-exclude_bind_hierarchies` to exclude cover items generated from bind statements.

JasperGold Apps Command Reference Manual

General Commands

-clear

Clear the coverage database and the `check_cov -init` settings.

Note: You must run `check_cov -init` again if coverage data needs to be extracted again.

-save [*file_name*] [-database_id *id*] [-uncompressed]
[-compute_coi] [-compute_proof_core]

Save the coverage database to a file.

- Use *file_name* to specify the file name. If you do not specify a file name, the `-save` command refers to `CoverageTmp.jcdb` by default.
- Use `-database_id` to specify the database to save. If you do not specify an ID, this command saves the default database (ID = 0).
- Use `-uncompressed` to save the database to a file without compression.
- Use `-compute_coi` to calculate the COI before saving.
- Use `-compute_proof_core` to calculate the proof core before saving.

-load [*file_name*] [-database_id *id* [-force]] [-uncompressed]

Load the coverage database from a saved session.

- Use *file_name* to specify the file name. If you do not specify a file name, the `-load` command refers to `CoverageTmp.jcdb` by default.
- Use `-database_id` to specify a database. If the destination database already contains content, the tool executes an auto-merge. Use `-force` to merge databases from incompatible designs. See `-merge` below for additional information.
- Use `-uncompressed` to load an uncompressed database file.

Note: You cannot load a compressed file if you have used the `-uncompressed` switch.

Note: You must run `analyze` and `elaborate` again before loading the database.

JasperGold Apps Command Reference Manual

General Commands

```
-merge -src_database_id id_list [-dest_database_id id]
[-cover_items] [-targets]
[-ta_prove_status [-negative]] [-ci_prove_status]
[-coi] [-proof_core]
[-bounded [-intersect_reports]]
[-waivers] [-force]
```

Merge the source database(s) with the specified destination database according to one or more specified switches.

Note:

- You must provide at least one source database. If you do not specify a destination database, the tool creates a new database to hold the merged list.
- If you do specify a destination database, the tool merges existing content with the source database content. It does not overwrite content.
- The `-merge` command returns the ID of the merged destination.
- If you do not specify otherwise, the tool merges the proof status by default.
- Use `-cover_items` to merge cover items only.
- Use `-targets` to merge targets only.
- Use `-ta_prove_status` to merge targets with their proof statuses, and use `-negative` to merge toward CEX instead of proven. By default, the tool merges targets toward the positive.
- Use `-ci_prove_status` to merge cover items with their proof statuses.
- Use `-coi` to merge the COIs.
- Use `-proof_core` to merge the proof cores.

Note: Merging the COI or proof core also merges the cover items and targets.

JasperGold Apps Command Reference Manual

General Commands

-merge (Continued)

- Use `-bounded` to merge both the cover items and targets with their proof statuses, and use `-intersect_reports` to calculate the bounded report for each database separately, intersect the results, and save the final cache at the destination database.

Note: The bounded report considers the cache if it is available.

- Use `-waivers` to merge waivers only.
- Use `-force` to merge databases from incompatible designs.

Note:

- Using the `-force` switch for merging may yield erroneous results.
- When the designs are compatible, this switch has no effect.

-close [-database_id *id*]

Close and remove coverage databases.

Use `-database_id` to specify the database to close. If you do not specify the ID, this command closes all databases except the default database (ID = 0).

-get_database_id

Return a list of IDs for all open databases.

JasperGold Apps Command Reference Manual

General Commands

```
-setup -model ([statement] [branch])+
[-hierarchies instance_list]
[-exclude_hierarchies instance_list]
[-exclude_cover_expression_calculation]
[-include_x_cover_items]
[-include_empty_branch_cover_items]
```

Extract cover items and store them in the coverage database. Use this command after elaboration.

- Use `-hierarchies` to specify a list of instances you want to extract.
- Use `-exclude_hierarchies` to specify a list of instances you want to exclude from extraction.
- Use `-exclude_cover_expression_calculation` to prevent the tool from calculating the expression of the cover items. Expressions are calculated by default.
- Use `-include_x_cover_items` to include cover items generated from default branches with X assignments or null statements in VHDL.
- Use `-include_empty_branch_cover_items` to include cover items generated from missing else branches and empty default case statements.

```
-prove [-no_auto] [-prove_opts prove_options_tcl_string]
```

Triggers `prove` or `autoprove` for Coverage properties, with a specified list of Tcl options.

- Use `-no_auto` to trigger the `prove` command.
- Use `-prove_opts` to specify `prove` or `autoprove` command options. See “[prove](#)” on page 741 and “[autoprove](#)” on page 207.

```
-add_assert target_list
```

Add the specified assertions (that is, targets) to the coverage database.

```
-remove_assert target_list
```

Remove the specified assertions (that is, targets) from the coverage database.

Note: Use this command after elaboration.

```
-add_user_cover cover_items
```

Add the specified cover items to the coverage database.

```
-remove_user_cover cover_items
```

Remove the specified cover items from the coverage database.

JasperGold Apps Command Reference Manual

General Commands

```
-type task
[-prove_options prove_options_tcl_string]
[-no_prove] [-autoprove]
[-include_hierarchies included_hierarchies]
[-exclude_hierarchies excluded_hierarchies]
[-task task_list
[ (-cover_items_id cover_items_list)
| (-cover_items_name cover_items_list)]
```

Convert cover items to cover properties.

- Type `task` is logged normally as the proof environment that is being checked.
 - Use `-prove_options` with a list of applicable options.
 - Refer to the `prove` command for possible options.
 - You must use valid `prove` options based on the type of the proof you are using.
 - Use `-no_prove` if you do not want the tool to issue the `prove` command immediately. When you use this switch, the tool exposes the cover items as cover properties and monitors their proof, which you can run at a later stage.
 - Use `-autoprove` if you want the tool to use autoprove for the proof. By default, the tool uses `prove`.
 - Use `-include_hierarchies` if you want the tool to generate cover properties for the specified hierarchies only.
 - Use `-exclude_hierarchies` if you do not want the tool to generate cover properties for the specified hierarchies.
 - Use `-task` to specify the tasks for which you want to expose cover items.
 - Use `-cover_items_id` to specify by ID the cover items you want to expose for the current or specified task.
 - Use `-cover_item_name` to specify by name the cover items you want to expose for the current or specified task.
-

Reporting Switches

```
-report -type
  ( unreachable | undetermined | coi | outcoi
  | bounded | proof_core | out_proof_core | custom | all)
[-database_id id]
[-merge] [-gui] [-filter_in expr_list [-regexp]]
[-filter_out expr_list [-regexp]]
[-bound override_bound]
[-exclude ([const] [assume] [reset] [waived] [unreachable])+]
[-assert {assert_name_list} [-detailed]]
[-task task_list |-snapshot snapshot_name]
[-reference reference_task_name]
{-limit N}
[-no_return]
[-status {status_list}]
[-extended_format]
[-query query_string]
[-summary]
[-report_file file_name [-xml] [-no_xml_stylesheet]]
[-no_merge_tasks] [-source_merge] [-expose]
[-model ([branch] [statement] [user])+]
```

Report cover items' unreachability status for your environment.

The tool reports bounded proof coverage for the bounded proof targets and COI proof coverage for the targets. Refer to the following `-report` sub-sections for additional definitions and to see supported switch combinations.

Stimuli Coverage Report

```
-report -type (unreachable | undetermined)
[-exclude ([const] [assume] [reset] [waived])+]
[-reference reference_task_name]
[-task task_list]
```

Report stimuli coverage.

Use `-type` and specify one of the following arguments:

- `unreachable` – Report unreachable cover items in the current or specified task.
- `undetermined` – Report undetermined cover items in the current or specified task.

Use the following switches with `unreachable` or `undetermined` to further refine the stimuli coverage report:

- Use `-exclude` with one or more of the following arguments to omit cover items that are not reachable due to propagation or initial simplification:
 - `const`: constant propagation
 - `assume`: assumption propagation
 - `reset`: reset propagation
- And use `-exclude waived` to exclude user-specified waivers.
- Use `-reference` to filter out unreachable cover items that are also unreachable in the reference task.

Bounded Coverage Report

```
-report -type bounded  
[-assert {assert_name_list} [-detailed] [-regexp]]  
[-filter_in expr_list [-regexp]] [-filter_out expr_list [-regexp]]  
[-bound override_bound]
```

Report bounded proof coverage.

This report includes the following information:

- Assertion statuses and a summary of the COI cover items for each assertion.
- The number of cover items covered beyond the assertions' bound and the max bound.
- The number of cover items undetermined (and not dead) and the max bound.

Use `-assert...` `-detailed` to report only the cover items in the COI of the specified assertions and only those with a bound that is higher than the assertion's proof bound or undetermined.

Note: `-assert...` `-detailed` supports regular expressions with the `-regexp` switch.

Use the following switches with `-type bounded` to further refine the bounded coverage report:

- Use `-filter_in expr_list` to include only cover items from the matching instance or module, and use `-regexp` to specify that the expression is a regular expression.
- Use `-filter_out expr_list` to exclude cover items from the matching instance or module, and use `-regexp` to specify that the expression is a regular expression.
- Use `-bound override_bound` to override the current bound with the specified bound.

Property Completeness Report

```
-report -type (coi | outcoi)
[-assert {assert_name_list} [-regexp]]
[-filter_in expr_list [-regexp]] [-filter_out expr_list [-regexp]]
[-task task_list]
```

Report property completeness.

Use `-type` and specify one of the following arguments:

- `coi` – Report cover items that are in the COI of assertions in the current or specified task.
- `outcoi` – Report cover items that are not in the COI of any assertions in the current or specified task.

Use the following switches with `coi` or `outcoi` to further refine the property completeness report:

- Use `-assert` to report only the cover items in the COI of the specified assertions.
- Use `-filter_in expr_list` to include only cover items from the matching instance or module, and use `-regexp` to specify that the expression is a regular expression.
- Use `-filter_out expr_list` to exclude cover items from the matching instance or module, and use `-regexp` to specify that the expression is a regular expression.

Proof Coverage Report

```
-report -type (proof_core | out_proof_core)
[-assert {assert_name_list} [-regexp]]
[-filter_in expr_list [-regexp]] [-filter_out exp_list [-regexp]]
[-limit N] [-no_return]
[-status {status_list}]
```

Report proof coverage.

Use `-type` and specify one of the following arguments:

- `proof_core` – Report cover items that are in the proof core of assertions that were added as targets. The proof core is the region in the design that was needed for the proof. This region is usually smaller than the COI.
- `out_proof_core` – Report cover items that are not in the proof core of any assertions that were added as targets.

Note: The coverage feature can collect proof core information only from the following engines: AB, AD, AG, AM, C, I, K, and N.

Use the following switches with `proof_core` or `out_proof_core` to further refine the proof coverage report:

- Use `-assert` to include only cover items in the proof coverage of the specified assertions.
- Use `-filter_in expr_list` to include only cover items from the matching instance or module, and use `-regexp` to specify that the expression is a regular expression.
- Use `-filter_out exp_list` to exclude cover items from the matching instance or module, and use `-regexp` to specify that the expression is a regular expression.
- Use `-limit N` to limit reporting to the first `N` items.
- Use `-no_return` if you do not want the tool to return a value for the Tcl interpreter.

Continued below.

Proof Coverage Report (Continued)

- Use `-status status_list` to report only those cover items of the targets with a status included in the statuses list. Possible statuses include `proven`, `proven_vacuous`, `proven_nonvacuous`, `marked_proven`, `undetermined`, `cex`, and `processing`.

Note:

- When you issue a proof-core report for multiple targets without specifying the status, the default is {`proven undetermined`}.
- The `proven_vacuous` status option reports targets that have unreachable related cover items.
- `proven` includes both vacuous and non-vacuous proofs.

Custom Coverage Report

```
-report -type custom -query query_string [-no_merge_tasks]
```

Generate the specified custom coverage report.

Use `-type custom` with `-query` and specify a string query. The tool applies the query to each cover item. If it evaluates to true for a specific cover item, then that cover item is included in the report.

Use `-no_merge_tasks` to specify that report results will not be shown as merged tasks and each cover item's proof status will be within the current task name.

The syntax for the query is a general expression with fields, functions, matching/comparison operators and logical operators in between, with free use of parenthesis. The format is case insensitive.

The query syntax is briefly described below. For additional details, see [“Help and Guidelines for Query Syntax”](#) on page 286.

■ Functions and arguments

`Status(arg)`, `Bound(arg)`, `Time(arg)`: `arg` is a task name or a task snapshot name.

`IN_COI_ANY(arg)`, `OUT_COI(arg)`, `IN_PROOF_CORE_ANY(arg)`, `OUT_PROOF_CORE(arg)`: `arg` is a target ID or target name including the task name.

`IN_COI_ALL(arg)`, `IN_COI_ANY(arg)`,
`IN_PROOF_CORE_ALL(arg)`, `IN_PROOF_CORE_ANY(arg)`: `arg` is a regular expression or a target list separated by commas.

■ Fields

`Status`, `Type`, `Name`, `Engine`, `Bound`, `Time`, `Instance`, `Cover Item ID`, `Module`, `File`, `Source Location`, `Waived`, `count`, `Caused By`

Note: These are the fields from the cover items table.

General Report Refinement Switches

```
-report -type
  ( unreachable | undetermined | coi | outcoi
  | bounded | proof_core | out_proof_core | custom | all)
  [-database_id id] [-merge] [-gui]
  [-filter_in expr_list [-regexp]]
  [-exclude (waived | unreachable)+]
  [-extended_format] [-summary]
  [-report_file file_name [-xml] [-no_xml_stylesheet]]
  [-merge] [-no_merge_tasks] [-source_merge] [-expose]
  [-model ([branch] [statement] [user])]+
```

Refine any of the coverage reports with the following switches:

- Use `-database_id` to specify a database other than the default database.
- Use `-merge` to report the coverage based on module+parameters and not on instance.
- Use `-gui` to sync the GUI to the session command. The tool applies all session command options to the report dialogs.

Note:

- This switch does not support the `-limit` switch.
- This switch does not support multiple tasks.
- Use `-filter_in` to include only cover items from the matching instance or module.
- Use `-exclude waived` to exclude user-specified waivers, which are not excluded from Coverage reports by default, and use `-exclude unreachable` to exclude unreachable cover items.
- The standard coverage report includes the cover item name and ID. Use `-extended_format` to expand the report to include the following additional information: module name, instance name, file name, line number, source location, and source code excerpt.

JasperGold Apps Command Reference Manual

General Commands

-report (Continued)

- Use `-summary` to print a report with the following information per instance:

- Number of cover items in the report
 - Total number of cover items
 - Number of excluded cover items

Note:

- This switch is applicable to the reachable, COI, and proof_core reports only. If you use the `-summary` switch for any other report type, the tool returns an error.
 - The `-summary` switch is applicable to the complete design only, that is, you cannot use switches like `-assert`, `-filter_in`, and `-filter_out` with `-summary`.

- Use `-report_file` to export the coverage report to the specified file.

- Use `-xml` to export the report to an XML style sheet.

Note: You cannot use `-xml` without using `-report_file file_name`.

- Use `-no_xml_stylesheet` to export the coverage report to an XML file.

Note: You cannot use `-no_xml_stylesheet` without using `-report_file file_name` or `-xml`.

- Use `-no_merge_tasks` to specify that report results will not be shown as merged tasks and each cover item's proof status will be within the current task name.

- Use `-source_merge` to merge all cover items with the same source location into one representative cover item, which will be the first reachable cover item. If there are no reachable cover items among the merged cover items, then the representative cover item can be any cover item from the set.

- Use `-expose` to expose all the cover items in the report result.

- Use `-model` to report only cover items from the specified model or models. By default, reports include all models.
-

Proof-Core Flops Report

```
-report -type proof_core -list flop  
[-assert {assert_name_list} [-regexp]]  
[-task task_list]
```

Report the flops in the proof core.

- Use `-assert` to list the flops in the specified targets.
- Use `-task` to list the flops in the specified tasks.

Utility Switches

```
-get_cover_item_info cover_item_id_list [-expression] [-database_id id]
```

Return the SVA expression for the cover items in the specified Tcl list.

Note: The `-expression` switch is mutually exclusive and cannot be used in conjunction with other `-get_cover_item_info` switches.

JasperGold Apps Command Reference Manual

General Commands

```
-get_cover_item_info cover_item_id_list [-task tasks_list] [-database_id id]
[-name] [-prove_status] [-exclusions] [-bound] [-engine] [-time]
[-instance] [-module] [-file] [-source_location] [-source_code] [-silent]
```

Return a summary and bracketed list of information about the specified cover item(s).

- Use `-database_id` to specify a database other than the default database.
- Use `-task` to return information about cover items in the specified tasks only. By default, the tool returns information for all tasks.
- Use one or more additional switches to return information for the specified fields only. By default, the tool returns information for all fields except proof time and source code.
- Use `-silent` to avoid printing the summary.

The return includes the following information:

```
index 0: netlist signal name
index 1: instance name
index 2: module name
index 3: file name
index 4: source location
index 5: task name
index 6: proof status
index 7: bound
index 8: engine
index 9: exclusions
```

JasperGold Apps Command Reference Manual

General Commands

```
-get_target_info target_id_list [-database_id id]
[-name] [-prove_status] [-bound] [-engine] [-task] [-time] [-silent]
```

Return information about targets in the specified Tcl list.

- Use `-database_id` to specify a database other than the default database.
- Use `-name`, `-prove_status`, `-bound`, `-engine`, `-task`, and/or `time` to return information on only the specified options. By default, `-get_target_info` returns information on all options.
- Use `-silent` to return Tcl return values without printing to the screen

The return includes the following information:

```
index 0: target full name
index 1: proof status
index 2: bound
index 3: engine
index 4: task
index 5: time
```

JasperGold Apps Command Reference Manual

General Commands

```
-get_cover_id [-name | -signal]
[ -source_signal {source_signal_arg_list} [-direction (fanin | fanout)+]
 | {target_id_list}]
[-database_id id]
```

Return a list of the cover item IDs.

- Use `-name` to return the full cover item name.

The specifics of the naming convention follow:

`_automatic_coveritem_<type>_condition_<subtype>_<cover_item_index>_<type>_<line>_<subtype_num>_<type_index>`

- `type` - branch or stmt
- `subtype` - Type of branch or statement cover item
- `cover item index` - Index of all cover items generated
- `type` - S for statement and B for branch
- `line number` - The location of the cover item
- `subtype number` - For conditions, 0 for true path and 1 for corresponding false path

Example: `if_expr_then is 0 while if_expr_else is 1`

Note: This information is irrelevant in the case of statements.

- `type index` - The index of cover items with the same subtypes, line numbers, and subtype number

Example: `mux_out = enable ? enable2 ? din_1: din2: din_0`; has two `if_expr_then` on the same line; therefore, their type indexes are 0 and 1.

- Use `-signal` to return the cover item names without the `COVER_ITEM_ID` suffix.
- Use `-source_signal` with `-direction` to return a list of cover item IDs in the fanin or fanout of the specified signal list
- Specify a `target_id_list` to return the cover items in the COI of the specified target ID list.
- Use `-database_id` to specify a database other than the default database.

JasperGold Apps Command Reference Manual

General Commands

```
-get_assert_id [-name] [-database_id id]
```

Return the target IDs in the database.

- Use `-name` to return the names of the targets.
- Use `-database_id` to specify a database other than the default database.

```
-snapshot (-list | ([-task task_name] [snapshot_name]) ) [-database_id id]  
[-silent]
```

Save a snapshot of the reachability status of cover items in the specified task.

- Use `-list` to query the list of saved snapshots.
- Use `-task` to specify a task other than the current task. If you do not also specify a name, the tool automatically generates one.
- Use `-silent` to return Tcl return values without printing to the screen. This option is helpful in Tcl scripts.
- Use `-database_id` to specify a database other than the default database.

Note: The snapshot is available for reporting with the `-report` command and in the GUI. In the GUI, the snapshot name appears with a `snapshot:` prefix in the pull-down combo box used to select the source of the report.

JasperGold Apps Command Reference Manual

General Commands

```
-waivers
  -import -file_name file_name [-force] [-silent]
            [-database_id id]
  |-export -file_name file_name [-force] [-silent]
            [-database_id id]
  |-add -comment comment [-database_id id]
    ([ -source_file file_name]
     [-start_line start_line] [-end_line end_line]
     [-start_column start_column] [-end_column end_column]
     [-instance instance]
     [-expression expression_arg]
     [-tag tag]
     [-type type]
     [-expression_only] | [-cover_item_id <id_list>])
    [-force] [-silent]
  |-remove (-all |-id waiver_id |-tag tag_name)
            [-database_id id]
  |-edit -id waiver_id [-database_id id]
    [-source_file file_name]
    [-start_line start_line] [-end_line end_line]
    [-start_column start_column] [-end_column end_column]
    [-instance instance]
    [-comment comment]
    [-expression expression]
    [-tag tag]
    [-type type]
    [-force] [-silent]
  |-get (-all |-id waiver_id_list |-tag tag_list)
            [-database_id id]
  |-list_matching_cover_items (-all |-id waiver_id)
            [-database_id id]
```

Handle waived properties as specified.

- Use `-import` to import waivers from the specified file or report to the Coverage App. This command marks each property included in the same location as waived. By default, if one of the imported waivers has data that has not been updated, the import stops with an error message.
 - Use `-force` to ignore waivers that have not been updated and waivers that have no data without stopping the import.

JasperGold Apps Command Reference Manual

General Commands

-waivers (Continued)

- Use `-silent` to return Tcl return values without printing to the screen.
 - Use `-database_id` to specify a database other than the default database.
 - Use `-export` to export waivers to the specified file.
 - Use `-force` to ignore waivers that have not been updated and waivers that have no data without stopping the export.
 - Use `-silent` to return Tcl return values without printing to the screen.
 - Use `-database_id` to specify a database other than the default database.
 - Use `-add` to add a waiver. This command returns a new waiver ID.
Note: Added waivers are not excluded from Coverage reports by default. Use `check_cov -report -exclude waived` to exclude user-specified waivers.
 - Use `-comment` to specify a comment for the new waiver.
Note: This field is mandatory.
 - Use `-database_id` to specify a database other than the default database.
 - Use `-source_file` to specify the file that contains the waiver.
 - Use `-start_line` to specify the start line of the waiver in the given file.
 - Use `-end_line` to specify the end line of the waiver in the given file.
 - Use `-start_column` to specify the start column of the waiver in the given file.
 - Use `-end_column` to specify the end column of the waiver in the given file.
 - Use `-instance` to specify the instance of the waiver in the given file.

JasperGold Apps Command Reference Manual

General Commands

-waivers (Continued)

- ❑ Use `-expression` to specify an expression or list of expressions that describe the waivers.
- ❑ Use `-tag` to specify a tag for the waiver.
- ❑ Use `-type` to specify the type of the waiver.
- ❑ Use `-expression_only` to add a waiver that matches by expression only.
- ❑ Use `-cover_item_id` to add waivers by cover item ID.
- ❑ Use `-force` to add the waiver even when it does not match any cover item.
- Use `-remove` to delete waivers.
 - ❑ Use `-all` to delete all waivers.
 - ❑ Use `-id` to delete the waiver with the specified ID.
 - ❑ Use `-tag` to delete waivers with the specified tag.
 - ❑ Use `-database_id` to specify a database other than the default database.
- Use `-edit -id` to edit the details of the waiver with the specified ID, and use `-database_id` to specify a database other than the default database.

Note:

- ❑ Use any of the switches listed under `-add` above to edit a field in an existing waiver.
- ❑ Use `-force` to add the waiver if the waiver matched cover items before the edit but failed to match any items after the edit.
- Use `-get` to get information about one or more waivers.
 - ❑ Use `-all` to get information about all waivers.
 - ❑ Use `-id` to get information about one or more waivers with the specified ID(s).

JasperGold Apps Command Reference Manual

General Commands

-waivers (Continued)

- ❑ Use -tag to get information about waivers with one or more of the specified tag(s).

Note: -get -id and -get -tag support Tcl lists.

- ❑ Use -database_id to specify a database other than the default database.

- Use -list_matching_cover_items to get information about one or more waivers.

- ❑ Use -all to get a list of the matching cover items for all the waivers.

- ❑ Use -id to get a list of the matching cover items for the specified waiver ID(s).

- ❑ Use -id to get information about one or more waivers with the specified ID(s).

- ❑ Use -database_id to specify a database other than the default database.

Return

A list of requested objects or the output of a requested action.

Help and Guidelines for Query Syntax*

The query language purpose is to simplify the process of retrieving information about cover items in the design. The language consists of expressions, which consist of fields, operators, logical operator, and functions. A field can be followed by an operator only, then an operator is followed by a value or a list of values.

Examples:

- Status = Covered
- Engine in { Ht, Hp }

The latter query expression will report all the cover items which have been proven by Hp or Ht engines, and it is equivalent to :Engine = Ht or Engine = Hp.

The other option for a query on a field is with a function return value, for example,

JasperGold Apps Command Reference Manual

General Commands

```
Waived = NOT IN_COI_ANY( _target_name / _target_id )
```

This query will yield all the cover items that hold as follows: (is Waived and is not in COI of tID) or (is not Waived and is in COI of tID)

A query may be composed by more than one expression combined together by comparison and logical operators.

Use `IN_COI_ALL` or `IN_PROOF_CORE_ALL` if you want to get a report with all the cover items that are in the COI or proof core of ALL the specified targets, and use `IN_COI_ANY` or `IN_PROOF_CORE_ANY` if you want to get a report with all the cover items that are in the COI or proof core of ANY of the specified targets.

Note:

- (“ and “) can be added wherever needed, as long they do not show up in value characters.
- (`NOT`) is the only available unary logical operator supported by the query.
- The grammar description below allows any field to be compared to any function return value, but the query language accepts only some of these comparisons. Fields' values may be compared only with functions that return the same type of values as these fields.
- The (~) operator means like (that is, `str1 ~ str2` if `str2` is a substring of `str1`).

A list of supported functions follows:

- `OUT_COI(target name / target ID)` – Return type: Boolean
- `OUT_PROOF_CORE(target name / target ID)` – Return type: Boolean
- `STATUS(task name)` – Return type: cover item status
- `BOUND(task name)` – Return type: Bound type (integer)
- `TIME(task name)` – Return type: Time type (string)
- `IN_COI_ALL(target name / target ID, ..., target name / target ID)` or `IN_COI_ALL(regular expression)` – Return type: Boolean.
- `IN_COI_ANY(target name / target ID, ..., target name / target ID)` or `IN_COI_ANY(regular expression)` – Return type: Boolean.
- `IN_PROOF_CORE_ALL(target name / target ID, ..., target name / target ID)` or `IN_PROOF_CORE_ALL(regular expression)` – Return type: Boolean.

JasperGold Apps Command Reference Manual

General Commands

- IN_PROOF_CORE_ANY(target name / target ID, ..., target name / target ID) or IN_PROOF_CORE_ANY(regular expression) – **Return type:** Boolean.
- BOUND_TARGET(target name / target ID) – **Return type:** Bound type (Integer)

[Table 4-2](#) on page 288 shows a list of supported functions for each field as a value (RHS):

Table 4-2 Supported Functions

[field]	[op]	[function]
status	[op]	STATUS
name	[op]	STATUS
waived	[op]	IN_COI_ANY, OUT_COI, IN_PROOF_CORE_ANY, OUT_PROOF_CORE
bound	[op]	BOUND, BOUND_TARGET
time	[op]	TIME

Note: Each field that does not appear above may not be in the prior pattern: [field] [op] [function].

A list of values that the STATUS field and function can accept follows:

- Unprocessed;
- Proven;
- Marked Proven;
- Covered;
- Covered in AR;
- CEX;
- Unreachable;
- CEX in AR;
- Undetermined;
- Processing;
- Error;

JasperGold Apps Command Reference Manual

General Commands

- Approved;
- Temporary;
- No Connection;

Note: A query may include any value, even if it is not defined above, but if the value is illegal, the evaluation will fail.

Custom Query Syntax as EBNF Grammar

```
<root_exp> ::= <expression> | "(" <expression> ")"
<expression> ::= <field_comparison> <exp> | <function_comparison> <exp> |
<function> <exp> | <unary_op> <root_exp>
<exp> ::= <logical_op> <root_exp> | <>>
<field_comparison> ::= <field> <binary_op> { <value> | <all_functions> } | <field>
<one_to_many_op> "{" <values_list> "}"
<all_functions> ::= { <function> | <function2> }
<function_comparison> ::= <function2> <binary_op> { <value> | <function> }
<function> ::= <function_name> "(" <values_list> ")"
<function2> ::= <function_name2> "(" <values_list> ")"
<function_name> ::= "IN_COI_ALL" | "IN_COI_ANY" | "OUT_COI" | "IN_PROOF_CORE_ALL"
| "IN_PROOF_CORE_ANY" | "OUT_PROOF_CORE"
<function_name2> ::= "STATUS" | "BOUND" | "TIME" | "BOUND_TARGET"
<field> ::= "Status" | "Type" | "Name" | "Cover Item ID" | "Module" | "File"
| "Source Location" | "Waived" | "Engine" | "Bound" | "Time" | "Instance" | "Count"
| "Caused By"
<logical_op> ::= "AND" | "OR"
<binary_op> ::= "=" | "!=" | ">" | ">=" | "<" | "<=" | "~" | "!" | "RegExpMatch"
<one_to_many_op> ::= "IN"
<unary_op> ::= "NOT"
<value> ::= { all characters - { ',', ' ', '\t', ')' , ')' } }
<values_list> ::= <value> | <value> "," <values_list>
```

Examples:

```
Status = unreachable AND IN_COI_ANY(target1) Status(taskA) = Covered OR
Status(taskB) = Unreachable

Status IN {Covered, Unreachable, Undetermined}

Status = Covered AND (IN_COI_ANY(targetA) OR IN_COI_ANY(TARGETB))
(Status(task1) = covered OR Bound = 2) AND Status = Covered
```

Note: *

- The above grammar shows the syntax of the query language supported in this type of report.

JasperGold Apps Command Reference Manual

General Commands

- Some general cases stated in the grammar are supported for future use, but their semantics are not yet defined. Using such cases produces 0 items in the report.

See Also

No related commands

check_csr

Description

Use `check_csr` to get information from the XML workbook file that defines the CSR registers format or to request actions related to this workbook. This command supports the JasperGold Apps CSR format and IEEE® 1685™ IP-XACT.

You can instantiate multiple CSR maps in each CSR verification execution flow in an analysis session. Each workbook is associated with one instance. For each instance, the tool generates properties for each field defined in the workbook, according to the field access type.

This command has three scopes, instance/workbook, register, and field.

Note: You must elaborate the design before using this command.

Syntax

```
check_csr -list instance

check_csr -list (register | addr_offset | addr_width | data_width)
    (-instance <instance_name>)

check_csr -list ( data_width | addr_offset | addr | access_type
    | reset_value | field)
    (-instance <instance_name>)
    (-register <register_name>)

check_csr -list (data_width | msb | lsb | access_type | reset_value)
    (-instance <instance_name>)
    (-register <register_name>)
    (-field <field>)

check_csr -merge -ipxact <ipxact_file_name>
    -csv <tsv_file_name> -ipxact_out <ipxact_out_file_name>

check_csr -generate_checker_instance -ipxact <ipxact_file_name>
    -bind_file <sv_bind_file_name>
    [-bind_to_instance <instance_name>]

check_csr -load (<xml_file_name> | <csv_file_name>)
    [-use_long_field_names]
    -instance <instance_name>
    [-auto_hr_info]
    [-unified_check]
    [-partitioned_checks]
    [-sequence_checks -reset_check (reset_only | non_reset_only)]
    [-init_gap (full | none)]
```

JasperGold Apps Command Reference Manual

General Commands

```
[-middle_gap (full | none)]
[-alias_covers]
[-addr_interaction_covers]
[-unassigned_field_checks (constant | zero)]  
check_CSR -clear [-instance <instance_name>]  
check_CSR -prove [-instance <instance_name>
                  [-register <register_name>
                   [-field <field_name>]
                  ]
                 ]
[-bg]  
check_CSR -generate_template <file> [-force]
[-csv | -xml]  
check_CSR -load_ipxact <ipxact_file_name> -generate_csv <csv_file_name>
[-maps <map_list>]
[-force]
[-component <spirit:component_name>]
[-no_debug]  
check_CSR -load_ipxact [<ipxact_file_name>] -instance <instance_name>
[-maps <map_list>]
[-unified_check]
[-partitioned_checks]
[-addr_interaction_covers]
[-unassigned_field_checks (constant | zero)]
[-component <spirit:component_name>]
[-no_debug]
```

Argument	Definition
----------	------------

List Switches



- Use `-instance` to get a list related to the specified instance.
- Use `-register` to get a list related to the specified register for the specified instance.
- Use `-field` to list only information defined for the specified field for the specified register for the specified instance.

`-list instance`

List the defined instance name.

JasperGold Apps Command Reference Manual

General Commands

```
-list (register | addr_offset | addr_width | data_width)
      -instance instance_name
```

List the information that matches the specified criteria.

The following switches are valid for the instance/workbook scope. They do not support the `-register` or `-field` scope.

- Use `register` to list all CSR names defined in the workbook.
- Use `addr_offset` to list the address offset defined for all CSR definitions.
- Use `addr_width` to list the address width defined for all CSR definitions.
- Use `data_width` to list the default data width defined for all CSR definitions. You can redefine the data width value and thus overwrite it in each CSR definition.

```
-list (data_width | addr_offset | addr | access_type | reset_value | field)
      -instance instance_name
      -register register_name
```

List the information that matches the specified criteria.

The following switches are valid for the `-register` scope.

- Use `data_width` to list the `data_width` for the specified register or field.
- Use `addr_offset` to list the address offset defined for the specified register.
- Use `addr` to list the address to check for each register.
- Use `access_type` to list the access type for the specified register or field.
- Use `reset_value` to list the reset value for the specified register.
- Use `field` to list all fields defined for the specified register.

JasperGold Apps Command Reference Manual

General Commands

```
-list (data_width | msb | lsb | access_type | reset_value)
      -instance instance_name
      -register register_name
      -field field_name
```

List the information that matches the specified criteria.

The following switches are valid for the `-field` scope.

- Use `data_width` to list the `data_width` for the specified register or field.
- Use `msb` to list the most significant bit value of each field.
- Use `lsb` to list the least significant bit value of each field.
- Use `access_type` to list the access type for the specified register or field.
- Use `reset_value` to list the reset value of each field.

Action Switches

```
-merge -ipxact ipxact_file_name
      -csv tsv_file_name -ipxact_out ipxact_out_file_name
```

Merge the IP-XACT input file with the CSV file in a fashion similar to that of the corresponding `merge` command of the IEV REG App.

```
-generate_checker_instance -ipxact ipxact_file_name
      -bind_file sv_bind_file_name
      [-bind_to_instance instance_name]
```

Generate a SystemVerilog CSR checker instantiation bind file.

You must analyze this bind file in the setup.

- Use `-ipxact` to specify the merged IP-XACT file.
- Use `-bind_to_instance` to specify the CSR checker instance name.

Note:

- This instance name is the one to be specified in the `check_csr -load` and `check_csr -load_ipxact` commands.
- If you do not specify this switch, the tool generates an instance with the name `inst1`.

JasperGold Apps Command Reference Manual

General Commands

```
-load (xml_file_name | csv_file_name)
[-use_long_field_names]
-instance instance_name
[-auto_hr_info]
[-unified_check]
[-partitioned_checks]
[-sequence_checks -reset_check (reset_only | non_reset_only)
 [-init_gap (full | none)] [-middle_gap (full | none)] [-alias_covers]]
[-addr_interaction_covers]
[-unassigned_field_checks (constant | zero)]
```

Load the properties generated by the specified workbook file in XML or CSV format.

- Use `-use_long_field_names` to postfix property names with `_msb_lsb`.
Note: This switch is particularly useful if you provide the same name for multiple fields, such as `reserved`.
- Use `-instance` to bind atomic-level properties to the glue logic of the CSR bus interface.
- Use `-auto_hr_info` to specify that you want the tool to elaborate with the extended HR handler.
- Use `-unified_check` to generate the single assertion `AST_DATA_INTEGRITY`, the unified data integrity check, which comprises the reset value check and read after write checks.
- Use `-partitioned_checks` to generate the partitioned checks:
 - `AST_RESET_VALUE`
 - `AST_BUS_READ_AFTER_BUS_WRITE`
 - `AST_BUS_READ_AFTER_INTERNAL_WRITE`
 - `AST_BUS_WRITE_BUS_READ_IMMEDIATE`
 - `AST_INTERNAL_WRITE_BUS_READ_IMMEDIATE`

JasperGold Apps Command Reference Manual

General Commands

-load (Continued)

Note:

- ❑ If you do not specify -unified_check or -partitioned_checks, the tool generates AST_RESET_VALUE and AST_DATA_INTEGRITY properties.
- ❑ AST_DATA_INTEGRITY comprises the following:
 - AST_RESET_VALUE
 - AST_BUS_READ_AFTER_BUS_WRITE
(AST_BUS_READ_AFTER_BUS_WRITE contains AST_BUS_WRITE_BUS_READ_IMMEDIATE)
 - AST_BUS_READ_AFTER_INTERNAL_WRITE (when internal write is present)
(AST_BUS_READ_AFTER_INTERNAL_WRITE contains AST_INTERNAL_WRITE_BUS_READ_IMMEDIATE)

■ Use -addr_interaction_covers to generate the following covers:

- ❑ For fields:
 - COV_BUS_READ_OTHER_ADDR_THEN_BUS_READ
 - COV_BUS_WRITE_OTHER_ADDR_THEN_BUS_READ
- ❑ For instances:
 - COV_BUS_READ_OUTSIDE_ADDR_BLOCKS
 - COV_BUS_WRITE_OUTSIDE_ADDR_BLOCKS

■ Use -unassigned_field_checks to generate the checks for the undefined fields according to the two possible semantics: read constant undefined value or read zero.

Note: -unassigned_field_checks is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

JasperGold Apps Command Reference Manual

General Commands

-load (Continued)

- Use -sequence_checks to generate the IEV REG App sequence-style properties in the CSR App.
- Use -reset_check with reset_only or non_reset_only to specify whether to generate the reset properties or the access policy properties.

It is not possible, in a single JasperGold Apps execution run/analysis session, to configure all access policy properties along with reset properties since they require different reset configuration in the setup.

- Use -init_gap with full or none to specify the REG App init gap configuration to be considered in the generated properties.
- Use -middle_gap with full or none to specify the REG App middle gap configuration to be considered in the generated properties.
- Use -alias_covers to generate the REG App alias covers.

You can instantiate multiple CSR maps, in each CSR verification execution flow in an analysis session. Each workbook is associated with one instance.

For each instance, the tool generates one property module subinstance for each field defined in the workbook. Each subinstance presents properties according to the access type defined for the field.

Note:

- The -load switch also loads the GUI.
- -load is not valid for the -register scope.
- Register and field names specified in the CSV file must use legal Verilog identifier syntax. That is, if you use spaces or any special characters (for example: /, [, or]) in register or field names, the tool issues an error during compilation of the file.

-clear [-instance *instance_name*]

Clear the data loaded for the CSR Verification App.

Use -instance to clear only the data related to the specified instance.

JasperGold Apps Command Reference Manual

General Commands

```
-prove [-instance instance_name [-register register_name  
[-field field_name]]] [-bg]
```

Prove the properties generated by CSR Verification App from the loaded workbook.

- Use `-instance` to prove only properties related to the specified instance.
- Use `-register` to prove only properties related to the specified register.
- Use `-field` to prove only properties related to the specified field for the specified register.
- Use `-bg` to run the command in the background and continue to accept and run new commands.

```
-generate_template file [-force] [-csv | -xml]
```

Generate a template in which to complete the CSR map.

- Use `-force` to overwrite an existing file.
- Use `-csv` to generate a template in comma-separated values format (default).
- Use `-xml` to generate a template in Microsoft Excel 2003 or 2004 XML worksheet format.

Note: Register and field names specified in the CSV file must use legal Verilog identifier syntax. That is, if you use spaces or any special characters (for example: /, [, or]) in register or field names, the tool issues an error during compilation of the CSV file.

IP-XACT Switches

```
-load_ipxact -generate_csv file [-maps maps] [-force]  
[-component spirit:component_name] [-no_debug]
```

Generate a CSV file with the specified file name for check_csr -load.

- Use -maps to limit the CSV file content to specific maps.
- Use -force to overwrite an existing CSV file.
- Use -component when the IP-XACT files describe multiple instances of the specified component.
- Use -no_debug to generate the map without any reference to the IP-XACT file names and line numbers.

Note:

- Use ipxact -load prior to using this command.
- The IP-XACT file must contain a single IP-XACT component description (Section 6 of the IEEE 1685).

JasperGold Apps Command Reference Manual

General Commands

```
-load_ipxact [ipxact_file_name] -instance instance
[-maps maps]
[-unified_check] [-partitioned_checks] [-sequence_checks]
[-addr_interaction_covers] [-unassigned_field_checks (constant | zero)]
[-component spirit:component_name] [-no_debug]
```

Load the maps in the IP-XACT file as if they are from a CSV file.

- Use `-maps` to limit the CSV file content to specific maps.
- Use `-component` when the IP-XACT files describe multiple instances of the specified component.
- Use `-unified_check` to generate the single assertion `AST_DATA_INTEGRITY`, the unified data integrity check, which comprises the reset value check and read after write checks.
- Use `-partitioned_checks` to generate the partitioned checks:
 - `AST_RESET_VALUE`
 - `AST_BUS_READ_AFTER_BUS_WRITE`
 - `AST_BUS_READ_AFTER_INTERNAL_WRITE`
 - `AST_BUS_WRITE_BUS_READ_IMMEDIATE`
 - `AST_INTERNAL_WRITE_BUS_READ_IMMEDIATE`

Note:

- If you do not specify `-unified_check` or `-partitioned_checks`, the default behavior is that of `-partitioned_checks`.
- `AST_DATA_INTEGRITY` comprises the following:
 - `AST_RESET_VALUE`
 - `AST_BUS_READ_AFTER_BUS_WRITE`
(`AST_BUS_READ_AFTER_BUS_WRITE` contains
`AST_BUS_WRITE_BUS_READ_IMMEDIATE`)
 - `AST_BUS_READ_AFTER_INTERNAL_WRITE` (when internal write is present)
(`AST_BUS_READ_AFTER_INTERNAL_WRITE` contains
`AST_INTERNAL_WRITE_BUS_READ_IMMEDIATE`)

-load_ipxact (Continued)

- Use -sequence_checks to generate the IEV REG App sequence-style properties in the CSR App.
- Use -addr_interaction_covers to generate the following covers:
 - For fields:
 - COV_BUS_READ_OTHER_ADDR_THEN_BUS_READ
 - COV_BUS_WRITE_OTHER_ADDR_THEN_BUS_READ
 - For instances:
 - COV_BUS_READ_OUTSIDE_ADDR_BLOCKS
 - COV_BUS_WRITE_OUTSIDE_ADDR_BLOCKS
- Use -unassigned_field_checks to generate the checks for the undefined fields according to the two possible semantics: read constant undefined value or read zero.
- Use -no_debug to load the map without any reference to the IP-XACT file names and line numbers.

Note:

- Use ipxact -load prior to using this command.
- The IP-XACT file must contain a single IP-XACT component description (Section 6 of the IEEE 1685).

Return

A list of requested objects or the output of a requested action.

Examples

```
% check_csr -list instance
% check_csr -list field -instance inst0 -register reg1
% check_csr -list msb -instance inst0 -register reg1 -field fA
% check_csr -list addr -instance inst0 -register reg1
% check_csr -load csr1.csv -instance csr1_inst
% check_csr -prove -instance inst0
% check_csr -prove -instance inst0 -register r1
% check_csr -prove -instance inst0 -register r2 -field fA
```

JasperGold Apps Command Reference Manual

General Commands

```
// Loading multiple maps
analyze -sv design.v
check_csr -load file1.xml -instance instance1
check_csr -load file2.xml -instance instance2
check_csr -load file3.xml -instance instance3
elaborate -top design
check_csr -prove
```

See Also

[ipxact](#)

check_dld

Description

Use the `check_dld` command to detect system-level deadlock situations on a network-on-chip (NOC) with user-configured interfaces.

This command sets the configurations of the NOC (including parameters, constraints, and interfaces) that are used during deadlock detection. It also creates deadlock properties and provides tools for creating helper assertions and for debugging.

Syntax

[Set Up Deadlock Models](#)

```
check_dld -initlib
  (-dir <dir_name>
   |-set_library <lib_path>
   |-get_library)

check_dld -init -top <top_module_name>

check_dld -parameters
  (-find {<rtl_name> <analyze_switches>}
   |-add -name <parameter_name> -value <parameter_value>
   |-show [<parameter_name>]
   |-set {<parameter_name> <parameter_value>}
   |-remove <parameter_name>
   |-clear
   |-elaborate)
```

[Topology Management](#)

```
check_dld -block -add
  -instance_name <block_name> [-module_name <module_name>]

check_dld -block
  (-remove <block_name>
   |-edit <block_name> -module_name <module_name>
   |-show <block_name>
   |-set_current <block_name>
   |-get_current
   |-list_interfaces <block_name>
   |-list_connections <block_name>
   |-sanity_check <block_name>)
```

JasperGold Apps Command Reference Manual

General Commands

```
| -list
| -clear
| -clear_block_interface <block_name>

check_dld -interface -name <ifc_name>
( -add
  -pattern <pattern> -protocol <protocol_name_tcl_list>
  -type <type> (MASTER | SLAVE) -block <block_name>
  -regexp
  |-set_parameter {<parameter_name> <parameter_value>}
  |-get_parameter <parameter_name>
  |-add_parameter {<parameter_name> <parameter_value>}
  |-remove_parameter <parameter_name>
  |-show_parameters
  |-set_pattern <pattern_name>
  |-set_regexp (true | false)
  |-set_mapping <map_name> -signal <signal_name>
  |-set_global {<interface_signal> <top_signal_name>}
  |-ignore_signals {<signal_list>})

check_dld -interface
( -remove <interface_name>
| -clear
| -constraints <interface_name>
| -get_parameters <interface_name>
| -enable <interface_name> -task <task_name>
| -disable <interface_name> -task <task_name>
| -get_global_signals <interface_name>
| -get_ignored_signals <interface_name>
| -show_protocols
| -show_types <protocol_name>
| -get_pattern <interface_name>
| -get_mapping <interface_name>
| -is_regexp <interface_name>
| -list)

check_dld -interface_constraint -add
  -name <name>
  (-interface {<interface_list>} | -model <model_name>)
  -expression <expression>
  [-tag <tag>]

check_dld -interface_constraint
( -remove [<interface_constraint>]
| -show <interface_constraint>
| -list
| -clear)

check_dld -connect_interfaces {<interface_name> <interface_name>}
check_dld -disconnect_interfaces {<interface_name> <interface_name>}
check_dld -connection -list
```

JasperGold Apps Command Reference Manual

General Commands

```
check_dld -draw_topology  
check_dld -export_topology
```

[DLD Setup](#)

```
check_dld -registers  
  (-add -name <name> |-set -name <name>)  
    -value <value>  
  
check_dld -registers  
  (-show [<register_name>]  
    |-clear  
    |-list  
    |-remove <register_name>  
    |-get <register_name>  
    |-file <file_name>)  
  
check_dld -network_constraint -add  
  -name <name>  
  -expression <expression>  
  [-tag <tag>]  
  
check_dld -network_constraint  
  (-show [<network_constraint>]  
    |-clear  
    |-remove <network_constraint>)  
  
check_dld -address  
  (-file <file_name>  
    |-show  
    |-add -master <master_ifc> -slave <slave_ifc>  
      (-begin <begin_addr> -end <end_addr> [-redirection <redirection>])  
    |-clear)
```

[Elaborate and Analyze Forward Progress](#)

```
check_dld -elaborate  
  [-block <block_name>  
   [-include_file <file_name>]  
   [-elaborate_opts <string>]  
   [-auto_ignore_missing_ifc_signals]]  
  
check_dld -elaborate  
  ([[-block <block_name>] -sanity_check  
   |-get_block]  
  
check_dld -task  
  (-create <task_name>  
   |-list_interface  
   |<task_name> -add_soft_assumptions)
```

JasperGold Apps Command Reference Manual

General Commands

```
check_dld -create -type (roundtrip | handshake)+  
    -task <task_name>  
    [-latency <number_of_cycles>]  
    [-use_internal_fwd_progress <expression>]  
    [-add_soft_assumptions]  
  
check_dld -prove  
    [(-property <property_name>  
        |-task <task_name>  
        |-bg)]
```

[Debug and Classify Failing Properties](#)

```
check_dld -debug -signal_list  
    [-include ( disabled | counters | exception | payload_ctrls  
                | payload_data | fanin)+]  
    [-window <window_name>]  
  
check_dld -debug -ifc_signal_list <interfaces>  
    [-file <file_name>]  
  
check_dld -debug -highlight (ctrls | payloads | all)  
    [-window <window_name>]  
    [-target <target_name>]  
  
check_dld -analyze_cex -property <property_name> [-extend]  
  
check_dld -extend -property <property>  
  
check_dld -counter  
    (-add -name <counter_name>  
     -expression <expression>  
     -clock <clock_signal>  
     (-reset <reset_signal>  
      |-inverted_reset <reset_signal>)  
     -width <width>  
     -init_value <initial_value>  
     -description <description>  
     [-interface <interface_name_tcl_list>]  
     [-model <model_name_tcl_list>] )  
  
check_dld -counter  
    (-show <counter_name>  
     |-remove <counter_name>  
     |-list  
     |-clear)  
  
check_dld -stall_condition -add  
    -name <stall_name>  
    -expression <expression>  
    (-interfaces <interface_name_tcl_list>  
     |-model <model_name_tcl_list>)  
    [-description <stall_description>]
```

JasperGold Apps Command Reference Manual

General Commands

```
[-target_channels <channels_tcl_list>]
[-target_interface (this | others | all)]
[-type (roundtrip | handshake)]  
  
check_dld -stall_condition -edit <stall_condition_name>
( -description <description>
| -expression <expression>)  
  
check_dld -stall_condition
( -show <stall_condition_name>
| -remove <stall_condition_name>
| -list
| -clear)
```

[Stall Prevention and Requirement Management](#)

```
check_dld -spr -add
( -name <spr_name>
( -protocol <protocol_name> -type (MASTER | SLAVE)
| -interface <ifc_name>
- precondition <precondition_expression>
- postcondition <postcondition_expression>
[-check_expression])  
  
check_dld -spr -auto  
  
check_dld -spr -edit <spr_name>
( -precondition <precondition_expression>
| -postcondition <postcondition_expression>)  
  
check_dld -spr
( -remove <spr_name>
| -list
| -export <file_name>
| -show [spr_name])  
  
check_dld -instantiate_spr
( -interface <interface_name>
| -all)
- assume_latency <value>
- assert_latency <value>
- enable_interfaces <interface_names_tcl_list>
[-use_internal_fwd_progress <expression>]  
  
check_dld -analyze_spr <spr_name>
```

[Dependency Management](#)

```
check_dld -dependency -add
( ( -source <source_spr_full_name>
| -source_symmetric <source_spr_reduced_name>)
[-source_block <block_name>])
```

JasperGold Apps Command Reference Manual

General Commands

```
( ( -destination <dest_spr_full_name>
    |-destination_symmetric <dest_spr_reduced_name>)
  [-destination_block <block_name>])

check_dld -dependency -remove
( ( -source <source_spr_full_name>
    |-source_symmetric <source_spr_reduced_name>)
 )
( ( -destination <dest_spr_full_name>
    |-destination_symmetric <dest_spr_reduced_name>)
 )

check_dld -dependency
( -remove -name <dependency_name>
 | -list
 | -export <file_name>
 | -check_loop [-silent])
```

[Manipulate the Deadlock Database](#)

```
check_dld -db
( -save <file_name> [-force]
 | -load <file_name>
 | -clear)
```

[Utility Switches](#)

```
check_dld -clear
```

Argument	Definition
Set Up Deadlock Models	
-initlib	<pre>(-dir dir_name -set_library lib_path -get_library)</pre>
	Set the specified directory as the root directory in which the if_defs and if_models reside.
-init -top top_module_name	Define the specified module as the top module.

JasperGold Apps Command Reference Manual

General Commands

```
-parameters
( -find {rtl_name analyze_switches}
  -add -name parameter_name -value parameter_value
  -show [parameter_name]
  -set {parameter_name parameter_value}
  -remove parameter_name
  -clear
  -elaborate)
```

Manipulate parameters as specified.

- Use `-find` to load a file with the parameter definitions.
- Use `-show` to see the parameters followed by their value.

Note: Specify a parameter name to see a single parameter name and value.

- Use `-set` to set a new value for a parameter.
- Use `-remove` to remove the specified parameter.
- Use `-clear` to remove all parameters.
- Use `-elaborate` to generate a comprehensive list of parameters in a format that can be used by the `elaborate` command.

Topology Management

```
-block -add
  -instance_name block_name [-module_name module_name]
```

Create a new block on the topology using the specified instance name.

- Use `-instance_name` to specify the name of the block.
- Use `-module_name` to specify the module of the block.

The DLD App can only synthesize a block if its module is elaborated right before DLD elaboration.

JasperGold Apps Command Reference Manual

General Commands

```
-block
( -remove block_name
  -edit -block block_name -module_name module_name
  -show block_name
  -set_current block_name
  -get_current
  -list_interfaces block_name
  -list_connections block_name
  -sanity_check block_name
  -list
  -clear
  -clear_block_interfaces)
```

Manipulate block information

- Use `-remove` to remove the specified block.
 - Use `-edit` to edit the `module_name` of the specified block.
 - Use `-set_current` to set the main block of your current analysis. This information will be used on DLD elaboration and when referring to interfaces without their full names.
 - Use `-get_current` to get the block currently being used.
 - Use `-list_interfaces` to list the interfaces of the specified block.
 - Use `-list_connections` to list the connections of the specified block.
 - Use `-list` to list all available blocks.
 - Use `-clear_block_interfaces` to remove all interfaces in a single block.
-

JasperGold Apps Command Reference Manual

General Commands

```
-interface -name ifc_name
  (-add
   -pattern pattern -protocol protocol_name_tcl_list
   -type type (MASTER | SLAVE) -block block_name
   -regexp
   -set_parameter {parameter_name parameter_value}
   -get_parameter parameter_name
   -add_parameter {parameter_name parameter_value}
   -remove_parameter parameter_name
   -show_parameters
   -set_pattern pattern_name
   -set_regexp (true | false)
   -set_mapping map_name -signal signal_name
   -set_global {interface_signal top_signal_name}
   -ignore_signals {signal_list})
```

Configure an interface port on the instance specified by the noc_name.

- Use `-name` to specify the interface port name.
- Use `-add` to create an interface on the NOC instance.
 - Use `-pattern` to specify the naming pattern (for example, S3%).
Note: % in pattern will be replaced by the mapping name specified in the configuration file (for example, ARVALID) to get all the signals for this interface.
 - Use `-type` to specify the interface type (for example, AXI).
- Use `-set_parameter` to configure the interface parameters.
- Use `-get_parameter` to get an interface parameter's information.
- Use `-add_parameter` to add a parameter to the interface.
- Use `-remove_parameter` to remove an interface parameter.
- Use `-show_parameters` to return a list of parameter name-value pairs, for example:

```
% check_dld -interface -name a -show_parameters
VERIFY_BLK 1 WMAXBURSTS 8 RMAXBURSTS 8...
```

JasperGold Apps Command Reference Manual

General Commands

-interface (Continued)

- Use `-set_pattern` to configure the interface pattern.
- Use `-set_regexp` to set the interface pattern as regular expression.
- Use `-set_mapping` to link the signal on the map to the signal on the design.
- Use `-set_global` to overwrite global signals' configurations that were defined in the `cfg_file`.
- Use `-ignore_signals` to remove signals from the list of signals of the selected interface.

-interface

```
( -remove interface_name
  -clear
  -constraints interface_name
  -get_parameters interface_name
  -enable interface_name -task task_name
  -disable interface_name -task task_name
  -get_global_signals interface_name
  -get_ignored_signals interface_name
  -show_protocols
  -show_types protocol_name
  -get_pattern interface_name
  -is_regexp interface_name
  -list)
```

Manipulate an interface port on the top module as specified.

- Use `-remove` to remove the specified interface.
- Use `-clear` to remove all interfaces.
- Use `-constraints` to list all constraints of the specified interface.
- Use `-enable` or `-disable` to enable or disable interfaces for the proof.
- Use `-show_protocols` to list all available protocols.
- Use `-show_types` to see the available interface types.
- Use `-get_pattern` to get the pattern of the specified interface.
- Use `-is_regexp` to check if the pattern of the specified interface uses regular expressions.

JasperGold Apps Command Reference Manual

General Commands

```
-interface_constraint -add
  -name name
  -interface {interface_list}
  (-expression expression |-model model_name)
  [-tag tag]
```

Configures additional interface constraints as specified.

- Use `-add` to add an interface constraint.
- Use `-name` to assign a name to this constraint.
- Use `-interface` to apply the constraint to a specified instance list.
- Use `-expression` to specify the constraint expression.
- Use `-tag` to associate a tag with this constraint.

```
-interface_constraint
( -show [interface_constraint]
  |-remove interface_constraint
  |-list
  |-clear)
```

Manipulate additional interface constraints as specified.

- Use `-show` to see the interface constraint information.
- **Note:** Specify an interface constraint name to get information for the specified constraint.
- Use `-remove` to remove the specified interface constraint.
- Use `-clear` to remove all interface constraints.

```
-connect_interfaces interface_name interface_name
```

Connect two interfaces.

Note: Interfaces must be of the same protocol and of opposite types.

```
-disconnect_interfaces interface_name interface_name
```

Disconnect two interfaces previously connected.

```
-connection -list
```

List all available connections.

```
-draw_topology
```

Draw the topology and open the diagram in a new window.

```
-export_topology
```

Export the topology to a Tcl script.

JasperGold Apps Command Reference Manual

General Commands

DLD Setup

```
-registers
  (-add -name name | -set -name name)
  -value value
```

Add or set the specified register with the specified name.

- Use `-add` and `-name` to add a new register.
- Use `-set` and `-name` to set an existing register.
- Use `-value` to define a value for the register.

```
-registers
  (-show [register_name]
  -remove register_name
  -file file_name
  -get register_name
  -clear
  -list)
```

Show or remove one or all registers' definitions.

- Use `-show` to see all registers' names and values.
Note: Specify a register name to see only its name and value.
- Use `-remove` to remove the specified register.
- Use `-get` to get the specified register.
- Use `-clear` to remove all registers.
- Use `-list` to list all registers.

```
-network_constraint -add
  -name name
  -expression expression
  [-tag tag]
```

Add the specified network constraint.

- Use `-name` to set the constraint name.
- Use `-expression` to set the network constraint expression.
- Use `-tag` to classify the constraint.

JasperGold Apps Command Reference Manual

General Commands

```
-network_constraint
( -show [network_constraint]
  |-remove network_constraint
  |-clear)
```

Manipulate network constraints as specified.

- Use `-show` to see the network constraint information.
Note: Specify a network constraint name to get information for the specified constraint.
- Use `-remove` to remove the specified network constraint.
- Use `-clear` to remove all network constraints.

```
-address
( -file file_name
  |-show
    |-add -master master_ifc -slave slave_ifc
      (-begin begin_addr -end end_addr [-redirection redirection])
  [-clear]
```

Manipulate an address map as specified.

- Use `-file` to load a file containing an address map.
- Use `-show` to see the address map.
- Use `-add` to add the specified address map.
- Use `-clear` to clear the address map.

Elaborate and Analyze Forward Progress

```
-elaborate
[-block block_name]
[-include_file file_name]
[-elaborate_opts string]
[-auto_ignore_ifc_signals]
```

Elaborate the design using information previously configured by the user, which is contained in the `include_file` and in the `elaborate_opts`.

Note: The block must have previously been elaborated so that it is possible to check the interface's correctness. If you do not provide the block name, the tool uses the current block or the only existent block (see `check_dld -block -set_current`).

- Use `-include_file` to define the Verilog include file.
- Use `-elaborate_opts` to set options to be used with the elaborate command.

JasperGold Apps Command Reference Manual

General Commands

```
-elaborate
( -sanity_check
  |-get_block
```

Manipulate elaboration information.

- Use `-sanity_check` to run the sanity check on the specified block. If you do not specify a block, the DLD App uses the current block or the only existent block (see `check_dld -block -set_current`).
- Use `-get_block` to get the current elaborated block.

```
-task ( -create task_name
  |-list_interface
    task_name -add_soft_assumptions)
```

Create a new DLD task.

Use `-add_soft_assumptions` to create soft assumptions for stable inputs.

```
-create -type (roundtrip | handshake) +
  -task task_name
  [-latency number_of_cycles]
  [-use_internal_forward_progress expression]
  [-add_soft_assumptions]
```

Create the deadlock property as specified.

- Use `-type` to specify the type of the deadlock property.
- Use `-task` to specify the name of the task where the properties will be created.
- Use `-latency` to specify the bound of the deadlock property.
- Use `-add_soft_assumptions` to create soft assumptions for stable inputs.

```
-prove
[ ( -property property_name
  | -task task_name
  | -bg) ]
```

Prove the specified deadlock properties.

- Use `-property` to prove the specified property.
- Use `-task` to prove all deadlock properties in the specified task.
- Use `-bg` to prove deadlock properties in the background.

JasperGold Apps Command Reference Manual

General Commands

Debug and Classify Failing Properties

```
-debug -signal_list  
  [-include ( disabled | counters | exception | payload_ctrls  
             | payload_data | fanin)+]  
  [-window window_name]
```

Clear all signals in the Visualize window and add groups of specific signals such as enabled interface signals.

- Use `-signal_list` to include groups of enabled interface signals.
- Use `-include` to add one or more of the following groups: disabled, counters, exception, payload_data, payload_ctrls, or fanin.
- Use `-window` to specify a Visualize window different than the current one.

```
-debug -ifc_signal_list interfaces [-file file_name]
```

TBD

```
-debug -highlight (ctrls | payloads | all)  
  [-window window_name]  
  [-target target_name]
```

Highlight payload controls, data, or both that are in the transitive why of the trace.

- Use `-highlight` to highlight all payload controls, data, or both that are in the transitive why of the trace.
- Use `-window` to specify a Visualize window different than the current one.

```
-analyze_cex -property property_name -extend
```

Analyze a counterexample to find a trace showing the signals that are related to the counterexample.

```
-extend [-property property]
```

Try to extend the current trace to find if the current situation is a deadlock or stall condition.

JasperGold Apps Command Reference Manual

General Commands

```
-counter
( -add -name counter_name
  -expression expression
  -clock clock_signal
  ( -reset reset_signal
    |-inverted_reset reset_signal)
  -width width
  -description description
  [-interface interface_name_tcl_list]
  [-model model_name_tcl_list] )
```

Manipulate a counter as specified.

- Use `-add` to add a counter.
- Use `-name` to set the counter name.
- Use `-expression` to set the counter expression.
- Use `-description` to set the counter description.
- Use `-clock` to set the counter clock signal.
- Use `-reset` to set the counter reset signal.
- Use `-width` to set the counter width.

```
-counter
( -show counter_name
  |-remove counter_name
  |-list
  |-clear)
```

Manipulate additional counters as specified.

- Use `-show` to see the counter information.
Note: Specify a counter name to get information for the specified counter.
- Use `-remove` to remove the specified counter.
- Use `-clear` to remove all counters.

JasperGold Apps Command Reference Manual

General Commands

```
-stall_condition
  -add
  -name stall_name
  -expression expression
  ( -interfaces interface_name_tcl_list
    | -model model_name_tcl_list)
  [-description stall_description]
  [-target_channels channel_tcl_list]
  [-target_interface (this | others | all)]
  [-type (roundtrip | handshake)]
```

Manipulate a stall condition as specified.

- Use **-add** to add a stall condition.
- Use **-name** to set the stall condition name.
- Use **-expression** to set the stall condition expression.
- Use **-description** to set the stall condition description.

```
-stall_condition -edit stall_name
  ( -description description
    | -expression expression)
```

TBD

```
-stall_condition
  ( -show [stall_condition_name]
    | -remove stall_condition_name
    | -list
    | -clear)
```

Manipulate additional stall conditions as specified.

- Use **-show** to see the stall condition information.
Note: Specify a stall condition name to get information for the specified stall condition.
- Use **-remove** to remove the specified stall condition.
- Use **-clear** to remove all stall conditions.

Stall Prevention and Requirement Management

```
-spr -add
  ( -name spr_name
    ( -protocol protocol_name -type (MASTER | SLAVE)
      | -interface ifc_name)
    -precondition precondition_expression
    -postcondition postcondition_expression
    [-check_expression] )
```

TBD

JasperGold Apps Command Reference Manual

General Commands

```
-spr -auto
```

TBD

```
-spr -edit spr_name
( -precondition precondition_expression
| -postcondition postcondition_expression)
```

TBD

```
-spr
( -remove spr_name
| -list
| -export file_name
| -show [spr_name])
```

TBD

```
-instantiate_spr
( -interface interface_name
| -all)
-assume_latency value
-assert_latency value
-enable_interfaces interface_names_tcl_list
[-use_internal_fwd_progress expression]
```

TBD

```
-analyze_spr spr_name
```

TBD

Dependency Management

```
-dependency -add
( ( -source source_spr_full_name
| -source_symmetric source_spr_reduced_name)
[ -source_block block_name])
( ( -destination dest_spr_full_name
| -destination_symmetric dest_spr_reduced_name)
[ -destination_block block_name])
```

TBD

```
-dependency -remove
( ( -source source_spr_full_name
| -source_symmetric source_spr_reduced_name)
)
( ( -destination dest_spr_full_name
| -destination_symmetric dest_spr_reduced_name)
)
```

TBD

JasperGold Apps Command Reference Manual

General Commands

```
-dependency
( -remove -name dependency_name
  | -list
  | -export file_name
  | -check_loop [-silent])
```

TBD

Manipulate the Deadlock Database

```
-db
( -save file_name [-force]
  | -load file_name
  | -clear)
```

Save, restore, or clear the deadlock database.

- Use `-save` to save all the information stored in the deadlock database to the specified file, and use `-force` to overwrite the file if it already exists.
- Use `-load` to restore the information from the specified file to the deadlock database.
- Use `-clear` to clear all the information stored in the deadlock database.

Utility Switches

```
-clear
```

TBD

Return

Various

Examples

```
% check_dld -initlib -dir "./deadlock_lib/"
% check_dld -block -add -instance_name top
% check_dld -interface -add -name sh -pattern "SH0%" -protocol AXI4 -type MASTER
-block top
% check_dld -elaborate -include "include.sv" -elaborate_opts "-bbox_a 256"
% check_dld -task -create DLD
% check_dld -create -type roundtrip -latency 10 -task DLD
```

JasperGold Apps Command Reference Manual

General Commands

See Also

No related commands

check_loop

Description

Use the `check_loop` command to search for a combinational loop starting from the specified signal or all loops in the cone of influence (COI) of a property. If the tool does not find a loop, it returns an empty string; otherwise, it returns a list of signals that contribute to the loop. If you specify a property, the command returns a list of all loops within the property's COI.

This command is helpful when you see an error message that identifies the presence or possible presence of a combinational loop in the COI of a signal. Specify the signal name reported in the error message as the starting point for a search to identify signals that contribute to the loop. Remove the loop in the design or add a stopat.

Syntax

```
check_loop <signal_name>
    [-task <task_name>] [-gui] [-latch_data]
check_loop <property_name> [-latch_data]
check_loop -global [-classify] [-latch_data] [-silent]
```

Argument	Definition
<code>signal_name</code>	Starting at <code>signal_name</code> , search for a combinational loop.
<code>-task task_name</code>	Consider all stopats and abstractions from task <code>task_name</code> . Note: If you do not use the <code>-task</code> switch, the only stopats this command considers are those specified with the <code>-env</code> switch.
<code>-gui</code>	Open the Source Browser in a highlight mode for loop investigation. The Source Browser in <code>check_loop</code> highlight mode includes a <i>Signals List</i> pane, which lists all the signals in the loop. The tool highlights the starting signal in the <i>Signals List</i> pane and its driver in the source code pane. Click on any listed signal name to move the source code highlight to its driver.

JasperGold Apps Command Reference Manual

General Commands

`-latch_data`

Include the data pin of latches in the search for combinational loops. By default, `check_loop` only considers loops coming through the enable pin of latches.

`property_name -latch_data`

Starting at the specified property, search for all combinational loops in its COI.

Use `-latch_data` to include the data pin of latches in the search for combinational loops.

`-global`

`[-classify]`
`[-latch_data]`
`[-silent]`

Search for combinational loops in the full design. This command prints out the loops on the console, but it also returns its output as a Tcl typed list as a convenience for users.



- If the tool finds loops, it prints them in the console, each one preceded by the string `Loop n:`, where `n` is an integer starting from 1.
- This command does not require a signal name.
- Use `-classify` to get more information about loops.
- Use `-latch_data` to include the data pin of latches in the search for combinational loops.
- Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

Note: `-classify` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

Return

For a signal, this command returns a list of signal names that contribute to a loop, or an empty string if the tool does not find a loop.

For a property or global search, this command returns a list of loops, and each loop is a list of signals.

Examples

```
% check_loop sigA  
% check_loop sig_A -gui  
% check_loop -global  
% check_loop -global -classify
```

See Also

[stopat](#)

check_lpv

Description

Use the `check_lpv` command to load a Unified Power Format (UPF) or Common Power Format (CPF) file, elaborate a power-aware design, and create and verify properties, that is, structural checks and assertions.

Syntax

```
check_lpv -load_upf <pf_file> [-allow_unescaped_bus_index]
check_lpv -load_cpf <pf_file>
check_lpv -generate_power_design
    [-invert_retention_event_semantics]
    [-exclude_modules <list_of_modules>]
    [-corrupt (off | on | ports_only)]
check_lpv -create assert_all
    [-task <task_name>]
    [-power_domain <power_domain>]
    [-after <N>]
check_lpv -create
    ( assert_iso_up_all
      | assert_iso_up_before
      | assert_iso_up_during
      | assert_iso_up_after)
    [-task <task_name>]
    [-port <port_name>]
    [-power_domain <power_domain>]
    [-after <N>]
check_lpv -create assert_iso_up_clk_stable
    [-task <task_name>]
    [-power_domain <power_domain>
        [-transitive]]
    [-after <N>]
check_lpv -create
    ( assert_iso_supply
      | assert_iso_change_signal_stable)
    [-task <task_name>]
    [-port <port_name>]
    [-power_domain <power_domain>]
    [-after <N>]
```

JasperGold Apps Command Reference Manual

General Commands

```
check_lpvs -create
( assert_supply_onehot
| assert_supply_parallel
| assert_clock_stable_after_restore
| assert_ret_supply_on_save
| assert_ret_supply_on_restore
| assert_ret_supply_on_from_save_to_restore
| assert_ret_no_save_on_restore
| assert_pd_supply_change_clk_stable)
[-task <task_name>]
[-power_domain <power_domain>]
[-after <N>]

check_lpvs create assert_rst_non_ret_flops
[-precond <combinational_expression>]
[-power_domain <power_domain>]
[-after <N>]

check_lpvs -create
( cover_power_switch_ctrl | cover_iso_ctrl
| cover_ret_save_ctrl | cover_ret_restore_ctrl
| cover_all)
[-power_domain <power_domain>]
[-after <N>]

check_lpvs -verify
( check_iso_clk_all
| [-use_declared_clocks] [-detailed]
| check_iso_clk_ctrl_data
| check_iso_clk_ctrl
| check_iso_outputs
)
[ -power_domain <power_domain>
| -instance <instance_name>]
[-report <file_name>]

check_lpvs -verify check_iso_inputs
[-power_domain <power_domain>]
[-report <file_name>]

check_lpvs -verify check_iso_default_value
[-exceptions <file_name>]
[-power_domain <power_domain>]
[-report <file_name>]

check_lpvs -verify
( check_iso_connections
| check_ret_connections
| check_power_switch_connections)
[-power_domain <power_domain>]
[-report <file_name>]
```

JasperGold Apps Command Reference Manual

General Commands

```
check_lpv -verify
  ( check_iso_ctrl_RST
  | check_ret_ctrl_RST
  | check_power_switch_ctrl_RST
  )
  [-power_domain <power_domain>]
  [-report <file_name>]

check_lpv -verify check_supply_parallel
  [-power_domain <power_domain>]
  [-report <file_name>]

check_lpv -get_status

check_lpv -sec (-spec |-imp)
```

Argument	Definition
-load_upf <i>pf_file</i> [-allow_unescaped_bus_index]	<p>Load and parse the specified power format file.</p> <p>Use <code>-allow_unescaped_bus_index</code> if your UPF does not use curly braces or escape signal names that contain square brackets.</p> <p>Note: <code>-allow_unescaped_bus_index</code> is an all-or-nothing switch. That is, the entire file must be unescaped or use no curly braces around bus indices.</p>
-load_cpf <i>pf_file</i>	Load and parse the specified power format file.

JasperGold Apps Command Reference Manual

General Commands

```
-generate_power_design  
[-invert_retention_event_semantics]  
[-exclude_modules list_of_modules]  
[-corrupt {off | on | ports_only}]
```

Build a power model of the design.

- Use `-invert_retention_event_semantics` to specify that you want the tool to consider high or low sensitivities declared in `set_retention_control` UPF commands as posedge or negedge, respectively.
- Use `-exclude_modules` with a Tcl list set of modules that should not be made power-aware (for example, verification modules or modules with explicit power modeling).
- Use `-corrupt` with one of the following arguments to specify the corruptions algorithm you want to use:
 - `off` – No corruption is done.
 - `on` – Use this argument to corrupt signals driven by active components within a disabled power domain. *This option is the default.*
 - `ports_only` – Use this argument to corrupt all power domain ports.

Note: When a power domain is disabled, the outputs of the active components in that domain are corrupted. Except for components that directly connect a domain's inputs and outputs, the tool recognizes components as "active" if they are reachable from any domain port.

Note:

- You must use `check_lpv -load_*` before using this command.
- `ports_only` is not recommended as it does not reflect the semantics described in the UPF LRM.

JasperGold Apps Command Reference Manual

General Commands

```
-create assert_all
[-task task_name]
[-power_domain power_domain]
[-after N]
```

Create all of the auto assertions, which are based on the design and the power intent. Refer to the `assert_*` definitions below.

- Use `-task` to apply the command to the specified task instead of the `<LowPower>` task.
- Use `-power_domain` to generate assertions for the specified power domain instead of all domains.
- Use `-after` to delay the property check by the specified number of cycles.

Note: This command generates assertions. You must manually run the proof of the assertions.

JasperGold Apps Command Reference Manual

General Commands

```
-create
( assert_iso_up_all | assert_iso_up_before
| assert_iso_up_during | assert_iso_up_after
[-task task_name]
[-port port_name | -power_domain power_domain]
[-after N]
```

Create the specified isolation cell auto assertions, which are based on the design and the power intent:

- `assert_iso_up_all` – Creates auto assertions for the following:
 - `assert_iso_up_before`
 - `assert_iso_up_during`
 - `assert_iso_up_after`
- `assert_iso_up_before` – Verifies that whenever a power domain of the signal driving the isolation cell is off and the power domain of the load of the isolation cell is on, the isolation conditions related to this isolation cell will be true strictly before the power is off.
- `assert_iso_up_during` – Verifies that whenever a power domain of the signal driving the isolation cell is off and the power domain of the load of the isolation cell is on, the isolation conditions related to this isolation cell are true.
- `assert_iso_up_after` – Verifies that whenever a power domain of the signal driving the isolation cell is off and the power domain of the load of the isolation cell is on, the isolation conditions related to this isolation cell are still true strictly after the power is turned on again.
- Use `-task` to apply the command to the specified task instead of the `<LowPower>` task.
- Use `-port` to specify a port in the RTL that is isolated. This switch is useful if you want the assertions to be generated only for specific isolation ports.

JasperGold Apps Command Reference Manual

General Commands

-create (Continued)

- Use `-power_domain` to generate assertions for the specified power domain instead of all domains.
- Use `-after` to delay the property check by the specified number of cycles.

Note:

- `-port` and `-power_domain` are mutually exclusive switches.
- This command generates assertions. You must manually run the proof of the assertions.

-create assert_iso_up_clk_stable

```
[-task task_name]
[-power_domain power_domain [-transitive]]
[-after N]
```

Create the auto assertion `assert_iso_up_clk_stable`, which verifies that the clocks in the power domain are stable when the power domain is up and its isolations are enabled.

With this command, the tool generates assertions for each clock of each power domain in the design. This command generates only one precondition property for each power domain.

- Use `-task` to apply the command to the specified task instead of the `<LowPower>` task.
- Use `-power_domain` to generate assertions for the specified power domain instead of all domains.
- Use `-transitive` to recursively consider flops inside nested power domains without isolation rules in order to generate the clocks of the power domain specified by the `-power_domain` switch.
- Use `-after` to delay the property check by the specified number of cycles.

JasperGold Apps Command Reference Manual

General Commands

```
-create (assert_iso_supply | assert_iso_change_signal_stable)
[-task task_name]
[-port port_name | -power_domain power_domain]
[-after N]
```

Create the specified isolation cell auto assertions:

- `assert_iso_supply` – Verifies that whenever a power domain of the signal driving the isolation cell is on, and the power domain of the load of the isolation cell is on, the power supply of the isolation cell is on.
- `assert_iso_change_signal_stable` – Verifies that an isolation control toggle does not generate a toggle between an isolation target and isolation output.
- Use `-task` to apply the command to the specified task instead of the `<LowPower>` task.
- Use `-port` to specify a port in the RTL that is isolated. This switch is useful if you want the assertions to be generated only for specific isolation ports.
- Use `-power_domain` to generate assertions for the specified power domain instead of all domains.
- Use `-after` to delay the property check by the specified number of cycles.

Note:

- `-port` and `-power_domain` are mutually exclusive switches.
- This command generates assertions. You must manually run the proof of the assertions.

JasperGold Apps Command Reference Manual

General Commands

```
-create
( assert_supply_onehot | assert_supply_parallel
  assert_clock_stable_after_restore
  assert_ret_supply_on_save | assert_ret_supply_on_restore
  assert_ret_supply_on_from_save_to_restore
  assert_ret_no_save_on_restore
  assert_pd_supply_change_clk_stable)
[-task task_name] [-power_domain power_domain] [-after N]
```

Create the specified auto assertions:

- `assert_supply_onehot` – Verifies that if a power supply net has the resolution semantic onehot, then there is never more than one active driver.
- `assert_supply_parallel` – Verifies that if a power supply net has resolution semantic parallel, then all the drivers that are in the one-state have the same voltage.
- `assert_clock_stable_after_restore` – Verifies that there is no clock change in the last cycle of a restore.
- `assert_ret_supply_on_save` – Verifies that whenever the value of an element is saved to the retention logic, the power supply of the corresponding retention logic is on.
- `assert_ret_supply_on_restore` – Verifies that whenever the value of an element is restored from the retention logic, the power supply of the corresponding retention logic is on.
- `assert_ret_supply_on_from_save_to_restore` – Verifies that the power supply of the retention logic is always on between save and restore.
- `assert_ret_no_save_on_restore` – Verifies that for each retention rule, the save signal and restore signals are not enabled at same time.
- `assert_pd_supply_change_clk_stable` – Verifies that if the power supply of a power domain is changing, then the clock is disabled.
- Use `-task` to apply the command to the specified task instead of the `<LowPower>` task.

JasperGold Apps Command Reference Manual

General Commands

-create (Continued)

- Use `-power_domain` to generate assertions for the specified power domain instead of all domains.
- Use `-after` to delay the property check by the specified number of cycles.

Note: This command generates assertions. You must manually run the proof of the assertions.

-create assert_rst_non_ret_flops

`[-precond combinational_expression]`
`[-power_domain power_domain] [-after N]`

Create an auto assertion that checks whether the values of the non-retention flops are equal to the reset values if the precondition is true.

- Use `-precond` to specify the precondition expression that must be true. If you do not use the `-precond` switch, the tool uses the restore signal of the retention rule as the precondition of the assertion.
- Use `-power_domain` to generate the assertion for the specified power domain instead of all domains.
- Use `-after` to delay the property check by the specified number of cycles.

JasperGold Apps Command Reference Manual

General Commands

```
-create
( cover_power_switch_ctrl | cover_iso_ctrl
| cover_ret_save_ctrl | cover_ret_restore_ctrl | cover_all)
[-power_domain power_domain] [-after N]
```

Create the specified cover properties to check control signals for power element toggles:

- `cover_power_switch_ctrl` – Checks whether control signals for power switches can toggle.
- `cover_iso_ctrl` – Checks whether control signals for isolations can toggle.
- `cover_ret_save_ctrl` – Checks whether save control signals for retentions can toggle.
- `cover_ret_restore_ctrl` – Checks whether restore control signals for retentions can toggle.
- `cover_all` – Creates all of the cover properties listed above.
- Use `-power_domain` to generate cover properties for the specified power domain instead of all domains.
- Use `-after` to delay the property check by the specified number of cycles.

Note: This command generates assertions. You must manually run the proof of the assertions.

JasperGold Apps Command Reference Manual

General Commands

```
-verify
(  check_iso_clk_all [-use_declared_clocks] [-detailed]
  |  check_iso_clk_ctrl_data
  |  check_iso_clk_ctrl
  |  check_iso_outputs)
  [-power_domain power_domain |-instance instance_name]
  [-report file_name]
```

Perform the specified structural checks. These checks target rules that should be followed when defining the power configurations. The tool provides a textual report showing violations to the rules.

- Use `check_iso_clk_all` to verify that all input clocks of a power domain are isolated with an isolation cell.
 - use `-use_declared_clocks` to show only instance inputs that are directly connected to declared clocks. By default, the violation report shows all instance inputs that lead to the clock/enable logic inside the instance.
 - Use `-detailed` to get a verbose textual report that includes, beyond all violations, a list of all instance ports that passed the check.
 - Use `check_iso_clk_ctrl_data` to verify that for each isolated signal, the clocks of the isolation control signal and the isolated signal are derived from the same primary clock.
- Note:** The tool warns “The clocks of the isolation element and the isolation control overlap but are not the same” if the two lists have some declared clocks in common but also have some different declared clocks.
- Use `check_iso_clk_ctrl` to verify that for each isolated signal that isolates a clock signal, the isolation control is derived from the same primary clock as the isolated clock.

JasperGold Apps Command Reference Manual

General Commands

-verify (Continued)

- Use `check_iso_outputs` to verify whether the outputs of the boundary instances of each power domain are isolated (OK), not isolated (ERROR) or partially isolated (PARTIALLY OK).
 - Use `-power_domain` to run the check for the specified power domain instead of running it for all domains.
 - Use `-instance` to perform the specified checks on the specified instance.
 - Use `-report` to export the report to a file instead showing it in the console message pane.
-

`-verify check_iso_inputs`

```
[-power_domain power_domain]  
[-report file_name]
```

Perform the `check_iso_inputs` structural check.

This check verifies whether the inputs of the boundary instances of each power domain are isolated (OK), not isolated (ERROR), or partially isolated (PARTIALLY OK).

- Use `-power_domain` to run the check for the specified power domain instead of running it for all domains.
 - Use `-report` to export the report to a file instead of showing it in the console message pane.
-

JasperGold Apps Command Reference Manual

General Commands

```
-verify check_iso_default_value  
[-exceptions file_name] [-power_domain power_domain]  
[-report file_name]
```

Perform the `check_iso_default_value` structural check.

This check verifies that the clamp value of an isolation cell coincides with the reset value of the isolated signal and returns a Tcl list with the signals that failed.

Use this command after `check_lpv -load_upf`.

You can use the `check_iso_default_value` check as a power structure check before the design is power aware. This usage does not model corruption of isolation cells; however, using the check in this way avoids requirements such as power sequence or constraints on the power signals. If you run `check_iso_default` after power transformation, you must provide constraints and/or a valid power sequence to generate realistic results.

- Use `-exceptions` to specify isolation values that are different from the reset value. The format of the specified file is as follows:

`(signal_name value [1 | 0] | signal_name cycle natural_number)`

- `signal_name value [1 | 0]`: Use this format to verify that the specified signal has the specified clamp value instead of the value from the last reset cycle.
 - `signal_name cycle natural_number`: Use this format to verify that the specified signal has the clamp value from a specified reset cycle instead of the last reset cycle.

Refer to the examples below.

- Use `-power_domain` to run the check for the specified power domain instead of running it for all domains.
- Use `-report` to export the report to a file instead of showing it in the console message pane.

JasperGold Apps Command Reference Manual

General Commands

```
-verify
( check_iso_connections | check_ret_connections
  | check_power_switch_connections)
[-power_domain power_domain]
[-report file_name]
```

Perform the specified structural checks.

These checks target inputs and outputs of power elements to ensure that they are not empty.

- Use `check_iso_connections` to verify that the power net, the isolation signal, and the isolation elements of an isolation rule are not empty.
- Use `check_ret_connections` to verify that the power net, the save and restore signals, and the retention elements of a retention rule are not empty.
- Use `check_power_switch_connections` to verify that the output supply net, the input supply nets, and the control signal of a power switch are not empty.
- Use `-power_domain` to run the check for the specified power domain instead of running it for all domains.
- Use `-report` to export the report to a file instead of showing it in the console message pane.

JasperGold Apps Command Reference Manual

General Commands

```
-verify
(  check_iso_ctrl_RST | check_ret_ctrl_RST
  | check_power_switch_ctrl_RST)
[-power_domain <power_domain>]
[-report <file_name>]
```

Perform the specified structural checks.

These checks verify that control of power elements are not X in the last cycle of reset.

- Use `check_iso_ctrl_RST` to verify that the power net and the isolation signal of an isolation rule are not X in the last cycle of reset.
- Use `check_ret_ctrl_RST` to verify that the power net and save and restore signals of a retention rule are not X in the last cycle of reset.
- Use `check_power_switch_ctrl_RST` to verify that the output supply net, input supply nets, and control signal of a power switch are not X in the last cycle of reset.
- Use `-power_domain` to run the check for the specified power domain instead of running it for all domains.
- Use `-report` to export the report to a file instead of showing it in the console message pane.

```
-verify check_supply_parallel
[-power_domain power_domain]
[-report file_name]
```

Verify that all sources of a parallel supply net share the same supply driver.

To perform this check, the tool traverses backward through the design from the parallel supply net to confirm that the same root supply driver exists for all possible drivers of the parallel supply net.

- Use `-power_domain` to run the check for the specified power domain instead of running it for all domains.
- Use `-report` to export the report to a file instead of showing it in the console message pane.

JasperGold Apps Command Reference Manual

General Commands

-get_status

Check whether the loaded design is power-aware. This command returns the following:

- An error when no design is elaborated.
- An empty string when a design is elaborated but has no power intent file loaded.
- POWER_INTENT_LOADED when you have used `check_lpvs -load_cpf` or `check_lpvs -load_upf` to load a power format file.
- POWER_AWARE when you have made the design power-aware with the command `check_lpvs -generate_power_design`.

-sec (-spec | -imp)

Populate the LPV App GUI (*Power Domain Info Browser* and *Design Hierarchy* pane) with content for the specified SEC side (spec or imp).

Refer to command documentation for “[check_sec](#)” on page 345, specifically `check_sec -spec_upf_file` and `check_sec -imp_upf_file`.

Return

`-get_status` returns the following:

- An error when no design is elaborated.
- An empty string when a design is elaborated but has no power intent file loaded.
- POWER_INTENT_LOADED when you have used `check_lpvs -load_cpf` or `check_lpvs -load_upf` to load a UPF or CPF file.
- POWER_AWARE when you have made the design power-aware with the command `check_lpvs -generate_power_design`.

Examples

```
% check_lpvs -load_cpf simple.cpf
% check_lpvs -create assert_all
% check_lpvs -generate_power_design

# Exceptions File Example
# -----
```

JasperGold Apps Command Reference Manual

General Commands

```
% check_lpv -verify check_iso_default_value -exceptions my_exceptions  
  
# my_exceptions contains the following lines:  
sig1 cycle 10  
sig2 value 1  
  
# In this example, the tool compares the clamp value for sig1 against the reset  
# value from 10 cycles before the end of the reset. In addition, it compares  
# the clamp value for sig2 against 1.
```

See Also

[set_xprop_use_low_power](#)

check_one_of

Description

Use this command to check whether an actual value matches the expectations. If it does not, `check_one_of` returns with an error code and aborts command interpretation unless you also use `catch`. The tool evaluates the optional `on_error` block before returning with an error code.

Note: This command is useful when creating a regression script.

Syntax

```
check_one_of <actual> <ok_tcl_list> [ <on_error> ]
```

Argument	Definition
<i>actual</i>	
	Actual value.
<i>ok_tcl_list</i>	
	List of values that is expected to include <i>actual</i> . This argument can be a Tcl list.
<i>on_error</i>	
	Evaluate when <i>actual</i> is not an element in <i>ok_list</i> .

Return

No value returned.

Examples

```
% check_one_of [trace -getValue x 1] {2'b01 2'b0x}
```

See Also

[check_code](#)

check_sec

Description

Use the `check_sec` command to formally compare two RTL designs, the specification (spec) and the implementation (imp). With this command, the Sequential Equivalence Checking (SEC) App analyzes and elaborates the two designs in a fashion that is similar to other JasperGold Apps, but in addition, it links (connects) them and creates one or more tasks for comparing pairs of signals from both models.

Syntax

[Setup](#)

```
check_sec -setup
  [-global_setup_file <global_setup_file>]
  -spec_dut <spec_dut_instance_name_list>
  (-spec_analyze_opts {<analyze_switches_and_target>})*
  [-spec_elaborate_opts {<elaborate_flags_list>}]
  [-spec_top <spec_top_name>]
  -imp_dut <imp_dut_instance_name_list>
  (-imp_analyze_opts {<analyze_switches_and_target>})*
  [-imp_elaborate_opts {<elaborate_flags_list>}]
  [-imp_top <imp_top_name>]
  (-ipk_opts {ipk_switches_and_arguments})+
  [-setup_file <setup_file>]
  [-no_auto_mapping]
  [ -spec_upf_file <upf_file>
    |-spec_cpf_file <cpf_file> ]
  [ -imp_upf_file <upf_file>
    |-imp_cpf_file <cpf_file> ]
  [-power_control_signals {<power_control_signals_list>}]]
  [-map_file <map_file_name>]
  [-exclude_internal]
  [-include_cells]
  [-auto_map_reset_x_values [-map_one_side_reset_x]]
  [-generate_named_x_wires [-auto_map_x_values]]]

check_sec -set_dut_instance
  -spec_dut <spec_dut_instance_name_list>
  -imp_dut <imp_dut_instance_name_list>
```

JasperGold Apps Command Reference Manual

General Commands

Clear

```
check_sec -clear  
[all | analyze | rule | mapping | proof]
```

Interface Check

```
check_sec -interface  
[-mapped |-unmapped]  
[-spec |-imp]  
[-signal_type ( primary_input | primary_output  
| bbox_input | bbox_output | undriven  
| setup_stopat | reset_x | x_value  
| x_default_value | x_lpv_value | x_md_abstract)]  
[-instance <instance_full_path>]  
[-task task_name]
```

Signals List

```
check_sec -list  
[ [-signal]  
  [-signal_type ( input | output | primary_input | primary_output  
| bbox_input | bbox_output | mpram_port | undriven  
| flop | latch | internal | stopat | reset_x  
| x_value | x_default_value | x_md_abstract)]  
  [-spec |-imp]  
  [-instance <instance_full_path>  
  | -name {<regexp_or_signal>} [-reg [-verbose] ]  
]
```

Hierarchy Mapping Rule List

```
check_sec -list -mapping -hierarchy  
[-return (text | id)]
```

Signal Mapping List

```
check_sec -list -mapping  
[-mapping_type ( connection_all | connection_primary_input  
| connection_bbox_output | connection_undriven  
| connection_stopat | connection_x_value  
| connection_x_default_value | connection_misc  
| ver_target_all | ver_target_primary_output  
| ver_target_mpram_port | ver_target_bbox_input  
| ver_target_misc | cutpoint_all | cutpoint_both  
| cutpoint_spec | cutpoint_imp | stopat_all  
| stopat_both | stopat_spec | stopat_imp  
| helper | init ) ]
```

JasperGold Apps Command Reference Manual

General Commands

```
[-signal_type ( input | output | primary_input | primary_output
                | bbox_input | bbox_output | mpram_port | undriven
                | flop | latch | internal | stopat | reset_x
                | x_value | x_default_value | x_md_abstract)]
[ (-instance <instance_full_path>)
| (-of_signal {<regexp_or_signal_list>}
  [ -spec_regex [-verbose]
    | -imp_regex [-verbose]]))
]
[-tag {<tag_name>}]
[-task {<task_name>} [-include_global_mapping]]
[-return (text | id | property)]
[-speculative (on | off)]
[-proof_status (proven | cex | unprocessed | undetermined)+]
```

Marked Signals List

```
check_sec -list -marked -forbid_init [-imp]
```

Cutpoint Status List

```
check_sec -list -mapping
(-unguaranteed_cutpoint |-unproved_cutpoint)
[-speculative (on | off)]
```

Disabled Cutpoints List

```
check_sec -list -mapping
-violating_cutpoint (all | clock_signal | init_diff | assumption_comb_coi)
[-force]
```

Disabled Init Mapping List

```
check_sec -list -mapping -violating_init_mapping
```

Rules List

```
check_sec -list -rule [-return (text | id | property)]
```

Rule Mapping List

```
check_sec -list -rule_mapping <rule_id>
[-return (text | id | property)]
```

JasperGold Apps Command Reference Manual

General Commands

Mapping Info

```
check_sec -mapping_info
  (-id <mapping_id_list>
   | (-spec {<spec_regexp_or_signal_list>}
   | -imp {<imp_regexp_or_signal_list>}))
  )
  [-regexp [-verbose]]
  [-signal_type ( input | output | primary_input | primary_output
                  | bbox_input | bbox_output | mpram_port | undriven
                  | flop | latch | internal | stopat | reset_x
                  | x_value | x_default_value | x_md_abstract))
  [-mapping_type ( connection_all | connection_primary_input
                  | connection_bbox_output | connection_undriven
                  | connection_stopat | connection_x_value
                  | connection_x_default_value | connection_misc
                  | ver_target_all | ver_target_primary_output
                  | ver_target_mpram_port | ver_target_bbox_input
                  | ver_target_misc | cutpoint_all | cutpoint_both
                  | cutpoint_spec | cutpoint_imp | stopat_all
                  | stopat_both | stopat_spec | stopat_imp
                  | helper | init ) ]
  [ (-instance <instance_full_path>)
  | (-of_signal {<regexp_or_signal_list>}
    [ -spec_regexp [-verbose]
    | -imp_regexp [-verbose]]))
  ]
  [-proof_status (proven | cex | unprocessed | undetermined)+]
  [-tag {<tag_name>}]
  [-task {<task_name>} [-include_global_mapping]]
  [-get_type |-get_task |-get_tag |-is_speculative]
```

Hierarchy Mapping Rule Info

```
check_sec -mapping_info -hierarchy
  -id <hierarchy_aliasing_rule_id_list>
```

Rule Info

```
check_sec -rule_info -id {<rule_id_list>}
```

Signal Mapping

```
check_sec -map
  (-auto
   [-spec {<spec_instance_list>}
   [-imp {<imp_instance_list>}]])
  [-signal_type ( input | output | primary_input | primary_output
```

JasperGold Apps Command Reference Manual

General Commands

```
| bbox_input | bbox_output | mpram_port | undriven
| flop | latch | internal | stopat | reset_x
| x_value | x_default_value | x_md_abstract]
[-bit_blast]
[-return (text | id | none)]
[-spec_delay <spec_delay_value>]
[-imp_delay <imp_delay_value>]
[-spec_condition {<spec_cond_expr>}]
[-spec_condition_delay <spec_cond_delay_value>]
[-imp_condition {<imp_cond_expr>}]
[-imp_condition_delay <imp_cond_delay_value>]
[-clock {<clocking_event>}]
[-disable {<disable_cond_expr>}]
[-cut (both | both_no_init | spec | imp)
|-stopat (both | spec | imp)
|-helper]
[-set_level <N>]
[-tag {<tag_name>}]
[-task {<task_name>}]
[-exclude {<signal_list_to_exclude>}]
[-speculative (on | off)]
)
|
| (-spec {<spec_regexp_or_signal_list>}
|   [-imp {<imp_regexp_or_signal_list>}])
[-regexp
|   [-verbose] [-exclude {<signal_list_to_exclude>}]]
[-bit_blast] [-inverse]
[-return (id | text | none)]
[-spec_delay <spec_delay_value>]
[-imp_delay <imp_delay_value>]
[-spec_condition {<spec_cond_expr>}]
[-spec_condition_delay <spec_cond_delay_value>]
[-imp_condition {<imp_cond_expr>}]
[-imp_condition_delay <imp_cond_delay_value>]
[-clock {<clocking_event>}]
[-disable {<disable_cond_expr>}]
[-cut (both | both_no_init | spec | imp)
|-stopat (both | spec | imp)
|-helper
|-connect]
[-set_level <N>]
[-tag {<tag_name>}]
[-task {<task_name>}]
[-inverse]
[-init]
[-speculative (on | off)]
)
|
| (-levelize_helpers
|   [ (-property <property_name>+
|       | -outputs <output_signal_name>+)
|     [-tag {<tag_name>}]
```

JasperGold Apps Command Reference Manual

General Commands

```
[ -task {<task_name>} ]
)
| (-hierarchy
  -spec {<spec_regex_or_hierarchy_list>}
  -imp {<imp_regex_or_hierarchy_list>}
  [-regexp]
)
|-map_file <map_file_name>
```

Signal Unmapping

```
check_sec -unmap
  ( -id <mapping_id_list>
    [-force]
    | ( -spec {<spec_regex_or_signal_list>}
      |-imp {<imp_regex_or_signal_list>} )
      [-regexp [-verbose]]
      [-mapping_type ( connection_all | connection_primary_input
                      | connection_bbox_output | connection_undriven
                      | connection_stopat | connection_x_value
                      | connection_x_default_value | connection_misc
                      | ver_target_all | ver_target_primary_output
                      | ver_target_mparam_port | ver_target_bbox_input
                      | ver_target_misc | cutpoint_all | cutpoint_both
                      | cutpoint_spec | cutpoint_imp | stopat_all
                      | stopat_both | stopat_spec | stopat_imp
                      | helper | init ) ]
      [ (-instance <instance_full_path>)
        | (-of_signal {<regexp_or_signal_list>}
          [ -spec_regex [-verbose]
            | -imp_regex [-verbose]] )
      ]
      [-proof_status (proven | cex | unprocessed | undetermined)+]
      [-signal_type ( input | output | primary_input | primary_output
                      | bbox_input | bbox_output | mparam_port | undriven
                      | flop | latch | internal | stopat | reset_x
                      | x_value | x_default_value | x_md_abstract) ]
      [-tag {<tag_name>} ]
      [-task {<task_name>} [-include_global_mapping]]
      [-force]
    )
    | ( -hierarchy
      -id <hierarchy_aliasing_rule_id_list>
      [-force]
    )
  )
check_sec -remove_rule -id <rule_id_list>
```

[Controlling Automatic Init Mapping](#)

```
check_sec -auto_map_reset_x_values (on | off)
```

[Setting Mapping Context](#)

```
check_sec -set_context
  ( -id <mapping_id_list>
  | ( -spec {<spec_regex_or_signal_list>}
    | -imp {<imp_regex_or_signal_list>} )
    [-regexp [-verbose]]
    [-mapping_type ( connection_all | connection_primary_input
                     | connection_bbox_output | connection_undriven
                     | connection_stopat | connection_x_value
                     | connection_x_default_value | connection_misc
                     | ver_target_all | ver_target_primary_output
                     | ver_target_mpram_port | ver_target_bbox_input
                     | ver_target_misc | cutpoint_all | cutpoint_both
                     | cutpoint_spec | cutpoint_imp | stopat_all
                     | stopat_both | stopat_spec | stopat_imp
                     | helper | init ) ]
    [ (-instance <instance_full_path>)
    | (-of_signal {<regexp_or_signal_list>}
      [ -spec_regex [-verbose]
      | -imp_regex [-verbose]] )
    ]
    [-proof_status (proven | cex | unprocessed | undetermined)+]
    [-signal_type ( input | output | primary_input | primary_output
                    | bbox_input | bbox_output | mpram_port | undriven
                    | flop | latch | internal | stopat | reset_x
                    | x_value | x_default_value | x_md_abstract) ]
    [-tag {<tag_name>}]
    [-task {<task_name>} [-include_global_mapping]]
    |-set_default
    |-clear_default
  )
  [-spec_delay <spec_delay_value>]
  [-imp_delay <imp_delay_value>]
  [-spec_condition {<spec_cond_expr>}]
  [-spec_condition_delay <spec_cond_delay_value>]
  [-imp_condition {<imp_cond_expr>}]
  [-imp_condition_delay <imp_cond_delay_value>]
  [-clock {<clocking_event>}]
  [-disable {<disable_cond_expr>}]
```

JasperGold Apps Command Reference Manual

General Commands

Setting Mapping Type

```
check_sec -set_mapping_type -id <mapping_id_list>
  [ -cut (both | both_no_init | spec | imp)
  | -stopat (both | spec | imp)
  | -helper]
  [-set_tag <tag_name>]
  [-speculative (on | off)]
```

Enabling and Disabling Rules and Mapping

```
check_sec (-enable | -disable)
  [-force]
  ( (-mapping |-rule) -id {<mapping_or_rule_id_list>}
  | ( -spec {<spec_regexp_or_signal_list>}
  | -imp {<imp_regexp_or_signal_list>} )
  [-regexp [-verbose]]
  [-mapping_type ( connection_all | connection_primary_input
  | connection_bbox_output | connection_undriven
  | connection_stopat | connection_x_value
  | connection_x_default_value | connection_misc
  | ver_target_all | ver_target_primary_output
  | ver_target_mpram_port | ver_target_bbox_input
  | ver_target_misc | cutpoint_all | cutpoint_both
  | cutpoint_spec | cutpoint_imp | stopat_all
  | stopat_both | stopat_spec | stopat_imp
  | helper | init ) ]
  [ (-instance <instance_full_path>)
  | (-of_signal {<regexp_or_signal_list>}
  [ -spec_regexp [-verbose]
  | -imp_regexp [-verbose]] )
  ]
  [-proof_status (proven | cex | unprocessed | undetermined)+]
  [-signal_type ( input | output | primary_input | primary_output
  | bbox_input | bbox_output | mpram_port | undriven
  | flop | latch | internal | stopat | reset_x
  | x_value | x_default_value | x_md_abstract) ]
  [-tag {<tag_name>}]
  [-task <task_name>] [-include_global_mapping]
  [-force]
  )
```

Marking Signals

```
check_sec -mark -forbid_init -sig_list {<signals_list>}
check_sec -unmark -forbid_init -sig_list {<signals_list>}
```

JasperGold Apps Command Reference Manual

General Commands

Waiving Signals

```
check_sec -waive -waive_signals {<signals_list>}
  |-unwaive_signals {<signals_list>}
  |-list_waived
```

Locking and Unlocking the Environment

```
check_sec -lock_env [-error_onViolation]
  |-unlock_env
```

Prove

```
check_sec -prove
  [-task <task_name> |-property <property_name_tcl_list> [-regexp]]
  [-no_auto] [-order_targets (by_gates | by_flops | none)]
  [ (-rule_id {<rule_id_list>} |-mapping_id {<mapping_id_list>})
    | (-spec {<spec_regex_or_signal_list>}
      |-imp {<imp_regex_or_signal_list>})
    [-regexp [-verbose]])
  [-signal_type ( input | output | primary_input | primary_output
                  | bbox_input | bbox_output | mpram_port | undriven
                  | flop | latch | internal | stopat | reset_x
                  | x_value | x_default_value | x_md_abstract)
    [-mapping_type ( connection_all | connection_primary_input
                     | connection_bbox_output | connection_undriven
                     | connection_stopat | connection_x_value
                     | connection_x_default_value | connection_misc
                     | ver_target_all | ver_target_primary_output
                     | ver_target_mpram_port | ver_target_bbox_input
                     | ver_target_misc | cutpoint_all | cutpoint_both
                     | cutpoint_spec | cutpoint_imp | stopat_all
                     | stopat_both | stopat_spec | stopat_imp
                     | helper | init ) ]
    [ (-instance <instance_full_path>)
      | (-of_signal {<regexp_or_signal_list>}
        [-spec_regex [-verbose]
        |-imp_regex [-verbose]])
    ]
    [-proof_status (proven | cex | unprocessed | undetermined)+]
    [-tag {<tag_name>}]
  ]
  [-prove_opts {<prove_options_list>}]
  [-time_limit <time_budget>]
  [-levelize_helpers
    [-levels <level_to_prove>+])
    [-tag {<tag_name>}]
    [-bg]
  [-no_auto]
```

JasperGold Apps Command Reference Manual

General Commands

```
[-silent | -verbosity <N>]
[-assumption_lifting]
[-suppress_traces]
[-cex_threshold <number_of_failures>]
```

Prove with Cutpoint Refinement

```
check_sec -prove -autoprove_mode (sscp | dscp | none)
[-task <task_name> |-property <property_name_tcl_list> [-regexp]]
[-cutpoints_on_latches]
[-cutpoints_on_internal_signals]
[-max_number_of_cutpoints <N>=0>]
[-bg]
[-silent | -verbosity <N>]
```

Generating Verification

```
check_sec -generate_verification
[-task <task_name>]
[-assume_equiv (by_assume | by_driver)]
```

Sign-Off Report

```
check_sec -signoff
[-task <task_name>]
[-file <file_name>]
[-silent]
[-summary]
[-detailed]
[-waive_x]
[-waive_consistency]
[-cp_bound [-include_cex_bound]]
```

Utility Switches

```
check_sec (-get_spec_top_instance | -get_imp_top_instance)
```

Setup

```
-setup
  [-global_setup_file global_setup_file]
  -spec_dut spec_dut_instance_name_list
    (-spec_analyze_opts {analyze_switches_and_target})*
      [-spec_elaborate_opts {elaborate_flags_list}]
      [-spec_top {spec_top_name}]
  -imp_dut imp_dut_instance_name_list
    (-imp_analyze_opts {analyze_switches_and_target})*
      [-imp_elaborate_opts {elaborate_flags_list}]
      [-imp_top {imp_top_name}]
  (-ipk_opts {ipk_switches_and_arguments})+
  [-setup_file setup_file]
  [-no_auto_mapping]
  [-spec_upf_file upf_file |-spec_cpf_file cpf_file]
  [-imp_upf_file upf_file |-imp_cpf_file cpf_file]
    [-power_control_signals {power_control_signals_list}]]
  [-map_file map_file_name]
  [-exclude_internal]
  [-include_cells]
  [-auto_map_reset_x_values [-map_one_side_reset_x]]
  [-generate_named_x_wires [-auto_map_x_values]]
```

Set up the specified spec and imp designs and perform auto mapping for primary_input, primary_output, bbox_input, bbox_output, undriven, stopat, and mram_port signals.

- Use `-global_setup_file` to specify a Tcl script with your global environment and settings. The tool sources this script before starting the setup step.
- Use `-spec_dut` to specify a list of one or more spec instances that will be used in the SEC analysis.
 - Use `-spec_analyze_opts` with a list of switches, their parameters, and an HDL or property file to specify the analyze command for the spec side.
 - Use a single string.

JasperGold Apps Command Reference Manual

General Commands

-setup (Continued)

- The contents of this string should be exactly what you would pass to an `analyze` command for the HDL or property file.
- You can use `-spec_analyze_opts` multiple times with the `-setup` switch to analyze several HDL or property files.
- Use `-spec_elaborate_opts` to pass the specified switches to the `elaborate` command for the spec side.
- Use `-spec_top` to specify the top instance when it is different from the instance specified with `-setup -spec_dut`. Use this option when you want to run SEC on a submodule of the elaborated design against some implementation of that submodule.
- Use `-imp_dut` to specify a list of one or more imp instances that will be used in the SEC analysis.
 - Use `-imp_analyze_opts` with a list of switches, their parameters, and an HDL or property file to specify the `analyze` command for the imp side.
 - Use a single string.
 - The contents of this string should be exactly what you would pass to an `analyze` command for the HDL or property file.
 - You can use `-imp_analyze_opts` multiple times with the `-setup` switch to analyze several HDL or property files.
 - Use `-imp_elaborate_opts` to pass the specified switches to the `elaborate` command for the imp side.
 - Use `-imp_top` to specify the imp top instance when it is different from the instance specified with `-setup -imp_dut`. This switch requires the `-spec_top` switch.
- Use `-ipk_opts` with a list of switches and their parameters to specify the `ipk` command.



- Use a single string.

-setup (Continued)

- The contents of this string should be exactly what you would pass to an `ipk` command.
 - You can use `-ipk_opts` multiple times with the `-setup` switch to analyze multiple IPKs.
 - Use `-setup_file` to specify a setup file to run after `analyze`, `elaborate`, and `miter` creation.
 - Use `-no_auto_mapping` to specify that you do not want the tool to perform auto mapping after signals collection.
 - Use `-spec_upf_file` or `-spec_cpf_file` to specify UPF or CPF files to load for low power treatment for the spec design.
 - Use `-imp_upf_file` or `-imp_cpf_file` to specify UPF or CPF files to load for low power treatment for the imp design.
 - Use `-power_control_signals` to specify a list of power control signals. For each of these signals, the tool generates a disabled assumption that states that the signal is constant 0. Later, you can enable these assumptions to activate low power control.
 - Use `-map_file` to run a user-defined Tcl file after setup and miter creation completes. The commands in this file specify user-defined mapping.
 - Use `-exclude_internal` to avoid analyzing wires that are not in the top module. Use this switch to reduce complexity.
 - Use `-include_cells` to analyze modules encapsulated in the ``celldefine`/`endcelldefine`. To reduce complexity, the tool does not analyze these modules by default.
-

-setup (Continued)

- Use `-auto_map_reset_x_values` to instruct SEC to automatically map flop and latch pairs that are nonresettable or uninitialized. And use `-map_one_side_reset_x` if you would also like to automatically map flops and latches that are nonresettable or uninitialized on just one side (spec or imp).

Note: In this mode, SEC automatically monitors reset updates and updates the init mappings accordingly. However, for this command to be effective, a reset command must follow it.

- Use `-generate_named_x_wires` to instruct SEC to generate names for the X-values in X-handling mode in elaboration. In this mode, Xs have the following naming convention:

- X-nets NOT in the default case:

```
<instance prefix>.:x_<line num>_<bit  
index>_<width>_<uniqueness ID>_<rtl identifier context>
```

- X-nets in the default case:

```
<instance prefix>.:x_dc_<line num>_<bit  
index>_<width>_<uniqueness ID>_<rtl identifier context>
```

The `rtl-identifier-context` is the left-hand side variable name (in the case of assignments) and array name (in the case of implicit Xs generated for-out-of bound indexing of arrays).

JasperGold Apps Command Reference Manual

General Commands

-setup (Continued)

In addition they have the following attributes:

- ❑ X-nets *not* in the default case are marked with the attribute `x_value`, which means that `-list -signal_type x_value` lists them and that, when mapped, they become assumptions.
- ❑ x-nets in the default case have the attribute `x_default_value`.
- ❑ Use `-auto_map_x_values` to set up auto-mapping for signals with X values. The following commands provide other options for mapping these signals:

```
check_sec -map -auto -signal_type x_value  
check_sec -map -auto -signal_type x_default_value
```

The setup auto-mapping does *not* map these signals. To specify that you want to map these signals, use a command such as the following:
`check_sec -map -auto -signal_type x_value`

Note: For the `-setup` sub-command,

- The `-spec` and `-imp` switches are mandatory.
- The switches for specifying `analyze` and `elaborate` are optional.

-set_dut_instance

```
-spec spec_dut_instance_name_list -imp imp_dut_instance_name_list
```

Change the previously specified DUT instances to the instances specified with this command.

Note: `-set_dut_instance` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments.

JasperGold Apps Command Reference Manual

General Commands

Clear

```
-clear [all | analyze | rule | mapping | proof]
```

Clear SEC settings.

- Use `all` to clear all SEC settings in the current session.
- Use `analyze` to clear all `analyze`, `elaborate`, `rule`, `mapping`, and `proof` information.
- Use `rule` to clear all rules, their mappings, and their proof properties.
- Use `mapping` to clear all mappings and their proof properties.
- Use `proof` to clear all proof information.

If you specify none of these options, the command clears all SEC settings in the current session.

Interface Check

```
-interface
  [-mapped | -unmapped]
  [-spec | -imp]
  [-signal_type ( primary_input | primary_output
                  | bbox_input | bbox_output | undriven
                  | setup_stopat | reset_x | x_value
                  | x_default_value | x_lpv_value | x_md_abstract )]
  [-instance instance_full_path]
  [-task task_name]
```

Run the interface check to determine the outcome of mapping.

Use this command without additional switches to report, for each signal type, whether the check passed or failed, and for failures, report how many signals of the specified type in the spec and imp are unmapped.

Or specify additional switches defined below to refine the report.

- Use `-mapped` to list mapped signals.
- Use `-unmapped` to list unmapped signals. This is the default if you do not specify `-mapped`.
- Use `-spec` to list only signals from the spec design. This is the default if you do not specify `-imp`.
- Use `-imp` to list only signals from the imp design.
- Use `-signal_type` to list only signals of the specified type.
- Use `-instance` to list only signals that are in the specified instance.
- Use `-task` to specify a task other than the current task.

List Switches

This section describes the many lists available with `check_sec`. Use these commands to list signals, hierarchy mapping, signal mapping, disabled cutpoints, rules, or rule mapping.

Note: The `-signal`, `-mapping`, `-rule`, and `-rule_mapping` switches are mutually exclusive. If you specify none of these switches (`check_sec -list`), the command lists signals.

Signals List

```
-list
[ [-signal]
  [-signal_type ( input | output | primary_input | primary_output
                  | bbox_input | bbox_output | mparam_port
                  | undriven | flop | latch | internal | stopat
                  | reset_x | x_value | x_default_value | x_md_abstract) ]
  [-spec | -imp]
  [ -instance instance_full_path
    | -name {regexp_or_signal} [-reg [-verbose]] ]]
```

`-list -signal` or just `-list` with no additional switches lists all signals.

Note: The `-name` and `-instance` switches are mutually exclusive.

- Use `-spec` or `-imp` to indicate whether the `-instance` or `-name` argument specifies an instance or signal(s) on the spec or imp side.
- Use `-name` to define the set of signal names on the specified side that you want to list.
 - Use `-regexp` to specify that the `-name` argument is a regular expression; otherwise, it is the name of a specific signal. Include the `-verbose` switch to print info messages detailing the regexp matching operation.
 - Use `-instance` to list all signals that are in the specified instance.
 - Use `-signal_type` to list only signals of the specified type.

Hierarchy Mapping Rule List

```
-list -mapping -hierarchy [-return (text | id)]
```

`-list -mapping -hierarchy` lists all hierarchy mapping rules.

- Use `-return text` to return the result as rules text. (This is the default behavior.)
- Use `-return id` to return the result as hierarchy mapping rules IDs.

JasperGold Apps Command Reference Manual

General Commands

Signal Mapping List

```
-list -mapping
  [-mapping_type ( connection_all | connection_primary_input
                    connection_bbox_output | connection_undriven
                    connection_stopat | connection_x_value
                    connection_x_default_value | connection_misc
                    ver_target_all | ver_target_primary_output
                    ver_target_mpram_port | ver_target_bbox_input
                    ver_target_misc | cutpoint_all | cutpoint_both
                    cutpoint_spec | cutpoint_imp | stopat_all
                    stopat_both | stopat_spec | stopat_imp
                    helper | init ) ]
  [-signal_type ( input | output | primary_input | primary_output
                    bbox_input | bbox_output | mpram_port
                    undriven | flop | latch | internal | stopat
                    reset_x | x_value | x_default_value | x_md_abstract )]
  [ (-instance {instance_full_path})
    | (-of_signal {regexp_or_signal_list}
      [ -spec_regex [-verbose]
        | -imp_regex [-verbose]] )
  ]
  [-tag {tag_name}]
  [-task {task_name} [-include_global_mapping]]
  [-return (text | id | property)]
  [-speculative (on | off)]
  [-proof_status (proven | cex | unprocessed | undetermined)+]
```

-list -mapping with no additional switches lists all mapping pairs.

Note: The -of_signal and -instance switches are mutually exclusive.

- Use -mapping_type to list only mapping pairs of the specified mapping type.
- Use -signal_type to list only mapping pairs with signals of the specified signal type.

JasperGold Apps Command Reference Manual

General Commands

-list -mapping (Continued)

- Use `-of_signal` to define the set of signal names whose mappings you want to list.
 - Use `-regexp` to specify that the `-of_signal` argument list is a regular expression list; otherwise, it is a list of specific names of signals, bits, or bit-vector expressions.
 - Use `-spec` or `-imp` to indicate whether the `-of_signal` argument list specifies regular expression list on the spec or imp side. the default is spec side.
 - Use `-verbose` to print info messages detailing the regexp matching operations.
 - Use `-instance` to specify that you want to list only mapping pairs of signals that are in the specified instance.
 - Use `-tag` to list only mapping pairs with the specified tag.
 - Use `-return` to specify the return type.
 - Use `-return text` to return the result as mapping text. (This is the default behavior.)
 - Use `-return id` to return the result as mapping IDs.
 - Use `-return property` to return the result as the names of all the properties of the selected mapping pairs.
 - Use `-speculative` to filter mapping entries according to their speculative status. The speculative status specifies that the cutpoint or helper was added by some heuristic, and its failure does not necessarily point to a bug. The default value of mapping entries is `on`. Since non-speculative cutpoints and helper assertions are not expected to fail, you should remove them only after careful consideration. Therefore, you must use the `-force` switch to disable or unmap them.

Note: You must debug any failure of non-speculative mapping entries.
 - Use `-proof_status` to filter mapping entries according to their proof status. This switch accepts one or more of the following arguments: `proven`, `cex`, `unprocessed`, `undetermined`.
-

JasperGold Apps Command Reference Manual

General Commands

Marked Signals List

```
-list -marked -forbid_init [-imp]
```

List the signals marked with `forbid_init`.

Cutpoint Status List

```
-list -mapping (-unguaranteed_cutpoint | -unproved_cutpoint)  
[-speculative (on | off)]
```

`-list -mapping` with one of the following switches lists cutpoint status.

- Use `-unguaranteed_cutpoint` to list all the cutpoints that failed (have CEX status).
- Use `-unproved_cutpoint` to list all the cutpoints that have not been proved.
- Use `-speculative` to filter cutpoints by their speculative status.

Note: The switches `-unguaranteed_cutpoint` and `-unproved_cutpoint` are mutually exclusive. Furthermore, you cannot use them with any other `-list -mapping` switches.

Disabled Cutpoints List

```
-list -mapping -violating_cutpoint { all | clock_signal | init_diff  
| assumption_comb_coi)  
[-force]
```

`-list -mapping -violating_cutpoint` with one of the following arguments lists cutpoints that were disabled due to rule violations.

- Use `all` to list all violating cutpoints.
- Use `clock_signal` to list cutpoints on clocks.
- Use `init_diff` to list cutpoints on pairs with different initial values.
- Use `assumption_comb_coi` to list cutpoints where the cut signal is under a combinational influence of a user-specified assumption. That is, there is a combinational trace from the cut signal to a signal that is used in the user-specified assumption.
- Use `-force` to disable non-speculative mapping entries. That is, mapping entries that were mapped with the `-speculative off` switch.

Note: Do not combine `-violating_cutpoint` with any other `-list -mapping` switches.

Disabled Init Mapping List

```
-list -mapping -violating_init_mapping
```

Lists init mapping pairs that were disabled due to a different init value.

Note: As long as the init values remain different, these mapping pairs will be disabled automatically.

Rules List

```
-list -rule [-return (text | id | property)]
```

-list -rule with no additional switches lists all mapping rules.

- Use -return text to return the result as rule text. This is the default behavior.
- Use -return id to return the result as rule IDs.
- Use -return property to return the result as the names of all the properties of the mapping pairs generated from the rules.

Rule Mapping List

```
-list -rule_mapping rule_id [-return (text | id | property)]
```

-list -rule_mapping with no additional switches lists all mappings generated from the rule with the specified ID.

- Use -return text to return the result as mapping text. This is the default behavior.
- Use -return id to return the result as mapping IDs.
- Use -return property to return the result as the names of all the properties of the mapping pairs generated from the rule with the specified ID.

Mapping Info

```
-mapping_info
( -id mapping_id_list
  | ( -spec {spec_regex_or_signal_list}
      | -imp {imp_regex_or_signal_list}
        [-regexp [-verbose]])
)
[-signal_type ( input | output | primary_input | primary_output
  | bbox_input | bbox_output | mpram_port | undriven
  | flop | latch | internal | stopat | reset_x
  | x_value | x_default_value | x_md_abstract))
[-mapping_type ( connection_all | connection_primary_input
  | connection_bbox_output | connection_undriven
  | connection_stopat | connection_x_value
  | connection_x_default_value | connection_misc
  | ver_target_all | ver_target_primary_output
  | ver_target_mpram_port | ver_target_bbox_input
  | ver_target_misc | cutpoint_all | cutpoint_both
  | cutpoint_spec | cutpoint_imp | stopat_all
  | stopat_both | stopat_spec | stopat_imp
  | helper | init ) ]
[ (-instance {instance_full_path})
  | (-of_signal {regexp_or_signal_list}
    [ -spec_regex [-verbose]
      | -imp_regex [-verbose]])]
]
[-proof_status (proven | cex | unprocessed | undetermined)+]
[-tag {tag_name}]
[-task {task_name} [-include_global_mapping]]
[-get_type |-get_task |-get_tag |-is_speculative]
```

List info for specific mapping pairs.

This command returns, for each specified mapping pair, the following tuple:

```
{ mapping pair ID, spec_signal, imp_signal,
  generating rule ID, assert/assume expression,
  property name, standard textual representation,
  is enabled, is verification pair, is inverse, is speculative }
```

- Use **-id** to specify the list of IDs of the specific mapping pairs.
- Use **-spec** to define the set of signal name(s) on the spec side whose mapping info you want to list.
- Use **-imp** to define the set of signal name(s) on the implementation side whose mapping info you want to list.

JasperGold Apps Command Reference Manual

General Commands

-mapping_info (Continued)

- Use `-regexp` to specify that the `-spec` or `-imp` argument list is a regular expression list; otherwise, it is a list of specific names of signals, bits, or bit-vector expressions. Include the `-verbose` switch to print info messages detailing the regexp matching operation.
 - Use `-signal_type` to list only mappings of signals of the specified type.
 - Use `-mapping_type` to list only mapping pairs of the specified mapping type.
 - Use `-of_signal` to define the set of signal names whose mappings you want to list.
 - Use `-spec_regexp` or `-imp_regexp` to specify that the `-of_signal` argument is a regular expression list on the spec or imp side. The default is spec side; otherwise, it is a list of specific names of signals, bits, or bit-vector expressions.
 - Use `-verbose` to print info messages detailing the regexp matching operations.
 - Use `-instance` to specify that you want to list only mapping pairs of signals that are in the specified instance.
 - Use `-proof_status` to list mapping entries according to their proof status. This switch accepts one or more of the following arguments: proven, cex, unprocessed, undetermined.
 - Use `-tag` to list only mappings with the specified tag.
 - Use `-task` to list only the task-related mappings without considering global or other tasks pairs.
 - If the `-task` switch was not specified, it will list only global mapping pair(s).
 - Use `-include_global_mapping` with `-task` to list both the global and task-specific mapping pair(s).
-

JasperGold Apps Command Reference Manual

General Commands

-mapping_info (Continued)

- Use -get_type to get the mapping type of the specified mapping entries.
- Use -get_task to get the task of the specified mapping entries.
- Use -get_tag to get the mapping tag of the specified mapping entries.
- Use -is_speculative to get mapping speculative status. The speculative status specifies that the cutpoint or helper was added by some heuristic, and its failure does not necessarily point to a bug.

Note: You must debug any failure of non-speculative mapping entries.

Hierarchy Mapping Rule Info

```
-mapping_info -hierarchy -id {hierarchy_aliasing_rule_id_list}
```

List information for specific hierarchy mapping rules.

This command returns, for each specified mapping rule ID, the following tuple:

```
{ hierarchy mapping rule ID, spec hierarchy expression, imp hierarchy expression, isRegEx }
```

Use -id to specify the list of IDs of the specific hierarchy mapping rules.

Rule Info

```
-rule_info -id {rule_id_list}
```

List info for specified rules.

This command returns, for each specified mapping pair, the following tuple:

```
{ rule_text, rule_source, is_enabled }
```

This command Tcl lists and lists of IDs for specific rules.

Signal Mapping

```
-map
  (-auto
    [-spec {spec_instance_list}
     [-imp {imp_instance_list}]]
    [-signal_type ( input | output | primary_input
                    | primary_output | bbox_input | bbox_output
                    | mpram_port | undriven | flop
                    | latch | internal | stopat | reset_x
                    | x_value | x_default_value | x_md_abstract)]
    [-bit_blast]
    [-return (text | id | none)]
    [-spec_delay spec_delay_value]
    [-imp_delay imp_delay_value]
    [-spec_condition {spec_cond_expr}]
    [-spec_condition_delay spec_cond_delay_value]
    [-imp_condition {imp_cond_expr}]
    [-imp_condition_delay imp_cond_delay_value]
    [-clock {clocking_event}]
    [-disable {disable_cond_expr}]
    [-cut (both | both_no_init | spec | imp)
      |-stopat (both | spec | imp)
      |-helper]
    [-set_level N]
    [-tag {tag_name}]
    [-task {task_name} [-include_global_mapping]]
    [-exclude {signal_list_to_exclude}]
    [-inverse] [-init]
    [-speculative (on | off)])
  )
  | (
    (-spec {spec_regexp_or_signal_list}
      [-imp {imp_regexp_or_signal_list}])
    [-regexp [-verbose] [-exclude {signal_list_to_exclude}]]
    [-bit_blast] [-inverse]
    [-return (id | text | none)]
    [-spec_delay spec_delay_value]
    [-imp_delay imp_delay_value]
    [-spec_condition {spec_cond_expr}]
    [-spec_condition_delay spec_cond_delay_value]
    [-imp_condition {imp_cond_expr}]
    [-imp_condition_delay imp_cond_delay_value]
    [-clock {clocking_event}]
    [-disable {disable_cond_expr}])
```

JasperGold Apps Command Reference Manual

General Commands

```
-map (Continued)
    [ -cut (both | both_no_init | spec | imp)
      | -stopat (both | spec | imp)
      | -helper
      | -connect]
    [-set_level N]
    [-tag {tag_name}]
    [-task {task_name} [-include_global_mapping]]
    [-inverse] [-init]
    [-speculative (on | off)]
  )
  | (-levelize_helpers
    [ (-property property_name+
        | -outputs output_signal_name+)
    [-tag {tag_name}]
    [-task {task_name} [-include_global_mapping]]
  )
  | (-hierarchy
    -spec {spec_regexp_or_hierarchy_list}
    -imp {imp_regexp_or_hierarchy_list}
    [-regexp]
  )
  | -map_file map_file_name
```

Map specification signals to implementation signals.

Auto-Mapping Switches

Use `-auto` to automatically map primary inputs and outputs, inputs and output signals of black boxes, and undriven signals that have the same names in the specification and implementation.

- Use `-spec` with `-auto` to specify the spec instance list whose signals are to be auto-mapped.
- Use `-imp` with `-auto` to specify the imp instance list whose signals are to be auto-mapped. If you use `-spec` but not `-imp`, the tool attempts to map the spec instance signals to imp signals in an instance of the same name, taking into account the imp prefix. If you use both `-spec` and `-imp` switches, the tool considers pairs of instances by matching each spec instance from the given list to the imp instance in the same position of the imp list. Therefore, the two supplied spec and imp lists must be of the same length.

-map (Continued)

- Use `-signal_type` and a signal type to auto-map only signals of the specified type.
- Use `-bit_blast` to create a separate mapping pair for each mapped bit. By default, mapping of vector signals or bit ranges creates atom pairs with each mapping a bit range in a spec signal to a bit range in an imp signal.
- Use `-tag` to add a tag (property name prefix) to the mapping(s).
- Use `-exclude` to specify a list of signals you want to exclude from auto-mapping.
- Use `-speculative on` to mark a mapping entry as speculative and `-speculative off` to mark a mapping entry as non-speculative. You can only mark cutpoints and helper assertions as speculative. The speculative status specifies that the cutpoint or helper was added by some heuristic, and its failure does not necessarily point to a bug.

Note: You must debug any failure of non-speculative mapping entries.

Auto-Levelize Helpers Mapping Switches

Use `-levelize_helpers` to automatically map signals that have the same names in the specification and implementation, and tag each mapping pair based on levels of distance from outputs.

- Use `-property` to specify the properties whose COIs will be used for the leveling calculation.
- Use `-outputs` to specify the outputs whose COIs will be used for the leveling calculation.
- Use `-tag` to add a tag (property name prefix) to the generated mapping(s). This tag is in addition to the `-auto_levelizing` tag described above.
- Use `-inverse` to create inverted mapping.
- Use `-init` to create init mapping (that is, bound 1 equivalence assumptions).

JasperGold Apps Command Reference Manual

General Commands

-map (Continued)

Hierarchy Mapping (Aliasing)

- Use `-hierarchy` to add hierarchy mapping rules. These hierarchy mapping rules help the tool automatically map primary inputs and outputs, input and output signals of black boxes, and undriven signals that have the same names but are under different hierarchy names in the spec and imp. Each time the auto mapping tries to find a matching imp signal of some spec signal, it applies these rules to it to find the matching imp signal. It applies the rules one by one in the same order you added them. Use `-spec` to define the name(s) or sub-name(s) of the spec hierarchies.
- Use `-imp` to define the name(s) or sub-name(s) of the imp hierarchies.
- Use `-regexp` to specify that the `-spec` and `-imp` argument lists are regular expression lists; otherwise, they are lists of specific hierarchy names.

Note:

- Any imp regular expression may contain references to group values matched by the spec regular expression: `$0, $1, $2, $3, ..., $9`
- Use `(?:RE)` pattern to group a sub-pattern but not capture the result in a reference (`$0, ..., $9`).

Manual Mapping Switches

You can also manually map the spec to the imp as follows.

- Use `-spec` to define the name(s) of the spec signals to map. If you also use `-regexp`, the tool uses the values of the matched groups used in the argument `{imp_regex_or_signal}`.
 - Use `(?:RE)` pattern to group a sub-pattern but not capture the result in a reference (`$0, ..., $9`).
 - The signal may be a bit-vector signal. In such cases, the imp signal must be a bit-vector signal of the same length. Specify sub-vectors of such bit-vector signals with an expression of the form `[left_index:right_index]`, for example, `[9:13]` or `[3:0]`. The bits are mapped according to their order from the right vector bound to the left one.

-map (Continued)

- ❑ Include `-verbose` to print info messages detailing the regexp matching operation.
- Use `-imp` to define the name(s) of the imp signals, bits, or bit-vector expressions.
 - ❑ If you use `-spec` but not `-imp`, the tool attempts to map the spec signals to imp signals of the same name, considering the imp prefix.
 - ❑ If you use both `-spec` and `-imp` switches, the tool considers pairs of `regexp_or_signal` by matching each spec `regexp_or_signal` from the given list to the imp `regexp_or_signal` in the same position of the imp list. Therefore, the two supplied spec and imp lists must be of the same length.
 - ❑ If you use `-regexp` with `-imp`, `imp_regexp_or_signal` is a regular expression. It may contain references to group values matched by `{spec_regexp_or_signal}`: \$0, \$1, \$2, \$3, ..., \$9
 - ❑ You may use multi-bit vector expressions as in `{spec_regexp_or_signal}`.
 - ❑ Include the `-verbose` switch to print info messages.
- Use `-bit_blast` to create a separate mapping pair for each mapped bit. By default, mapping of vector signals or bit ranges creates atom pairs with each mapping a bit range in a spec signal to a bit range in an imp signal.
- Use `-inverse` to specify that the mapped signals are the inverse (not) of each other. The tool generates assume or assert properties accordingly.
- Use `-set_level` to map signals with a specified level. You can then use this level to perform an iterative levelize prove on signals.
- Use `-tag` to add a tag (property name prefix) to the generated mapping(s).
- Use `-exclude` with `-regexp` to specify a list of signals you want to exclude from mapping.
- Use `-inverse` to create inverted mapping.

-map (Continued)

- Use `-init` to create init mapping, that is, bound 1 equivalence assumptions.
- Use `-speculative on` to mark a mapping entry as speculative and `-speculative off` to mark a mapping entry as non-speculative. You can only mark cutpoints and helper assertions as speculative. The speculative status specifies that the cutpoint or helper was added by some heuristic, and its failure does not necessarily point to a bug.

Note: You must debug any failure of non-speculative mapping entries.
- Use `-connect` to create connection mapping, that is, to connect the signal pair by adding equivalence assumptions on them.

Task-Related Mapping

With either auto or manual mapping, use `-task` to relate the generated mapping to a task, that is, to expose the mapping pairs as properties for the specified task only. If the signal is undriven, a primary input, a black-box output, or a stopat, it generates a global mapping pair instead of a task-specific mapping pair. Global mapping pairs can be exposed to any task.

Note: If you do not specify the `-task` switch, the tool generates global mapping pairs that can be exposed to any task.

Return Type

With either auto or manual mapping, use `-return` and one of the following arguments to return information:

- Use `id` to return the result as mapping IDs.
- Use `text` to return the result as mapping text. This is the default behavior.
- Use `none` to return nothing (empty list of strings).

Mapping Pairs as Helpers

With either auto or manual mapping, use `-helper` to create helper mapping pairs. Helper mapping pairs direct the tool to create helper assertions (see [Generating Verification](#) on page 395)

-map (Continued)

The purpose of generating helpers is to prevent any unintended change in the environment. For example, if you add a new mapping pair on a primary input or an undriven signal, the tool generates a new `-env` assumption (even if that was not your intention); however, using `-helper` prevents the tool from generating an assumption and only exposes helper assertions.

Mapping Pairs as Cutpoints

With either auto or manual mapping, use `-cut` to create cutpoint pairs.

- Use `-cut both` to add cutpoints on the signals on both sides.
- Use `-cut both_no_init` to add cutpoints on the signals on both sides without the concrete value init assumption.
- Use `-cut spec` to add a cutpoint only on the signal on the spec side.
- Use `-cut imp` to add a cutpoint only on the signal on the imp side.

Mapping Pairs as Stopats

With either auto or manual mapping, use `-stopat` to create stopat mapping pairs. During the generation phase (see [Generating Verification](#) on page 395), the tool adds stopats to mapped signals (see [“stopat”](#) on page 840) and generates an equivalence assumption on them.

- Use `-stopat both` to add stopats on the signals on both sides.
- Use `-stopat spec` to add a stopat only on the signal on the spec side.
- Use `-stopat imp` to add a stopat only on the signal on the imp side.

Temporal and Logical Context

Use the following switches to specify temporal and logical context for new mappings. This set of switches generates various combinations of temporal and logical conditions on spec and imp equivalence.

- `-spec_delay`
- `-imp_delay`
- `-spec_condition`

JasperGold Apps Command Reference Manual

General Commands

-map (Continued)

- -spec_condition_delay
- -imp_condition
- -imp_condition_delay
- -clock
- -disable



- If you do not specify delay values, the default is 0.
- All condition values default to 1 (true).
- Delay values are non-negative integers.
- The following condition expressions are valid SVA expressions, and each of them may contain signals on both sides.
 - {spec_cond_expr}
 - {imp_cond_expr}
 - {disable_cond_expr}
- {clocking_event} is negedge or posedge plus an SVA expression for clock signals.
- If you supply all of these values along with the spec_signal mapped to the imp_signal, the tool generates an expression equivalent to the following:

```
@ (clocking_event)
    disable iff (disable_cond_expr)
    (
        ( (spec_cond_expr ##spec_cond_delay_value 1) and
        (imp_cond_expr ##imp_cond_delay_value 1) )
        | ->
        ( ##spec_delay_value (spec_signal ==
$past(imp_signal, (spec_delay_value - imp_delay_value)) )
    )
```

-map (Continued)

Note:

- This expression specifies a condition on spec_signal and imp_signal that is checked on every {clocking_event} and when {disable_cond_expr} is false.
- This expression is built out of the antecedent on the left of the implication ($| \rightarrow$) and the consequent on its right.
- The equality between the mapped signals is the consequent, but it may be more than a simple equality; that is, the mapped signals may differ on the relative time when they are sampled.
- All the delays in the context expression are relative to the time of the event of the implication (the $| \rightarrow$ time).
- The spec_cond_delay_value and imp_cond_delay_value delays in the antecedent define how many clock cycles before the implication the conditions {spec_cond_expr} and {imp_cond_expr} should be true.
- {spec_cond_expr} and {imp_cond_expr} can be any Boolean expression; that is, it is not necessary for them to contain signals only from one side.
- The consequent is an equality between the mapped signals such that the spec signal is sampled spec_delay_value time cycles after the implication time and the imp signal is sampled imp_delay_value time cycles after the implication time.
- If $\text{spec_delay_value} \geq \text{imp_delay_value}$, then the consequent is of the following form:

```
( ##spec_delay_value ( spec_signal == $past(imp_signal,  
(spec_delay_value - imp_delay_value) ) )
```

Otherwise, it is of the following form:

```
( ##imp_delay_value ( $past(spec_signal, (imp_delay_value -  
spec_delay_value)) == imp_signal )
```

Thus, the equivalence is assumed or asserted after the greater time value after the implication.

- The whole implication condition is valid only if, at implication time, {disable_cond_expr} is false.

-map (Continued)

- The time frames are counted according to the {clocking_event}.

Manual Mapping with a Map File

Use `-map_file` to run a user-defined Tcl file that specifies mapping.

Signal Unmapping

```
-unmap
( -id mapping_id_list
  [-force]
  | ( -spec {spec_regex_or_signal}
      | -imp {imp_regex_or_signal} )
      [-regexp [-verbose]]
      [-mapping_type ( connection_all | connection_primary_input
                      | connection_bbox_output | connection_undriven
                      | connection_stopat | connection_x_value
                      | connection_x_default_value | connection_misc
                      | ver_target_all | ver_target_primary_output
                      | ver_target_mpram_port | ver_target_bbox_input
                      | ver_target_misc | cutpoint_all | cutpoint_both
                      | cutpoint_spec | cutpoint_imp | stopat_all
                      | stopat_both | stopat_spec | stopat_imp
                      | helper | init ) ]
      [ (-instance <instance_full_path>)
          | (-of_signal {<regexp_or_signal_list>}
              [ -spec_regex [-verbose]
                | -imp_regex [-verbose]] )
      ]
      [-proof_status (proven | cex | unprocessed | undetermined)+]
      [-signal_type ( input | output | primary_input
                      | primary_output | bbox_input
                      | bbox_input | bbox_output | mpram_port
                      | undriven | flop | latch
                      | internal | stopat | reset_x
                      | x_value | x_default_value | x_md_abstract) ]
      [-tag {tag_name}]
      [-task {task_name} [-include_global_mapping]]
      [-force]
    )
  | ( -hierarchy
      -id {hierarchy_aliasing_rule_id_list} )
      [-force]
  )
)
```

Signal Unmapping

Cancel a specified mapping pair or all mappings that involve signals that match *spec_regex_or_signal_list* or *imp_regex_or_signal_list*. The tool also cancels all properties that might have been generated from these mappings.

- Use **-id** with a list of IDs for the mapping pairs you want to cancel.
- Use **-spec** to define the set of signal name(s) on the spec side that you want to unmap.
- Use **-imp** to define the set of signal name(s) on the imp side that you want to unmap.

JasperGold Apps Command Reference Manual

General Commands

-unmap (Continued)

- ❑ Use `-regexp` to specify that the `-spec` or `-imp` argument list is a regular expression list; otherwise, it is a list of specific names of signals, bits, or bit-vector expressions. Include the `-verbose` switch to print info messages detailing the `regexp` matching operation.
- ❑ Use `-mapping_type` to unmap the mapping pairs of the specified mapping type.
- ❑ Use `-of_signal` to define the set of signal names whose mappings you want to unmap.
 - Use `-spec_regexp` or `-imp_regexp` to specify that the `-of_signal` argument is a regular expression list on the spec or imp side. The default is spec side; otherwise, it is a list of specific names of signals, bits, or bit-vector expressions.
 - Use `-verbose` to print info messages detailing the `regexp` matching operations.
- ❑ Use `-instance` to specify that you want to unmap mapping pairs of signals that are in the specified instance.
- ❑ Use `-proof_status` to unmap mapping entries according to their proof status. This switch accepts one or more of the following arguments: proven, cex, unprocessed, undetermined.
- ❑ Use `-signal_type` to unmap only signals of the specified type. You can only use this switch with `-spec` or `-imp` and `-regexp`. That is, if you specify mapping IDs, all types are unmapped.
- ❑ Use `-tag` to unmap only mappings with the specified tag.

JasperGold Apps Command Reference Manual

General Commands

-unmap (Continued)

- ❑ Use `-task` to list only the task related mappings without considering global or other tasks pairs. If the `-task` switch was not specified, it will list only global mapping pair(s). And use `-include_global_mapping` with `-task` to list both the global and task-specific mapping pair(s).
- ❑ Use `-force` to disable non-speculative cutpoints. That is, cutpoints for which you used the `-speculative off` option.

Hierarchy Unmapping

Use `-hierarchy` to remove hierarchy aliasing rules, and use `-id` with a list of IDs for the hierarchy aliasing rules you want to cancel.

`-remove_rule -id rule_id_list`

Cancel the specified rules and all their associated mappings and the properties generated from them.

Controlling Automatic Init Mapping

`-auto_map_reset_x_values (on | off)`

Turn on or off the automatic mapping of pairs of uninitialized signals. When used with `on`, SEC checks all automatically identified mapping pairs and creates init mapping for pairs with uninitialized signals. Auto init mapping is completely automatic, which means it adds or removes init mapping when the reset changes. The only control you have over auto init mapping is to turn it on or off.

Setting Mapping Context

```
-set_context
( -id mapping_id_list
  | ( -spec {spec_regex_or_signal}
    | -imp {imp_regex_or_signal} )
  [-regexp [-verbose]]
  [-signal_type ( input | output | primary_input
                  | primary_output | bbox_input | bbox_output
                  undriven | flop | latch | internal
                  stopat | reset_x | x_value
                  x_default_value | x_md_abstract) ]
  [-mapping_type ( connection_all | connection_primary_input
                  connection_bbox_output | connection_undriven
                  connection_stopat | connection_x_value
                  connection_x_default_value | connection_misc
                  ver_target_all | ver_target_primary_output
                  ver_target_mparam_port | ver_target_bbox_input
                  ver_target_misc | cutpoint_all | cutpoint_both
                  cutpoint_spec | cutpoint_imp | stopat_all
                  stopat_both | stopat_spec | stopat_imp
                  helper | init ) ]
  [ (-instance {instance_full_path})
    | (-of_signal {regexp_or_signal_list}
      [ -spec_regex [-verbose]
        | -imp_regex [-verbose]] )
  ]
  [-proof_status (proven | cex | unprocessed | undetermined)+]
  [-tag {tag_name}]
  [-task {task_name} [-include_global_mapping]]
  |-set_default
  |-clear_default
)
[-spec_delay spec_delay_value]
[-imp_delay imp_delay_value]
[-spec_condition {spec_cond_expr}]
[-spec_condition_delay spec_cond_delay_value]
[-imp_condition {imp_cond_expr}]
[-imp_condition_delay imp_cond_delay_value]
[-clock {clocking_event}]
[-disable {disable_cond_expr}]
```

Set or change the temporal and logical context of a set of mappings.

- Use **-id** with a list of IDs for the mapping pairs whose context you want to change.
- Use **-spec** to define the set of signal names on the spec side whose mapping context you want to change.

JasperGold Apps Command Reference Manual

General Commands

-set_context (Continued)

- Use `-imp` to define the set of signal names on the imp side whose mapping context you want to change.
 - Use `-regexp` to specify that the `-spec` or `-imp` argument list is a regular expression list; otherwise, it is a list of specific names of signals, bits, or bit-vector expressions. Include the `-verbose` switch to print info messages detailing the regexp matching operation.
 - Use `-signal_type` to set the context only for mappings of signals of the specified type. You can only use this switch with `-spec` or `-imp` and `-regexp`. That is, if you specify mapping IDs, the context is set for all types.
 - Use `-mapping_type` to set the context only for mapping pairs of the specified mapping type.
 - Use `-of_signal` to define the set of signal names whose mappings you want to set the context for.
 - Use `-spec_regexp` or `-imp_regexp` to specify that the `-of_signal` argument is a regular expression list on the spec or imp side. The default is spec side; otherwise, it is a list of specific names of signals, bits, or bit-vector expressions.
 - Use `-verbose` to print info messages detailing the regexp matching operations.
 - Use `-instance` to specify that you want to set the context only for mapping pairs of signals that are in the specified instance.
 - Use `-proof_status` to set the context for mapping entries according to their proof status. This switch accepts one or more of the following arguments: `proven`, `cex`, `unprocessed`, `undetermined`.
 - Use `-tag` to set the context only for mappings with the specified tag.
 - Use `-task` to list only the task related mappings without considering global or other tasks pairs. If the `-task` switch was not specified, it will list only global mapping pair(s). And use `-include_global_mapping` with `-task` to list both the global and task-specific mapping pair(s).

-set_context (Continued)

- Use -set_default to set the default context. Once you set the default context, every new mapping with no explicit context has the default context. To use the default context in the setup for automatic mapping, use check_sec -setup... -setup_file....
Note: The default context applies to all mapping, including pairs of inputs and clock and reset signals.
 - Use -clear_default to clear the default context.
 - Use -spec_delay, -imp_delay, -spec_condition, -spec_condition_delay, -imp_condition, -imp_condition_delay, -clock, and -disable to specify temporal and logical context. This set of switches generates various combinations of temporal and logical conditions on spec and imp equivalence. For additional information, refer to "[Temporal and Logical Context](#)" on page 376.
-

JasperGold Apps Command Reference Manual

General Commands

Setting Mapping Type

```
check_sec -set_mapping_type -id mapping_id_list
  [ -cut (both | both_no_init | spec | imp)
    | -stopat (both | spec | imp)
    | -helper
  [-set_tag tag_name]
```

Set or change the type and tag of a set of mappings.

The tool removes the existing properties of the changed mapping pairs, and replaces them with new properties according to the new mapping type.

- Use `-id` with a list of IDs for the mapping pairs whose type you want to change.
- Use `-cut` to set the mapping type to cutpoint.
- Use `-cut both` to add cutpoints on the signals on both sides.
- Use `-cut both_no_init` to add cutpoints on the signals on both sides without the concrete value init assumption.
- Use `-cut spec` to add a cutpoint only on the signal on the spec side.
- Use `-cut imp` to add a cutpoint only on the signal on the imp side.
- Use `-stopat` to set the mapping type to stopat.
- Use `-stopat both` to add stopats on the signals on both sides.
- Use `-stopat spec` to add a stopat only on the signal on the spec side.
- Use `-stopat imp` to add a stopat only on the signal on the imp side.
- Use `-helper` to set the mapping type to helper.
- Use `-set_tag` to set a new tag for the mapping pairs.

Note: The command `check_sec -set_mapping_type -id mapping_id_list` without any other switches sets the `mapping_type` based on the signal attributes.

Enabling and Disabling Rules and Mapping

```
-enable | -disable
  ( (-mapping | -rule) -id {mapping_or_rule_id_list}
    [-force]
    | ( -spec {spec_regexp_or_signal}
        | -imp {imp_regexp_or_signal} )
        [-regexp [-verbose]]
        [-mapping_type ( connection_all | connection_primary_input
                         | connection_bbox_output | connection_undriven
                         | connection_stopat | connection_x_value
                         | connection_x_default_value | connection_misc
                         | ver_target_all | ver_target_primary_output
                         | ver_target_mpram_port | ver_target_bbox_input
                         | ver_target_misc | cutpoint_all | cutpoint_both
                         | cutpoint_spec | cutpoint_imp | stopat_all
                         | stopat_both | stopat_spec | stopat_imp
                         | helper | init ) ]
        [ (-instance {instance_full_path})
          | (-of_signal {regexp_or_signal_list}
            [ -spec_regexp [-verbose]
              | -imp_regexp [-verbose]] )
        ]
        [-proof_status (proven | cex | unprocessed | undetermined)+]
        [-signal_type ( input | output | primary_input
                        | primary_output | bbox_input | bbox_output
                        | mpram_port | undriven | flop | latch
                        | internal | stopat | reset_x
                        | x_value | x_default_value | x_md_abstract) ]
        [-tag {tag_name}]
        [-task {task_name} [-include_global_mapping]]
        [-force]
    )
  )
```

Enable or disable specific rules or mappings.

If you enable or disable rules, all their mapping pairs and properties are enabled or disabled accordingly. If you enable or disable mappings, their properties are enabled or disabled accordingly.

- Use **-mapping** to enable or disable specific mapping pairs.
- Use **-rule** to enable or disable specific rules.
- Use **-id** to specify the list of IDs of the specific mapping pairs you want to enable or disable.
- Use **-spec** instead of IDs to define the set of signal name(s) on the spec side whose mappings you want to enable or disable.
- Use **-imp** instead of IDs to define the set of signal name(s) on the imp side whose mappings you want to enable or disable.

JasperGold Apps Command Reference Manual

General Commands

- ❑ Use `-regexp` to specify that the `-spec` or `-imp` argument list is a regular expression list; otherwise, it is a list of specific names of signals, bits, or bit-vector expressions. Include the `-verbose` switch to print info messages detailing the regexp matching operation.
 - ❑ Use `-signal_type` to enable or disable only mappings of signals of the specified type. You can only use this switch with `-spec` or `-imp` and `-regexp`. That is, if you specify mapping IDs, all types are enabled or disabled.
 - ❑ Use `-mapping_type` to enable or disable only mapping pairs of the specified mapping type.
 - ❑ Use `-of_signal` to define the set of signal names whose mappings you want to enable or disable.
 - Use `-spec_regexp` or `-imp_regexp` to specify that the `-of_signal` argument is a regular expression list on the spec or imp side. The default is spec side; otherwise, it is a list of specific names of signals, bits, or bit-vector expressions.
 - Use `-verbose` to print info messages detailing the regexp matching operations.
 - ❑ Use `-instance` to specify that you want to enable or disable only mapping pairs of signals that are in the specified instance.
 - Use `-proof_status` to enable or disable mapping entries according to their proof status. This switch accepts one or more of the following arguments: proven, cex, unprocessed, undetermined.
 - Use `-tag` to enable or disable only the specified mappings.
 - Use `-task` to list only the task related mappings without considering global or other tasks pairs. If the `-task` switch was not specified, it will list only global mapping pair(s). And use `-include_global_mapping` with `-task` to list both the global and task-specific mapping pair(s).
 - Use `-force` to disable non-speculative cutpoints. That is, cutpoints for which you used the `-speculative off` option.
-

JasperGold Apps Command Reference Manual

General Commands

Marking Signals

```
-mark -forbid_init -sig_list {signals_list}
```

Mark signals for the SEC App.

Note: The only property that is currently available for marking is `forbid_init`, which forbids the SEC App from putting init mapping on then marked signals.

List marked signals with `check_sec -list -marked`

```
-unmark -forbid_init -sig_list {signals_list}
```

Unmark signals for the SEC App.

Waiving Signals

```
-waive -waive_signals {signals_list} [-description]
      | -unwaive_signals {signals list}
      | -list_waived [-verbose]
```

Manage waivers on signals.

Note: The tool does not map waived signal and the `signoff` command does not report them.

- Use `-waive_signals` to waive a list of signals. This command removes any existing mapping from these signals.
- Use `-description` to add a description of the waiver.
- Use `-unwaive_signals` to unwaive a list of signals. The tool does not restore any previous mapping.
- Use `-list_waived` to list all waived signals and use `verbose` to see the descriptions for the waivers.

Locking and Unlocking the Environment

```
-lock_env [-error_onViolation] | -unlock_env
```

Lock or unlock the environment.

If you use `-lock_env`, the tool ignores any action that attempts to change the locked environment. If you also specify `-error_onViolation`, the tool stops with an error message.

Use `-unlock_env` to open the environment for changes.

JasperGold Apps Command Reference Manual

General Commands

Prove

```
-prove
  [-task task_name | -property property_name_tcl_list [-regexp]]
  [-no_auto] [-order_targets (by_gates | by_flops | none)]
  [ (-rule_id {rule_id_list}) |-mapping_id {mapping_id_list}
    | ( -spec {spec_regex_or_signal}
      | -imp {imp_regex_or_signal})
    [-regexp [-verbose]])
  [-signal_type ( input | output | primary_input
    primary_output | bbox_input | bbox_output
    mpram_port | undriven | flop | latch
    internal | stopat | reset_x
    x_value | x_default_value | x_md_abstract)
    | -connection_all | connection_primary_input
    | connection_bbox_output | connection_undriven
    | connection_stopat | connection_x_value
    | connection_x_default_value | connection_misc
    | ver_target_all | ver_target_primary_output
    | ver_target_mpram_port | ver_target_bbox_input
    | ver_target_misc | cutpoint_all | cutpoint_both
    | cutpoint_spec | cutpoint_imp | stopat_all
    | stopat_both | stopat_spec | stopat_imp
    | helper | init ) ]
  [ (-instance {instance_full_path})
    | (-of_signal {regexp_or_signal_list}
      [ -spec_regex [-verbose]
        | -imp_regex [-verbose]])
    ]
  [-proof_status (proven | cex | unprocessed | undetermined)+]
  [-tag {tag_name}]
]
[-prove_opts {prove_options_list}]
[-time_limit time_budget]
[-cex_threshold number_of_failures]
[-levelize_helpers
  [-levels levels_arg]
  [-tag {tag_name}]]
[-bg]
[-no_auto]
[-silent |-verbosity N]
[-assumption_lifting]
[-suppress_traces]
```

Create a task that contains conditions, assumptions, and assertions to verify equivalence and launch a `prove` command on the task.

If you do not specify `-rule_id`, `-mapping_id`, `-spec`, or `-imp`, the tool generates all such properties; otherwise, it generates only the properties that are related to specified rules or mapping pairs.

- Use `-task` to generate the properties to the specified task. If you do not specify a task, this command applies to the current task.

-prove (Continued)

- Use `-property` to specify the names of the properties you want to prove. These properties may be a subset of the exposed SEC properties, but they may also be properties generated by other means. By default, the `check_sec -prove` subcommand attempts to prove all the provable properties in the task.
- Use `-no_auto` to run a low-level `prove` command with no special settings.
- Use `-order_targets` to specify that you want the app to attempt to order the `prove` dispatch for the helper assertions based on internal ranking of what is best to start with rather than ordering them one by one. The latter can be a time-consuming process that results in a time-out on hard-to-prove helper assertions. With the `-order_targets` mode, the easy-to-prove helper assertions will converge quickly, thus benefiting proof convergence.
- Use `-rule_id` to generate properties only from mapping pairs generated from rules of the specified IDs.
- Use `-mapping_id` to generate properties only from mapping pairs of the specified IDs.
- Use `-spec` instead of IDs to define the set of signal name(s) on the spec side whose properties you want to generate.
- Use `-imp` instead of IDs to define the set of signal name(s) on the imp side whose properties you want to generate.
 - Use `-regexp` to specify that the `-spec` or `-imp` argument list is a regular expression list; otherwise, it is a list of specific names of signals, bits, or bit-vector expressions. Include the `-verbose` switch to print info messages detailing the `regexp` matching operation.
 - Use `-signal_type` to generate properties only for mappings of signals of the specified type. You can only use this switch with `-spec` or `-imp` and `-regexp`. That is, if you specify mapping or rule IDs, the tool generates properties for all types.
 - Use `-mapping_type` to prove mapping pairs of the specified mapping type.

JasperGold Apps Command Reference Manual

General Commands

-prove (Continued)

- ❑ Use `-of_signal` to define the set of signal names whose mappings you want to prove.
 - Use `-spec_regexp` or `-imp_regexp` to specify that the `-of_signal` argument is a regular expression list on the spec or imp side. The default is spec side; otherwise, it is a list of specific names of signals, bits, or bit-vector expressions.
 - Use `-verbose` to print info messages detailing the regexp matching operations.
 - ❑ Use `-instance` to specify that you want to prove only mapping pairs of signals that are in the specified instance.
 - ❑ Use `-proof_status` to prove mapping entries according to their proof status. This switch accepts one or more of the following arguments: proven, cex, unprocessed, undetermined
 - ❑ Use `-tag` to generate properties only from mappings with the specified tag.
 - Use `-prove_opts` to specify additional `prove` options.
 - Use `-time_limit` to override the default two hours `prove_time_limit` for `autoprove`. Specify `time_budget` values as an integer followed by s, m, or h (no spaces).
 - ❑ s = seconds
 - ❑ m = minutes
 - ❑ h = hours
-

-prove (Continued)

- Use `-levelize_helpers` to specify an iterative, level-by-level proof over the levels that were previously calculated with the command `-map -levelize_helpers`.
 - Use `-levels` to specify what levels you want to prove. If you do not use this switch, the tool proves all levels.
 - Use `-tag` to specify the tag whose levels you want to prove.
 - Use `-bg` to run the command in the background and continue to accept and run new commands.
 - Use `-no_auto` to run the main proof flow with user-specified settings instead of the autoprove settings.
 - **Note:** The tool does not support `-bg` in combination with `-prove -no_auto`.
 - Use `-silent` for `-verbosity 0` (see below).
 - Use `-verbosity N` to override the current `prove_verbosity` for this `-prove` command. See “[set_prove_verbosity](#)” on page 1192 for a summary of the verbosity levels.
 - Use `-assumption_lifting` to only include a relevant subset of assumptions that is sufficient for the proof.
 - **Note:** This option might be useful in cases with too many assumptions.
 - Use `-suppress_traces` to suppress trace generation.
 - Use `-cex_threshold` to specify that you want the prove flow to stop the proof job after it reaches a certain threshold of failures on SEC properties.
-

Prove with Cutpoint Refinement

```
-prove -autoprove_mode (sscp | dscp)
[-task task_name |-property property_name_tcl_list [-regexp]]
[-cutpoints_on_latches]
[-cutpoints_on_internal_signals]
[-max_number_of_cutpoints N>=0]
[-bg]
[-silent |-verbosity N]
```

Run a proof with cutpoint refinement loop handling.

- Use `sscp` to run the proof with single-sided cutpoints.
- Use `dscp` to run the proof with double-sided cutpoints.
- Use `none` to run the proof without cutpoints.
- Use `-task` to generate the properties to the specified task. If you do not specify a task, this command applies to the current task. In the two cases it will generate properties from the task specific mapping pair(s) and from the global pair(s).
- Use `-property` to specify the names of the properties you want to prove. These properties may be a subset of the exposed SEC properties, but they may also be properties generated by other means. By default, the `check_sec -prove` subcommand attempts to prove all the provable properties in the task.
- Use `-cutpoints_on_latches` to add cutpoints on latches.
- Use `-cutpoints_on_internal_signals` to add internal cutpoints on selected signals based on the density algorithm.
- Use `-max_number_of_cutpoints` to specify that you want the tool to automatically add no more than the specified number of cutpoints. It includes flops and latches (if you use `-cutpoints_on_latches`) and internal signals.

JasperGold Apps Command Reference Manual

General Commands

-prove (Continued)

- Use `-bg` to run the command in the background and continue to accept and run new commands.
- Use `-silent` for `-verbosity 0` (see below).
- Use `-verbosity N` to override the current `prove_verbosity` for this `-prove` command. See “[set prove verbosity](#)” on page 1192 for a summary of the verbosity levels.

Note: Prove with cutpoint refinement is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

Generating Verification

```
-generate_verification  
  [-task task_name]  
  [-assume_equiv (by_assume | by_driver)]
```

Create a task that contains all the conditions, assumptions, and assertions to verify equivalence.

- Use `-task` to generate the properties to the specified task. If you do not specify a task, this command applies to the current task.
- Use `-assume_equiv` to specify how you want the tool to assume equality for the spec and imp pair of signals when those signals are `bbox_output`, `undriven`, or `primary input`:
 - `by_driver`: Replace the driver of the imp signal with the spec driver. This is the default.
 - `by_assume`: Generate an assumption property
`assume {spec_signal == imp_signal}.`

JasperGold Apps Command Reference Manual

General Commands

Sign-Off Report

```
-signoff  
  [-task task_name] [-file file_name]  
  [-silent]  
  [-summary] [-detailed]  
  [-waive_x] [-waive_consistency]  
  [-cp_bound [-include_cex_bound]]
```

Writes a report that summarizes the SEC proof status for a specific task. The sign-off report highlights the possible risks to the validity of the SEC proof. The risks are associated with a list of categories, where for every category, the report sets a grade.

- The grades are as follows:
 - Complete: This category poses no risk to the SEC proof.
 - Incomplete: The SEC proof can be valid, but that depends on further action or user knowledge. For example, an SEC proof can be valid when you disable X-handling. However, the validity of the SEC proof depends on the assumption that Xs do not propagate to valid outputs in both spec and imp. You should check this assumption, possibly with the X-Propagation Verification App.
 - Invalid: The SEC proof is invalid, and you must take further actions.
- Use `-task` to generate a sign-off report for a specified task. By default, the tool generates a report for the current task.
- Use `-file` to specify a file to which you want to write the report. By default, the name is `SecEquivSignoff + taskName + .txt`.
- Use `-silent` to get only the returned status.
- Use `-summary` to print only the “Signoff Summary” section.
- Use `-detailed` to get a detailed report that includes the signals that make the SEC proof incomplete or invalid.
- Use `-waive_x` to turn off the report for undriven or X mapping failures.

Note: Only use this switch for cases when the report makes the SEC proof incomplete or invalid.

-signoff (Continued)

- Use `-waive_consistency` to turn off the report from `check_assumptions -conflict`.
- Use `-cp_bound` to get a list of verification targets (outputs, bbox inputs, and so forth) that have unconverged or failing cutpoints in their COI. This command lists the minimal cutpoint bound in the COI. And use `include_cex_bound` if you want the minimum bound calculation to consider the actual bound of the failing cutpoints. By default, the bound of a failing cutpoint is calculated as `-1`.

Utility Switches

`-get_spec_top_instance | -get_imp_top_instance`

Return the name of the spec or imp top instance.

Return

Various

Examples

```
% check_sec -setup -spec modelA -spec_analyze_opts "-sv -f modelA.vfile" -imp modelB -imp_analyze_opts "-sv -f modelB.vfile" -setup_file setup.file
% check_sec -setup -spec modelA -imp modelB -setup_file setup.file -ipk_opts "-jdb $IPK_DIR/IPK/jasper_amba4_ace.jdb"
% check_sec -clear
% check_sec -clear proof
% check_sec -clear rule
% check_sec -interface
% check_sec -interface -mapped
% check_sec -interface -imp
% check_sec -interface -mapped -spec -signal_type primary_input
% check_sec -interface -unmapped -imp -signal_type bbox_output -inst modelB.inst1
% check_sec -list -signal_type output -inst modelA
% check_sec -list -imp -signal_type bbox_input -inst modelB_imp.inst5
% check_sec -list -mapping -mapping_type connection_primary_input
% check_sec -list -mapping -signal_type undriven
% check_sec -list -rule
% check_sec -map -auto
% check_sec -map -spec {inst1.sig1\([(0-9)+]\]} -imp {modelB_imp.inst1.sig1\$1\}
-regexp
```

JasperGold Apps Command Reference Manual

General Commands

```
% check_sec -map -spec {inst1.sig2[2]} -imp {modelB_imp.inst1.sig2[2]}
% check_sec -map -spec {inst1.sig3} -imp {modelB_imp.inst1.sig3}
% check_sec -map -spec {specsig(_i|_o|_reg)\[0:2\]} -imp
{modelB_imp.impsig$1\[2:4\]} -regexp
% check_sec -unmap -id 97
% check_sec -unmap -imp {modelB_imp.impsig(_i|_o|_reg)[2:4]} -regexp
% check_sec -remove_rule -id 100
% check_sec -disable -spec {inst1.sig3}
% check_sec -enable -imp {modelB_imp..*.sig1\[0-2\]} -regexp
% check_sec -prove auto

# Hierarchy aliasing examples:
# -----
% check_sec -map -hierarchy -spec {instA_1.instB} -imp {modelB_imp.instX_1.instY}
% check_sec -list -hierarchy -return id
% check_sec -mapping_info -hierarchy -id 5
% check_sec -unmap -hierarchy -id {7 3}

# Use regexp to automatically map spec signals to imp signals when their names
# are the same but their hierarchies are different. Example:
% check_sec -map -hierarchy -spec {instA_\([0-9]\).instB} -imp
{modelB_imp.instX_\$1.instY} -regex

# In the example above, consider 2 instances of instA_*: instA_4 and instA_5.
# The command maps signals found in, for example, instances {instA_4.instB} and
# {instA_5.instB} and maps them accordingly to the imp side using the matching
# instance numbers:
% {modelB_imp.instX_4.instY} and {modelB_imp.instX_5.instY}
```

See Also

No related commands

check_sps

Description

Use the `check_sps` command to manipulate property candidates obtained from the Structural Property Synthesis (SPS) App property extraction process. Each property candidate has a type and a classification. The tool uses intelligent heuristics to determine the desirable default type.

The following list shows the types and classifications.

- Types – assert, assume, cover
- Classifications – waived, dont_care, unclassified

Initially, all extracted properties are unclassified, thus allowing you to determine the relative significance of these properties.

Syntax

```
check_sps -init

check_sps <property_id_tcl_list>
    -set_class (dont_care | unclassified)
    [-silent]

check_sps <message_id_tcl_list>
    -set_class unclassified
    [-silent]

check_sps <property_id_tcl_list>
    -set_active_task <task_name>
    [-silent]

check_sps <property_id_tcl_list>
    -get (type | class | expression | representation
          | scope | status | waiver | task | fpv_property | label
          | severity | tag | category)
    [-baseline] [-silent]

check_sps <message_id_tcl_list>
    -get ( class | expression | scope | label | waiver
          | severity | tag | category)
    [-silent]

check_sps -export
    (-type (assert | assume | cover))*
    (-class (waived | dont_care | unclassified))*
    (-status (all | cex | ar_cex | proven | ar_covered | unprocessed
```

JasperGold Apps Command Reference Manual

General Commands

```
| covered_reset | unreachable | undetermined | partial_cex
| partial_ar_cex | partial_proven | partial_ar_covered
| partial_covered_reset | partial_unreachable | partial_undetermined) )*
[-check_type (all | unique_case | priority_case
| arithmetic_overflow | out_of_bound_indexing | dead_code | fsm
| signals | contention_bus | floating_bus | x_assignment
| default_case) ]
[-module <sva_module_name>]
[-task <task_name>]
[-extended_property_name]
[-baseline]
[-silent]

check_sps -export
  -property <property_id_tcl_list>
  [-module <sva_module_name>]
  [-task <task_name>]
  [-extended_property_name]
  [-baseline]
  [-silent]

check_sps -extract
  [-instances <instance_list> |-hierarchies <instance_list>
   |-exclude_hierarchies <instance_list> |-top]

check_sps -filter
  [-engine <jg_engine>]
  [-time_limit <jg_time_limit>]
  [-iter <jg_iteration_limit>]
  [-env <setup_file_name>]
  [-task <task_name>]
  [-property <property_id_tcl_list>]
  [-debug]

check_sps -show <property_id_tcl_list>
  [-baseline] [-silent]

check_sps -show <message_id_tcl_list>
  [-silent]

check_sps -set_label <label> (<message_id_tcl_list> | <property_id_tcl_list>)

check_sps -list
  [-type (assert | assume | cover)]*
  [-class (waived | dont_care | unclassified)]*
  [-regexp <property_expression_regexp>]
  [-status (all | cex | ar_cex | proven | ar_covered | unprocessed
            | covered_reset | unreachable | undetermined| partial_cex
            | partial_ar_cex | partial_proven | partial_ar_covered
            | partial_covered_reset | partial_unreachable | partial_undetermined) ]*
  [-check_type (all | unique_case | priority_case
| arithmetic_overflow | out_of_bound_indexing | dead_code | fsm
| signals | contention_bus | floating_bus | x_assignment
| default_case) ]
```

JasperGold Apps Command Reference Manual

General Commands

```
[-scope <poi_tcl_list>]
[-severity (error | warning | note)]
[-category <category>]
[-tag <tag>]
[-signal (<signal>)+]
[-label <label>]
[-baseline] [-silent]

check_sps -highlight <id>

check_sps -update_fsm_idle_state <id_tcl_list>
    [-idle_state_value <idle_state_value>]
```

Argument	Definition
-init	<p>Initialize the settings required by the Structural Property Synthesis App.</p>
<code>property_id_tcl_list -set_class (dont_care unclassified)</code> [-silent]	<p>Change the class of the specified property candidates. Use this switch to control which candidates should be exported.</p> <p>Note: If the classification of a property is changed to waived using the <code>check_sps_waiver -add</code> command or the <i>Property Candidates</i> pane context menu, you cannot change this classification with the <code>check_sps -set_class</code> command.</p> <p>Use <code>-silent</code> to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.</p>
<code>message_id_tcl_list -set_class unclassified</code> [-silent]	<p>Set the class of the specified messages to unclassified.</p> <p>Use <code>-silent</code> to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.</p>
<code>property_id_tcl_list -set_active_task task_name</code> [-silent]	<p>Change the active export task of the specified property candidates.</p> <p>Use <code>-silent</code> to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.</p>

JasperGold Apps Command Reference Manual

General Commands

```
property_id_tcl_list -get
( type | class | expression | representation | scope | status
| waiver | task | fpv_property | label | severity | tag | category)
[-baseline]
[-silent]
```

Report the specified attribute of the specified property candidates.

- Use `type` to report the property candidate type: assertion, assumption, or coverage_hole.
- Use `class` to report the class for property candidates: waived, dont_care, or unclassified.
- Use `expression` to report the expressions for property candidates.
- Use `representation` to report available representations.
- Use `scope` to report the scope (POI IDs) for the property candidates.
- Use `status` to report the proof status for the property candidates.
- Use `waiver` to report waivers for the property candidates.
- Use `task` to report the tasks for which the property candidates were exported.
- Use `fpv_property` to report the Formal Property Verification property names in the active exported tasks for the property candidates.
- Use `label` to report labels for the specified property candidates.
- Use `severity` to report the severity of any violation messages for the specified property candidates.
- Use `tag` to report the message tag for the specified property candidates.
- Use `category` to report the message category for the specified property candidates.
- Use `-baseline` to show the baseline property candidates.
- Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

JasperGold Apps Command Reference Manual

General Commands

```
message_id_tcl_list -get ( class | expression | scope | label | waiver  
                           | severity | tag | category)  
[-silent]
```

Report the specified attribute of the specified messages.

- Use `class` to report the class for the specified messages.
- Use `expression` to report the expressions for the specified messages.
- Use `scope` to report the scope (POI) IDs for the specified messages.
- Use `label` to report the labels for the specified messages.
- Use `waiver` to report the waivers for the specified messages.
- Use `severity` to report the severity of the specified messages.
- Use `tag` to report the tag for the specified messages.
- Use `category` to report the category for the specified messages.

JasperGold Apps Command Reference Manual

General Commands

```
-export
-type (assert | assume | cover)+
-class (waived | dont_care | unclassified) +
-status ( all | cex | ar_cex | proven | ar_covered | unprocessed
          | covered_reset | unreachable | undetermined
          | partial_cex | partial_ar_cex | partial_proven
          | partial_ar_covered | partial_covered_reset
          | partial_unreachable | partial_undetermined) +
-check_type ( all | unique_case | priority_case
              | arithmetic_overflow | out_of_bound_indexing
              | dead_code | fsm | signals | contention_bus
              | floating_bus | x_assignment | default_case) +
[-module sva_module_name]
[-task task_name]
[-extended_property_name]
[-baseline]
[-silent]
```

JasperGold Apps Command Reference Manual

General Commands

Export the property candidates to the Formal Property Verification App.

- Use `-type` to specify a type of property candidate to export. The default is `assert` and `cover`.
- Use `-class` to specify a class of property candidate to export. The default is `unclassified`.
- Use `-status` to specify a status of property candidate to export. The default is `all`.
- Use `-check_type` to specify specific checks type for export. The default is `all`.
- Use `-module` to specify a module name for SVA export. This command overwrites the default name.
- Use `-task` to export the property candidates to the specified task instead of creating a new task for each check type.

Note: Replace `task_name` with a period (.) to specify the current task.

- Use `-extended_property_name` to specify the following naming convention for checks:

```
\<instance>_<file><start line><end line><check  
type>poi<poi id>prop<prop id>[more indexes]
```

The naming convention uses Verilog escaped identifiers, that is, they begin with a backslash (\).

- Use `-baseline` to use the property candidates in the baseline.
 - Use `-silent` to suppress messages and generate only tcl return values.
-

JasperGold Apps Command Reference Manual

General Commands

```
-export -property property_id_tcl_list
[-module sva_module_name]
[-task task_name]
[-extended_property_name]
[-baseline]
[-silent]
```

Export the property candidates to the Formal Property Verification App.

- Use `-property` to specify a set of property candidate IDs to export.
 - Use `-module` to specify a module name for SVA export. This command overwrites the default name.
 - Use `-task` to export the property candidates to the specified task instead of creating a new task for each check type.
- Note:** Replace *task_name* with a period (.) to specify the current task.
- Use `-extended_property_name` to specify the following naming convention for checks:

```
\<instance>_<file><start line><end line><check
type>poi<poi id>prop<prop id>[more indexes]
```

The naming convention uses Verilog escaped identifiers, that is, they begin with a backslash (\).

- Use `-baseline` to use the property candidates in the baseline.
- Use `-silent` to suppress messages and generate only tcl return values.

```
-extract
[ -instances instance_list | -hierarchies instance_list
 | -exclude_hierarchies instance_list | -top]
```

Analyze the RTL to extract POIs and then generate properties from these scopes.

- Use `-instances` to limit extraction to the specified instances.
- Use `-hierarchies` to limit extraction to the specified hierarchies.
- Use `-exclude_hierarchies` to exclude the specified hierarchies.
- Use `-top` to limit extraction to the top instance.

JasperGold Apps Command Reference Manual

General Commands

```
-filter
[-engine jg_engine]
[-time_limit jg_time_limit]
[-iter jg_iteration_limit]
[-env setup_file_name]
[-task task_name]
[-property property_id_tcl_list]
[-batch]
[-debug]
```

Rank candidate properties by functional complexity.

- Use `-engine` to specify an engine.
- Use `-time_limit` to specify a per property time limit.
- Use `-iter` to specify an iteration limit.
- Use `-env` to specify a Tcl setup file to run before proof in the context of the proof task.
- Use `-task` to select a task from which to read embedded properties. The default is `<embedded>`.
- Use `-property` to select the properties to filter. The default is `unclassified`.
- Use `-debug` to print debugging messages.

```
-show property_id_tcl_list
[-baseline]
[-silent]
```

Show the details of the property candidates.

- Use `-baseline` to show the baseline property candidates.
- Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

```
-show message_id_tcl_list
[-baseline]
[-silent]
```

Show the details of the specified messages.

Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

```
-set_label label property_id_tcl_list | message_id_tcl_list
```

Add a label for properties or messages that persists across different runs.

Note: The label must be a non-negative integer.

JasperGold Apps Command Reference Manual

General Commands

```
-list
[-type (assert | assume | cover)+]
[-class (waived | dont_care | unclassified)+]
[-regexp property_expression_regexp | message_expression_regexp]
[-status ( all | cex | ar_cex | proven | ar_covered
           unprocessed | covered_reset | unreachable
           undetermined | partial_cex | partial_ar_cex
           partial_proven | partial_ar_covered
           partial_covered_reset | partial_unreachable
           partial_undetermined)+]
[-check_type ( all | unique_case | priority_case
               arithmetic_overflow | out_of_bound_indexing
               dead_code | fsm | signals | contention_bus
               floating_bus | x_assignment | default_case)+]
[-scope poi_tcl_list]
[-signal signal]
[-severity (error | warning | note)]
[-tag tag]
[-category category]
[-label label]
[-baseline] [-silent]
```

List the IDs for the property candidates or messages that match the specified criteria.

- Use `-type` to list properties of a specific type.
- Use `-class` to list properties and messages of a specific classification.
- Use `-regexp` to filter properties or messages by expression using regular expressions.
- Use `-status` to list properties of a specific status. The default is `all`.
- Use `-check_type` to list properties and messages of a specific check type.
- Use `-scope` to list only properties and messages derived from the specified POIs.
- Use `-signal` to list only properties that involve at least one of the specified signals.

JasperGold Apps Command Reference Manual

General Commands

-list (Continued)

- Use `-severity` to list only properties and messages of the specified severity.
- Use `-tag` to list only properties and messages with the specified tag.
- Use `-category` to list only the properties and messages in the specified category.
- Use `-baseline` to use the baseline property candidates.
- Use `-label` to list properties and messages with the specified label.
- Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

-highlight *id*

Highlight the property with the specified ID in the *Property Candidates* pane. (The tool takes no action if the ID is invalid or it is currently invisible due to filtering.)

-update_fsm_idle_state *id_tcl_list*

`[-idle_state_value idle_state_value]`

Update the idle state of FSM POIs.

Use `-idle_state_value` to specify the updated idle state value.

Return

Context-sensitive return values

Examples

```
% check_sps -extract  
% check_sps -list -check_type dead_code  
% check_sps -list -regexp {.*my_instance_name.*}  
% check_sps -export -property {1 2 3 4 5}  
% check_sps -filter -task assumptions_task
```

See Also

[check_sps_configure](#)

check_sps_configure

Description

Use the `check_sps_configure` command to configure the Structural Property Synthesis (SPS) App flow. Some of the available checks take effect during module elaboration; therefore, you must run this command before running the `elaborate` command. A list of checks that require module re-elaboration follows:

- case assertions
- arithmetic overflow
- out of bound indexing
- dead code checks



- FSM reachable multi-transitions check –
 - The FSM reachable multi-transitions check generates property macros like the following:

```
`jasper_reachable_multi_transitions(brdg.byte_num, {0 |=> 1 |=> 0, 0 |=> 1 |=> 2, 1 |=> 0 |=> 1, 1 |=> 2 |=> 0, 1 |=> 2 |=> 3, 2 |=> 0 |=> 1, 2 |=> 3 |=> 0, 3 |=> 0 |=> 1})`
```
 - For this example, the generated property is the following:

```
cover( S1 ##1 S2[*1:$] ##1 S3 )
```

Note: In this example, the FSM may stay in state `S2` for more than one cycle.

- FSM deadlock check –
 - The FSM deadlock check generates property macros like the following:

```
!`jasper_deadlock(brdg.byte_num, 0-3)
```
 - For this example, the POI name and property are formatted as follows:

```
SPS_fsm_state_deadlock_poi_294_prop_297_1_3 : assert property
(((brdg_byte_num == s) |=> ##[0:$] (brdg_byte_num != s)));
```

■ **FSM livelock check –**

- The FSM livelock check generates property macros like the following:

```
!`jasper_livelock(brdg.byte_num, 0-3)
```

- For the init state S_i , the POI name and property are formatted as follows:

```
SPS_fsm_state_livelock_poi_294_prop_296_0_4 : assert property (( ##[0:$] (brdg_byte_num == Si)) );
```

- For each other state S , they are formatted as follows:

```
SPS_fsm_state_livelock_poi_294_prop_296_2_4 : assert property (((brdg_byte_num == S) |=> ##[0:$] (brdg_byte_num == Si)) );
```

Note:

- The init state is detected as the state with the lowest encoding number of the main FSM states group (eliminating the initial sequence states and states in a deadlock). You can examine the init state using the `scope -show <poi_id>` command. If the init state is incorrect, you can update it using `check_sps -update_fsm_init_state`. See “[check_sps](#)” on page 399.
- This check detects livelock within one FSM path; therefore, if an FSM transitions from init state to another state, it will not have a livelock and will revert back to init state. However, if an FSM transition from the init state to another state turns unreachable, this goes undetected. This behavior is intended to reach a balance between adding constraints complexity and covering all potential livelock detection.

Syntax

```
check_sps_configure ((-enable |-disable)
  ( all
    | case_assertions
    | floating_bus
    | contention_bus
    | out_of_bound_indexing
    | arithmetic_overflow
    | stuckat_signals
    | toggle_rise_signals | toggle_fall_signals | toggle_stable_signals
    | fsm_reachable_states | fsm_reachable_transitions
    | fsm_reachable_multi_transitions
    | fsm_deadlock | fsm_livelock
    | dead_code
```

JasperGold Apps Command Reference Manual

General Commands

```
| x_assignment  
| default_case  
| elaboration_checks)+  
  
check_sps_configure (-enable |-disable hal_checks)  
  (-hal_rule_file <file_name>)+  
  (-hal_msg_file <file_name>)+  
  (-hal_design_info_file <file_name>)+  
  (-hal_incisiv_path <path>)  
  [-hal_extra_options <options_list>]
```

Argument Definition

-enable -disable	(all case_assertions floating_bus contention_bus out_of_bound_indexing arithmetic_overflow stuckat_signals toggle_rise_signals toggle_fall_signals toggle_stable_signals fsm_reachable_states fsm_reachable_transitions fsm_reachable_multi_transitions fsm_deadlock fsm_livelock dead_code x_assignment default_case elaboration_checks)+
--------------------	--

Enable or disable the specified check type. Use `all` to denote the entire set of checks.

Note:

- Case assertions are available for SystemVerilog and Verilog only.
- Use one command to enable and disable multiple checks by preceding each check enum with the `-enable` or `-disable` switch, as shown in [“Examples”](#) on page 413. The tool parses `-enable` options before `-disable` options unless you have specified `-disable all`. When that is the case, the tool parses `-disable` options first.

`-enable | -disable hal_checks`

Note: With the Superlint flow, that is, when you run SPS with the Incisive frontend and enable both HAL lint checks and SPS advanced lint checks, `arithmetic_overflow`, `default_case`, `contention_bus`, and `floating_bus` checks are not available.

Enable or disable HAL lint checks.

`-hal_rule_file rule_files_list`

Add the specified HAL rule files.

`-hal_message_file message_files_list`

Add the specified HAL message files.

JasperGold Apps Command Reference Manual

General Commands

```
-hal_design_into_file design_info_files
```

Add the specified HAL design information files.

```
-hal_incisiv_path path
```

Provide the path to the Incisive installation.

```
-hal_extra_options options_list
```

Specify the desired options.

Return

No value returned

Examples

```
% check_sps_configure -enable all
% check_sps_configure -disable all
% check_sps_configure -enable dead_code -disable out_of_bound_indexing
% check_sps_configure -disable all -enable dead_code -enable arithmetic_overflow
% check_sps_configure -enable all -disable dead_code -disable arithmetic_overflow
% check_sps_configure -disable dead_code -enable arithmetic_overflow
% check_sps_configure -enable hal_checks -hal_rule_file my_rule.def -hal_msg_file
my_msg1.htx -hal_msg_file my_msg2.htx
```

See Also

[check_sps](#)
[set_sps_arithmetic_overflow_style](#)
[set_sps_no_duplicates_per_check](#)
[set_sps_signals_flops](#)
[set_sps_signals_outputs](#)
[set_sps_stuck_at_compact](#)

check_sps_waiver

Description

In some cases, when using the SPS App, you might debug a failing check and conclude it does not represent a bug, and no RTL modification is warranted. In these cases, you can waive the check so that when you re-run the tool, it will not flag the failure. Use the `check_sps_waiver` command to specify, review, and save waivers for future use. For additional information, see Appendix G, “Persistent Waivers with the SPS App” in the *JasperGold Apps User’s Guide*.

Syntax

```
check_sps_waiver -add -ids (<message_id_tcl_list | property_id_tcl_list>)
    -comment <comment>
    [-silent]
    [-force]

check_sps_waiver -add
    (-[-check_type
        ( unique_case | priority_case
        | arithmetic_overflow | out_of_bound_indexing | dead_code
        | fsm | signals | contention_bus
        | floating_bus | x_assignment | default_case) ]
    [-instance <instance_name>]
    [-top]
    [-expression <expr>]
    [-category <message_category>]
    [-tag <message_tag>]
    [-message <message>]
    [-module <module_name>]
    [-source_file <file_name>
        [-start_line <line_num>] [-end_line <line_num>]
    ]
    )+
    -comment <comment>
    [-silent]
    [-force]

check_sps_waiver -remove <waiver_id_tcl_list>
    [-silent]

check_sps_waiver -show <waiver_id_tcl_list>
    [-silent]

check_sps_waiver -list
    (-[-check_type
        ( all | unique_case | priority_case
```

JasperGold Apps Command Reference Manual

General Commands

```
| arithmetic_overflow | out_of_bound_indexing | dead_code
| fsm | signals | contention_bus
| floating_bus | x_assignment | default_case)]
[-instance <instance_name>]
[-top]
[-expression <expr>]
[-category <message_category>]
[-tag <message_tag>]
[-message <message>]
[-module <module_name>]
[-source_file <file_name>]
[-silent]

check_sps_waiver -list_matching_checks <waiver_id_tcl_list>
[-silent]

check_sps_waiver -export -file_name <file_name>
<waiver_id_tcl_list>

check_sps_waiver -import -file_name <file_name>
[-silent] [-force]

check_sps_waiver -accept_update <waiver_id_tcl_list>
[-silent]
```

Argument	Definition
<code>-add -ids (message_id_tcl_list property_id_tcl_list) -comment comment</code> [-silent] [-force]	<p>Add waivers on the specified messages or properties.</p> <ul style="list-style-type: none">■ Use <code>-comment</code> to add a comment to the waiver.■ Use <code>-silent</code> to suppress output and generate only Tcl return values.■ Use <code>-force</code> to add waivers even if there are no matching messages or properties.

JasperGold Apps Command Reference Manual

General Commands

```
-add
```

```
( [-check_type
  ( unique_case | priority_case | arithmetic_overflow
    | out_of_bound_indexing | dead_code | fsm | signals
    contention_bus | floating_bus | x_assignment
    default_case) ]
  [-instance instance_name] [-top]
  [-expression expr]
  [-category message_category]
  [-tag message_tag]
  [-message message]
  [-module module_name]
  [-source_file file_name
    [-start_line line_num]
    [-end_line line_num]
  ]
)
+
-comment comment
[-silent]
[-force]
```

Add waivers as specified and return an ID list of the added waivers.

Note: You must use one or more of the following switches: `-check_type`, `-instance`, `-expression`, and `-source_file`.

- Use `-check_type` to add waivers on properties of the specified types.
- Use `-instance` to add waivers on properties of the specified instance. This switch supports wildcards as follows:
 - You can use a string such as `a.*.b` but not a string such as `a*`.
 - Use `*` to add waivers on properties of all instances and `.*` to exclude the top instance.
- Use `-top` to add waivers on properties of the top instance.
- Use `-expression` to add waivers on properties with the specified expression.
- Use `-category` to add waivers on messages in the specified category.
- Use `-tag` to add waivers on messages with the specified tag.
- Use `-message` to add waivers on the specified message.
- Use `-module` to add waivers on messages from the specified module.

JasperGold Apps Command Reference Manual

General Commands

-add (Continued)

- Use `-source_file` to add waivers on properties from the specified file.
 - Use `-start_line` to add waivers on properties after the specified line.
 - Use `-end_line` to add waivers on properties before the specified line.
- Use `-comment` to add a required comment to the waiver.
- Use `-silent` to suppress output and generate only Tcl return values.
- Use `-force` to add waivers even if there are no matching messages or properties.

`-remove waiver_id_tcl_list`
[-silent]

Remove the specified waivers and return an ID list of the waivers removed.
Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

`-show waiver_id_tcl_list`
[-silent]

Show details for the specified waivers.
Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

JasperGold Apps Command Reference Manual

General Commands

```
-list
( [-check_type
    unique_case | priority_case | arithmetic_overflow
    out_of_bound_indexing | dead_code | fsm | signals
    contention_bus | floating_bus | x_assignment
    default_case]
  [-instance instance_name]
  [-top]
  [-expression expr]
  [-category message_category]
  [-tag message_tag]
  [-message message]
  [-module module_name]
  [-source_file file_name]
  [-silent]
```

List the IDs of the waivers that match the specified criteria.

- Use `-check_type` to list waivers added on properties of the specified types.
- Use `-instance` to list waivers added on properties of the specified instances. This switch supports wildcards.
 - You can use a string such as `a.*.b` but not a string such as `a*`.
 - Use `*` to add waivers on properties of all instances and `.*` to exclude the top instance.
- Use `-top` to add waivers on properties of the top instance.
- Use `-expression` to list waivers added on properties with the specified expression.
- Use `-category` to add waivers on messages in the specified category.
- Use `-tag` to add waivers on messages with the specified tag.
- Use `-message` to add waivers on the specified message.
- Use `-module` to add waivers on messages from the specified module.
- Use `-source_file` to list waivers added on properties from the specified file.
- Use `-silent` to suppress output and generate only Tcl return values.

```
-list_matching_checks waiver_id_tcl_list
[-silent]
```

List the IDs of properties or messages that match the specified waivers.

Use `-silent` to suppress messages and generate only Tcl return values.

JasperGold Apps Command Reference Manual

General Commands

```
-export -file_name file_name [waiver_id_tcl_list]
```

Export waivers to the specified file. Use *waiver_id_tcl_list* to export only the listed waivers.

Note: This command generates two files as follows:

- The specified *file_name.txt*, which you can edit, add, or update.
- A tool-generated XML file *filename_persistency_data.xml*, which helps the tool when importing waivers after source file changes. To preserve persistence keep the XML file in the same directory as the text file when importing.

```
-import -file_name file_name  
[-force]  
[-silent]
```

Specify the path to the waivers file to be imported.

- Use *-force* to import waivers even if there are no matching properties.
- Use *-silent* to suppress messages and generate only Tcl return values.

Note: When you import waivers after the design has changed, the tool adds waivers that do not match any property to a file named *filename_unmatched.txt*. If you import waivers using the tool-generated XML file *filename_persistency_data.xml*, which is generated on export, the tool tries to update waivers automatically. Waivers that are automatically imported and updated are added to a separate file named *filename_updated.txt*. However, If the tool-generated XML file has an invalid location as a result of a design change, then the tool will not be able to automatically update imported waivers even if you use the *-force* switch. In this case, the imported waivers are still added to *filename_unmatched.txt*.

```
-accept_update waiver_id_tcl_list  
[-silent]
```

Accept the update to the specified waivers and return an ID list of the matching properties or messages..

Use *-silent* to suppress output and generate only Tcl return values.

Return

Context-sensitive return values

Examples

```
% check_sps_waiver -add -property {1 2 3} -comment "Expected to fail in this configuration"  
% check_sps_waiver -add -instance a.b.* -check_type fsm -comment "Waive all properties for type FSM under a.b sub-instances"  
% check_sps_waiver -add -check_type dead_code -source_file "/a/b/c.v" -start_line 100 -comment "Waive unreachable code in the /a/b/c.v file from line 100"  
% check_sps_waiver -remove {1 2 3}  
% check_sps_waiver -remove [check_sps_waiver -list]  
% check_sps_waiver -export -file "/a/b/mywaivers.txt"  
% check_sps_waiver -export -file "/a/b/mywaivers.txt" [check_sps_waiver -list -check_type dead_code]  
% check_sps_waiver -import -file "/a/b/c/mywaivers.txt"  
% check_sps_waiver -list_matching_properties {1 2}  
% check_sps_waiver -list -source_file /a/b/c.v -expression ~a.rstN
```

See Also

[check_sps](#)
[export](#)

check_spv

Description

Use the `check_spv` command to check the existence of functional paths between specified source and destination signals.

This command creates security properties, properties that check the existence of functional paths between two signals, and provides some tools, such as abstractions, to enable these verifications.

Syntax

```
check_spv -abstract
  (-instances <instance_tcl_list>
   |-inputs <input_tcl_list> -outputs <output_tcl_list>
   |-bbox_instances)
    -mode ( auto [-ignore_clock_connections]
            | comb | seq
            | manual -connections <connection_spec_tcl_list>
            | N[:M])
    [-clock <clock_name>]
    [-ignore_inputs <input_tcl_list>]
    [-ignore_outputs <output_tcl_list>]
    [-env
     |-task <task_name>]

check_spv -abstract -remove <abstraction_id_tcl_list>
check_spv -abstract -clear
check_spv -abstract -list [-silent]

check_spv -create
  -from <signal_tcl_list>
  -to <signal_tcl_list>
  [-from_precond <expression>]
  [-to_precond <expression>]
  [-no_preconditions]
  [-through <signal_tcl_list>]
  [-not_through <signal_tcl_list>]
  [-include_control_path |-exclude_control_path]
  [-task <task_name> |-name <name> |-name <task_name::name>]

check_spv -prove
  [-all |-task <task_name> |-property <property_name>
   [-strategy (parallel | forward | backward)] [-extend_graph_exploration]]
  [-bg]
```

JasperGold Apps Command Reference Manual

General Commands

```
check_spv -list (connection_specs | properties)
[-silent]

check_spv -consistency
[ -task <regexp_task_list>
| -property <property_list>
|-from <signal_list> -to <signal_list>
[-context <task_name>]]
[-list (instance | signal)]

check_spv -highlight <property_list>

check_spv -clear
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<pre>-abstract (-instances <i>instance_tcl_list</i> -inputs <i>input_tcl_list</i> -outputs <i>output_tcl_list</i> -bbox_instances) -mode (auto [-ignore_clock_connections] comb seq manual -connections <i>connection_spec_tcl_list</i> N[:M]) [-clock <i>clock_name</i>] [-ignore_inputs <i>input_tcl_list</i>] [-ignore_outputs <i>output_tcl_list</i>] [-env -task <i>task_name</i>]</pre>	<p>Create an abstraction that does not eliminate functional paths for the specified instances or inputs and outputs. This command returns one abstraction ID when using <code>-inputs</code> and <code>-outputs</code> or one for each instance abstracted when using <code>-instances</code>.</p> <p>Note: Black-boxed modules must be abstracted to enable functional paths through them.</p> <ul style="list-style-type: none">■ Use <code>-instances</code> to abstract the specified instances. This switch is mutually exclusive with the <code>-inputs</code>, <code>-outputs</code>, and <code>-bbox_instances</code> switches.■ Use <code>-inputs</code> to specify signals that will form the initial barrier of the abstraction. You must use this switch with <code>-outputs</code>.■ Use <code>-outputs</code> to specify the signals that will form the end barrier of the abstraction. You must use this switch with <code>-inputs</code>.■ Use <code>-bbox_instances</code> to abstract all instances black boxed during elaborate. This switch is mutually exclusive with <code>-inputs</code>, <code>-outputs</code>, and <code>-instances</code>.

-abstract (Continued)

■ Use `-mode` to specify the abstraction mode:

- `auto`: Performs a structural analysis on the abstraction target and replaces the original structure with a new structure containing combinational and sequential connections based on this analysis.
- Note:** Use `-ignore_clock_connections` with `-mode auto` to disable analysis of paths that go through flop-clock and latch-enable pins.
- `comb`: Replaces the abstraction target with a structure containing only combinational connections. When you use this switch, the data takes `0:$` cycles to propagate through the abstraction.
- `seq`: Replaces the abstraction target with a structure containing only sequential connections. When you use this switch, the data takes `1:$` cycles to propagate through the abstraction.
- `manual`: Replaces the abstraction target with a structure specified with the `-connections` switch.

Note: You must use `-connections` with `-mode manual` and specify a `connection_spec_list`. The specification must have the following format:

```
{ ( output_port { (input_port (comb|seq))+ } )+ }
```

Also see the "[Examples](#)" on page 431 section below.

- `N[:M]`: Replaces the abstraction target with a structure containing connections that delays the data propagation for a minimum of `N` cycles and a maximum of `M` cycles. `N` must be greater than zero and `M` must be greater than `N`.

Note: Using `M` as `$` indicates that the data propagation will take any number of cycles greater than or equal to `N` to propagate through the abstraction.

■ Use `-clock` to specify the abstraction clock:

JasperGold Apps Command Reference Manual

General Commands

-abstract (Continued)

- Use `-ignore_inputs` to exclude the specified signals from the initial barrier of the abstraction.
- Use `-ignore_outputs` to exclude the specified signals from the end barrier of the abstraction.
- Use `-env` to add an abstraction that affects all tasks. Use `-task` to add an abstraction that affects only the specified task.

Note:

- An abstraction created with `-env` takes precedence over any `-task` abstraction for the same target (that is, `-env` overrides `-task`).
- If a target already has an `-env` abstraction, and you attempt to create an abstraction for that same target, the tool issues an error.
- The same target may be abstracted differently for each task.
- If you do not use `-env` or `-task`, the command applies to the current task.

```
-abstract -remove abstraction_id_tcl_list
```

Remove the specified abstractions.

```
-abstract -clear
```

Remove all abstractions performed by `check_spv`.

```
-abstract -list [-silent]
```

Lists and returns all abstractions with their abstraction IDs, inputs, outputs, connection specs, task, and whether it is an environment abstraction. The tool shows inputs and outputs of an instance if the abstraction was performed over an instance.

JasperGold Apps Command Reference Manual

General Commands

```
-create
  -from signal_tcl_list
  -to signal_tcl_list
  [-from_precond expression] [-to_precond expression]
  [-no_preconditions]
  [-through signal_tcl_list] [-not_through signal_tcl_list]
  [-include_control_path |-exclude_control_path]
  [-task task_name |-name name |-name task_name::name]
```

Create security properties to check for functional paths between source and destination signals.

Note: This command ignores user-defined clocks and reset as source signals.

- Use `-from` to define the specified signals as source signals.
- Use `-to` to define the specified signals as destination signals.
- Use `-from_precond` to define an expression that determines when the value on the `-from` point will be allowed to propagate.
- Use `-to_precond` to define an expression that must be true when the value that was on the `-from` has propagated to the `-to`.

Note: When you define `-from_precond` and `-to_precond` expressions, the tool creates related covers by default. Use the `-no_precondition` switch to disable the creation of related covers when defining `-from_precond` and `-to_precond` expressions.

- Use `-no_preconditions` to disable the creation of related covers when using the `-from_precond` and `-to_precond` switches.
- Use `-through` to specify a list of intermediary points for the security path that might be found.

Note: A security path found by the engine will be considered valid if at least one of the signals on the through list is on the path.

- Use `-not_through` to define the specified signals as signals that cannot be on the security path that might be found.

JasperGold Apps Command Reference Manual

General Commands

-create (Continued)

- Use `-include_control_path` to analyze control paths during the property proof even if the `spv_include_control_path` variable is set to `off`. And use `-exclude_control_path` to specify that control paths not be analyzed during the property proof even if the `spv_include_control_path` variable is set to `on`, which is the default. See `set_spv_include_control_path` for additional information.
 - Use `-task` to apply the command to the specified task instead of the current task.
 - Use `-name` to assign a name to the property.
-

JasperGold Apps Command Reference Manual

General Commands

```
-prove [-all | -task task_name | -property property_name  
       [-strategy (parallel | forward | backward)]  
       [-extend_graph_exploration]]  
       [-bg]
```

Note: check_spv -prove is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments.

Start the proof of all or specified security properties. check_spv -prove with no additional arguments proves all SPV properties in the background.

- Use `-all` to prove all SPV properties in all tasks.
- Use `-task` to prove all SPV properties in the specified task.
- Use `-property` to prove the specified SPV property.
- Use `-property` with `-strategy` to choose a strategy that leverages the datapath graph to enhance proof convergence on the specified property.
 - Use `parallel` to run the proof for all generated helper properties simultaneously.
 - Use `forward` to run the proof for helper properties level by level in a forward direction.
 - Use `backward` to run the proof for helper properties level by level in a backward direction.

Note: `forward` and `backward` work as follows. The proof runs for a set of nodes/properties `s1`, then for a set of nodes/properties `s2` (which are connected to nodes/properties in `s1` that were reached), then for a set of nodes/properties `s3` (which are connected to nodes/properties in `s2` that were reached), and so forth.

- If the strategy is `forward`, the first set of nodes/properties includes the ones connected to the `from` property, and every cycle after the new set of nodes/properties includes nodes that are closer to the `to` property.

JasperGold Apps Command Reference Manual

General Commands

-prove (Continued)

- If the strategy is backward, the first set of nodes/properties includes the ones connected to the property `to`, and every cycle after the new set of nodes/properties includes nodes/properties that are closer to the `from` property.
 - Use `-extend_graph_exploration` to run the proof until all helper properties have been determined.
 - Use `-bg` to run the proof in the background.

```
-list (connection_specs | properties)
[-silent]
```

List useful information regarding security abstractions and properties.

- Use exactly one of the following arguments with each command:
 - `connection_specs`: Lists and returns the name and connection specs for all performed abstractions with the same format required when specifying a manual abstraction.
 - `properties`: Lists and returns all security properties with the corresponding `-from` and `-to` signals and `-from` and `-to` preconditions.
- Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

JasperGold Apps Command Reference Manual

General Commands

-consistency

```
[ -task regexp_task_list
  |-property property_list
  |-from signal_list -to signal_list
  [-context task_name]]
  [-list (instance | signal)]
```

Check whether any black box on the path of any security property was not handled by an SPV abstraction.

Note: Use this command to identify black boxes that have not been abstracted. This command can help you avoid a false positive proof result.

- Use one of the following switches to run the command only for a given target:

- Use `-task` with a list of tasks. This command supports regular expressions.
 - Use `-property` with a list of properties.
 - Use `-from` and `-to` to specify a list of from signals and a list of to signals.

Note: In this case, the tool assumes only the environment abstractions. If you want the tool to consider a task's abstractions also, use `-context` with the desired task name as its argument.

- Use `-list` to report either the instances' names or the instances' ports. If you do not use `-list`, the command lists the properties that do not have abstracted black boxes along with their names and ports.

-highlight *property_list*

Generates highlight information for each property in the *property_list*. This information will be shown in the SPV Analysis Browser as follows:

- `-from fanout` – Light purple
- `-to fanin` – Light blue
- `-from fanout intersection with -to fanin` – Light green

-clear

Clear everything related to security.

Return

No value returned

Examples

```
% check_spv -create -from {source} -to {destination}
% check_spv -abstract -instances {instance_0 instance_1} -mode auto
% check_spv -abstract -instances {instance_0} -mode manual -connections {
  instance0.out1 { instance_0.inp1 comb instance_0.inp2 seq } instance_0.out2
  { instance_0.inp3 seq instance_0.inp4 seq } }
% check_spv -list abstractions -silent
% check_spv -prove
```

See Also

[elaborate](#)

check_unr

Description

Use this command to find unreachable block, toggle, or expression signals in a design.

Syntax

```
check_unr -setup
    [-coverage {block | expr | toggle | all}]
    [-task <task_name>]

check_unr -enable_property
    {<properties>}
    [-regexp]
    [-task <task_name>]

check_unr -disable_property
    {<properties>}
    [-regexp]
    [-task <task_name>]

check_unr -list
    [<properties>]
    [-type ( reachable | unreachable | bounded | unprocessed
              | enabled | disabled)]
    [-regexp <regular_expression>]
    [-task <task_name>]

check_unr -clear
    [-task <task_name>]

check_unr -prove
    [-all |-property <property_name_list> |-task <task_name>]
    [-prove_opts <prove_option_list>]
    [-silent]

check_unr -covgen
    [<run_name>]
    [-force]
    [-interval <time>]
    [-launch_imc]
    [-task <task_name>]
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<pre>-setup [-coverage {block expr toggle all}] [-task task_name]</pre>	<p>Extract and enable all UNR coverage properties in the current task.</p> <p>Use <code>-coverage</code> to filter the UNR coverage properties you want to enable.</p> <p>Use <code>-task</code> to enable UNR coverage properties in a specified task.</p> <p>Note:</p> <ul style="list-style-type: none">■ By default, the UNR App creates its own task named <code>unr</code>.■ In a UNR task, the user property or assumption is not enabled by default.■ No task is created if you have already created a task and launched this command on that task.■ No environment will be copied from the <embedded> task.■ If a UNR task already exists, then the integer number is suffixed in the UNR task name.■ Once the setup is successful, you cannot provide another <code>-setup</code> command without using the <code>-clear</code> command.
<pre>-enable_property {properties} [-regexp] [-task task_name]</pre>	<p>Enable the specified UNR coverage properties.</p> <ul style="list-style-type: none">■ Use <code>-regexp</code> to enable the UNR coverage properties that match the specified regular expression.■ Use <code>-task</code> to enable UNR coverage properties in a specified task.

JasperGold Apps Command Reference Manual

General Commands

```
-disable_property {properties}  
  [-regexp]  
  [-task task_name]
```

Disable the specified UNR coverage properties.

- Use `-regexp` to disable the UNR coverage properties that match the specified regular expression.
- Use `-task` to disable UNR coverage properties in a specified task.

```
-list
```

```
[properties]  
  [-type ( reachable | unreachable | bounded | unprocessed  
           | enabled | disabled )]  
  [-regexp regular_expression]  
  [-task task_name]
```

Display all UNR properties along with their status as described below:

- BND** – Bounded: No conclusive result found for property during UNR analysis.
- DSB** – Disabled: Property is not active for UNR analysis.
- RCH** – Reachable: Property is reachable.
- UNP** – Unprocessed: Property is not processed for UNR analysis.
- UNR** – Unreachable: Property is not reachable.
- Use `-type` to filter out UNR coverage properties based on the specified status.
- Use `-task` to list the UNR coverage properties of the specified task.
- Use `-regexp` to list the UNR coverage properties that match the specified regular expression.

```
-clear
```

```
  [-task task_name]
```

Clear all UNR coverage properties from the current task.

Use `-task` to clear UNR coverage properties in a specified task.

JasperGold Apps Command Reference Manual

General Commands

-prove

```
[-all |-property property_name_list |-task task_name]  
[-prove_opts prove_option_list]  
[-silent]
```

Find the reachable UNR coverage properties using an auto-prove strategy.

- Use `-all` to prove all UNR coverage properties in all tasks.
 - Use `-property` to prove the specified UNR coverage properties.
 - Use `-task` to prove UNR coverage properties of a given task.
 - Use `-prove_opts` to specify additional prove options you want to use during autoprove execution.
 - Use `-silent` to ignore prove messages.
-

-covgen

```
[run_name]  
[-force]  
[-interval time]  
[-launch_imc]  
[-task task_name]
```

Generate the UNICOV database for unreachable UNR coverage properties in the directory specified by the run name.

- Use `-force` to overwrite the existing UNR database.
- Use `-interval` to enable periodic generation of the UNR coverage database during `check_unr -prove` command. Specifying time as 0 disables the periodic database generation.
- Use `-launch_imc` to invoke IMC tool to view the currently generated UNR coverage database.
- Use `-task` to generate a UNR coverage database for the properties in a specified task.

Note:

- By default, the tool uses the current task name as the run name.
 - This command also generates a default IMC script to merge and analyze the UNICOV database.
-

Return

Various

Examples

```
% check_unr -setup -coverage {block expr toggle}  
% check_unr -disable_property .* -regexp  
% check_unr -enable_property .*block.* -regexp  
% check_unr -list  
% check_unr -prove  
% check_unr -covgen  
% check_unr -clear
```

See Also

No related commands

check_xprop

Description

Use this command to perform X-propagation (X-Prop) analysis on one or more signals.

This command provides a set of flows that:

- Check the clocks and resets to ensure that they are free of Xs
- Check the control structures to see which areas of the code Xs can propagate to
- Check data assignment structures to ensure that they are free of Xs
- Check the outputs to see if Xs can be propagated out of the design
- Check the black-boxed inputs to see if Xs can propagate to black-boxed instances
- Use assume-guarantee reasoning to achieve convergence on difficult X-Prop proofs

In addition to assume-guarantee reasoning, the flow also provides property grouping to take advantage of parallel processing provided by engine H and distributed processing provided by ProofGrid™. This capability is helpful when the `prove` command alone is not converging on X-Prop assertions.

Important

- This command does not guarantee convergence, but it helps in some cases. Be aware that it can require some time to complete.
- It is possible to open and interact with traces while the flow is still running.

Syntax

```
check_xprop ( ( -init_control |-init_all_control) (true | false)
              [-sequential |-expression])
              ((-init_data |-init_all_data) (true | false))
            ) +
check_xprop -create
            [-task <task_name> [-no_prefix]]
            [-from <signal_name>]
            [-to <signal_name>]
            [-clocks_and_resets]
            [-control [-instance_file_naming]]
```

JasperGold Apps Command Reference Manual

General Commands

```
[-no_flops_with_reset_pin [-include_sync_resets]]]
[-data [-instance_file_naming]]
[-outputs]
[-bbox_inputs]
[-flops_with_reset_pin [-include_sync_resets]]
[-flops_with_reset_value [-include_sync_resets]]
[-counters]
[-fsms]
[-precond <boolean_expression>]
[-from_file <file_name>]
[-bit_blast |-no_bit_blast]
[-instance <instance_name>+]
[-filter <filter_expression> [-regexp] [-case_sensitive]]]

check_xprop -prove
  [-task <task_name>]
  [-all]
  [-decompose]
  [-no_flop_sort]
  [-no_output_sort]
  [-time_limit <time_limit>]
  [-prove_opts {<prove_options_list>}]
  [-suppress_trace]
  [-detailed]

check_xprop -report
  ( source
    [ -property <property_name>]
  | reason
    [ -include_failing_properties
      |-include_location])
  [-task <task_name>]
  [-file file_name [-csv]]
  [-detailed] [-silent]

check_xprop -highlight_failure_path [-clear]
  [-task <task_name>]
  [-property <property_name>]

check_xprop -clear
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<code>-init_control (true false) [-sequential -expression]</code>	<p>Enable or disable analysis of the control structures.</p> <ul style="list-style-type: none">■ true – Enable the analysis.■ false – Disable the analysis.■ Use <code>-sequential</code> to check for X-Prop only on sequential always blocks.■ Use <code>-expression</code> to check for X-Prop by comparing all case items with case variables (still generating one property per case statement) instead of the default one property per variable in case variables or case items. <p>Note: You must run <code>check_xprop</code> with <code>-init_control</code> before loading the design.</p>
<code>-init_all_control (true false) [-sequential -expression]</code>	<p>Enable or disable analysis of the control structures, including structures of the type <code>a?b:c</code>, which are not done by <code>-init_control</code>:</p> <ul style="list-style-type: none">■ true – Enable the analysis.■ false – Disable the analysis.■ Use <code>-sequential</code> to check for X-Prop only on sequential always blocks.■ Use <code>-expression</code> to check for X-Prop by comparing all case items with case variables (still generating one property per case statement) instead of the default one property per variable in case variables or case items. <p>Note: You must run <code>check_xprop</code> with <code>-init_all_control</code> before loading the design.</p>

JasperGold Apps Command Reference Manual

General Commands

```
-init_data (true | false)
```

Enable or disable analysis of the data assignment structures:

- true – Enable the analysis.
- false – Disable the analysis.

Note: You must run `check_xprop` with `-init_data` before loading the design.

```
-init_all_data (true | false)
```

Enable or disable analysis of the data assignment structures without filtering data assignments of constants where X cannot propagate in any way:

- true – Enable the analysis.
- false – Disable the analysis.

Note: You must run `check_xprop` with `-init_all_data` before loading the design.

Creating X-Prop Assertions

-create

Create the specified X-Prop assertions in the specified task.

You can specify which signals to target with the following switches:

- -clocks_and_resets
- -control
- -data
- -outputs
- -bbox_inputs
- -flops_with_reset_pin
- -flops_with_reset_value
- -counters
- -fsms
- -to <sig>

If you do not specify target signals, this command creates assertions for the outputs of the design.

-task *task_name* -no_prefix

Create the specified X-Prop assertions in the specified task.

- Replace *task_name* with a period (.) to specify the current task.
- If you use -create without -task, the tool generates default task names, according to the type of properties it creates. For example, for -outputs, the tool creates the task XP_outputs. This naming convention makes it easy to identify the property type. However, if you specify a task, the type of the properties can be difficult to determine; therefore, in these cases, the tool adds a prefix to the name of each property to identify its type. If you do not want the tool to add this prefix, use -no_prefix.

JasperGold Apps Command Reference Manual

General Commands

-from *signal_name*

Allow the specified signal to be a source of Xs.



- Signals used with the -from switch must be inputs or stopats. They cannot be internal signals driven by some logic.
- Xs in the design are allowed to propagate in accordance with how the design was elaborated (-disable_x_handling) and the current setting for set_xprop_use_x_assignments.
- Besides signals specified in -from, sources of X are also captured from configurations defined by the following commands:
 - [set_xprop_use_all_undriven](#)
 - [set_xprop_use_bbox_outputs](#)
 - [set_xprop_use_inputs](#)
 - [set_xprop_use_internal_undriven](#)
 - [set_xprop_use_low_power](#)
 - [set_xprop_use_reset_abstraction](#)
 - [set_xprop_use_reset_state](#)
 - [set_xprop_use_stopats](#)
 - [set_xprop_use_x_assignments](#)
- Multiple drivers are always a source of X.
- Reset abstracted flops are always a source of X.

-to *signal_name*

Perform X-Prop analysis on the specified signal.

-clocks_and_resets

Perform X-Prop analysis on the clocks and resets in the design.

JasperGold Apps Command Reference Manual

General Commands

```
-control [-instance_file_naming]
[-no_flops_with_reset_pin [-include_sync_resets]]
```

Perform X-Prop analysis on the control structures in the design.

Note:

- The default name of control properties has the following format:
instance__signalName1_signalName2..signalNameN__line_x__col_y
- Use **-instance_file_naming** to specify the following format:
instance__fileName__line_x__col_y
- Use **-no_flops_with_reset_pin** to omit control checks for asynchronous flops with reset pins. And use **-include_sync_resets** to also omit them for synchronous flops.

```
-data [-instance_file_naming]
```

Perform X-Prop analysis on the data assignment structures in the design.

Note:

- The default name of data properties has the following format:
instance__signalName1_signalName2..signalNameN__line_x__col_y
- Use **-instance_file_naming** to specify the following format:
instance__fileName__line_x__col_y

```
-outputs
```

Perform X-Prop analysis on the outputs of the design.

Note: This is the default behavior.

```
-bbox_inputs
```

Perform X-Prop analysis on the inputs of any black-boxed modules.

```
-flops_with_reset_pin [-include_sync_resets]
```

Perform X-Prop analysis on flops with a reset pin.

Use **-include_sync_resets** to include flops with a synchronous reset in the X-Prop analysis. By default, the tool excludes them.

```
-flops_with_reset_value [-include_sync_resets]
```

Perform X-Prop analysis on flops with a reset value.

Use **-include_sync_resets** to include flops with a synchronous reset in the X-Prop analysis. By default, the tool excludes them.

JasperGold Apps Command Reference Manual

General Commands

-counters

Perform X-Prop analysis on counter flops of the design.

-fsm

Perform X-Prop analysis on finite state machine (FSM) flops of the design.

-precond *boolean_expression*

Limit X-Prop checking to situations where the precondition (triggering) expression is true; that is, instead of checking `hasXInOutput`, check `preconditionExpression & hasXInOutput`.

Note: The precondition must be a Boolean expression; that is, the result must be true or false.

-from_file *file_name*

Treat the signals in the specified file as allowable sources of X.



Construct the file with one signal per line.

-bit_blast

Analyze the target signals and bit blast each signal before performing X-Prop analysis.

By default, `check_xprop` does not bit blast each signal before performing X-Prop analysis. Using this switch may greatly increase the number of properties created and can lead to slower performance.

-no_bit_blast

Analyze the target signals at the word level.

This is the default behavior.

-instance *instance_name...*

Only analyze targets of the specified type from within the specified instance or instances.

By default, the tool analyzes the entire design.

JasperGold Apps Command Reference Manual

General Commands

```
-filter filter_expression [-regexp] [-case_sensitive]
```

Generate properties matching *filter_expression* based on the property names using case-insensitive wildcards (default).

- Use `-regexp` to change the filter expression to a case-insensitive regular expression.
 - Use `-case_sensitive` to filter output using case-sensitive wildcards or regular expressions.
-

Proving X-Prop Assertions

```
-prove
```

```
[-task task_name] [-all] [-decompose] [-no_flop_sort]  
[-no_output_sort] [-time_limit time_limit]  
[-prove_opts {prove_options_list}] [-suppress_trace] [-detailed]
```

Attempt to prove the X-Prop assertions.

- Use `-task` to prove the X-Prop assertions in the specified task.
 - Use `-all` to prove the X-Prop assertions in all tasks.
 - Use `-decompose` to instruct the tool to perform assume-guarantee reasoning.
- Note:** By default, the tool does not perform assume-guarantee reasoning. Use this switch to turn on this feature.
- Use `-no_flop_sort` to instruct the tool not to sort the registers in the COI of a `-to` signal before beginning the analysis.
 - Use `-no_output_sort` to instruct the tool not to sort the `-to` signals before beginning the analysis.
 - Use `-time_limit` to restrict the `check_xprop` analysis to the specified wall-time limit.
-

Note:

- `check_xprop` typically performs many proofs, and the time used for each proof is still governed by the `prove_time_limit` and `per_property_time_limit*` configuration variables.
 - By default, the `check_xprop` time limit is set to unlimited.
 - The `set_cmd_time_limit` command has no effect on `check_xprop -prove`.
-

JasperGold Apps Command Reference Manual

General Commands

-prove (Continued)

- Use `-prove_opts` to specify additional `prove` options you want to use during `autoprove` execution.
 - Use `-suppress_trace` to suppress traces for counterexamples.
Note: To generate traces on demand, use the `visualize` command. This switch is a performance optimization that might help with difficult proofs.
 - Use `-detailed` to provide information as to which registers are X-propagatable and which are not.
-

X-Prop Reporting

```
-report source {-property property_name |-task task_name]  
[-detailed] [-file file_name [-csv]] [-silent]
```

Report the X sources configuration the tool will use in the X-Prop analysis for the current task.

This report prints the status (on or off) of each kind of X source that can be enabled or disabled with `set_xprop_use_*` Tcl commands and, for each enabled kind, the names of the signals that will be sources of X during X-Prop analysis. It also prints the sources of X that cannot be disabled (multiple drivers, reset-abstracted flops, X-Prop assumptions, and X-Prop properties with a specific `-from` signal).

- Use `-property` to list just the sources in the analysis of the specified property.
- Use `-task` to list the X sources for a specified task. If you do not specify a task, this command applies to the current task.
- Use `-detailed` to print all possible sources of X, including those for disabled kinds of X source. With this switch, the report can include sources that will not be used in the X-Prop analysis.
- Use `-file` to save the report output to a specified file and use `-csv` to save the report output in CSV format.
- Use `-silent` to return the report as Tcl return values without printing the report on the screen. This option is helpful in Tcl scripts.

The output format is as follows:

```
<x source type:>  
  <x signal>
```

Also see the [“Examples”](#) on page 450 section below.

JasperGold Apps Command Reference Manual

General Commands

```
-report reason  
[-task task_name]  
[-include_failing_properties |-include_location |-detailed]  
[-file file_name [-csv]] [-silent]
```

Report the reasons for X-Prop failures.

This report prints the x signal name, its type (as specified with the commands `set_xprop_use_*`, from multiple drivers, or from reset-abstracted flops), and the number of properties that failed due to this signal.

- Use `-task` to list the X sources for a specified task. If you do not specify a task, this command applies to the current task.
- Use `-include_failing_properties` to include a list of the failing properties.
- Use `-include_location` to include a list of the failing properties and the source code file and line number of the X assignment for each property that failed due to X assignment.
- Use `-detailed` to create a report that includes all of the information specified by the switches above.
- Use `-file` to save the report output to a specified file and use `-csv` to save the report output in CSV format.
- Use `-silent` to return the report as Tcl return values without printing the report on the screen. This option is helpful in Tcl scripts.

The output format is as follows:

```
<x signal> (<x reason>, #Failing Properties: <number of  
properties failing due to this signal>) [<list of failing  
properties> (# cycles)]
```

Also see the [“Examples”](#) on page 450 section below.

JasperGold Apps Command Reference Manual

General Commands

```
-highlight_failure_path [-clear]
[-task task_name]
[-property property_name]
```

Compute failure path highlights for display in the *X-Propagation Analysis Browser*.

- Use `-task` to compute the failure path highlight of all properties of the specified task. If you do not specify a task, this command applies to the current task.
- Use `-property` to compute the failure path highlights for the specified property.
- Use `-clear` to clear all computed failure path highlights, highlights of specific tasks (`-task`), or highlights of specific properties (`-property`).



- Failure path highlights represent the recursive *Why* of a violation trace. They indicate the X path in the source code with red highlights.
- Note:** These highlights depend on the generated trace.
- The failure path highlight is not computed automatically. It occurs when you do one of the following:
 - Request failure path highlights with this command.
 - Click the button *Compute X Sources* in the X-Prop Failure Analysis dialog.
 - Choose the menu option *Application – Highlight – Compute All Failure Path Highlights*.

```
-clear
```

Clear the `check_xprop` data structures.

Return

No return value.

Examples

```
% check_xprop -init_control true -sequential
% check_xprop -create -clocks_and_resets -control
% check_xprop -create -to data_out -precond data_out_valid -from data_in
% check_xprop -prove
% check_xprop -clear

# X Sources Report Example:
[<embedded>] % check_xprop -report source -task XP_outputs
X Sources Report
-----
Target: TASK: XP_outputs
Uninitialized Registers (on): 0
BBoxed Outputs (on): 0
Primary Inputs (off)
Stopats (off)
Internal Undriven (on): 1
    cout
Low Power (off)
X-Assignments (on): 0
-----
Multiple Driver: 0
Flop Reset Abstractions: 0
Assumes: 0
XProp Properties with specific -from: 0

# Detailed X Sources Report Example:
[<embedded>] % check_xprop -report source -task XP_outputs
X Sources Report
-----
Target: TASK: XP_outputs
Uninitialized Registers (on): 0
BBoxed Outputs (on): 0
Primary Inputs (off): 1
    ain
Stopats (off): 0
Internal Undriven (on): 1
    cout
Low Power (off): 0
X-Assignments (on): 0
```

JasperGold Apps Command Reference Manual

General Commands

```
-----
Multiple Driver: 0
Flop Reset Abstractions: 0
Assumes: 0
XProp Properties with specific -from: 0

# X Reasons Report Example:
[<embedded>] % check_xprop -report reason
X Reasons Report
-----
Target: TASK: <embedded>

brdg.int2ig_data[0] (BBoxed Outputs, #Failing Properties: 4)
brdg.int2ig_data[10] (BBoxed Outputs, #Failing Properties: 4)
brdg.int2ig_data[11] (BBoxed Outputs, #Failing Properties: 4)

.
.

.

eg.eg_ad_dataout[5] (Uninitialized Registers, #Failing Properties: 1)
eg.eg_ad_dataout[6] (Uninitialized Registers, #Failing Properties: 1)
size0[0] (Primary Inputs: 1)
size0[1] (Primary Inputs: 1)
valid0 (Primary Inputs: 4)

# Detailed X Reasons Report Example:
[<embedded>] % check_xprop -report reason -detailed
X Reasons Report
-----
Target: TASK: <embedded>

brdg.int2ig_data[0] (BBoxed Outputs, #Failing Properties: 4)
XP_outputs::rdata3[0] (2 cycles)
XP_outputs::rdata2[0] (2 cycles)
XP_outputs::rdata1[0] (2 cycles)
XP_outputs::rdata0[0] (1 cycles)

.
.

.

valid0 (Primary Inputs, #Failing Properties: 4)
XP_outputs::ready3 (2 cycles)
XP_outputs::eg_valid (2 cycles)
```

JasperGold Apps Command Reference Manual

General Commands

```
XP_outputs::ready0 (6 cycles)
XP_outputs::ready2 (2 cycles)
valid1 (Primary Inputs, #Failing Properties: 1)
XP_outputs::ready1 (3 cycles)

.
.
.
```

See Also

[elaborate](#)

[set_xprop_use_all_undriven](#)
[set_xprop_use_bbox_outputs](#)
[set_xprop_use_inputs](#)
[set_xprop_use_internal_undriven](#)
[set_xprop_use_low_power](#)
[set_xprop_use_reset_abstraction](#)
[set_xprop_use_reset_state](#)
[set_xprop_use_stopats](#)
[set_xprop_use_x_assignments](#)

clear

Description

Use this command to clear the proof and/or GUI information. If you do not use any switches the tool preserves the analyzed and elaborated design and some caches, for example, the proof simplification cache and the engine L cache. Be aware that if you previously specified clocks or reset, `clear` without switches also clears them, so you will need to specify them again before moving forward with your session.

Use switches to customize the clear operation, for example, to clear the analysis region and any prior proof information for a specified task or to clear a cache.

Syntax

```
clear [-task <task_name>] [-result]
      | [-engineL_cache]
      | [-cache_proof_simplification]
      | [-all]
```

Argument	Definition
<code>-task task_name</code>	Clear the analysis region and any prior proof information in the specified task. Note: <ul style="list-style-type: none">■ This command clears task stopats, justifies, and counter abstractions.■ Replace <i>task_name</i> with a period (.) to specify the current task.
<code>-result</code>	Clear the proof and Visualize results without clearing the setup (for example, the analysis region).
<code>-engineL_cache</code>	Clear the engine L cache. See “ set_engineL_cache ” on page 1060.
<code>-cache_proof_simplification</code>	Clear the proof simplification cache. See “ set_cache_proof_simplification ” on page 960.

JasperGold Apps Command Reference Manual

General Commands

-all

Clear GUI windows launched from the current session, such as the Visualize window and the Source Browser.



- This command applies to the current session, that is, if you are running multiple sessions, the others are not affected.
- This command has no effect on the database.
- This command does not clear Proof Accelerator data structures. Use the Proof Accelerator command's `-clear` switch instead.

Return

No value returned

Examples

```
% clear  
% clear -result  
% clear -task .
```

See Also

[set_cache_proof_simplification](#)
[set_engineL_cache](#)

clock

Description

Use this command to specify a global clock configuration. You can also use this command to list all clock configurations and to analyze the clock tree. The analysis can provide the information you need to specify the clock environment properly.

Before you can run a proof or Visualize a target, you must specify at least one of the following commands:

- Use `clock <clk>` to declare a signal as a clock.
- Use `clock -none` for a free-running clock environment.
- Use `clock -infer` to automatically infer primary clocks from every flop it encounters.

The `clock` command only considers global stopats and assumptions (specified with `assume -env`). Task-specific information, including assumption and stopat information, does not affect inferred clocks.

Important

- Use signal names for posedge clocks and signal negations for negedge clocks.
- For clocks with both posedge and negedge activities, use the `-both_edges` option. Refer to the `-both_edges` definition for additional considerations.
- Do not use expressions.

Syntax

```
clock <primary_clock> [-both_edges]
clock <clock_1> <clock_tcl_list> [<factor> [<phase>]]
    [-both_edges]
clock -rate <input_tcl_list> <clock> [-both_edges]
clock <clock_name> -factor <N> [-phase <N>]
clock <clock_name> -both_edges -pattern <pattern>
clock -clear
clock -analyze [-include_latches] [-typed_list] [-silent]
```

JasperGold Apps Command Reference Manual

General Commands

```
clock -list [signal | configuration] [-silent]
clock -infer
clock -none
clock -domain
( -signal_list <clocking_signal_list> [-target_domain <domain_ID>]
| -list [-target_domain <clocking_signal>]
| -clear)
```

Argument	Definition
Specifying a Single Clock	
<i>primary_clock</i> [-both_edges]	<p>The specified clock is the primary clock for the design.</p> <p>Implementation guidance:</p> <ul style="list-style-type: none">■ For Verilog designs, use <code>~<primary_clock></code> for negedge clocks. For VHDL designs, use <code>clock {not <primary_clock>}</code> unless you elaborate with <code>-mode verilog</code>.■ Use <code>-both_edges</code> if the analysis region has one of the following conditions:<ul style="list-style-type: none">□ Some flops trigger on the posedge of the clock and others trigger on the negedge of the clock.□ The high phase of <code>clk_in</code> enables a latch. <p>Note: Only use <code>-both_edges</code> if the tool prompts you to do so when running the <code>prove</code> or <code>visualize</code> command.</p>

Specifying Multiple Clocks

```
clock_1 clock_tcl_list  
[factor [phase]]  
[-both_edges]
```

Specifies the relationship of the specified clock to one or more specified clocks:

- *clock_1* – Base clock for the current analysis.
- *clock_tcl_list* – One or more derived clocks specified for the current analysis.
- *factor* – Integer ratio of base clock frequency to the frequency of one or more derived clocks (default 1).
- *phase* – Use an integer to indicate the relative phase between clocks. The derived clock will have its first positive edge at the Nth positive edge of the base clock (default 1).

Note:

- When used with *clock_1 clock_tcl_list*, *-both_edges* applies to the listed clocks.
- This command accepts lists of signals and Tcl lists.

Configuring the Rate of Change of an Input to an Analysis Region

```
-rate [input_tcl_list] [clock] [-both_edges]
```

The specified *input_tcl_list*, which is a boundary net to the current analysis region, changes value at the rate of the specified *clock*.

Use *-both_edges* to specify that you want inputs rated on both edges.



- Specified signals can be either signals or regular expressions.
- Specified signals must be free inputs or environmental stopats.
- In the case of buffers, the tool uses their equivalent free inputs or environmental stopats.
- Any signal that does not exist, triggers a warning.
- This command accepts lists of signals and Tcl lists.
- By default, *clock -rate* applies to the rising edge of the specified clocks. To apply the command to the falling edge, invert the clock (*~clk*).

Return: The signals that were added to the *-rate* list.

Specifying Clock Ratios

```
clock_name -factor N [-phase N]
```

The specified *clock_name* has the specified factor.

Use *-phase* with an integer to specify the relative phase between clocks.

Note:

- If you have declared a primary clock, the specified clock has its first positive edge at the Nth positive edge of the primary clock.
- If you have not declared the primary clock, the specified clock has its first positive edge between cycles N and N+1.

Specifying Clock Behavior with Patterns

`clock_name -both_edges -pattern pattern`

The specified `clock_name` behaves like the wave described by `pattern`.

The argument `pattern` is a sequence of 0's and 1's, where each "0" corresponds to a cycle where `clock_name` will remain low and each "1" corresponds to a cycle where `clock_name` will be high.

Currently, every clock declared with `clock -pattern` must be allowed to sample registers on both of its edges. Therefore, the switch `-both_edges` is mandatory.

Note:

- Clock inversion is not supported when using the `-pattern` switch. For example, the following command triggers an error: `clock ~clk -both_edges -pattern 01`
- A patterned clock cannot be the base clock for another clock declaration. For example, the following command set is not supported:
`% clock clk -both_edges -pattern 01`
`% clock clk clk1`
- Pattern clocks do not rate inputs by default. When you use `clock -pattern` and you want an input to change only on the rising edge of the clock, use `clock -rate <input> <pattern_clk_name>`. Without `clock -rate`, inputs are allowed to change at every cycle, that is, at every digit of the pattern.

Utility Switches

`-clear`

Remove all existing clock configurations from the global environment.

Use this command to clear your clock declarations before attempting to specify a new base clock.

JasperGold Apps Command Reference Manual

General Commands

```
-analyze [-include_latches] [-typed_list] [-silent]
```

Analyze the design's clock structure and return a report that includes inferred clocks for the global environment.

- Use `-include_latches` to include clocking signals from latches in the analysis report and the Tcl return.
- Use `-typed_list` to return a list of all information contained in the clock report summarized as Tcl return values.
- Use `-silent` to return the list as Tcl return values without printing the list on the screen. This option is helpful in Tcl scripts.

Note:

- The tool automatically extracts inferred clocks when you use the command `clock -infer`. They are based on setup (that is, environment assumptions and stopats and `clock` command options).
- The Tcl return includes the following:
 - All declared clocks that are in the combinational COI of the clocking signals.
 - All clocking signals, including constant clocks, gated clocks, and those driven by a complex logic clock.

The clock report includes the following information:

- Signal name
 - Status – defined, undefined, or propagated
 - Type – simple clock, AND-gated tree, and so forth
 - Number of PSL/SVA flops connected to the clock
 - Number of design flops connected to the clock
 - Names and types of signals driving the clock
-

JasperGold Apps Command Reference Manual

General Commands

```
-list [signal | configuration] [-silent]
```

List the user-defined clock configurations for the global environment.

- Use `signal` to list all clock signal names, suppressing the sampling edge information.
- Use `configuration` to list the full clock configuration, including clock phase, periods, clock patterns, and rated inputs. You can use each element in the list as an argument to the `clock` command to recreate the clock configuration.
- Use `-silent` to turn off output to the screen and return clock configurations with a Tcl return value. This option is helpful in Tcl scripts.

```
-infer
```

Execute clock inference explicitly and declare and return all inferred clocks for the global environment.

The tool automatically extracts all inferred clocks by analyzing primary inputs that are connected to flip-flop clock pins.

```
-none
```

Allow the tool to run with free clocks.

Note: To run a proof or Visualize a target, you must either have declared your clocks or used `clock -none`.

JasperGold Apps Command Reference Manual

General Commands

-domain

```
( -signal_list clocking_signal_list [-target_domain domain_ID]
  | -list [-target_domain clocking_signal]
  | -clear)
```

Assign specified signals to a new clock domain, list information about the current clock domain, or clear previous reassignments.

- Use **-signal_list** to specify the list of clocking signals that will be removed from their current clock domain and added to another domain. If you do not specify a domain with **-target_domain**, the tool creates a new clock domain.
 - **Note:** Use this command if you want to redefine clock domains, that is, change the clock domain definitions that were built based on the clock tree analysis.
 - Use **-list** to list the current clock domain's information. And use the **switch -target_domain** to return the clock domain ID of the specified clocking signal.
 - Use **-clear** to revert your previous reassignments.
-

Return

`clock -analyze` returns a Tcl list containing inferred clock signals. If procedural assignments are sensitive to both edges of a clocking signal, the return follows the example below:

```
+{<clocking_signal> ~<clocking_signal>}
```

Examples

```
% clock clk {in1 in2}
% clock clk {in1 in2} 2 1

// Supported formats for clock -rate:
% clock -rate sig1 sig2 sig3 clk
% clock -rate {sig1 sig2 sig3} clk
% clock -rate {sig1} {sig2} {sig3} clk
% clock -rate {sig1 sig2 sig3} clk -both_edges

// In the following command, the ratio of the periods of clk1 to clk2 is 4:1,
// and both the primary and derived clocks tick together right after reset.
% clock clk1 clk2 4 1
```

JasperGold Apps Command Reference Manual

General Commands

```
// In the following command, -both_edges applies to clkB
% clock clkA clkB 2 1 -both_edges

// In the following command, the ratio of the periods of clk2 to clk3 is 2:3.
% clock clk2 -factor 2
% clock clk3 -factor 3

// The following clock declaration will force clk to behave like a square wave
// with a 25% duty cycle.
% clock clk -pattern 0100 -both_edges
```

See Also

[abstract](#)
[assume](#)
[prove](#)
[reset](#)
[sanity check](#)

configure

Description

Use this command to reset all or specified limits to defaults or list the current values of all JasperGold Apps variables.

Note: Use the `set_*` and `get_*` commands to set and list individual variables. Refer to [Chapter 5, “Configuration Commands”](#).

Syntax

```
configure -default  
        |-list  
        |-list_changed  
        |(-unlimit [ all | proof | trace | cmd | synthesize ] )
```

Argument	Definition
-default	Reconfigure all settings to their default values.
-list	List the current values of all JasperGold-specific variables.
-list_changed	List the settings that the user has changed.

JasperGold Apps Command Reference Manual

General Commands

```
-unlimit [all | proof | trace | cmd | synthesize]
```

Remove limits using the following settings:

- all (*This option is the default.* It removes limits in all of the following categories.)
 - proof
 - prove_time_limit = 0s
 - prove_per_property_time_limit = 0s
 - trace
 - max_trace_length = 0
 - cmd
 - cmd_time_limit = 7200s
 - synthesize
 - task_compile_time_limit = 120s
 - property_compile_time_limit = 5s

Note: In the case of the configuration variables `task_compile_time_limit` and `property_compile_time_limit`, the `-unlimit` switch resets the limit to the highest supported value, which is 10000s.

Return

No value returned

Examples

```
configure -default  
configure -unlimit cmd  
configure -list_changed  
configure -unlimit proof
```

See Also

Refer to the commands in [Chapter 5, “Configuration Commands.”](#)

connect

Description

Use this command to do one of the following:

- Instantiate and connect a module or entity containing a set of requirements from analyze -req.
- Instantiate and bind a module or entity to the top module or an instance inside the top module.
- Instantiate and connect a Proof Accelerator (PA) to the design.

Note: This command clears any previous proof and Visualize™ results.

Syntax

```
connect <unit_name> <instance_name> [-auto]
      (-connect <formal_name> <actual_name>) *
```

-bind Switches

```
connect -bind <unit_name> <instance_name> [<instance_to_bind_to>]
      (-parameter <param_name> <param_value>)*
      [-auto] (-connect <formal_name> <actual_name>)*
      [-elaborate] [-vhdl]
```

Utility Switches

```
connect -elaborate [-vhdl]
connect -list [-silent] [<instance_name>]
connect -remove <instance_name>
connect -clear
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<pre>unit_name instance_name [-auto] (-connect formal_name actual_name) *</pre>	<p>Instantiate and connect the specified module or entity containing a set of requirements from analyze -req.</p> <ul style="list-style-type: none">■ <i>unit_name</i> is the name of an elaborated module or entity.■ <i>instance_name</i> is the unique instance name you want to use to instantiate an elaborated module or entity.■ Use <i>-auto</i> to automatically connect the input port of the specification module to the design signal with the same name. <p>Note: The tool does not autoconnect signals of different widths.</p> <ul style="list-style-type: none">■ Use <i>-connect</i> to connect the specified port (<i>formal_name</i>) from the specification module to a specified signal or port in the design (<i>actual_name</i>). <p>Note: If you are connecting to a signal or port of an instance, you cannot specify <i>actual_name</i> signals that are in the top module.</p>

-bind Switches

Implementation guidance for -bind:

- This command supports connections to constants, multibits, and expressions.
- When connecting to an expression, use the full hierarchical path for signals. (The command does not support relative paths.)
- You must specify the `-auto` switch to automatically connect the input port of the specified module to the design signal with the same name.
- It is not possible to connect a Verilog and a VHDL module in the same `connect -elaborate` command. You must use two separate `connect -elaborate` commands.

Example:

```
% analyze -verilog top.v
% analyze -verilog vlogmod.v
% analyze -vhdl vhdlmod.vhdl
% elaborate -top top
% connect -bind vlogmod mod1 -auto -elaborate
% connect -bind vhdlmod mod2 -auto -elaborate -vhdl
```

`-bind unit_name instance_name [instance_to_bind_to]`

```
(-parameter param_name param_value) +
[-auto] (-connect formal_name actual_name) +
[-elaborate] [-vhdl]
```

Instantiate and bind the specified module or entity to the top module or an instance inside the top module.

- *unit_name* is the name of an elaborated module or entity.
- *instance_name* is the unique instance name you want to use to instantiate an elaborated module or entity.
- *instance_to_bind_to* is the unique instance name for the instance you want to bind the new instance to.
- Use `-parameter` to specify the parameter values for the instance that is being created.
- Use `-auto` to automatically connect the input port of the specification module to the design signal with the same name.

Note: The tool does not autoconnect signals of different widths.

JasperGold Apps Command Reference Manual

General Commands

-bind (Continued)

- Use `-connect` to connect the specified port (*formal_name*) from the specification module to a specified signal or port in the design (*actual_name*).

Note: If you are connecting to a signal or port of an instance, you cannot specify *actual_name* signals that are in the top module.

- Use `-elaborate` to execute the `-bind` specification you entered previously and use `-vhdl` to bind a VHDL module.

After a successful elaboration, you can end the separate frontend process (`jg_frontend`) to free the associated memory with the `analyze -clear` command; however, if you end the `jg_frontend` process beforehand, subsequent `connect -elaborate` commands result in an error.

Utility Switches

`-elaborate [-vhdl]`

Execute the `-bind` specification you entered previously.

Use `-vhdl` to bind a VHDL module.

After a successful elaboration, you can end the separate frontend process (`jg_frontend`) to free the associated memory with the `analyze -clear` command; however, if you end the `jg_frontend` process beforehand, subsequent `connect -elaborate` commands result in an error.

`-list [-silent] [instance_name]`

List information for the instances that are connected with the `connect` command.

- Use the `-silent` switch to show on the screen only the list of instances connected and return them as Tcl return values. This option is helpful in Tcl scripts.
- Supply an instance name if you want to list connection information for a specified *instance_name*.

Note: This switch does not list instances connected with `-bind`.

`-remove instance_name`

Remove a previously created `connect` instance.

Note: This switch does not remove instances connected with `-bind`.

JasperGold Apps Command Reference Manual

General Commands

-clear

Note: Remove all connect instances.

This switch also removes instances connected with -bind, but you must run connect -elaborate to complete the clear process.

Return

This command returns the specified instance name.

Examples

```
% connect pci_Requirement Requirement_inst \
-connect clk PCI_CLK_IN \
-connect gntN PCI_GNTn_IN \
-connect idsel PCI_IDSEL_IN \
-connect mAd PCI_AD_OUT \
-connect sAd PCI_AD_IN \
-connect mAd_enable 1'b1
% connect -remove Requirement inst
% // Binding a Formal Scoreboard to the DUV
% connect -bind jasper_scoreboard_2 my_pa -connect {clk1} {clk} \
{LOG_PKT_LATENCY} {5} -parameter {PKT_LATENCY} {7} -elaborate
```

See Also

[analyze](#)
[elaborate](#)

cover

Description

Use this command to add a cover directive about the signals in the design. This command does *not* initiate an attempt to analyze whether the expression is covered. Use the `prove` command to verify the cover directive.

By default, this command applies to the current task and does not automatically update the database. Use `-task` to specify a different task.

Syntax

[Creating a Cover](#)

```
cover [-task <task_name> |-name <name> |-name <task_name>::<name>]
      <expression>
      [-label <label>] [-annotation '{'<annotation>'}']
      [-update_db |-update_all_sessions |-suppress_db_update]
```

[Editing a Cover](#)

```
cover -replace
      -name <name>
      |-name <task_name>::<name>
      <expression>
      [-label <label>] [-annotation '{'<annotation>'}']
      [-update_db |-update_all_sessions |-suppress_db_update]
```

[Path Sensitizing Analysis](#)

```
cover -path
      -from <input_signal_tcl_list>
      [-to <barrier_signal_tcl_list>]
      [-task <task_name> |-name <name> |-name <task_name>::<name>]
      [-annotation '{'<annotation>'}']
```

[Utility Switches](#)

```
cover -list [-silent] [-type path]
            [-task <task_name>]

cover -clear
      [-task <task_name>]
```

JasperGold Apps Command Reference Manual

General Commands

```
cover -show <name>
| -remove <name>+ [-regexp]
| -disable <name>+ [-regexp]
| -enable <name>+ [-regexp]
| -rename <old_name> <new_name>
  [-update_db | -update_all_sessions | -suppress_db_update]
```

Argument	Definition
Creating a Cover	
	<pre>-task task_name -name name -name task_name::name expression -label label -annotation {annotation} -update_db -update_all_sessions -suppress_db_update</pre> <p>Create a cover directive for the specified Boolean expression that evaluates to either true or false.</p> <p>Use the <code>prove</code> command to verify this cover.</p> <p>Note: JasperGold Apps supports SVA expressions on the command line. Refer to Appendix D, “Tcl Support for SVA Expressions” for guidance.</p> <p>Implementation guidance:</p> <ul style="list-style-type: none">■ Use <code>-task</code> to apply the command to the specified task instead of the current task.■ Use <code>-name</code> to assign a name to the property.■ Use <code>-label</code> to add an element heading. The GUIs will display this heading instead of the property name.■ Use <code>-annotation</code> to annotate the specified property with the description enclosed by the curly braces. You can view annotations in the Visualize window <i>Waveforms</i> pane by positioning the mouse to hover over the property and reveal a tooltip. And you can print annotations to the screen with the <code>get_annotation</code> command.■ Use <code>-update_db</code> to add the specified assumption to the database.

Creating a cover (Continued)

- Use `-update_all_sessions` when you are running multiple sessions and you want to add the specified assumption to all of them and the database.
- Use `-suppress_db_update` when you have used `set_property_auto_update_db` on to set the tool to update the database automatically but you do not want to include the specified property.

Note:

- Do not use * or ? in property names.
- Do not use the `-task` and `-name` switches in the same command. To specify both a `name` and `task_name`, use the `-name` switch and prepend the property name with `<task>::`. (Replace `<task>` with an actual task name.) Example:

```
cover -name taskA::A {~rdy => ~trx}
```

Editing a Cover

```
-replace
  -name name
  |-name task_name:::name
  expression
  -label label -annotation {annotation}
  -update_db |-update_all_sessions |-suppress_db_update
```

Replace the expression for the specified cover.

Use this command to edit a cover used as a Visualize target without triggering a `-clear_all`. Replot the trace after you use this command.

Implementation guidance:

- Use `-name` to specify the cover you are editing.
- Use `-label` to add or change the property's element heading.
- Use `-annotation` to add or change the property's annotation.
- Use `-update_db` to add the specified assumption to the database.
- Use `-update_all_sessions` when you are running multiple sessions and you want to add the specified assumption to all of them and the database.
- Use `-suppress_db_update` when you have used `set_property_auto_update_db on` to set the tool to update the database automatically but you do not want to include the specified property.

Path Sensitizing Analysis

```
-path
  -from input_signal_tcl_list
  -to barrier_signal_tcl_list
  -task task_name |-name name |-name task_name::name
  -annotation {annotation}
```

Generate a cover property that checks whether the value of a signal can be propagated to another signal, that is, whether one signal can control another signal.

- Use `-from` to select the originating signals for the path.
- Use `-to` to select the end points for the path.
- If you do not use `-to`, the tool uses the list of output signals.
- Use `-task` to assign the property to a task other than the current task.
- Use `-name` to assign a name to the property.
- Do not use * or ? in property names.
- Do not use both `-task` and `-name`. To specify both, use the `-name` switch and prepend the property name with `<task>::`. (Replace `<task_name>` with an actual task name.)
- Use `-annotation` to annotate the specified property with the description enclosed by the curly braces. You can view annotations in the Visualize window *Waveforms* pane by positioning the mouse to hover over the property and reveal a tooltip. And you can print annotations to the screen with the `get_annotation` command.

JasperGold Apps Command Reference Manual

General Commands

-path (Continued)



- You can use ProofGrid to prove multiple -path properties in parallel. To enforce serial processing, put properties in different tasks.
- Path properties are not supported with tunneling. The tool ignores these properties when verifying tasks that contain them.
- The -from and -to switches support Tcl lists and lists of signals that can be formed using wildcards, for example:

```
cover -path -from io_dram?_data_valid -to dram_io_cas0_1
cover -path -from io_dram*_data_valid -to dram_io_cas0_1
```
- If you use lists, the tool generates one Cover (path) property. You can interpret its proof results as follows:
 - Covered: There is a path from ONE of the -from signals to ONE of the -to signals.
 - Unreachable: There is no path from ANY of the -from signals to ANY of the -to signals.
- Signals used with the -from switch must be inputs or stopats. They cannot be internal signals driven by some logic.

Utility Switches

```
-list [-silent] [-type path]
[-task task_name]
```

Show the list of cover directives in the current or specified task.

- Use -silent to turn off output to the screen and return cover directives with a Tcl return value. This option is helpful in Tcl scripts.
- Use -type path to return only the list of covers generated by path sensitizing analysis.
- Use -task to apply the command to the specified task instead of the current task.

```
-clear {-task task_name}
```

Remove all interactive cover directives (that is, covers you specified on-the-fly).

JasperGold Apps Command Reference Manual

General Commands

-show name

Show the actual expression for the cover directive with the specified name.

Refer to the definitions for `cover -name`.

-remove name+ [-regexp]

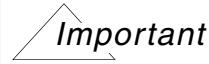
Remove the cover directive with the specified name.



- Specify one or more covers.
- Wildcards and regular expressions are supported; for example:
`cover -remove {^.*$} -regexp`
- You can only remove properties created with the GUI or on the command line. That is, you cannot remove properties added automatically (for example, SVA and PSL embedded properties). However, you can disable and enable these properties.

-disable name+ [-regexp]

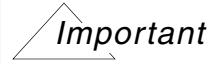
Keep the cover directive, but do not include it in any analysis.



- Specify one or more covers.
- Wildcards and regular expressions are supported; for example:
`cover -disable {^.*$} -regexp`

-enable name+ [-regexp]

Re-enable the specified (previously disabled) cover directive.



- Specify one or more covers.
- Wildcards and regular expressions are supported; for example:
`cover -enable {^.*$} -regexp`

JasperGold Apps Command Reference Manual

General Commands

```
-rename old_name new_name
  [ -update_db
    | -update_all_sessions
    | -suppress_db_update]
```

Change *old_name* cover directive to *new_name*.

- Use `-update_db` to add the specified assumption to the database.
- Use `-update_all_sessions` when you are running multiple sessions and you want to add the specified assumption to all of them and the database.
- Use `-suppress_db_update` when you have used `set_property_auto_update_db` on to set the tool to update the database automatically but you do not want to include the specified property.

Return

Returns a string (identifier name) for the cover directive.

Examples

```
% cover a==b
% cover -name illegal_size {wr_size != 2'b11}
% cover {scan_enable == 1'b0}
% cover -name illegal_size {WR_SIZE /= "11"}
% cover {SCAN_ENABLE = '0'}
```

Note: JasperGold Apps supports the implication overlapping operator `->` on the command line to declare properties. This operator indicates that the expressions on both sides happen at the same iteration.

```
% cover {~rdy -> ~trx}
% cover {(rdy = '1') -> (trx = '1')}
```



```
# Pattern-matching with regular expressions:
# -----
# Disable cover properties that match zero or more characters followed by
# either "_ended" or "_started"
% cover -disable {.*(_ended|_started)} -regexp
```



```
# Remove cover properties that match "trans" followed by 1 or more numbers and
```

JasperGold Apps Command Reference Manual

General Commands

```
# followed by zero or more characters
% cover -remove {trans([0-9]+).*} -regexp
```

See Also

[assert](#)
[assume](#)

custom_gui_action

Description

Use the `custom_gui_action` command to add, remove, or list custom button(s) from the main JasperGold Apps window or Visualize windows.

Note: This feature is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version

Syntax

```
custom_gui_action -add <button_name>
    -window ( visualize | arch | bps | conn | cov | csr | dld
               | fpv | lpv | rtld | sec | spec | sps | spv | unr
               | xprop)
    -script <tcl_script>
    [-tooltip <tooltip_string>]
    [-iconfile <path_of_icon_file>]

custom_gui_action -remove <button_name>
    -window ( visualize | arch | bps | conn | cov | csr | dld
               | fpv | lpv | rtld | sec | spec | sps | spv | unr
               | xprop)

custom_gui_action -list
    -window ( visualize | arch | bps | conn | cov | csr | dld
               | fpv | lpv | rtld | sec | spec | sps | spv | unr
               | xprop)
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<pre>-add button_name -window (visualize arch bps conn cov csr dld fpv lpv rtld sec spec sps spv unr xprop) -script tcl_script [-tooltip tooltip_string] [-iconfile path_of_icon_file]</pre>	<p>Add a custom button to the specified window.</p> <ul style="list-style-type: none">■ Use <code>-window</code> to specify the window or app view to add this button to.<ul style="list-style-type: none">□ <code>visualize</code> – JasperGold Visualize window□ <code>arch</code> – Architectural Modeling App□ <code>bps</code> – Behavioral Property Synthesis App□ <code>conn</code> – Connectivity Verification App□ <code>cov</code> – Coverage App□ <code>csr</code> – CSR Verification App□ <code>dld</code> – Deadlock Detection App□ <code>fpv</code> – Formal Property Verification App□ <code>lpv</code> – Low Power Verification App□ <code>rtld</code> – RTL Development App□ <code>sec</code> – Sequential Equivalence Checking App□ <code>spec</code> – Executable Specification App□ <code>sps</code> – Structural Property Synthesis App□ <code>spv</code> – Security Path Verification App□ <code>unr</code> – UNR Analysis App□ <code>xprop</code> – X-Propagation Verification App■ Use <code>-script</code> to specify the Tcl script you want the tool to run when you click the button.

JasperGold Apps Command Reference Manual

General Commands

-add (Continued)

- Use `-tooltip` to specify the tooltip you want the tool to display on mouse hover over the button.
- Use `-iconfile` to specify the PNG image you want to use for the button icon.

-remove *button_name*

```
-window ( visualize | arch | bps | conn | cov | csr | dld  
          | fpv | lpv | rtld | sec | spec | sps | spv | unr  
          | xprop)
```

Remove the specified custom button from the specified window.

Use `-window` to specify the window or app view you want to remove this button from.

-list

```
-window ( visualize | arch | bps | conn | cov | csr | dld  
          | fpv | lpv | rtld | sec | spec | sps | spv | unr  
          | xprop)
```

Return a list of button names for the specified window or app view.

Use `-window` to specify the window or app view that you want a list for.

Return

Returns a list of files that match the regular expression.

Examples

```
% custom_gui_action -add VisualizeCover -window fpv -script {visualize -cover  
-property $propertyName -bg}  
% custom_gui_action -add Explore -window visualize -script {visualize -explore  
$signalName} -iconfile ~/explore.png  
% custom_gui_action -remove Explore -window visualize  
% custom_gui_action -list -window fpv
```

See Also

No related commands

database

Description

Use this command to access and manipulate the Executable Specification database.

Syntax

[Manipulate the Navigation/Document Pane](#)

```
database -get_focus  
database -set_focus <element_id>  
database -expand <element_id>  
database -collapse <element_id>  
database -hide <element_id> [-subtree]  
database -show <element_id> [-subtree]  
database -move_before <element_id> <before_this_id>  
database -move_after <element_id> <after_this_id>  
database -move_as_child <element_id> <parent_id>
```

[Add Various Data](#)

```
database -add_section <heading>  
    [-description <description>] [-batch]  
database -add_question <heading>  
    [-description <description>] [-resolution <resolution>] [-batch]  
database -add_high_level_functionality <heading>  
    [-description <description>] [-batch]  
database -add_signal <signal_name>  
    [-label <label>] [-description <description>] [-batch]
```

[Export Various Data](#)

```
database -export <file_name>  
database -export_VCDs -dir <dir_name> [-baseline]
```

JasperGold Apps Command Reference Manual

General Commands

Access Database Elements

```
database -get_all_elements  
database -get_elements (section | hlf | question | recipe | signal | behavior  
| assumption | assertion)  
database -get_elements -property <property_name>  
database -get_children <element_id>  
database -delete_element <element_id>
```

Access Reports

```
database -get_report_list  
database -get_report_elements <report_name>  
database -export_report <report_name> <file_name>
```

Access Data Files

```
database -get_datafile_list  
database -get_datafile_location <data_file>
```

Access Element Attributes

```
database -get_type <element_id>  
database -get_label <element_id>  
database -get_description <element_id>  
database -get_resolution <element_id>  
database -get_signal_name <element_id>  
database -get_unique_property_name <element_id>
```

Modify Element Attributes

```
database -set_label <element_id> <text>  
database -set_description <element_id> <text>  
database -set_resolution <element_id> <text>
```

Manipulate Element Data Files

```
database -get_modifiable_datafile <element_id>
```

JasperGold Apps Command Reference Manual

General Commands

```
database -get_READONLY_datafile <element_id>
database -get_diagram <element_id>
database -clear_modifiable_datafile <element_id>
database -clear_READONLY_datafile <element_id>
database -clear_diagram <element_id>
database -import_modifiable_datafile <element_id> <data_file>
database -import_READONLY_datafile <element_id> <data_file>
database -import_diagram <element_id> <data_file>
```

[Synchronize Sessions with the Database](#)

```
database -sync
```

[Generate a Behavioral Analysis](#)

```
database (-ba | -behavioral_analysis)
    [-include_behaviors] [-confirm_vcd] [-print_summary] [-highlight_coverage]
    [-batch] [-cumulative] [-num_sessions <num_session>] [-use_server_farm]
database (-ba | -behavioral_analysis) -set_baseline
database (-ba | -behavioral_analysis)
    (-clear_baseline | -clear_latest | -clear_all)
```

[Generate a Structural Analysis](#)

```
database (-sa | -structural_analysis) [-all] [-define] [-ifdef]
    [-analyzed_file] [-reset] [-clock] [-input] [-output]
    [-counter] [-fsm] [-array] [-flop] [-latch]
    [-threshold <threshold>] [-include_nd]
    [-batch]
database (-sa | -structural_analysis)
    (-set_baseline | -clear_baseline | -clear_latest | -clear_all)
database (-sa | -structural_analysis)
    (-purge_obsolete | -purge_recently_delete) [-force]
```

[Capture and View BPS and SPS Setup Scripts](#)

```
database -set_setup [-import <tcl_file>]
database -get_setup
```

JasperGold Apps Command Reference Manual

General Commands

[Set and Restore BPS and SPS Baseline](#)

```
database -set_baseline (-bps | -sps)  
database -revert_to_baseline (-bps | -sps)
```

[Initialize and Export Unicov-Based Database](#)

```
database -init_unicov (-coverage <type> | -write_metrics)  
    [-work_dir <work>] [-scope <scope>] [-test <test>]  
    [-waveform] [-overwrite]  
  
database -export_unicov
```

Argument	Definition
Manipulate the Navigation/Document Pane	
-get_focus	Return the element ID of the current focus of the navigation pane.
-set_focus <i>element_id</i>	Move the navigation pane focus to the element with the specified ID and return the ID of the element after setting the focus. If the specified ID is invalid, the tool sets the focus to the root element and returns -1.
-expand <i>element_id</i>	Expand the specified element to show its children.
-collapse <i>element_id</i>	Collapse the specified element to hide its children.
-hide <i>element_id</i> -subtree	Hide the specified element from the navigation pane and document pane. Use the -subtree switch to also hide the element's subtree.
-show <i>element_id</i> -subtree	Show the specified element and its subtree in the navigation pane and document pane. Use the -subtree switch to also hide the element's subtree.

JasperGold Apps Command Reference Manual

General Commands

```
-move_before element_id before_this_id
```

Make the element with first ID a sibling of the element with the second ID, and position it before the second element.

```
-move_after element_id after_this_id
```

Make the element with first ID a sibling of the element with the second ID, and position it after the second element.

```
-move_as_child element_id parent_id
```

Make the element with first ID a child of the element with the second ID, and position as the last sibling.

Add Various Data

```
-add_section heading  
-description description  
-batch
```

Add a new section with the specified *heading*.

- Use *-description* to add a description.
- Use *-batch* to suppress the dialog for selecting the location of the new element.

The return value is the ID of the new element.

```
-add_question heading  
-description description  
-resolution resolution  
-batch
```

Add a new question with the specified heading.

- Use *-description* to add a description.
- Use *-resolution* to add the resolution.
- Use *-batch* to suppress the dialog for selecting the location of the new element.

The return value is the ID of the new element.

JasperGold Apps Command Reference Manual

General Commands

```
-add_high_level_functionality heading
  -description description
  -batch
```

Add a new high-level functionality element with the specified heading.

- Use `-description` to add a description.
- Use `-batch` to suppress the dialog for selecting the location of the new element.

The return value is the ID of the new element.

```
-add_signal signal_name
  -label label
  -description description
  -batch
```

Search for the specified signal in the database; if not present, add a new signal element.

- Use `-label` to add an element heading.
- Use `-description` to add a description.
- Use `-batch` to add the new element without asking for confirmation.

Export Various Data

```
-export file_name
```

Export the current database to an HTML report.

```
-export_VCDs
  -dir dir_name
  -baseline
```

Export VCD files associated with any behavior or recipe element into the specified directory.

Use `-baseline` to export the baseline VCD files instead of the latest VCD files.

Access Database Elements

```
-get_all_elements
```

Return the IDs of the elements in the database.

JasperGold Apps Command Reference Manual

General Commands

```
-get_elements
```

```
    section | hlf | question | recipe | signal | behavior  
    | assumption | assertion
```

Return the IDs of the elements with the specified type in the database.

```
-get_elements -property property_name
```

Return the element ID of the specified property.

This switch is useful for scripting.

```
-get_children element_id
```

Return the IDs of the children of the element with the specified ID.

```
-delete_element element_id
```

Delete the element with the specified ID and return `true` if successful.

Access Reports

```
-get_report_list
```

Return the names of the reports in the database.

```
-get_report_elements report_name
```

Return the IDs of the elements in the specified report and return an empty list if the report name is invalid.

```
-export_report report_name file_name
```

Export a report into a file using a comma-separated-value format.

Access Data Files

```
-get_datafile_list
```

Return the names of the data files in the database.

```
-get_datafile_location data_file
```

Return the location of the data file in the database and return an empty string if there is no such data file.

Use this switch with the `source` Tcl command for a Tcl script data file and with other applications for other types of files.

Access Element Attributes

```
-get_type element_id
```

Return the element type of the specified ID.

```
-get_label element_id
```

JasperGold Apps Command Reference Manual

General Commands

Return the element label of the specified ID.

```
-get_description element_id
```

Return the element description of the specified ID.

```
-get_resolution element_id
```

Return the resolution of the Question element with the specified ID and return an empty string if the element is not a Question element.

```
-get_signal_name element_id
```

Return the signal name of the Signal element with the specified ID and return an empty string if the element is not a Signal element.

```
-get_unique_property_name element_id
```

Return the unique property name of an element with the specified ID and return an empty string if the element does not have a unique property name.

Modify Element Attributes

```
-set_label element_id text
```

Modify the label of the element with the specified ID and return `true` if successful.

```
-set_description element_id text
```

Modify the description of the element with the specified ID and return `true` if successful.

```
-set_resolution element_id text
```

Modify the resolution of the Question element with the specified ID and return `true` if successful.

Manipulate Element Data Files

```
-get_modifiable_datafile element_id
```

Return the location of the modifiable data file of the element with the specified ID.

Use this command with the `source` Tcl command for a Tcl script data file and with other applications for other types of files.

```
-get_READONLY_datafile element_id
```

JasperGold Apps Command Reference Manual

General Commands

Return the location of the read-only data file for the element with the specified ID.

Use this command with the `source` Tcl command for Tcl script data files and with other applications for other types of files.

`-get_diagram element_id`

Return the location of the diagram data file for the element with the specified ID.

Use this command to pass on to external applications for viewing.

`-clear_modifiable_datafile element_id`

Clear the modifiable data file attribute from the element with the specified ID and return `true` if successful.

`-clear_READONLY_datafile element_id`

Clear the read-only data file attribute from the element with the specified ID and return `true` if successful.

`-clear_diagram element_id`

Clear the diagram data file attribute from the element with the specified ID and return `true` if successful.

`-import_modifiable_datafile element_id data_file`

Import the specified data file into the database, set the modifiable data file attribute of the element with the specified ID to point to the resulting data file, and return `true` if successful.

`-import_READONLY_datafile element_id data_file`

Import the data file into the database, set the read-only data file attribute of the element with the specified ID to point to the resulting data file, and return `true` if successful.

`-import_diagram element_id data_file`

Import the data file into the database, set the diagram data file attribute of the element with the specified ID to point to the resulting data file, and return `true` if successful.

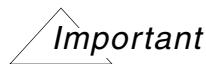
If this data file has the `.gif`, `.jpg`, or `.jpeg` file name extension and is in GIF or JPG format, it is displayed in the document pane.

Synchronize Sessions with the Database

-sync

Examine the property elements in the database to determine which ones are on-the-fly properties created by the assert, assume, and cover commands, and re-run them in the analysis session.

Use this command if you create on-the-fly properties in one analysis session (which will automatically populate the database if you have used set_property_auto_update_db on) and you want to access these properties from another analysis session.



It is not possible to synchronize two sessions with different design elaborations that use different languages (VHDL versus Verilog) because attempting to add the properties from one session to the other triggers a syntax error. Consider the following three options:

- Avoid synchronization in this case.
 - Avoid different syntax in the properties if they are to be synchronized.
 - Accept that properties with different syntax will not be synchronized and avoid using the same name for incompatible properties in different sessions.
-

Generate a Behavioral Analysis

```
-ba | -behavioral_analysis  
[-include_behaviors]  
[-confirm_vcd]  
[-print_summary]  
[-highlight_coverage]  
[-batch]  
[-cumulative]  
[-num_sessions num_session]  
[-use_server_farm]
```

Extract the behavioral information based on applying the recipes in the database to the current design.

- Use `-include_behaviors` to also generate waveforms from behaviors and not just recipes.
- Use `-confirm_vcd` to also apply stimuli from the baseline analysis and check whether the corresponding recipes are still satisfied.
- Use `-print_summary` to list the signals not exercised by the recipes at the end of the analysis.
- Use `-highlight_coverage` to bring up the Design Space Coverage window with cumulative trace coverage highlights.
- Use `-batch` to suppress the final dialog and allow uninterrupted Tcl scripting.
- Use `-cumulative` to revise new results without clearing old ones.
- Use `-num_sessions` to run the analysis with multiple sessions.
- Use `-use_server_farm` to launch these sessions with the server farm setting (for example, the one you specified with `session -set_remote_mode`).

```
-ba | -behavioral_analysis  
-set_baseline
```

Set the current behavioral analysis results as the baseline for future comparison.

JasperGold Apps Command Reference Manual

General Commands

```
-ba | -behavioral_analysis  
(-clear_baseline | -clear_latest | -clear_all)
```

Clear the behavioral analysis results

- Use `-clear_baseline` to clear only the baseline results.
- Use `-clear_latest` to clear only the latest results.
- Use `-clear_all` to clear all behavioral analysis results.

Generate a Structural Analysis

```
-sa | -structural_analysis  
[-all] [-define] [-ifdef]  
[-analyzed_file] [-reset] [-clock] [-input] [-output]  
[-counter] [-fsm] [-array] [-flop] [-latch]  
[-threshold threshold] [-include_nd]  
[-batch]
```

Extract the structural information based on current design understanding.

- Use `-all` to extract all options.
- Use `-define` to extract `define definitions.
- Use `-ifdef` to extract `ifdef definitions.
- Use `-analyzed_file` to extract the RTL file list.
- Use `-reset` to extract current reset expressions.
- Use `-clock` to extract current clock expressions.
- Use `-input` to extract a list of the top module's inputs.
- Use `-output` to extract a list of the top module's outputs.
- Use `-counter` to extract a list of the design's counters.
- Use `-fsm` to extract a list of the design's FSMs.
- Use `-array` to extract a lists of the design's arrays.
- Use `-flop` to extract a list of the design's flops.
- Use `-latch` to extract a list of the design's latches.
- Use `-batch` to suppress the final dialog and allow uninterrupted Tcl scripting.

JasperGold Apps Command Reference Manual

General Commands

-structural_analysis (Continued)

Important: For better overall performance, if structural analysis extracts more than 1000 signals in the same category, the tool does not attempt to populate the database.

Use `-threshold` to change the default threshold. Use 0 to suppress the threshold.

Important: For better overall performance, structural analysis does not attempt to extract counters and FSMs from N-D arrays.

Use `-include_nd` to include N-D arrays in counter and FSM extraction.

```
-sa |-structural_analysis  
-set_baseline
```

Set the current structural analysis result as the baseline for future comparison.

```
-sa |-structural_analysis  
(-clear_baseline |-clear_latest |-clear_all)
```

Clear the structural analysis results.

- Use `-clear_baseline` to clear the classification for baseline and obsolete elements.
- Use `-clear_latest` to clear the classification for recently added, deleted, and modified elements.
- Use `-clear_all` to clear the classification for all elements.

```
-sa |-structural_analysis  
(-purge_obsolete |-purge_recently_delete [-force])
```

Remove elements and their children from the database.

- Use `-purge_obsolete` to remove all obsolete elements and their children from the database.
- Use `-purge_recently_delete` to remove all recently deleted elements and their children from the database.
- Use `-force` to suppress the confirmation dialog.

JasperGold Apps Command Reference Manual

General Commands

Capture and View BPS and SPS Setup Scripts

```
-set_setup [-import tcl_file].
```

Capture an RTL setup script into the database.

- Use `-import` to copy an explicit Tcl file as the setup script.
- Use `-set_setup` without additional arguments to capture the settings in the current session.

```
-get_setup
```

Return the commands stored in the setup script.

Set and Restore BPS and SPS Baseline

```
-set_baseline -bps | -sps
```

Set the current BPS/SPS database contents as the baseline.

```
-revert_to_baseline
```

```
-bps | -sps
```

Restore current BPS/SPS database contents to the baseline content.

Note: All changes since the last database `-set_baseline` command will be lost; however, this command does not undo elaboration data when the baseline is set before analyzing and elaborating the design.

Initialize and Export Unicov-Based Database

```
-init_unicov (-coverage type | -write_metrics)
  [-work_dir work]
  [-scope scope]
  [-test test]
  [-waveform]
  [-overwrite]
```

Initialize the Unicov database setup.

- Use `-coverage` to specify the type of coverage to export to Unicov. Only `f`, `all`, and `conn` are valid types. Use `f` or `all` for FPV checks and `conn` for Connectivity checks.
- Use `-work_dir` to specify the work directory.
- Use `-scope` to specify the scope directory.
- Use `-test` to specify the test directory.
- Use `-write_metrics` to generate the `.vsوف` file. In the VManager-based flow, this is the default.
- Use `-waveform` to export waveforms along with the `.vsوف` file.
- Use `-overwrite` to overwrite the Unicov database.

```
-export_unicov
```

Export the Unicov database for the selected task based on the `-init_unicov` specifications.

Return

Various

Examples

```
% database -get_focus
% edit [database -get_datafile_location umax.sv]

% database -sa
% database -ba -include_behaviors -print_summary
% save -jdb my_db.jdb

// Parallel behavioral analysis with Platform LSF(R):
% session -set_remote_mode lsf
```

JasperGold Apps Command Reference Manual

General Commands

```
% database -ba -num_sessions 5 -use_server_farm

// Parallel behavioral analysis with a shell script:
% session -set_remote_mode shell
% session -set_remote_shell "my.script -arg abcd"
% database -ba -num_sessions 5 -use_server_farm
```

See Also

[session](#)

dglob

Description

Use this Tcl command to sort and list files that match your specified regular expression argument.

The tool alphabetically sorts the list of files this command produces. For example, use `dglob` to determine which files should be analyzed.

Syntax

`dglob <expression>`

Argument	Definition
<i>expression</i>	A shell-style regular expression used to match a directory or set of files. See a Tcl reference manual for details.

Return

Returns a list of files that match the regular expression.

Examples

% `dglob/design/*.v`

See Also

[analyze](#)
[help](#)

edit

Description

Use this command to edit a file using vi in a new xterm session. To use a different editor, set the EDITOR environment variable (see “[EDITOR](#)” on page 73).

Syntax

`edit <file_name>`

Argument	Definition
<i>file_name</i>	Edit the specified file.

Return

No value returned

Examples

% edit my_file.v

See Also

No related commands

elaborate

Description

Use this command to synthesize and read the netlist and extract verification tasks for Verilog® and VHDL files that have been analyzed using the `analyze` command.



- After a successful elaboration, you can end the separate frontend process (`jg_frontend`) to free the associated memory with the `analyze -clear` command; however, subsequent `elaborate` commands result in an error if you end the `jg_frontend` process beforehand.
- After an unsuccessful elaboration, correct the issues that caused the elaboration error, re-analyze the design, and elaborate again.
- If your design has inout ports in the top module, the tool splits them into two ports: one input port and one output port.
- When using the extended functions `next` and `driver` in expressions on the command line, you must prepend them as shown below to avoid name clashes with signal names.
 - Verilog expression mode – use `$next` and `$driver`.
 - VHDL expression mode – use `::next` and `::driver`.
- During elaboration, the tool recognizes several types of memories, including memories with asynchronous or synchronous reset pins, multidimensional arrays, multiple read and write ports, and so forth. As a consequence, you might encounter some tool-generated signals. The original net in the RTL that represented a RAM appears with the naming convention `<ram_net>.ram_net`. In addition, you might see the following signals, for example, in the Visualize window or the `get_design_info` return:
 - For each write port of a RAM:
 - `<ram_name>.clk_#`
 - `<ram_name>.write_enable_#`
 - `<ram_name>.write_address_#`
 - `<ram_name>.write_data_#`
 - For each asynchronous write port of a RAM:

JasperGold Apps Command Reference Manual

General Commands

- <ram_name>.async_write_enable_#
- <ram_name>.async_write_address_#
- <ram_name>.async_write_data_#
- For each read port of a RAM:
 - <ram_name>.read_enable_#
 - <ram_name>.read_address_#
 - <ram_name>.read_data_#
- JasperGold Apps supports SystemVerilog configurations, as stated in the LRM (for example, Section 33 of the SystemVerilog '09 LRM). Elaborate the configuration cell as the top module. For example, if the configuration is called cfg1, use the following elaborate command:

```
elaborate -top cfg1
```

See "[analyze](#)" on page 160 for additional information about SystemVerilog configurations.
JasperGold Apps also supports VHDL configurations, and as with SystemVerilog configurations, you must elaborate the configuration as the top module, for example:

```
elaborate -vhdl -top CONFIG_ADDSUB_STRUCT
```
- If you use `iRun` with `-elab_options_jg`, you must add a leading space to the options between the quotation marks. For example:

```
-elab_options_jg " -create_related_covers witness  
-disable_auto_bbox"
```

Note: JasperGold Apps cumulatively print, as warnings, all the instances ignored during elaborate. The tool records these under the *Lint* tab in the main window.

Syntax

```
elaborate [-vhdl]  
          [-sv09_expression_mode]  
          [-bbox (0 | 1)]  
          [-bbox_m <string>]  
          [-no_bbox_m <string>]  
          [-bbox_i <string>]  
          [-no_bbox_i <string>]  
          [-bbox_a <value>]  
          [-bbox_mul <value>]  
          [-bbox_div <value>]
```

JasperGold Apps Command Reference Manual

General Commands

```
[ -bbox_mod <value>]
[ -bbox_pow <value>]
[ -disable_auto_bbox]
[ -ignore_translate_off]
[ -top [lib_name.]string['('arch_name')']]
[ -parameter <param_name> <param_value>]
[ -req]
[ -no_psl_tasks]
[ -no_type_properties]
[ -no_preconditions]
[ -create_related_covers
    [late_precondition | precondition | witness]+]
[ -mode (verilog | vhdl [-disable_sva_vhdl])]
[ -weaken_embedding]
[ -sv09_strong_embedding]
[ -multiple_clock]
[ -enable_sva_isunknown]
[ -enable_dpram]
[ -enable_assert_action_block]<
[ -enable_sv_type_properties]
[ -enable_unnamed_generate_naming]
[ -disable_x_handling]
[ -x_triple_equal]
[ -x_value (0 | 1)]
[ -extract_case_assertions [-parallel_equal_assignment]]
[ -ignore_if_case_qualifiers]
[ -ignore_embedded_sva]
[ -abstract_ram] [-mram_init_value (0 | 1)]
[ -max_array_dim_addr_size <N>]
[ -loop_limit <size>]
[ -global_embedded_assumes]
[ -enable_new_hierref_handler]
[ -auto_hr_info |-hr_info_file <file_name>]
[ -unshare_instances]
[ -add_package [top | root]]
[ -keep_logfile]
[ -shared_internal_inout_wire]
[ -ignore_driver_strength]
[ -handle_mos_using_tri]
[ -unbuffered_concatenation]
[ -use_expanded_module_name]
[ -f <file_name>]*
[ (-L <package_name>)+ <package_declaration.sv> |-L <library_name>]
[ -cov_ignore_m <string>]
[ -use_universal_automata]

elaborate -load_cpi <cpi_design_name>
[ -hdlvar <hdl_var_file>]
[ -cdslib <cds_lib_file>]
[ -nclibdirpath <nc_lib_dir_path>]
[ -cds_implicit_tmpdir <cpi_data_storage_location>]
```

JasperGold Apps Command Reference Manual

General Commands

```
[-mode (verilog | vhdl [-disable_sva_vhdl])]  
[-bbox_mul <value>]  
[-bbox_div <value>]  
[-bbox_mod <value>]  
[-bbox_pow <value>]  
[-disable_auto_bbox]  
[-weaken_embedding]  
[-multiple_clock]  
[-no_preconditions]  
[-create_related_covers  
    [late_precondition | precondition | witness]+]  
[-global_embedded Assumes]  
[-sv09_expression_mode]  
[-sv09_strong_embedding]  
[-use_universal_automata]  
[-enable_source_browser]  
[-disable_mpram]
```

```
elaborate -log
```

```
elaborate -clear [-req]
```

Argument	Definition
no arguments	<p>Elaborate all Verilog and SystemVerilog modules and...</p> <ul style="list-style-type: none">■ If <code>-top</code> is not specified, choose one module as top. (The top module will be the last module elaborated that is not instantiated by any other module.)■ Elaborate any other modules or VHDL entities that are needed by instantiations. <p>The default assumption is that the top-level module is from Verilog or SystemVerilog.</p>
<code>-vhdl</code>	

JasperGold Apps Command Reference Manual

General Commands

Elaborate all VHDL entities and...

- If `-top` is not specified, choose one entity-architecture pair as top. (The top module will be the last entity elaborated that is not instantiated by any other architecture.)
- Elaborate any other modules or entities that are needed by instantiations.
- If `-mode` is not specified, set the expression mode to VHDL.

`-sv09_expression_mode`

Elaborate all Verilog/SystemVerilog modules and support Tcl expressions that follow the SystemVerilog 2009 (IEEE Std 1800-2009) semantic for implicit embedding.

`-bbox 0 | 1`

By default, all missing modules are errors. Use this option to override the default behavior.

- 0 = missing modules are errors
- 1 = missing modules are automatically black boxed

Note: The `analyze` command also supports this switch, and in some cases, you will want to use it there so the modules are black boxed earlier in the flow.

`-bbox_m string`

Black box the specified set of modules.

Note:

- Specify modules with a wildcard string. For example, the following argument black boxes all modules whose names match the wildcard string.
`-bbox_m FIFO_*_TWO_PORT`
- If you specify a name with an escape character, you must surround the name with curly braces, for example:
`elaborate... -bbox_m {\arb$().z}`
- The tool does not black box the `-top` module when it is a match for the wildcard string.
- The `analyze` command also supports this switch, and in some cases, you will want to use it there so the modules are black boxed earlier in the flow.

JasperGold Apps Command Reference Manual

General Commands

`-no_bbox_m string`

Do not black box the specified set of modules.

Note:

- Specify modules with a wildcard string. For example, the following argument black boxes all modules whose names match the wildcard string.
`-bbox_m FIFO_*_TWO_PORT`
 - If you specify a name with an escape character, you must surround the name with curly braces, for example:
`elaborate... -no_bbox_m {\arb$.()z}`
 - `-no_bbox_m` takes priority over `-bbox_m`; therefore, if you specify a module for both `-bbox_m` and `-no_bbox_m`, the tool does not black box it.
 - Using `-no_bbox_m` without `-bbox_m` has no effect.
 - The `analyze` command also supports this switch, and in some cases, you will want to use it there to avoid black boxing earlier in the flow.
-

`-bbox_i string`

Black box the specified set of instances.

Note:

- Specify instances with a wildcard string. For example, the following command argument black boxes all instances whose names match the wildcard string `-bbox_i inst?_*`
 - The top module is implied; therefore, do not include it in the instance name. For example, instance `arb` is under top module `top`, so the command argument is `-bbox_i arb`
 - If you specify a name with an escape character, you must surround the name with curly braces, for example:
`elaborate... -bbox_i {\arb$.()z}`
-

JasperGold Apps Command Reference Manual

General Commands

`-no_bbox_i string`

Do not black box the specified set of instances.

Note:

- Specify instances with a wildcard string. For example, the following command argument black boxes all instances whose names match the wildcard string `-bbox_i inst?_*`
 - The top module is implied; therefore, do not include it in the instance name. For example, instance `arb` is under top module `top`, so the command argument is `-bbox_i arb`
 - If you specify a name with an escape character, you must surround the name with curly braces, for example:
`elaborate... -no_bbox_i {\arb$.()z}`
 - `-no_bbox_i` takes priority over `-bbox_i`; therefore, if you specify an instance for both `-bbox_i` and `-no_bbox_i`, the tool does not black box it.
 - Using `-no_bbox_i` without `-bbox_i` has no effect.
-

`-bbox_a value`

Black box arrays if their size exceeds the specified `value`.

Note: If you do not use this switch, the tool black boxes arrays whose size exceeds 2048.

`-bbox_mul value`

Black box multipliers if their size (sum of input bits) exceeds the specified `value`.

By default, JasperGold Apps black boxes all multiplier “gates” whose size is greater than or equal to a threshold of 1.

Note: In most cases, each multiplication has a multiplier “gate.” However, power-of-two multiplication (for example, `sig * 4`) is the exception to this rule. In these cases, the multiplier is implemented as a shift (`sig << 2`) operation and there is no multiplier to black box.

JasperGold Apps Command Reference Manual

General Commands

-bbox_div value

Black box divide operators if their size exceeds the specified *value*.



- If you do not use this switch, the tool black boxes all dividers.
- Signed dividers are always black boxed. As a workaround, convert the operands to unsigned types, make the operation, and then apply common signal rules for the result.
- The output size is the size of the dividend. For example, for a/b , the output has the size of a .

-bbox_mod value

Black box modulus operators if their size exceeds the specified *value*.
The default value is 1.



- If you do not use this switch, the tool black boxes all modulus operators whose size exceeds the default value.
- Signed modulus operators are always black boxed. As a workaround, convert the operands to unsigned types, make the operation, and then apply common signal rules for the result.
- The output size is the size of the right operand. For example, in the operation $a \% b$, the output has the size of b .

JasperGold Apps Command Reference Manual

General Commands

-bbox_pow *value*

Black box power operators if their size exceeds the specified *value*.



- If you do not use this switch, the tool black boxes all power operators.
- The output size is the size of the left operand or the size of the left operand times two to the power of the size of the right operand, whichever is smaller. For example, in the operation $a \wedge b$, the output has the size of a or the size of $(a * (2 \wedge b))$, whichever is smaller.

-disable_auto_bbox

Disable all automatic black boxing.

Note:

- You cannot use -disable_auto_bbox in combination with switches that change the default threshold for automatic black boxing (for example, -bbox_a and -bbox_mul).
- Using this switch with very large designs increases the chances of erroring out due to insufficient memory.

-ignore_translate_off

Ignore synthesis pragmas when reading files from libraries during elaboration.

JasperGold Apps Command Reference Manual

General Commands

```
-top lib_name.string(arch_name)
```

Use the specified module, entity, or configuration as the “top level” for synthesis and analysis.

- If you do not include a `-top` option, the tool elaborates, synthesizes, and reads in all modules.
- Supplying the library name is optional. (The default library is `work`.)
- `string` is an entity or configuration name for VHDL or a module name for Verilog.
- Supplying the architecture name is optional. If you do not specify one, the tool selects a suitable architecture using the latest architecture it analyzed.

Note:

- The tool only supports `arch_name` when you use the `-vhdl` option and a string that is an entity.
- Using `-top` in `-f` files is prohibited. It triggers an error.

```
-parameter param_name param_value
```

Specifies the parameter values for the top-level module.

Note: Only use this option when you also use `-top` to specify the top-level module for analysis.

```
-req
```

Only elaborate the requirement files.

Note:

- Use `-top` with `-req` to precisely control what is elaborated into the requirements namespace.
- Using `-req` in `-f` files is prohibited. It triggers an error.

```
-no_psl_tasks
```

Do not generate a separate task for each vunit. Instead, create properties in the `<embedded>` task.

JasperGold Apps Command Reference Manual

General Commands

-no_type_properties

Do not generate type properties.

The tool supports VHDL type properties. During elaboration, it extracts type properties on the signals of the design from the allowed values and places them in the <constraints> task. By default, all the other tasks inherit the properties in the <constraints> task.

Refer to [Appendix I, “Automatically Extracted Type Properties.”](#)

Note: -no_type_properties and -enable_sv_type_properties are mutually exclusive.

-no_preconditions

Disable the precondition check.

By default, the precondition check is enabled. This check creates new cover points (trigger points) in the assertion, which create cover statements in the task.



If you choose not to run precondition checking, vacuous proofs can go undetected.

Note: It is an error to use this switch in combination with the switch -create_related_covers precondition (see below).

Refer to [Appendix I, “Automatically Extracted Type Properties.”](#)

JasperGold Apps Command Reference Manual

General Commands

```
-create_related_covers
```

```
[late_precondition | precondition | witness] +
```

Enable the precondition check for one or more specified types of cover statements.

The name of the precondition property indicates the precondition type, for example, `propertyA:witness1`.

The tool creates related covers as follows. In these examples, `s1` and `s2` are sequences.

- The late precondition of an SVA implication `s1 | => s2` will be a cover of the sequence (`s1 #\$1 1'b1`).
- The witness cover of an SVA implication `s1 | => s2` will be the cover of the sequence (`s1 #\$1 s2`).
- The witness cover of an SVA implication `s1 | -> s2` will be the cover of the sequence (`s1 #\$0 s2`).

Note:

- In many cases, when a property contains a negated sequence, for example, `B | -> not (A[*4:4])`, the tool does not produce a witness cover.
- The tool does not support witness precondition covers for properties with local variable assignments.
- By default, the tool creates precondition covers.
- It is an error to use `-create_related_covers` precondition in combination with the switch `-no_preconditions`.
- This command accepts Tcl lists.
- Using `-create_related_covers` in `-f` files is prohibited. It triggers an error.

JasperGold Apps Command Reference Manual

General Commands

```
-mode verilog | vhdl [-disable_sva_vhdl]
```

Interpret expressions used in commands (for example, assert, assume, cover, and visualize) as follows:

- verilog – interpret expressions using Verilog format.
- vhdl – interpret expressions using VHDL format.

The VHDL parser accepts SVA operators. To avoid name clashes with VHDL operators, suppress SVA keywords in the VHDL expression parser with `-disable_sva_vhdl`. Otherwise, define SVA keywords and, or, and not, as s_and, s_or, and s_not.

Note:

- Use `-mode` when the expression mode is different from that of the top module entity.
- Using `-mode` in `-f` files is prohibited. It triggers an error.

```
-weaken_embedding
```

During elaboration, convert strongly embedded SVA sequences into weakly embedded ones.

This switch can be useful when you want to avoid creating a liveness assertion unnecessarily.

```
-sv09_strong_embedding
```

During elaboration, treat all embedded SVA sequences defined in SystemVerilog 2009 files as strongly embedded SVA sequences.



- In SystemVerilog 2009, all embedded SVA sequences are treated as weakly embedded SVA sequences by default, which is the opposite of the SystemVerilog 2005 behavior.
- This option does not affect embedded SVA sequences specified with an explicit strong or weak keyword.

JasperGold Apps Command Reference Manual

General Commands

-multiple_clock

Elaborate PSL properties sampled by multiple clocks.

Note:

- SVA properties with multiple clocks are always elaborated.
 - Using this switch can, in some cases, increase the complexity of a proof or visualization.
-

JasperGold Apps Command Reference Manual

General Commands

-enable_sva_isunknown

Enable elaboration for SVA properties with \$isunknown to support X-propagation checks for these properties.



JasperGold Apps supports X-Prop checkers for SVA properties with \$isunknown, provided the property follows one of these patterns:

- assert property (@(posedge clk) ! \$isunknown(<user-defined signal>));
- assert property (@(posedge clk) not \$isunknown(<user-defined signal>));
- assert property (@(posedge clk) <pre-condition> |-> ! \$isunknown(<user-defined signal>));
- assert property (@(posedge clk) <pre-condition> |-> not \$isunknown(<user-defined signal>));
- assume property (\$isunknown(<signal>));
- assume property (<pre-condition> |-> ! \$isunknown(<signal>));

Note:

- The tool does not support clocking statements in SVA assumptions that use \$isunknown, for example:

```
assume property (@(posedge clk2) $isunknown(<signal>));
```

Instead, it uses the fastest clock.
- If there is a default clocking statement in the module, the following property is not supported because it inherits the clock from the default:

```
default clocking cb1 @(posedge clk); endclocking assume  
property ($isunknown(<signal>));
```
- Using -enable_sva_isunknown in -f files is prohibited. It triggers an error.

JasperGold Apps Command Reference Manual

General Commands

-enable_dpram

Enable dual port RAM detection during synthesis.



- This switch provides an alternative RAM detection mode in the event you experience undesirable tool behavior.
- This is the RAM detection algorithm that was available prior to version 2012.08.
- You must use this switch in *all* `analyze` commands *and* in the `elaborate` command. (You cannot mix RAM detection algorithms.)

-enable_assert_action_block

Enable assert action blocks.

By default, the tool ignores assert action blocks.

-enable_sv_type_properties

Generate SystemVerilog type properties for enum variables to ensure that enum values are obeyed.

When you use this switch, the tool extracts SystemVerilog type properties on the signals of the design from the allowed values and places them in the `<constraints>` task. By default, all the other tasks inherit the properties in the `<constraints>` task.

Note:

- If an enum is complete (that is, all values are possible), the tool does not create any properties for it.
- `-no_type_properties` and `-enable_sv_type_properties` are mutually exclusive.
- `elaborate -enable_sv_type_properties` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

JasperGold Apps Command Reference Manual

General Commands

-enable_unnamed_generate_naming

Automatically name un-named generate blocks following the rules defined in Section 27.6 of the SystemVerilog 2009 LRM.

Note: You must use this switch with both the `analyze` and `elaborate` commands.

-disable_x_handling

Do not treat Xs as free variables (that is, do not assign 0 to Xs).

Note:

- By default, the tool ensures a “pessimistic” verification by allowing signals assigned X to take all potential values.
- Using `-disable_x_handling` in `-f` files is prohibited. It triggers an error.

-x_triple_equal

Treat `==` and `!=` as `==` and `!=`.

If you elaborate with `-x_triple_equal` (and without `-disable_x_handling`), handling for the SVA operators is as follows:

- If constants in `==` have X or Z values, the tool evaluates `==` and `!=` as `==` and `!=` and uses the X-handling algorithm for values X and Z.
- If constants in `==` do not have X or Z values, the tool evaluates `==` and `!=` as `==` and `!=`.

-x_value 0 | 1

In assignment statements, treat Xs as specified:

- 0 – Assign 0 to Xs.
- 1 – Assign 1 to Xs.

Note: By default, the tool treats Xs as free variables; therefore, this switch has no effect unless you have also used `-disable_x_handling`.

JasperGold Apps Command Reference Manual

General Commands

```
-extract_case_assertions [-parallel_equal_assignment]
```

Automatically extract assertions for priority, unique, and unique0 case statements and add them to the <embedded> task.

The naming convention is <path>._automatic_unique_case_N.

<path>._automatic_unique0_case_N.

<path>._automatic_priority_case_N.

N is the automatically extracted assertion's number. Numbers start at 0.

The tool generates unique_case assertions if the design includes any of the following:

- UNIQUE keyword
- full_case and parallel_case pragma
- SystemVerilog full_case and parallel_case attribute

The tool generates unique0_case assertions if the design includes any of the following:

- UNIQUE0 keyword
- parallel_case pragma without full_case pragma
- SystemVerilog parallel_case attribute without full_case attribute

The tool generates priority_case assertions if the design includes no unique_case assertion as described above and any of the following:

- PRIORITY keyword and no default case
- full_case pragma
- SystemVerilog full_case attribute

Use -parallel_equal_assignment to prove parallelism between the assertions extracted for unique and unique0 case statements that pass the following test:

- All case items have only one basic assignment statement.
- The left-hand assignment variable is the same for all basic assignments.
- Although some case items have the same conditions, they also have the same assignment expressions for the same variable.

JasperGold Apps Command Reference Manual

General Commands

-ignore_if_case_qualifiers

Do not use SystemVerilog if/case qualifiers and Synopsys pragmas (full_case and parallel_case) during synthesis.

-ignore_embedded_sva

Ignore embedded SystemVerilog assertions during synthesis.

-abstract_ram [-mpram_init_value (0 | 1)]

Replace automatically extracted dual-port and multiport RAMs in the DUV with jasper_model_mpram Proof Accelerator instances.

Use -mpram_init_value to specify the initial value of RAMs this command abstracts.

Note: Use this switch if you are using the visualize RAM abstraction switches. See “[RAM Abstraction](#)” on page 925 for further details.

-max_array_dim_addr_size N

Increase the maximum dimension of a vector or array to the specified size (in terms of address bits).

Note:

- N must be less than or equal to 31.
- The default value is 20.

-loop_limit size

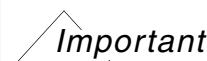
Change the loop limit to the specified size.

By default, the tool can elaborate a loop size up to 5000. Use this option to override the default behavior.

Note: Use the value 0 to re-enable the default behavior.

-global_embedded_assumes

Use embedded assumptions as environment assumptions.



- Use this option to leverage embedded assumptions for reset analysis.
- This command applies to assumptions in the <embedded> task. It does not apply to assumptions in vunits.

JasperGold Apps Command Reference Manual

General Commands

-enable_new_hierref_handler

Use the hierarchical reference (HR) handler released with version 2015.03.

Note: If you use the new HR handler, you must also use the switch -enable_unnamed_generate_naming in both the elaborate and analyze commands.

-auto_hr_info | -hr_info_file *file_name*

Use the specified hierarchical references (HR) handling.

By default, the tool uses the two-phase extended HR handler.

Implementation guidance:

- Use -auto_hr_info to specify the extended HR handler.

The extended HR handler uses a two-phase elaboration process. In the first phase, the tool collects information about all HRs present in the design and attempts to resolve them. If it detects a wrong size inference when resolving the HRs, it issues a warning and enables the second phase. If not, elaboration concludes.

In the second phase, the tool automatically calls elaborate -clear and starts a new elaboration using *jgproject/auto_hier_info.txt* (the internal file with the correct HR sizes).

- Use -hr_info_file to specify the file with the design's correct HR sizes and avoid the first phase.

Note: The extended HR handler always creates the file *hr_info.txt* with the correct HR sizes in the project dir. You can copy this file (or the *auto_hier_info.txt* file) to your area and use it with later elaborate commands and this switch to avoid running the two-phase elaboration required by elaborate -auto_hr_info.

Note: When running two elaborate commands without an elaborate -clear (or analyze -clear) between them, the extended HR handler verifies whether there is any potential synthesis problem due to the incremental elaboration. If so, the extended HR handler executes an elaborate -clear and starts the elaborate command from the very beginning.

JasperGold Apps Command Reference Manual

General Commands

-unshare_instances

Enable netlist duplication in earlier stages of `elaborate` commands to try to avoid later conflicts in the HR solver.

You must use this switch when requested by the `elaborate` command.

-add_package [top | root]

Support signals in packages and the global area (outside modules in SV).

- Use `top` to specify that the packages are children of the top module. This argument is the default if you specify `-add_package` without additional arguments.
- Use `root` to add the `$root` instance as the root of the instance tree and make the packages children of the `$root` instance.

Note:

- `elaborate -add_package` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.
- Using `-add_package` in `-f` files is prohibited. It triggers an error.

-keep_logfile

Save the `elaborate` log file.



- Only use this switch when requested by your Cadence representative. It is designed for tool debugging only.
- The tool stores the log in the following file:
`<proj>/sessionLogs/<session_identifier>/elaborate*.log`

-shared_internal_inout_wire

Connect two (or more) inout ports that share the same internal wire.

JasperGold Apps Command Reference Manual

General Commands

-ignore_driver_strength

Ignore strength information during synthesis and treat tri0 and tri1 as tri nets.



- The tool synthesizes all gates as if they have the same strength regardless of the RTL declaration.
- The tool ignores pullup and pulldown modeling for tri0 and tri1 nets, and treats them as tri nets.
- trireg nets will always have a medium charge strength.

-handle_mos_using_tri

Synthesize MOS switch gates (pmos, rpmos, nmos, rnmos, cmos, rcmos). This synthesis makes an approximated modeling as TRI, which is a pessimistic approach because it injects Z when a driver is not active.

Note: elaborate -handle_mos_using_tri is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

-unbuffered_concatenation

Do not add "Why" buffers inside concatenate operations.

Use this switch when "Why" buffers are breaking constant propagation and preventing the tool from elaborating the design.



If you use this switch, Why results will not highlight the constant values in concatenates.

-use_expanded_module_name

Do not rename excessively long module names.

Note: By default, JasperGold Apps renames modules if their length is excessively long, and the new name includes the string renamed_due_excessive_length.

JasperGold Apps Command Reference Manual

General Commands

`-f file_name`

Specifies a file listing HDL sources, options, and other `-f` calls.

Note:

- By default, the tool starts its search for modules in the directory where it last found a module and returns to the beginning of the directory list if necessary. See also `+librescan` and `+libnonamehide`.
- Use full switch names inside `-f` files as the tool has no prefix-matching or auto-completion feature for the file contents. For example, use `-no_preconditions` instead of `-no_precondition`.
- Only unknown and prohibited switches in `-f` files trigger messages.
- This command does not support file names that include whitespace or the character \$.

`(-L package_name)+ package_declaration.sv | -L library_name`

Use `-L` to specify Verilog packages or a VHDL or Verilog library for the tool to search. Also see “[set_automatic_library_search](#)” on page 958.

`-cov_ignore_m string`

Ignore coverage generation for the specified set of modules.

Note:

- Specify modules with a wildcard string.
- If you specify a name with an escape character, you must surround the name with curly braces. See the following example:

`elaborate... -cov_ignore_m {\arb$.()z}`

`-use_universal_automata`

Use the universal automata algorithm.

Note: For some properties, compilation can take much longer than expected. This algorithm can deliver a considerable property compilation and proof time reduction for many of those cases.

JasperGold Apps Command Reference Manual

General Commands

```
-load_cpi cpi_design_name
[-hdlvar hdl_var_file]
[-cdslib cds_lib_file]
[-nclibdipath nc_lib_dir_path]
[-cds_implicit_tmpdir cpi_data_storage_location]
[-mode (verilog | vhdl [-disable_sva_vhdl]) ]
[-bbox_mul value]
[-bbox_div value]
[-bbox_mod value]
[-bbox_pow value]
[-disable_auto_bbox]
[-weaken_embedding]
[-multiple_clock]
[-no_preconditions]
[-create_related_covers
    [late_precondition | precondition | witness]+]
[-global_embedded Assumes]
[-sv09_expression_mode]
[-sv09_strong_embedding]
[-use_universal_automata]
[-enable_source_browser]
[-disable_mpram]
```

Synthesize and read the netlist from a specified CPI file and load the corresponding design inside JasperGold Apps.

- Use `-hdlvar` to specify the `hdl.var` file to be used.
- Use `-cdslib` to specify the `cds.lib` file to be used.
- Use `-nclibdipath` to specify a relative path where libraries should be created.
- Use `-cds_implicit_tmpdir` to specify the location for design data storage.
- Use `-mode` to specify the language type you want the tool to use when interpreting expressions. The default language mode is determined by the top module's HDL language.

And use `-disable_sva_vhdl` with the `vhdl` argument to avoid name clashes between VHDL operators and SVA operators in property expressions.

Continued below.

JasperGold Apps Command Reference Manual

General Commands

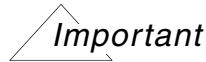
-load_cpi (Continued)

- Use -bbox_mul to black box multipliers if their size (sum of input bits) exceeds the specified value.

By default, JasperGold Apps black boxes all multiplier "gates" whose size is greater than or equal to a threshold of 1.

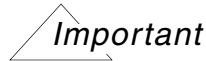
Note: In most cases, each multiplication has a multiplier "gate". However, power-of-two multiplication (for example, `sig * 4`) is the exception to this rule. In these cases, the multiplier is implemented as a shift (`sig << 2`) operation and there is no multiplier to black box.

- Use -bbox_div to black box divide operators if their output exceeds the specified value.



- If you do not use this switch, the tool black boxes all dividers.
 - Signed dividers are always black boxed. As a workaround, convert the operands to unsigned types, make the operation, and then apply common signal rules for the result.
 - The output size is the size of the dividend. For example, for `a/b`, the output has the size of `a`.

- Use -bbox_mod to black box modulus operators if their output exceeds the specified value. The default value is 1.



- If you do not use this switch, the tool black boxes all modulus operators whose size exceeds the default value.
 - Signed modulus operators are always black boxed. As a workaround, convert the operands to unsigned types, make the operation, and then apply common signal rules for the result.
 - The output size is the size of the right operand. For example, in the operation `a%b`, the output has the size of `b`.

Continued below.

-load_cpi (Continued)

- Use `-bbox_pow` to black box power operators if their output exceeds the specified value.



- If you do not use this switch, the tool black boxes all power operators.
 - The output size is the size of the left operand or the size of the left operand times two to the power of the size of the right operand, whichever is smaller. For example, in the operation $a ^ b$, the output has the size of a or the size of $(a * (2 ^ b))$, whichever is smaller.
- Use `-disable_auto_bbox` to disable all automatic black boxing.

Note:

- You cannot use `-disable_auto_bbox` in combination with switches that change the default threshold for automatic black boxing (for example, `-bbox_a` and `-bbox_mul`).
 - Using this switch with very large designs increases the chances of erroring out due to insufficient memory.
- Use `-weaken_embedding` to convert strongly embedded SVA sequences into weakly embedded ones. This switch can be useful when you want to avoid creating a liveness assertion unnecessarily.
- Use `-multiple_clock` to elaborate PSL properties sampled by multiple clocks.

Continued below.

JasperGold Apps Command Reference Manual

General Commands

-load_cpi (Continued)

- Use `-no_preconditions` to disable the precondition check.

By default, the precondition check is enabled. This check creates new cover points (trigger points) in the assertion or assumption, which create cover statements in the task.

CAUTION: If you choose not to run precondition checking, vacuous proofs can go undetected.

Note: It is an error to use this switch in combination with the switch `-create_related_covers` precondition (see below).

- Use `-create_related_covers` to enable the precondition check for one or more specified types of cover statements (late_precondition, precondition, witness).

The name of the precondition property indicates the precondition type, for example, `propertyA:witness1`.

The tool creates related covers as follows. In these examples, `s1` and `s2` are sequences.

- ❑ The late precondition cover of an SVA implication `s1 | => s2` will be a cover of the sequence (`s1 ##1 1'b1`).
- ❑ The witness cover of an SVA implication `s1 | => s2` will be the cover of the sequence (`s1 ##1 s2`).
- ❑ The witness cover of an SVA implication `s1 | -> s2` will be the cover of the sequence (`s1 ##0 s2`).

Note:

- ❑ In many cases, when a property contains a negated sequence, for example, `B | -> not(A[*4:4])`, the tool does not produce a witness cover.
- ❑ The tool does not support witness precondition covers for properties with local variable assignments.
- ❑ By default, the tool creates precondition covers.
- ❑ It is an error to use `-create_related_covers` precondition in combination with the switch `-no_preconditions`.

Continued below.

JasperGold Apps Command Reference Manual

General Commands

-load_cpi (Continued)

- This command accepts Tcl lists.
- Using `-create_related_covers` in `-f` files is prohibited. It triggers an error.
- Use `-global_embedded_assumes` to specify that you want the tool to use embedded assumptions as environment assumptions.



- Use this option to leverage embedded assumptions for reset analysis.
- This command applies to assumptions in the <embedded> task. It does not apply to assumptions in vunits.
- Use `-sv09_expression_mode` to elaborate all Verilog/SystemVerilog modules and support Tcl expressions that follow the SystemVerilog 2009 (IEEE Std 1800-2009) semantic for implicit embedding.
- Use `-sv09_strong_embedding` to specify that you want the tool to treat all embedded SVA sequences defined in SystemVerilog 2009 files as strongly embedded SVA sequences.



- In SystemVerilog 2009, all embedded SVA sequences are treated as weakly embedded SVA sequences by default, which is the opposite of the SystemVerilog 2005 behavior.
- This option does not affect embedded SVA sequences specified with an explicit strong or weak keyword.
- Use `-use_universal_automata` to specify that you want the tool to use the universal automata algorithm.

Note: For some properties, compilation can take much longer than expected. This algorithm can deliver a considerable property compilation and proof time reduction for many of those cases.

JasperGold Apps Command Reference Manual

General Commands

-load_cpi (Continued)

- Use -enable_source_browser to enable the Source Browser with appropriate color coding for signals.
- By default, the CPI flow black boxes memories. To change the default behavior (that is, avoid black boxing memories), use -disable_mpram along with the -BIT_BLAST argument of formalbuild in the `irun` command.

-log

Print the log file generated by the `elaborate` command.

The log file from the `elaborate` command is written to the directory for temporary files. See “[Environment Variables](#)” on page 73.

-clear

Clear the effects of the last elaboration.

Use this option in cases where you need to change the parameters of an elaboration.

For example:

```
// Elaborate the design.  
% elaborate  
[WARN] foo.v(11): multiply mult_40u_40u is BLACK BOXED  
  
// This multiplier is 40 x 40 bits.  
// Clear the effects of elaborate so you  
// can tell the tool to elaborate again  
// and only black box multipliers if they  
// are larger than 80 bits.  
% elaborate -clear  
  
// Elaborate again and only black box  
// multipliers larger than 80 bits.  
% elaborate -bbox_mul 80
```

-clear -req

Clear the effects of the previous requirements elaboration.

Note: In general, you will use this option only if you want to clear the effects of the last requirement file elaboration so you can elaborate again with different command options or modified requirements.

Return

Returns the name of the top module selected for analysis, including any existing parameters and their values. If there are multiple top-level candidates (more than one module not instantiated inside another module), and you did not use the `-top` option, the tool selects an arbitrary one.

Examples

```
% elaborate -vhdl -top ALU
% elaborate +incdir+../include+.../include -top DUV
% elaborate -vhdl -top ALU -mode verilog
% elaborate -create_related_covers {precondition late_precondition witness}

// Use -clear to change the parameters of an elaboration as follows:
// Elaborate the design.
% elaborate
[WARN] foo.v(11): multiply mult_40u_40u is BLACK BOXED
// This multiplier is 40 x 40 bits. Clear the effects of elaborate so you can // tell the tool to elaborate again and only black box multipliers if they are
// larger than 80 bits.
% elaborate -clear
// Elaborate again and only black box multipliers larger than 80 bits.
% elaborate -bbox_mul 80
```

See Also

[analyze](#)
[dglob](#)
[help](#)

exit

Description

Use this command to exit the tool.

Syntax

```
exit [<return_value>] [-force]
```

Argument	Definition
<i>return_value</i>	<p>Exit with the specified exit status.</p> <p> <i>Important</i></p> <ul style="list-style-type: none">■ If you provide this argument, the tool uses it as the exit status to return to the parent process. For example, you can use <code>exit 1</code> to indicate a session that included an error.■ If you do not provide an argument, the tool exits with a status of 0.■ Use 0, 1, or a code greater than 10.■ Do not use exit codes 2–10 as the tool uses these codes internally.■ Refer to Table 4-3 on page 531 for additional information on available exit codes.
<code>-force</code>	<p>Exit immediately without confirmation dialogs.</p>

Table 4-3 Exit Codes

Exit Code	Definition
0	Represents normal exit.
1	Represents generic failure (not license specific).

Table 4-3 Exit Codes

2	Indicates failure to check out a license at startup for any reason other than those related to exit codes 3 and 4 below.
3	Indicates failure to check out a license at startup because the FlexNet Publisher option <code>MAX</code> limit has been reached. ¹
4	Indicates failure to check out a license at startup because the current user/host/... is excluded (FlexNet Publisher option <code>EXCLUDE/EXCLUDEALL</code>) or is not included (option <code>INCLUDE/INCLUDEALL</code>). ¹
5	Indicates failure to reconnect to the license server after successful license checkout. This may happen if you have not specified the <code>-no_wait</code> switch, and it is expected when a “nice” session has been killed by a “non-nice” session. Refer to “ -no_wait ” on page 71 and “ -nice ” on page 72 for additional information.
6-10	Used internally by the tool and not available to the user.
0-1 and >10	Exit codes available to the user.

1. For additional information about licensing issues, consult your **FlexNet Publisher®** documentation.

Return

No value returned

Examples

```
% exit  
% exit 1
```

See Also

No related commands

export

Description

Use the `-export` command to export the Tcl properties, that is, assertions, assumptions, and covers, to an SVA or PSL requirements file. If you do not use the `-task` switch to export properties from a specified task, this command exports all on-the-fly properties.

Syntax

```
export -to_sva <file_name> |-to_psl <file_name>
  [-force] [-task <task_name>] [-env]
```

[Exporting BPS App Property Candidates by Category](#)

```
export -bps
  -to_sva <file_name>
    [-to_tcl <tcl_file_name>]
    [-extended_property_name] [-remove_duplicate_properties]
    [-module <sva_module_name>]
    [-reset_condition <sva_reset_condition>]
    [-initial_block <sva_initial_block>]
    [-action_block <sva_action_block>]
    (-type (assert | assume | exercised_cover | coverage_hole))*
    (-class (certified | dont_care | unclassified))*
    ([-status ( all | cex | ar_cex | proven | ar_covered | unprocessed
      | covered_reset | unreachable | undetermined
      | partial_cex | partial_ar_cex | partial_prove
      | partial_ar_covered | partial_covered_reset
      | partial_unreachable | partial_undetermined))*
    (-progress_state (new | obsolete | hole | covered | covered_now_hole
      | assert | violated | certified_hole | certified_assert | dont_care))*
    [-focus (high | medium | low)]
    [-baseline]
    [-silent]

export -bps
  -to_html <file_name>
    (-type (assert | assume | exercised_cover | coverage_hole))*
    (-class (certified | dont_care | unclassified))*
    ([-status ( all | cex | ar_cex | proven | ar_covered | unprocessed
      | covered_reset | unreachable | undetermined
      | partial_cex | partial_ar_cex | partial_proven
      | partial_ar_covered | partial_covered_reset
      | partial_unreachable | partial_undetermined))*
    (-progress_state (new | obsolete | hole | covered | covered_now_hole
      | assert | violated | certified_hole | certified_assert | dont_care))*
```

JasperGold Apps Command Reference Manual

General Commands

```
[-focus (high | medium | low) ]  
[-baseline]  
[-silent]
```

[Exporting BPS App Property Progress Report](#)

```
export -bps -progress  
    -to_html <file_name> | -to_csv <file_name>
```

[Exporting BPS App Property Candidates by Property ID](#)

```
export -bps  
    -to_sva <file_name>  
        [-to_tcl <tcl_file_name>]  
        [-extended_property_name] [-remove_duplicate_properties]  
        [-module <sva_module_name>]  
        [-reset_condition <sva_reset_condition>]  
        [-initial_block <sva_initial_block>]  
        [-action_block <sva_action_block>]  
    -property <id_tcl_list>  
    [-baseline]  
    [-silent]  
  
export -bps  
    -to_html <file_name>  
    -property <id_tcl_list>  
    [-baseline]  
    [-silent]
```

[Exporting SPS App Properties by Category](#)

```
export -sps  
    -to_sva <file_name>  
        [-to_tcl <tcl_file_name>]  
        [-extended_property_name] [-remove_duplicate_properties]  
        [-module <sva_module_name>]  
        [-initial_block <sva_initial_block>]  
        [-action_block <sva_action_block>]  
    (-type (assert | assume | exercised_cover | coverage_hole))*  
    (-class (certified | dont_care | unclassified))*  
    ([-status ( all | cex | ar_cex | proven | ar_covered | unprocessed  
            | covered_reset | unreachable | undetermined  
            | partial_cex | partial_ar_cex | partial_prove  
            | partial_ar_covered | partial_covered_reset  
            | partial_unreachable | partial_undetermined ))*)  
    [-check_type ( all | unique_case | priority_case  
            | arithmetic_overflow | out_of_bound_indexing | dead_code  
            | fsm | signals | contention_bus | floating_bus
```

JasperGold Apps Command Reference Manual

General Commands

```
| x_assignment | default_case) ]  
[-baseline]  
[-silent]  
  
export -sps  
( -to_html <file_name> [-launch_html_browser]  
| -to_xml <file_name> |-to_text <file_name>)  
(-type (assert | assume | exercised_cover | coverage_hole))*  
(-class (certified | dont_care | unclassified))*  
([-status ( all | cex | ar_cex | proven | ar_covered | unprocessed  
| covered_reset | unreachable | undetermined  
| partial_cex | partial_ar_cex | partial_proven  
| partial_ar_covered | partial_covered_reset  
| partial_unreachable | partial_undetermined))*)  
[-check_type ( all | unique_case | priority_case  
| arithmetic_overflow | out_of_bound_indexing | dead_code  
| fsm | signals | contention_bus | floating_bus  
| x_assignment | default_case)]  
[-compact]  
[-baseline]  
[-silent]
```

[Exporting SPS App Properties by Property ID](#)

```
export -sps  
-to_sva <file_name>  
[-to_tcl <tcl_file_name>]  
[-extended_property_name] [-remove_duplicate_properties]  
[-module <sva_module_name>]  
[-initial_block <sva_initial_block>]  
[-action_block <sva_action_block>]  
-property <id_tcl_list>  
[-baseline]  
[-silent]  
  
export -sps  
( -to_html <file_name> [-launch_html_browser]  
| -to_xml <file_name> |-to_text <file_name>)  
-property <id_tcl_list>  
[-compact]  
[-baseline]  
[-silent]
```

[Exporting SPS App Linting Messages](#)

```
export -sps -violation_report  
( -to_html file_name [-launch_html_browser]  
| -to_xml file_name  
| -to_text file_name)  
(-linting_type (lint | superlint))*
```

JasperGold Apps Command Reference Manual

General Commands

```
(-class (waived | unclassified))*  
(-severity (error | warning | info))*  
(-order (category | tag | filename | severity | label | instance))
```

Argument	Definition
<code>-to_sva <i>file_name</i></code>	<p>Export Tcl properties to an SVA requirements file.</p> <p>Bind the resulting SVA to the RTL file with SVA bind.</p>
<code>-to_psl <i>file_name</i></code>	<p>Export Tcl properties to a PSL requirements file.</p> <p>Analyze the resulting PSL file along with the design to see the extracted properties in their environment.</p>
Common Switches	
<code>-force</code>	If the specified SVA or PSL file already exists, overwrite it.
<code>-task <i>task_name</i></code>	<p>Export the Tcl properties located in the specified task.</p> <p>If you do no specify a task, the tool uses the current task.</p>
<code>-env</code>	Include global properties created with <code>-env</code> .

Exporting BPS App Property Candidates by Category

```
-bps
  -to_sva file_name
    [-to_tcl tcl_file_name]
    [-extended_property_name] [-remove_duplicate_properties]
    [-module sva_module_name]
    [-reset_condition sva_reset_condition]
    [-initial_block sva_initial_block]
    [-action_block sva_action_block]
  (-type (assert | assume | exercised_cover | coverage_hole))*
  (-class (certified | dont_care | unclassified))*
  ([-status ( all | cex | ar_cex | proven | ar_covered
              unprocessed | covered_reset | unreachable
              undetermined | partial_cex | partial_ar_cex
              partial_proven | partial_ar_covered
              partial_covered_reset | partial_unreachable
              partial_undetermined)])*
  (-progress_state ( new | obsolete | hole | covered
                     covered_now_hole | assert | violated
                     certified_hole | certified_assert
                     dont_care))*
  [-focus (high | medium | low)]
  [-baseline]
  [-silent]
```

Export property candidates as SVA modules.

- Use `-to_tcl` to create a Tcl script for loading the exported SVA file in JasperGold Apps.
- Use `-extended_property_name` to include the instance path name or the signal name.
- Use `-remove_duplicate_properties` to export properties with the same expression and clock only once.
- Use `-module` to specify a module name for SVA export. This command overwrites the default name.

-bps (Continued)

- Use `-reset_condition` to disable exported properties during reset. This switch adds `disable iff (sva_reset_condition)` clause to all exported SVA properties.
- Use `-initial_block` to add an initial block to the exported SVA file. For example,

```
export -bps -to_sva ... -initial_block {  
    begin  
        $display("Loaded JasperGold assertions.")  
    end  
}
```

Note: Use the `set_export_sva_target_environment` command to change the target environment to simulation or emulation when you use this switch.

- Use `-action_block` to add an action block to assertions and assumptions in the exported SVA file. For example,

```
export -bps -to_sva ... -action_block {  
    begin  
        `UVM_ERROR("Assertion %name% failed.");  
        $assertoff(%name%);  
    end  
}
```

Note:

- In this example, the tool replaces `%name%` in the action block with the property name.
- You must use the `set_export_sva_target_environment` command to change the target environment to simulation or emulation when you use this switch.
- Use `-type` to specify a type of property candidate to export. The default is `assert` and `cover`.
- Use `-class` to specify a class of property candidates to export. The default is `certified`.

JasperGold Apps Command Reference Manual

General Commands

-bps (Continued)

- Use `-status` to specify a status of property candidate to export. The default is all.
- Use `-focus` to select how many property candidates to export.
 - `high` – Export only high-rank property candidates.
 - `medium` – Export both high- and medium-rank property candidates.
 - `low` – Export all property candidates.
- The default is high.
- Use `-progress_state` to specify a progress report state of property candidate to export. The default is all.
- Use `-baseline` to use the property candidates in the baseline.
- Use `-silent` to suppress messages and generate only Tcl return values.

-bps

```
-to_html file_name
(-type (assert | assume | exercised_cover | coverage_hole))*
(-class (certified | dont_care | unclassified))*
([-status ( all | cex | ar_cex | proven | ar_covered
            | unprocessed | covered_reset | unreachable
            | undetermined | partial_cex | partial_ar_cex
            | partial_proven | partial_ar_covered
            | partial_covered_reset | partial_unreachable
            | partial_undetermined)])*
(-progress_state ( new | obsolete | hole | covered
                    | covered_now_hole | assert | violated
                    | certified_hole | certified_assert
                    | dont_care))*
[-focus (high | medium | low)]
[-baseline]
[-silent]
```

Export property candidates as HTML.

- Use `-type` to specify a type of property candidate to export. The default is `assert` and `cover`.
- Use `-class` to specify a class of property candidates to export. The default is `certified`.

JasperGold Apps Command Reference Manual

General Commands

-bps (Continued)

- Use `-status` to specify a status of property candidate to export. The default is all.
 - Use `-focus` to select how many property candidates to export.
 - `high` – Export only high-rank property candidates.
 - `medium` – Export both high- and medium-rank property candidates.
 - `low` – Export all property candidates.
- The default is high.
- Use `-progress_state` to specify a progress report state of property candidate to export. The default is all.
 - Use `-baseline` to use the property candidates in the baseline.
 - Use `-silent` to suppress messages and generate only Tcl return values.

Exporting BPS App Property Progress Report

`-bps -progress -to_html file_name | -to_csv file_name`

Export the property progress report to an HTML or CSV file.

Exporting BPS App Property Candidates by Property ID

```
-bps
  -to_sva file_name
    [-to_tcl tcl_file_name]
    [-extended_property_name] [-remove_duplicate_properties]
    [-module sva_module_name]
    [-reset_condition sva_reset_condition]
    [-initial_block sva_initial_block]
    [-action_block sva_action_block]
  -property id_tcl_list
  [-baseline]
  [-silent]
```

Export property candidates as SVA modules.

- Use `-to_tcl` to create a Tcl script for loading the exported SVA file in JasperGold Apps.
- Use `-extended_property_name` to include the instance path name or the signal name.
- Use `-remove_duplicate_properties` to export properties with the same expression and clock only once.
- Use `-module` to specify a module name for SVA export. This command overwrites the default name.
- Use `-reset_condition` to disable exported properties during reset. This switch adds `disable iff (sva_reset_condition)` clause to all exported SVA properties.

JasperGold Apps Command Reference Manual

General Commands

-bps (Continued)

- Use `-initial_block` to add an initial block to the exported SVA file. For example,

```
export -bps -to_sva ... -initial_block {  
    begin  
        $display("Loaded JasperGold assertions.")  
    end  
}
```

Note: Use the `set_export_sva_target_environment` command to change the target environment to simulation or emulation when you use this switch.

- Use `-action_block` to add an action block to assertions and assumptions in the exported SVA file. For example,

```
export -bps -to_sva ... -action_block {  
    begin  
        `UVM_ERROR("Assertion %name% failed.");  
        $assertoff(%name%);  
    end  
}
```

Note:

- In this example, the tool replaces `%name%` in the action block with the property name.
- You must use the `set_export_sva_target_environment` command to change the target environment to simulation or emulation when you use this switch.
- Use `-property` to specify a set of property candidate IDs to export.
- Use `-baseline` to use the property candidates in the baseline.
- Use `-silent` to suppress messages and generate only Tcl return values.

JasperGold Apps Command Reference Manual

General Commands

```
-bps
  -to_html file_name
  -property id_tcl_list
  [-baseline]
  [-silent]
```

Export specified property candidates as HTML.

- Use `-property` to specify a set of property candidate IDs to export.
- Use `-baseline` to use the property candidates in the baseline.
- Use `-silent` to suppress messages and generate only Tcl return values.

Exporting SPS App Properties by Category

```
-sps
  -to_sva file_name
    [-to_tcl tcl_file_name]
    [-extended_property_name] [-remove_duplicate_properties]
    [-module sva_module_name]
    [-initial_block sva_initial_block]
    [-action_block sva_action_block]
    (-type (assert | assume | exercised_cover | coverage_hole))* 
    (-class (certified | dont_care | unclassified))* 
    ([-status ( all | cex | ar_cex | proven | ar_covered
      | unprocessed | covered_reset | unreachable
      | undetermined | partial_cex | partial_ar_cex
      | partial_proven | partial_ar_covered
      | partial_covered_reset | partial_unreachable
      | partial_undetermined)])*
    [-check_type ( all | unique_case | priority_case
      | arithmetic_overflow | out_of_bound_indexing
      | dead_code | fsm | signals
      | contention_bus | floating_bus
      | x_assignment | default_case)]
  [-baseline] [-silent]
```

Export structural properties as SVA modules.

- Use `-to_tcl` to create a Tcl script for loading the exported SVA file in JasperGold Apps.

JasperGold Apps Command Reference Manual

General Commands

-sps (Continued)

- Use `-extended_property_name` to specify the following naming convention for checks:

```
\<instance>_<file><start line><end line><check  
type>poi<poi id>prop<prop id>[more indexes]
```

Note: The naming convention uses Verilog escaped identifiers, that is, they begin with a backslash (\).

- Use `-remove_duplicate_properties` to export properties with the same expression and clock only once.
- Use `-module` to specify a module name for SVA export. This command overwrites the default name.
- Use `-initial_block` to add an initial block to the exported SVA file. For example,

```
export -sps -to_sva ... -initial_block {  
    begin  
        $display("Loaded JasperGold assertions.")  
    end  
}
```

Note: Use the `set_export_sva_target_environment` command to change the target environment to simulation or emulation when you use this switch.

JasperGold Apps Command Reference Manual

General Commands

-sps (Continued)

- Use `-action_block` to add an action block to assertions and assumptions in the exported SVA file. For example,

```
export -sps -to_sva ... -action_block {  
    begin  
        `UVM_ERROR("Assertion %name% failed.");  
        $assertoff(%name%);  
    end  
}
```

Note:

- In this example, the tool replaces `%name%` in the action block with the property name.
 - You must use the `set_export_sva_target_environment` command to change the target environment to simulation or emulation when you use this switch.
 - Use `-type` to specify a type of property to export. The default is `assert` and `cover`.
 - Use `-class` to specify a class of property to export. The default is `certified`.
 - Use `-status` to specify a status of property to export. The default is `all`.
 - Use `-check_type` to specify specific check types for export. The default is `all`.
 - Use `-baseline` to use the structural properties in the baseline.
 - Use `-silent` to suppress messages and generate only Tcl return values.
-

JasperGold Apps Command Reference Manual

General Commands

```
-sps
( -to_html file_name [-launch_html_browser]
  |-to_xml file_name |-to_text file_name)
(-type (assert | assume | exercised_cover | coverage_hole))* 
(-class (certified | dont_care | unclassified))* 
([-status ( all | cex | ar_cex | proven | ar_covered
            | unprocessed | covered_reset | unreachable
            | undetermined | partial_cex | partial_ar_cex
            | partial_proven | partial_ar_covered
            | partial_covered_reset | partial_unreachable
            | partial_undetermined)])*
[-check_type ( all | unique_case | priority_case
               | arithmetic_overflow | out_of_bound_indexing
               | dead_code | fsm | signals
               | contention_bus | floating_bus
               | x_assignment | default_case)]
[-compact]
[-baseline] [-silent]
```

Export structural properties as HTML, XML, or text reports.

- Use `-to_html` to report results as HTML, and use `-launch_html_browser` to launch an HTML browser containing results.
- Use `-to_xml` to report results as XML.
- Use `-to_text` to report results as text.
- Use `-type` to specify a type of property to export. The default is `assert` and `cover`.
- Use `-class` to specify a class of property to export. The default is `certified`.
- Use `-status` to specify a status of property to export. The default is `all`.
- Use `-check_type` to specify specific check types for export. The default is `all`.
- Use `-compact` to compact the results.
- Use `-baseline` to use the properties in the baseline.
- Use `-silent` to suppress messages and generate only Tcl return values.

JasperGold Apps Command Reference Manual

General Commands

Exporting SPS App Properties by Property ID

```
-sps
  -to_sva file_name
    [-to_tcl tcl_file_name]
    [-extended_property_name] [-remove_duplicate_properties]
    [-module sva_module_name]
    [-initial_block sva_initial_block]
    [-action_block sva_action_block]
  -property id_tcl_list
  [-baseline] [-silent]
```

Export property candidates as SVA modules.

- Use `-to_tcl` to create a Tcl script for loading the exported SVA file in JasperGold Apps.
- Use `-extended_property_name` to specify the following naming convention for checks:

```
\<instance>_<file><start line><end line><check
type>poi<poi id>prop<prop id>[more indexes]
```

Note: The naming convention uses Verilog escaped identifiers, that is, they begin with a backslash (\).

- Use `-remove_duplicate_properties` to export properties with the same expression and clock only once.
- Use `-module` to specify a module name for SVA export. This command overwrites the default name.

JasperGold Apps Command Reference Manual

General Commands

-sps (Continued)

- Use `-initial_block` to add an initial block to the exported SVA file. For example,

```
export -sps -to_sva ... -initial_block {  
    begin  
        $display("Loaded JasperGold assertions.")  
    end  
}
```

Note: Use the `set_export_sva_target_environment` command to change the target environment to simulation or emulation when you use this switch.

- Use `-action_block` to add an action block to assertions and assumptions in the exported SVA file. For example,

```
export -sps -to_sva ... -action_block {  
    begin  
        `UVM_ERROR("Assertion %name% failed.");  
        $assertoff(%name%);  
    end  
}
```

Note:

- In this example, the tool replaces `%name%` in the action block with the property name.
- You must use the `set_export_sva_target_environment` command to change the target environment to simulation or emulation when you use this switch.
- Use `-property` to specify a set of property IDs to export.
- Use `-baseline` to use the properties in the baseline.
- Use `-silent` to suppress messages and generate only Tcl return values.

JasperGold Apps Command Reference Manual

General Commands

```
-sps
( -to_html file_name [-launch_html_browser]
  |-to_xml file_name |-to_text file_name)
-property id_tcl_list
[-compact]
[-baseline]
[-silent]
```

Export the specified structural properties to an HTML, XML, or text report.

- Use `-to_html` to report results as HTML, and use `-launch_html_browser` to launch an HTML browser containing results.
- Use `-to_xml` to report results as XML.
- Use `-to_text` to report results as text.
- Use `-property` to specify a set of property candidate IDs to export.
- Use `-compact` to compact the results.
- Use `-baseline` to use the properties in the baseline.
- Use `-silent` to suppress messages and generate only Tcl return values.

Exporting SPS App Linting Messages

```
-sps -violation_report
  (-to_html file_name [-launch_html_browser]
   | -to_xml file_name
   | -to_text file_name)
  (-linting_type (lint | superlint))*
  (-class (waived | unclassified))*
  (-severity (error | warning | info))*
  (-order (category | tag | filename | severity | label | instance))
```

Export violation messages to an HTML, XML, or text report..

- Use `-to_html` to report results as HTML, and use `-launch_html_browser` to launch an HTML browser containing results.
- Use `-to_xml` to report results as XML.
- Use `-to_text` to report results as text.
- Use `-linting_type` to specify the export of lint or superlint messages. The default is both types.
- Use `-class` to specify the export of waived or unclassified messages. The default is unclassified.
- Use `-severity` to specify the export of error, warning, or information messages. The default is all.
- Use `-order` to specify the criteria by which you would like the report ordered. The default is category.

Return

No return value.

Examples

```
% export -to_sva properties.sva -task my_task -force
% export -bps -to_html my_design_bps.html -property {2 4 8 22}
% export -sps -to_sva my_design_sps.sva -check_type dead_code
  -check_type arithmetic_overflow
```

See Also

[assert](#)
[assume](#)

JasperGold Apps Command Reference Manual

General Commands

[cover](#)

formal_bring_up

Description

`formal_bring_up` is an automatic property synthesis flow that uses the BPS App to create an initial formal environment. This environment includes basic set up, clock configuration, and assumptions export. If you did not configure the BPS App prior to calling the `formal_bring_up` command, then this command runs the BPS App default configuration. However, if you had previously configured the BPS App with the `scope`, `scan`, or `waveform` commands or by setting configuration variables, then the `formal_bring_up` flow respects that configuration.

Note: The configuration of the `scan_default_assume_representation_policy` variable affects the POI creation in the `formal_bring_up` flow.

Syntax

```
formal_bring_up -trace (<file_list>)+  
  | [-shm (<directory>)+]  
  | -hier_path <path>  
  | [-file <file_name>]  
  | [-task <task_name>]
```

Argument	Definition
<code>-trace (file_list)+</code>	Create assumptions from the specified trace files.
<code>[-shm (directory_list)+]</code>	Create assumptions from the specified SHM files.
<code>-hier_path path</code>	Specify the hierarchical path.
<code>[-file file_name]</code>	Generate an FPV App setup file (Tcl file and SVA file).
<code>[-task task_name]</code>	Export the property candidates to the specified task

Return

Various

Examples

```
% formal_bring_up -trace {t1.fsdb t2.fsdb} -hier_path path  
% formal_bring_up -shm {t1.shm t2.shm t3.shm} -hier_path path -task myTask  
% formal_bring_up -trace {t1.fsdb t2.fsdb} -hier_path path -file mySetup
```

See Also

[check_bps](#)
[get scan default assume representation policy](#)
[scan](#)
[scope](#)
[set scan seek mode](#)
[set_scope_clock_edge](#)
[task](#)
[waveform](#)

fsm_checks

Description

Use this command to generate properties for proving the reachability of:

- State machine transitions
- Combinations of states across pairs of state machines (crosses)

To perform FSM checks, the command automatically detects state variables or accepts user-specified state variables and extracts cover properties you can prove or Visualize.

Syntax

```
fsm_checks [-add_fsm <fsm_spec>]
            [-add_cross <cross_spec>]
            [-cross [-auto [-depth <N>]]]
            [-trans]
```

Argument	Definition
no arguments	Print a list of FSMs, their states, and any crosses that will be acted upon if the <code>fsm_checks</code> command is run with <code>-trans</code> or <code>-cross</code> .
<code>[-add_fsm fsm_spec]</code>	Define a new FSM to be used when creating the properties.  <i>Important</i> <ul style="list-style-type: none">■ You must supply an FSM spec of the form <code>{<fsm_signal> <list of FSM states>}</code>■ <code>list of FSM states</code> is an alternating list of values and labels.
<code>[-add_cross cross_spec]</code>	Define the pair of FSMs to “cross,” that is, prove the reachability of each combination of states across the two FSMs. Supply a <code>cross_spec</code> of the form <code>{<FSM1> <FSM2>}</code>

JasperGold Apps Command Reference Manual

General Commands

`[-cross [-auto [-depth N]]]`

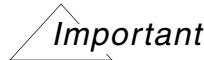
Generate the specified cross-state properties.



- It creates a new task for each state machine pair.
- The task naming format is:
`CROSS_<full path to FSM1>_<full path to FSM2>`
- The automatically extracted property name format is:
`<value>_&&_<value>`
- If you use `-auto`, the tool attempts to automatically determine which FSMs should be crossed. In this mode, the tool evaluates the fanin of each FSM, and if there is another FSM within *N* ranks of registers (as defined by the `-depth` switch and with a default value of 1), then those two FSMs are crossed.
- The value of `-depth` is the level of registers to look through. It must be a positive integer.

`[-trans]`

Generate properties for state transition reachability for both user-defined and automatically identified FSMs.



- It creates a new task for each state machine pair.
- The task naming format is:
`TRANS_<full path to FSM>`
- The automatically extracted property name format is:
`<value>_to_<value>`

Return

No return value.

Examples

```
// Define a new FSM to be used when creating the properties
% fsm_checks -add_fsm {my_fsm {0 `IDLE 1 `WRITE 2 `READ 3 `WAIT}}
```

```
// Define the pair of FSMs to "cross"
% fsm_checks -add_cross {my_fsm1 my_fsm2}
```

```
// Construct a conceptual FSM from the concatenation of sig1, sig2, and sig3
% fsm_checks -add_fsm {{{sig1,sig2,sig3}}} {val lab val lab val lab}}
```

See Also

No related commands

get_annotation

Description

Use the `get_annotation` command to print annotations to the screen.

Syntax

```
get_annotation (-property <property_name>) | (-signal <signal_name>)
```

Argument	Definition
<code>-property</code> <i>property_name</i>	Print the annotation for the specified property.
<code>-signal</code> <i>signal_name</i>	Print the annotation for the specified signal.

Default

None

Return

`get_annotation` returns the user-defined annotation for the specified property or signal.

Examples

```
% get_annotation -signal clk
Global clock
```

See Also

[set_annotation](#)

get_clock_info

Description

Use the `get_clock_info` command to launch the Clock Information window where you can get detailed information regarding your clock configuration and environment. This window presents information regarding the clocking signals and their connected drivers, related registers, related properties, and aliases. It also reports clock characteristics and runs a sanity check on each clock. The window also provides information about declared clocks, including their reference and derived clocks and inputs rated by them.

The window also provides access to simplified clock tree visualizations with the Schematic Viewer and Graph GUIs.

Note: In the current version, the sanity checks for gated glitch, flops whose input pin logic involves the flop clocking signal, and waivers for the sanity check are disabled.

Syntax

```
get_clock_info -gui
```

Argument	Definition
<code>-gui</code>	Launches a GUI that displays all the information in an easy-to-read manner.

Default

None

Return

No return values

Examples

```
// Launch Clock Information window  
% get_clock_info -gui
```

JasperGold Apps Command Reference Manual

General Commands

See Also

[clock](#)

[sanity_check](#)

get_decomposed_expression

Description

Use the `get_decomposed_expression` command to get the sub-expressions of the provided argument.

Note:

- List of decomposable Verilog operators:
 - Logical operators: `&&`, `||`, `!`
 - Bitwise operators: `&`, `|`, `^`, `~`, `~&`, `~|`, `~^`
 - Sequential operators: `##`, `|=>`, `|->`
 - Conditional (ternary '?') operator: `COND_EXPR ? THEN_EXPR : ELSE_EXPR`
 - Comparison operators: `==`, `!=`, `====`, `!==`, `>`, `<=`, `<`, `>=`
- The remaining Verilog operators can appear in a given expression, but the tool will not decompose them.
- `get_decomposed_expression` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

Syntax

```
get_decomposed_expression ( <signal_name> | <property_name>
                           | <literal_expression>)
                           [-max_depth <N>]
```

Argument	Definition
<i>signal_name</i> <i>property_name</i> <i>literal_expression</i> [-max_depth <i>N</i>]	<p>Get the sub-expressions of the provided argument.</p> <ul style="list-style-type: none">■ <i>signal_name</i>: The command returns the sub-expressions of the specified signal's driving expression.■ <i>property_name</i>: The command returns the sub-expressions of the specified property's expression. <p>Note: This command does not decompose embedded property expressions. In this case, the command returns the specified property's name.</p> <ul style="list-style-type: none">■ <i>literal_expression</i>: The command returns the sub-expressions of the specified expression. See examples below. <p>Note: VHDL expressions are not supported.</p> <ul style="list-style-type: none">■ Use <i>-max_depth</i> to get the sub-expressions up to depth, where <i>N</i> is an unbounded non-negative integer (0, 1, 2, ..., \$). <p>Note: The default depth value is \$ (returns all sub-expressions in all depths).</p>

Default

None

Return

`get_decomposed_expression` returns a list of expression strings corresponding to the sub-expressions of the provided argument.

Examples

```
% get_decomposed_expression { a ##2 b ##[5:7] c } -max_depth 0
# Returns a list containing the following expressions:
#     a ##2 b ##[5:7] c
```

JasperGold Apps Command Reference Manual

General Commands

```
% get_decomposed_expression { a ##2 b ##[5:7] c } -max_depth 1
# Returns a list containing the following expressions:
#   a ##2 b ##[5:7] c
#   a ##2 b
#   c

% get_decomposed_expression { a ##2 b ##[5:7] c } -max_depth 2
# OR
% get_decomposed_expression { a ##2 b ##[5:7] c } -max_depth 3
# OR
% get_decomposed_expression { a ##2 b ##[5:7] c } -max_depth $
# OR
% get_decomposed_expression { a ##2 b ##[5:7] c }
# Returns a list containing the following expressions:
#   a ##2 b ##[5:7] c
#   a ##2 b
#   c
#   a
#   b

% get_decomposed_expression { a & ~( b | ( c & ~d ) ) } -max_depth 0
# Returns a list containing the following expressions:
#   (a & ~b & (~c | d))

% get_decomposed_expression { a & ~( b | ( c & ~d ) ) } -max_depth 1
# Returns a list containing the following expressions:
#   (a & ~b & (~c | d))
#   (a & ~b)
#   (a & (~c | d))
#   { (~b & (~c | d))

% get_decomposed_expression { a & ~( b | ( c & ~d ) ) } -max_depth 2
# Returns a list containing the following expressions:
#   (a & ~b & (~c | d))
#   (a & ~b)
#   (a & (~c | d))
#   { (~b & (~c | d))
#   a
#   ~b
#   a
```

JasperGold Apps Command Reference Manual

General Commands

```
#      (~c | d)
#      ~b
#      (~c | d)

% get_decomposed_expression { a & ~( b | ( c & ~d ) ) } -max_depth 3
# OR
% get_decomposed_expression { a & ~( b | ( c & ~d ) ) } -max_depth 4
# OR
% get_decomposed_expression { a & ~( b | ( c & ~d ) ) } -max_depth $#
# OR
% get_decomposed_expression { a & ~( b | ( c & ~d ) ) }
# Returns a list containing the following expressions:
#      (a & ~b & (~c | d))
#      (a & ~b)
#      (a & (~c | d))
#      { (~b & (~c | d))
#      a
#      ~b
#      a
#      (~c | d)
#      ~b
#      (~c | d)
#      ~c
#      d
#      ~c
#      d
```

See Also

No related commands

get_design_info

Description

Design Information is a feature that gives insight into the formal computational complexity of the design. Access this feature through the GUI interface (for example, right-click on a property in a *Property Table* and choose *Design Information*) or the `get_design_info` Tcl command.

This command provides various structural and functional information for a specific scope within the given design. It provides:

- Structural metrics, including the number of gates, flops, and so forth
- Functional metrics, including the set of counters, any special values of the counters, the set of finite-state machines, and so forth.

In addition, for modules and instances, this command provides the following:

- Module source information, including the number of RTL lines, instances, and embedded properties
- Current task information, including the number of stopats in the current task.

Note: This command reports stopats that meet the following requirements:

- In the current task
- In the specified scope
- In the COI of properties in the current task and scope

For a more general list of stopats, use the `stopat -list` command (see “[stopat](#)” on page 840).



- You can specify the scope (focus) for which this information is provided. For example, you might want to get the list of flops in the transitive fanin of all the properties in a task or the list of counters feeding to an output of the design. You can also control the exact set of information provided through options.
- If you do not specify arguments, this command returns information for the top instance.

- The `-list` switch supports multiple enum types in a single Tcl command, but you must supply at least one enum. For example, you can specify the following:
`get_design_info -instance myInstance -list output input`
In this example, the command returns the analysis information for outputs and inputs. And the return value (the single line at the end) is a list of output names followed by input names. Use `-typed_list` to add a type element before each of the requested lists.
- Use the commands
`set_design_exploration_include_connected_assumptions` and
`set_design_exploration_include_clock_logic` to control the logic the tool will traverse when gathering information.

TIP: Use tab-completion to form these commands.

Use the following links to jump to a scope or functionality that interests you:

- [Listing Design Info for the Top-Instance Scope](#) on page 568
- [Listing Design Info for the “-instance” Scope](#) on page 576
- [Listing Design Info for the “-module” Switch](#) on page 583
- [Listing Design Info for the “-signal” Scope](#) on page 590
- [Listing Design Info for the “-property” Scope](#) on page 595
- [Listing Design Info for the “-task” Scope](#) on page 600
- [Additional Optional Switches for Customizing Lists](#) on page 606
- [FSM and Counter Detection](#) on page 608
- [Cached-Info Clearing](#) on page 608
- [Symmetry Check](#) on page 609
- [Depth Calculation](#) on page 609

Syntax

```
get_design_info

get_design_info -list
  ([parameter] [local_parameter] [constant] [enum] [output] [input] [inout]
   [stopat] [undriven] [undriven_internal] [multiple_driven] [bbox_mod]
   [bbox_inst] [bbox_in] [bbox_out] [bbox inout] [bbox_in_instance_port]
   [bbox_out_instance_port] [bbox inout_instance_port]
   [basic] [counter] [fsm] [array [-no_aggregate]]
   [register] [flop] [latch] [gate]
   [multiplier] [divider] [modulus]
```

JasperGold Apps Command Reference Manual

General Commands

```
[module] [module_name [module_name_no_param]]  
[instance [-depth <N>]] [architecture]  
[signal] [wire] [property]  
[assert] [assume] [cover] [macro]  
) +  
[-verbosity (high | low | silent)]  
[-typed_list]  
[-silent]  
[-file <file_name> [-force]]  
[-filter <filter_expression>  
[-regexp] [-case_sensitive]]  
[-gui]  
[-include_hier_path]  
  
get_design_info -instance <instance_name>  
[-list  
 ([parameter] [local_parameter] [constant] [enum] [output] [input] [inout]  
 [stopat] [undriven] [undriven_internal] [multiple_driven]  
 [bbox_mod] [bbox_inst] [bbox_in] [bbox_out] [bbox inout]  
 [bbox_in_instance_port] [bbox_out_instance_port]  
 [bbox inout_instance_port])  
 [basic] [counter] [fsm] [array [-no_aggregate]]  
 [register] [flop] [latch] [gate]  
 [multiplier] [divider] [modulus]  
 [module] [module_name [module_name_no_param]]  
 [instance [instance [-depth <N>]]  
 [architecture] [signal] [wire]  
 [property] [assert] [assume] [cover]  
 [parent] [empty_inst]  
) +  
]  
[-verbosity (high | low | silent)]  
[-typed_list]  
[-silent]  
[-file <file_name> [-force]]  
[-filter <filter_expression>  
[-regexp] [-case_sensitive]]  
[-gui]  
[-include_hier_path]  
  
get_design_info -module <module_name>  
[-list  
 ([parameter] [constant] [enum] [output] [input] [inout] [stopat]  
 [undriven] [undriven_internal] [multiple_driven] [bbox_mod]  
 [bbox_inst] [bbox_in] [bbox_out]  
 [bbox inout] [bbox_in_instance_port]  
 [bbox_out_instance_port] [bbox inout_instance_port])  
 [basic] [counter] [fsm] [array [-no_aggregate]]  
 [register] [flop] [latch] [gate]  
 [multiplier] [divider] [modulus]  
 [module] [module_name [module_name_no_param]]  
 [instance] [architecture] [signal] [wire]
```

JasperGold Apps Command Reference Manual

General Commands

```
[property] [assert] [assume] [cover]
)++
]
[-verbosity (high | low | silent)]
[-typed_list]
[-silent]
[-file <file_name> [-force]]
[-filter <filter_expression>
[-regexp] [-case_sensitive]]
[-gui]
[-include_hier_path]

get_design_info -signal <signal_name>+
[-list
 ([basic] [counter] [fsm] [array [-no_aggregate]]]
 [register] [flop] [latch] [gate] [input]
 [undriven] [undriven_internal]
 [connected_assumptions]
 [special_values] [-module]
 [instance] [signal] [wire]
 [-parent]
 )+
]
[-fanout]
[-verbosity (high | low | silent)]
[-typed_list]
[-silent]
[-file <file_name> [-force]]
[-filter <filter_expression>
[-regexp] [-case_sensitive]]
[-gui]

get_design_info -property <property_name>+
[-regexp] [-region |-coi]
[-list
 ([basic] [counter] [fsm] [array [-no_aggregate]]]
 [register] [flop] [latch] [gate] [input]
 [undriven] [undriven_internal]
 [connected_assumptions]
 [module] [instance] [signal] [wire] [property]
 [assert] [assume] [cover] [stopat]
 )+
]
[-verbosity (high | low | silent)]
[-typed_list] [-silent]
[-file <file_name> [-force]]
[-filter <filter_expression> [-regexp] [-case_sensitive]]
[-gui]

get_design_info -task <task_name>+
[-region |-coi] [-compute_all]
[-list
```

JasperGold Apps Command Reference Manual

General Commands

```
([basic] [counter] [fsm] [array [-no_aggregate]]  
[register] [flop] [latch] [gate] [input]  
[undriven] [undriven_internal]  
[connected_assumptions]  
[module] [instance] [signal] [wire]  
[property] [assert] [assume] [cover]  
) +  
]  
[-verbosity (high | low | silent)]  
[-typed_list] [-silent]  
[-file <file_name> [-force]]  
[-filter <filter_expression> [-regexp] [-case_sensitive]]  
[-gui]  
  
get_design_info -signal <signal_name>+  
    -is_counter |-is_fsm  
    [-verbosity high]  
  
get_design_info -clear  
  
get_design_info -symmetry -from <from_bus> -to <to_bus>  
    [-full |-quick]  
  
get_design_info (-signal <name>+ |-property <name>+)  
    -list (flop | latch | register)  
    -depth <N>  
    [-boundary] [-silent]
```

Argument	Definition
Listing Design Info for the Top-Instance Scope	
no arguments	<p>If you do not specify arguments, this command returns the following statistics for the top instance:</p> <ul style="list-style-type: none">■ Instance source info, such as the number of RTL instances, embedded properties by type, and the number of lines of code■ Instance structural metrics such as the number of ports, registers, and gates <p>The output of this command is the sum of all registers (flops and latches).</p>

JasperGold Apps Command Reference Manual

General Commands

```
-list
([parameter] [local_parameter] [constant] {enum} [output] [input] [inout]
[stopat] [undriven] [undriven_internal] [multiple_driven] [bbox_mod]
[bbox_inst] [bbox_in] [bbox_out] [bbox inout] [bbox_in_instance_port]
[bbox_out_instance_port] [bbox inout_instance_port]
[basic] [counter] [fsm] [array [-no_aggregate]]
[register] [flop] [latch] [gate]
[multiplier] [divider] [modulus]
[module] [module_name [module_name_no_param]]
[instance [-depth N]] [architecture]
[signal] [wire] [property]
[assert] [assume] [cover] [macro]
) +
[-verbosity (high | low | silent)]
[-typed_list]
[-silent]
[-file file_name [-force]]
[-filter filter_expression
[-regexp] [-case_sensitive]]
[-gui]
[-include_hier_path]
```

Get design information for the top instance.



- This analysis ignores hard stopats, instance stopats, and soft stopats.
- Below you will find definitions of the supported -list enums for the top-instance scope.
- To refine your list, for example to use it in scripting, refer to [“Additional Optional Switches for Customizing Lists”](#) on page 606.

Supported “-list” Enums for the Top-Instance Scope

- **parameter:** Get the parameters listed in a design.
The tool returns a string of parameter-value pairs. Example:
`[param1] [value1] [param2] [value2]`.
- **local_parameter:** Get the local parameters listed in a design.
The tool returns a string of parameter-value pairs. Example:
`[param1] [value1] [param2] [value2]`.

JasperGold Apps Command Reference Manual

General Commands

Top Instance (Continued)

- **constant:** Get the VHDL constants declared in the top instance. The tool returns a string of constant-value pairs. Example:
[const1] [value1] [const2] [value2].
- **enum:** List all enums instantiated in the top instance scope, displaying their names followed by their mnemonics and their values. This command returns a list of space-separated enum names. Example: [enum1] [enum2] [enum3].
- **output:** List the names of outputs and their sizes.
 - This command returns a list of space-separated output names.
 - Use `-verbosity high` to also list the number of flops and gates in their COI and minimum and maximum depth. (For this command, depth means the number of visible gates the tool must traverse to arrive from the output into a primary input.)
- **input:** List the number of inputs followed by their names and sizes.
 - Inputs may be PIs, signals with hard or soft stopats, or output signals of instance stopats.
 - This command returns a list of space-separated input names.
- **inout:** List the number of inout followed by their names and sizes. This command returns a list of space-separated inout names.
- **stopat:** List all the stopats that are in the top-instance scope and in the current task.

Note: This command reports stopats that meet these requirements:

- In the current task
- In the top-instance scope
- In the COI of properties in the current task and top-instance

For a more general list of stopats, use the `stopat -list` command (see "[stopat](#)" on page 840).

- **undriven:** List the names and sizes of all undriven signals in the scope of the top instance, including its sub-instances. This command returns a list of space-separated names of undriven signals.

Top Instance (Continued)

- `undriven_internal`: List the names and sizes of all undriven signals that are not primary inputs and that are in the scope of the top instance, including its sub-instances.
This command returns a list of space-separated names of undriven internal signals.
- `multiple_driven`: List the names and sizes of all multiple driven signals in the scope of the top instance, including its sub-instances.
This command returns a list of space-separated names of multiple driven signals.
- `bbox_mod`: List all modules that have been black boxed under the specified instance.
- `bbox_inst`: List all instances that have been black boxed under the specified instance.
- `bbox_in`
`bbox_out`
`bbox inout`
List the names and sizes of the signals connected to the specified kind of port (input, output, or inout) of black-boxed modules. (Ports that behave as both input and output ports are inout ports.)
This command returns a list of space-separated signal names.
- `bbox_in_instance_port`
`bbox_out_instance_port`
`bbox inout_instance_port`
List triples of the specified kind of black-boxed ports (input, output, or inout), black-boxed instance names, and black-boxed connection ports for black-boxed modules in the top-instance scope.
This command returns a list of triples.
- `basic`: List the default set of information for the top-instance scope.
 - The Tcl command returns the number of flops in the top-instance scope.
 - The command `get_design_info -list basic` has the same effect as `get_design_info` with no additional switches.
- `counter`: identify counters in the vector of flops detected in the top-instance scope.

Top Instance (Continued)

- ❑ In addition to listing the names of flops, this command can print the following information when you use the `-verbosity high` switch:
 - Number of bits – flop width.
 - Size logic – number of gates inside the flop's COI.
 - Max depth – maximum number of flops from the original flop to a primary input.
 - Special values – values that are critical to the counter.
 - Direct flops – list of flops directly connected to the original flop. This list includes flop types (FSM, COUNTER, or OTHER).
 - COI flops – list of flops in the COI of the original flop.
- This command returns a list of space-separated counter names.`fsm`: identify FSMs in the vector of flops detected in the top-instance scope.
 - ❑ In addition to listing the names of flops, this command can print the following information when you use the `-verbosity high` switch:
 - Number of bits – flop width.
 - Size logic – number of gates inside the flop's COI.
 - Max depth – maximum number of flops from the original flop to a primary input.
 - Special values – values that are critical to the FSM.
 - Direct flops – list of flops directly connected to the original flop. This list includes flop types (FSM, COUNTER, or OTHER).
 - COI flops – list of flops in the COI of the original flop.
 - ❑ This command returns a list of space-separated FSM names.
- `array`: identify multidimensional arrays (including FIFOs) in the vector of flops detected in the top-instance scope.
 - ❑ Use `-no_aggregate` to get a bit-blasted list.

Top Instance (Continued)

- ❑ In addition to listing the names of flops, this command can print the following information when you use the `-verbosity high` switch:
 - Number of bits – flop width.
 - Size logic – number of gates inside the flop's COI.
 - Max depth – maximum number of flops from the original flop to a primary input.
 - Direct flops – list of flops directly connected to the original flop. This list includes flop types (FSM, COUNTER, or OTHER).
 - COI flops – list of flops in the COI of the original flop.
 - ❑ This command returns a list of space-separated array names.

■ `register`: List all flops and latches in the top-instance scope, displaying their names and sizes (in bits).
This command returns a list of space-separated flop and latch names.

■ `flop`: List all flops in the top-instance scope, displaying their names and sizes (in bits).

 - ❑ The list includes flop numbers shown as $x(y)(z$ property flop bits).
 - x is the total number of flops.
 - y is the sum of bits of the flops, not including property flop bits.
 - z is the sum of bits of property flops, which are generated as observers for SVA and PSL properties.
 - ❑ This command returns a list of space-separated flop names.

■ `latch`: List all latches in the top-instance scope, displaying their names and sizes (in bits).
This command returns a list of space-separated latch names.

■ `gate`: List the number of gates and gate bits for the top instance.
This command returns the number of gates.

JasperGold Apps Command Reference Manual

General Commands

Top Instance (Continued)

- `multiplier`
`divider`
`modulus`
List all multipliers, dividers, or modulus in the top-instance scope.
This command lists two kinds of the specified operator:
 1. Black-boxed – each multiplier, divider, or modulus is described by three signals: {`out_internal` `left_internal` `right_internal`}
The right and left internal signals are the operands, and `out_internal` is the resulting signal.
 2. Cell-defined – each multiplier, divider, or modulus that has not been black boxed is cell defined. By default, all multipliers, dividers, and modulus are black boxed; however, you can use `elaborate -bbox_mul`, `-bbox_div`, and `-bbox_mod` to avoid black boxing them. `out_internal` is the resulting signal.
 - Each operator is described by three signals:
{`out_internal` `left_internal` `right_internal`}
 - The right and left internal signals are the operands, and This functionality is useful for manual abstraction.
 - This command returns a list of space-separated triples of signals.
- `module`: List all modules in the top-instance scope, displaying their names. This command returns a list of space-separated module names.
- `module_name`: return the module name for the top instance.
- `module_name_no_param`: Return the specified module name without the non-default parameter values.
- `instance`: List all instances in the top-instance scope, displaying their names.
This command returns a list of space-separated instance names. The list does not include black-boxed instances (see `bbox_inst`).
Use `-depth` to filter the list of instances using the depth level, listing the instances until the chosen depth. Depth 0 is the top instance.
- `architecture`: List all architectures in the top-instance scope, displaying their names.
This command returns a list of space-separated architecture names.

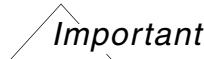
Top Instance (Continued)

- **signal:** List all signals in the top-instance scope, displaying their names and sizes in bits.
 - Signals are the set of all registers, primary inputs, primary outputs, and wires (signals = registers + PIs + POs + wires).
 - This command returns a list of space-separated signal names.
 - **wire:** List all wires in the top-instance scope, displaying their names and sizes in bits.
 - Wires are all the signals not classified as registers, primary inputs, or primary outputs (wires = signals - registers - PIs - POs).
 - This command returns a list of space-separated signal names.
 - **property**
assert
assume
cover
List the current task's embedded and Tcl properties of the specified type (all embedded and Tcl properties, embedded and Tcl assertions, embedded and Tcl assumptions, or embedded and Tcl cover directives) in the top-instance scope.
This command lists the number and names of properties and returns a list of space-separated property names.
 - **macro:** List all macros in the elaborated design.
This command returns a list of space-separated macro names with their values.
-

Listing Design Info for the “-instance” Scope

```
-instance instance_name
  [-list
    ([parameter] [local_parameter] [constant] [enum] [output] [input] [inout]
     [stopat] [undriven] [undriven_internal] [multiple_driven]
     [bbox_mod] [bbox_inst] [bbox_in] [bbox_out] [bbox inout]
     [bbox_in_instance_port] [bbox_out_instance_port]
     [bbox inout_instance_port])
    [basic] [counter] [fsm] [array [-no_aggregate]]
    [register] [flop] [latch] [gate]
    [multiplier] [divider] [modulus]
    [module] [module_name [module_name_no_param]]
    [instance [instance [-depth N]]]
    [architecture] [signal] [wire]
    [property] [assert] [assume] [cover]
    [parent] [empty_inst]
  )+
]
  [-verbosity (high | low | silent)]
  [-typed_list]
  [-silent]
  [-file file_name [-force]]
  [-filter filter_expression
  [-regexp] [-case_sensitive]]
  [-gui]
  [-include_hier_path]
```

Get design information for the specified instance.



- This analysis ignores hard stopats, instance stopats, and soft stopats.
- Below you will find definitions of the supported -list enums for the -instance scope.
- To refine your list, for example to use it in scripting, refer to [“Additional Optional Switches for Customizing Lists”](#) on page 606.

Supported “-list” Enums for the “-instance” Scope

- parameter: Get the parameters listed in a design.
The tool returns a string of parameter-value pairs. Example:
[param1] [value1] [param2] [value2].

-instance (Continued)

- **local_parameter**: Get the local parameters listed in a design. The tool returns a string of parameter-value pairs. Example: [param1] [value1] [param2] [value2].
- **constant**: Get the VHDL constants declared in the specified instance. The tool returns a string of constant-value pairs. Example: [const1] [value1] [const2] [value2].
- **enum**: List all enums instantiated in the specified instance's scope, displaying their names followed by their mnemonics and their values. This command returns a list of space-separated enum names. Example: [enum1] [enum2] [enum3].
- **output**: List the names of outputs and their sizes.
 - This command returns a list of space-separated output names.
 - Use -verbosity high to also list the number of flops and gates in their COI and minimum and maximum depth. (For this command, depth means the number of visible gates the tool must traverse to arrive from the output into a primary input.)
- **input**: List the number of inputs followed by their names and sizes.
 - Inputs may be PIs, signals with hard or soft stopats, or output signals of instance stopats.
 - This command returns a list of space-separated input names.
- **inout**: List the number of inout followed by their names and sizes. This command returns a list of space-separated inout names.
- **stopat**: List all the stopats that are in the -instance scope and in the current task.

Note: This command reports stopats that meet these requirements:

- In the current task
- In the -instance scope
- In the COI of properties in the current task and -instance

For a more general list of stopats, use the `stopat -list` command (see "[stopat](#)" on page 840).

JasperGold Apps Command Reference Manual

General Commands

-instance (Continued)

- **undriven:** List the names and sizes of all undriven signals in the scope of the specified instance, including its sub-instances.
This command returns a list of space-separated names of undriven signals.
- **undriven_internal:** List the names and sizes of all undriven signals that are not primary inputs and that are in the scope of the specified instance, including its sub-instances.
This command returns a list of space-separated names of undriven internal signals.
- **multiple_driven:** List the names and sizes of all multiple driven signals in the scope of the specified instance, including its sub-instances.
This command returns a list of space-separated names of multiple driven signals.
- **bbox_mod:** List all modules that have been black boxed under the specified instance.
- **bbox_inst:** List all instances that have been black boxed under the specified instance.
- **bbox_in**
bbox_out
bbox inout
List the names and sizes of the signals connected to the specified kind of port (input, output, or inout) of black-boxed modules. (Ports that behave as both input and output ports are inout ports.)
This command returns a list of space-separated signal names.
- **bbox_in_instance_port**
bbox_out_instance_port
bbox inout_instance_port
List triples of the specified kind of black-boxed ports (input, output, or inout), black-boxed instance names, and black-boxed connection ports for black-boxed modules in the top-instance scope.
- **basic:** List the default set of information for the -instance scope.
 - The Tcl command returns the number of flops in the -instance scope.
 - The command `get_design_info -list basic` has the same effect as `get_design_info` with no additional switches.

JasperGold Apps Command Reference Manual

General Commands

-instance (Continued)

- **counter:** identify counters in the vector of flops detected in the -instance scope.
 - In addition to listing the names of flops, this command can print the following information when you use the `-verbosity high` switch:
 - Number of bits – flop width.
 - Size logic – number of gates inside the flop's COI.
 - Max depth – maximum number of flops from the original flop to a primary input.
 - Special values – values that are critical to the counter.
 - Direct flops – list of flops directly connected to the original flop. This list includes flop types (FSM, COUNTER, or OTHER).
 - COI flops – list of flops in the COI of the original flop.
 - This command returns a list of space-separated counter names.
 - **fsm:** identify FSMs in the vector of flops detected in the -instance scope.
 - In addition to listing the names of flops, this command can print the following information when you use the `-verbosity high` switch:
 - Number of bits – flop width.
 - Size logic – number of gates inside the flop's COI.
 - Max depth – maximum number of flops from the original flop to a primary input.
 - Special values – values that are critical to the FSM.
 - Direct flops – list of flops directly connected to the original flop. This list includes flop types (FSM, COUNTER, or OTHER).
 - COI flops – list of flops in the COI of the original flop.
 - This command returns a list of space-separated FSM names.
-

JasperGold Apps Command Reference Manual

General Commands

-instance (Continued)

- **array:** identify multidimensional arrays (including FIFOs) in the vector of flops detected in the -instance scope.
 - Use `-no_aggregate` to get a bit-blasted list.
 - In addition to listing the names of flops, this command can print the following information when you use the `-verbosity high` switch:
 - Number of bits – flop width.
 - Size logic – number of gates inside the flop's COI.
 - Max depth – maximum number of flops from the original flop to a primary input.
 - Direct flops – list of flops directly connected to the original flop. This list includes flop types (FSM, COUNTER, or OTHER).
 - COI flops – list of flops in the COI of the original flop.
 - This command returns a list of space-separated array names.
 - **register:** List all flops and latches in the -instance scope, displaying their names and sizes (in bits).
This command returns a list of space-separated flop and latch names.
 - **flop:** List all flops in the -instance scope, displaying their names and sizes (in bits).
 - The list includes flop numbers shown as $x(y)(z$ property flop bits).
 - x is the total number of flops.
 - y is the sum of bits of the flops, not including property flop bits.
 - z is the sum of bits of property flops, which are generated as observers for SVA and PSL properties.
 - This command returns a list of space-separated flop names.
 - **latch:** List all latches in the -instance scope, displaying their names and sizes (in bits).
This command returns a list of space-separated latch names.
 - **gate:** List the number of gates and gate bits for the top instance.
This command returns the number of gates.
-

-instance (Continued)

- `multiplier`
`divider`
`modulus`

List all multipliers, dividers, or modulus in the `-instance` scope.

This command lists two kinds of the specified operator:

1. Black-boxed – each multiplier, divider, or modulus is described by three signals: `{out_internal left_internal right_internal}`

The right and left internal signals are the operands, and `out_internal` is the resulting signal.

2. Cell-defined – each multiplier, divider, or modulus that has not been black boxed is cell defined. By default, all multipliers, dividers, and modulus are black boxed; however, you can use `elaborate -bbox_mul`, `-bbox_div`, and `-bbox_mod` to avoid black boxing them.

- Each operator is described by three signals:
`{out_internal left_internal right_internal}`
- The right and left internal signals are the operands, and `out_internal` is the resulting signal. This functionality is useful for manual abstraction.
- This command returns a list of space-separated triples of signals.

- `module`: List all modules in the `-instance` scope, displaying their names. This command returns a list of space-separated module names.

- `module_name`: return the module name for the top instance.

- `module_name_no_param`: Return the specified module name without the non-default parameter values.

- `instance`: List all instances in the `-instance` scope, displaying their names.

This command returns a list of space-separated instance names. The list does not include black-boxed instances (see `bbox_inst`).

Use `-depth` to filter the list of instances using the depth level, listing the instances until the chosen depth. Depth 0 is the top instance.

JasperGold Apps Command Reference Manual

General Commands

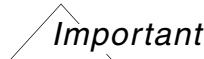
-instance (Continued)

- **architecture**: List all architectures in the top-instance scope, displaying their names.
This command returns a list of space-separated architecture names.
- **signal**: List all signals in the -instance scope, displaying their names and sizes in bits.
 - Signals are the set of all registers, primary inputs, primary outputs, and wires (signals = registers + PIs + POs + wires).
 - This command returns a list of space-separated signal names.
- **wire**: List all wires in the -instance scope, displaying their names and sizes in bits.
 - Wires are all the signals not classified as registers, primary inputs, or primary outputs (wires = signals - registers - PIs - POs).
 - This command returns a list of space-separated signal names.
- **property**
assert
assume
cover
List the current task's embedded and Tcl properties of the specified type (all embedded and Tcl properties, embedded and Tcl assertions, embedded and Tcl assumptions, or embedded and Tcl cover directives) in the -instance scope.
This command lists the number and names of properties and returns a list of space-separated property names.
- **parent**: List the parent of the specified instance.
- **empty_inst**: List the sub-instances of the specified instance that have no internal logic, just ports.

Listing Design Info for the “-module” Switch

```
-module module_name
[-list
  ([parameter] [constant] [enum] [output] [input] [inout] [stopat]
   [undriven] [undriven_internal] [multiple_driven] [bbox_mod] [bbox_inst]
   [bbox_in] [bbox_out] [bbox inout] [bbox_in_instance_port]
   [bbox_out_instance_port] [bbox inout instance_port])
  [basic] [counter] [fsm] [array [-no_aggregate]]
  [register] [flop] [latch] [gate]
  [multiplier] [divider] [modulus]
  [module] [module_name [module_name_no_param]]
  [instance] [architecture] [signal] [wire]
  [property] [assert] [assume] [cover]
) +
]
[-verbosity (high | low | silent)]
[-typed_list]
[-silent]
[-file file_name [-force]]
[-filter filter_expression]
[-regexp] [-case_sensitive]
[-gui]
[-include_hier_path]
```

Get design information for the specified module.



- This analysis ignores hard stopats, instance stopats, and soft stopats.
- Below you will find definitions of the supported -list enums for the -module switch.
- To refine your list, for example to use it in scripting, refer to [“Additional Optional Switches for Customizing Lists”](#) on page 606.

Supported “-list” Enums for the “-module” Switch

- **parameter:** Get the parameters listed in a design.
The tool returns a string of parameter-value pairs. Example:
`[param1] [value1] [param2] [value2]`.
- **constant:** Get the VHDL constants declared in the specified module.
The tool returns a string of constant-value pairs. Example:
`[const1] [value1] [const2] [value2]`.
- **enum:** List all enums instantiated in the specified module’s scope,
displaying their names followed by their mnemonics and their values.
This command returns a list of space-separated enum names.
Example: `[enum1] [enum2] [enum3]`.
- **output:** List the names of outputs and their sizes.
 - This command returns a list of space-separated output names.
 - Use `-verbosity high` to also list the number of flops and gates
in their COI and minimum and maximum depth. (For this
command, depth means the number of visible gates the tool must
traverse to arrive from the output into a primary input.)
- **input:** List the number of inputs followed by their names and sizes.
 - Inputs may be PIs, signals with hard or soft stopats, or output
signals of instance stopats.
 - This command returns a list of space-separated input names.
- **inout:** List the number of inout followed by their names and sizes.
This command returns a list of space-separated inout names.
- **stopat:** List all the stopats that are in an instance of the specified
module.

Note: This command reports stopats that meet these requirements:

- In the current task
- In the specified module scope
- In the COI of properties in the current task and specified
module

For a more general list of stopats, use the `stopat -list` command
(see “[stopat](#)” on page 840).

JasperGold Apps Command Reference Manual

General Commands

-module (Continued)

- **undriven:** List the names and sizes of all undriven signals in the scope of the instance of the specified module, including its sub-instances.
This command returns a list of space-separated names of undriven signals.
 - **undriven_internal:** List the names and sizes of all undriven signals that are not primary inputs and that are in the scope of the specified instance of the specified module, including its sub-instances.
This command returns a list of space-separated names of undriven internal signals.
 - **multiple_driven:** List the names and sizes of all multiple driven signals in the scope of the instance of the specified module, including its sub-instances.
This command returns a list of space-separated names of multiple driven signals.
 - **bbox_mod:** List all modules that have been black boxed under the specified instance.
 - **bbox_inst:** List all instances that have been black boxed under the specified module.
 - **bbox_in**
bbox_out
bbox inout
List the names and sizes of the signals connected to the specified kind of port (input, output, or inout) of black-boxed modules. (Ports that behave as both input and output ports are inout ports.)
This command returns a list of space-separated signal names.
 - **bbox_in_instance_port**
bbox_out_instance_port
bbox inout_instance_port
List triples of the specified kind of black-boxed ports (input, output, or inout), black-boxed instance names, and black-boxed connection ports for black-boxed modules in the specified module scope.
This command returns a list of triples.
 - **basic:** List the default set of information for the specified module. The Tcl command returns the number of flops in the specified module. The command `get_design_info -list basic` has the same effect as `get_design_info` with no additional switches.
-

JasperGold Apps Command Reference Manual

General Commands

-module (Continued)

- **counter:** identify counters in the vector of flops detected in the specified module scope.
 - In addition to listing the names of flops, this command can print the following information when you use the `-verbosity high` switch:
 - Number of bits – flop width.
 - Size logic – number of gates inside the flop's COI.
 - Max depth – maximum number of flops from the original flop to a primary input.
 - Special values – values that are critical to the counter.
 - Direct flops – list of flops directly connected to the original flop. This list includes flop types (FSM, COUNTER, or OTHER).
 - COI flops – list of flops in the COI of the original flop.
 - This Tcl command returns a list of space-separated counter names.
- **fsm:** Identify FSMs in the vector of flops detected in the specified module.
 - In addition to listing the names of flops, this command can print the following information when you use the `-verbosity high` switch:
 - Number of bits – flop width.
 - Size logic – number of gates inside the flop's COI.
 - Max depth – maximum number of flops from the original flop to a primary input.
 - Special values – values that are critical to the FSM.
 - Direct flops – list of flops directly connected to the original flop. This list includes flop types (FSM, COUNTER, or OTHER).
 - COI flops – list of flops in the COI of the original flop.
 - This Tcl command returns a list of space-separated FSM names.

JasperGold Apps Command Reference Manual

General Commands

-module (Continued)

- **array:** Identify multidimensional arrays (including FIFOs) in the vector of flops detected in the specified module.
 - Use `-no_aggregate` to get a bit-blasted list.
 - In addition to listing the names of flops, this command can print the following information when you use the `-verbosity high` switch:
 - Number of bits – flop width.
 - Size logic – number of gates inside the flop's COI.
 - Max depth – maximum number of flops from the original flop to a primary input.
 - Direct flops – list of flops directly connected to the original flop. This list includes flop types (FSM, COUNTER, or OTHER).
 - COI flops – list of flops in the COI of the original flop.
 - This Tcl command returns a list of space-separated array names.
- **register:** List all flops and latches in an instance of the specified module, displaying their names and sizes (in bits).
This Tcl command returns a list of space-separated flop and latch names.
- **flop:** List all flops in an instance of the specified module, displaying their names and sizes (in bits).
 - The list includes flop numbers shown as $x(y)(z$ property flop bits).
 - x is the total number of flops.
 - y is the sum of bits of the flops, not including property flop bits.
 - z is the sum of bits of property flops, which are generated as observers for SVA and PSL properties.
 - This Tcl command returns a list of space-separated flop names.
- **latch:** List all latches in an instance of the specified module, displaying their names and sizes (in bits).
This Tcl command returns a list of space-separated latch names.

JasperGold Apps Command Reference Manual

General Commands

-module (Continued)

- **gate**: List the number of gates and gate bits for the top instance.
This command returns the number of gates.
- **multiplier**
divider
modulus
List all multipliers, dividers, or modulus in the specified module.
This command lists two kinds of the specified operator:
 1. Black-boxed – each multiplier, divider, or modulus is described by three signals: {out_internal left_internal right_internal}
The right and left internal signals are the operands, and out_internal is the resulting signal.
 2. Cell-defined – each multiplier, divider, or modulus that has not been black boxed is cell defined. By default, all multipliers, dividers, and modulus are black boxed; however, you can use elaborate -bbox_mul, -bbox_div, and -bbox_mod to avoid black boxing them.Each multiplier, divider, and modulus is described by three signals:
{out_internal left_internal right_internal}
 - The right and left internal signals are the operands, and out_internal is the resulting signal.
 - This functionality is useful for manual abstraction.
 - This command returns a list of space-separated triples of signals.
- **module**: List all modules in the specified module, displaying their names. This command returns a list of space-separated module names.
- **module_name**: Return the specified module name.
- **module_name_no_param**: Return the specified module name without the non-default parameter values.
- **instance**: List all instances in the specified module, displaying their names.
This command returns a list of space-separated instance names. The list does not include black-boxed instances.

JasperGold Apps Command Reference Manual

General Commands

-module (Continued)

- **architecture**: List all architectures in the top-instance scope, displaying their names.
This command returns a list of space-separated architecture names.
- **signal**: List all signals in an instance of the specified module, displaying their names and sizes in bits.
 - Signals are the set of all registers, primary inputs, primary outputs, and wires (signals = registers + PIs + POs + wires).
 - This command returns a list of space-separated signal names.
- **wire**: List all wires in an instance of the specified module, displaying their names and sizes in bits.
 - Wires are all the signals not classified as registers, primary inputs, or primary outputs (wires = signals - registers - PIs - POs).
 - This command returns a list of space-separated signal names.
- **property**
assert
assume
cover
List the current task's embedded properties of the specified type (all embedded properties, embedded assertions, embedded assumptions, or embedded cover directives) in the specified module.
This command lists the number and names of properties and returns a list of space-separated property names.

Listing Design Info for the “-signal” Scope

```
-signal signal_name
[-list
  ([basic] [counter] [fsm] [array [-no_aggregate]]
   [register] [flop] [latch] [gate] [input]
   [undriven] [undriven_internal]
   [connected_assumptions]
   [special_values] [-module]
   [instance] [signal] [wire]
   [-parent]
  )+
]
[-fanout]
[-verbosity (high | low | silent)]
[-typed_list]
[-silent]
[-file file_name [-force]]
[-filter filter_expression
[-regexp] [-case_sensitive]]
[-gui]
```

Get design information for one or more specified signals.



- This scope supports multiple signals except in the case of `special_values`, which supports a single signal.
- Below you will find definitions of the supported `-list` enums for the `-signal` scope.
- To refine your list, for example to use it in scripting, refer to [“Additional Optional Switches for Customizing Lists”](#) on page 606.

Supported “-list” Enums for the “-signal” Scope

- **basic:** List the default set of information for the -signal scope.
 - The Tcl command returns the number of flops in the -signal scope.
 - The command `get_design_info -list basic` has the same effect as `get_design_info` with no additional switches.
- **counter:** identify counters in the vector of flops detected in the -signal scope.
 - In addition to listing the names of flops, this command can print the following information when you use the `-verbosity high` switch:
 - Number of bits – flop width.
 - Size logic – number of gates inside the flop's COI.
 - Max depth – maximum number of flops from the original flop to a primary input.
 - Special values – values that are critical to the counter.
 - Direct flops – list of flops directly connected to the original flop. This list includes flop types (FSM, COUNTER, or OTHER).
 - COI flops – list of flops in the COI of the original flop.
 - This command returns a list of space-separated counter names.
- **fsm:** identify FSMs in the vector of flops detected in the -signal scope.
 - In addition to listing the names of flops, this command can print the following information when you use the `-verbosity high` switch:
 - Number of bits – flop width.
 - Size logic – number of gates inside the flop's COI.
 - Max depth – maximum number of flops from the original flop to a primary input.
 - Special values – values that are critical to the FSM.

-signal (Continued)

- Direct flops – list of flops directly connected to the original flop. This list includes flop types (FSM, COUNTER, or OTHER).
- COI flops – list of flops in the COI of the original flop.
- This command returns a list of space-separated FSM names.
- array: identify multidimensional arrays (including FIFOs) in the vector of flops detected in the -signal scope.
 - Use `-no_aggregate` to get a bit-blasted list.
 - In addition to listing the names of flops, this command can print the following information when you use the `-verbosity high` switch:
 - Number of bits – flop width.
 - Size logic – number of gates inside the flop's COI.
 - Max depth – maximum number of flops from the original flop to a primary input.
 - Direct flops – list of flops directly connected to the original flop. This list includes flop types (FSM, COUNTER, or OTHER).
 - COI flops – list of flops in the COI of the original flop.
 - This command returns a list of space-separated array names.
- register: List all flops and latches in the -signal scope, displaying their names and sizes (in bits).
 - Use the `-fanout` switch with this argument to list all flops and latches in the fanout of the specified signal.
 - This command returns a list of space-separated flop and latch names.

JasperGold Apps Command Reference Manual

General Commands

-signal (Continued)

- **flop:** List all flops in the -signal scope, displaying their names and sizes (in bits).
 - The list includes flop numbers shown as x(y)(z property flop bits).
 - x is the total number of flops.
 - y is the sum of bits of the flops, not including property flop bits.
 - z is the sum of bits of property flops, which are generated as observers for SVA and PSL properties.
 - Use the –fanout switch with this argument to list all flops in the fanout of the specified signal.
 - This command returns a list of space-separated flop names.
- **latch:** List all latches in the -signal scope, displaying their names and sizes (in bits).
 - Use the –fanout switch with this argument to list all latches in the fanout of the specified signal.
 - This command returns a list of space-separated latch names.
- **gate:** List the number of gates and gate bits for the top instance. This command returns the number of gates.
- **input:** List the number of inputs followed by their names and sizes.
 - Inputs may be PIs, signals with hard or soft stopats, or output signals of instance stopats.
 - This command returns a list of space-separated input names.
- **undriven:** List the names and sizes of all undriven signals in the scope of the specified signals.
This command returns a list of space-separated names of undriven signals.
- **undriven_internal:** List the names and sizes of all undriven signals that are not primary inputs and that are in the scope of the specified signals.
This command returns a list of space-separated names of undriven internal signals.

JasperGold Apps Command Reference Manual

General Commands

-signal (Continued)

- `connected_assumptions`: identify and list relevant assumptions that may affect the analysis region of the specified signals.
 - This command returns a list of space-separated connected assumption names.
 - The connected assumptions are in the transitive fanin.

Note: This command does not include connected assumptions in statistics. To include them in computations, use the following command before running this command:

```
set_design_exploration_include_connected_assumptions
```
 - `special_values`: List all counter and FSM special values detected.
 - **Limitation:** This argument applies to a single signal.
 - Each special value is listed as an integer between parentheses and a textual description, if one exists, for example, (0) SOFA.
 - This command returns a list of space-separated special values.
 - `module`: List all modules in the `-signal` scope, displaying their names. This command returns a list of space-separated module names.
 - `instance`: List all instances in the `-signal` scope, displaying their names.

This command returns a list of space-separated instance names. The list does not include black-boxed instances.
 - `signal`: List the specified signal.

This command returns a list of space-separated signal names.
 - `wire`: List all wires in the `-signal` scope, displaying their names and sizes in bits.
 - Wires are all the signals not classified as registers, primary inputs, or primary outputs (wires = signals - registers - PIs - POs).
 - This command returns a list of space-separated signal names.
 - `parent`: List the parent instance of the specified signals.
-

Listing Design Info for the “-property” Scope

```
-property property_name+ [-regexp] [-region | -coi]
  [-list
    ([basic] [counter] [fsm] [array [-no_aggregate]]
     [register] [flop] [latch] [gate] [input]
     [undriven] [undriven_internal]
     [connected_assumptions]
     [module] [instance] [signal] [wire] [property]
     [assert] [assume] [cover] [stopat]
    )+
  ]
  [-verbosity (high | low | silent)]
  [-typed_list] [-silent]
  [-file file_name [-force]]
  [-filter filter_expression [-regexp] [-case_sensitive]]
  [-gui]
```

Get design information for one or more specified properties.



- This analysis considers hard stopats, instance stopats, and soft stopats in the corresponding task.
- This command supports wildcards and regular expressions in property names.
When using a regular expression, use the `-regexp` switch after the pattern:
`get_design_info -property {pattern} -regexp`
- Use the optional `-region` or `-coi` switch as follows:
 - Use `-region` to limit the analysis to the analysis region specified in the corresponding task. This is the default setting.
 - Use `-coi` to use the full cone of influence in the analysis.

- Information available for more than one property is: number and list of flops, latches, registers and inputs (PIs and stopats).
 - Below you will find definitions of the supported -list enums for the -property scope.
 - To refine your list, for example to use it in scripting, refer to [“Additional Optional Switches for Customizing Lists”](#) on page 606.
-

Supported “-list” Enums for the “-property” Scope

- basic: List the default set of information for the -property scope.
 - The Tcl command returns the number of flops in the -property scope.
 - The command `get_design_info -list basic` has the same effect as `get_design_info` with no additional switches.
 - counter: identify counters in the vector of flops detected in the -property scope.
 - In addition to listing the names of flops, this command can print the following information when you use the `-verbosity high` switch:
 - Number of bits – flop width.
 - Size logic – number of gates inside the flop's COI.
 - Max depth – maximum number of flops from the original flop to a primary input.
 - Special values – values that are critical to the counter.
 - Direct flops – list of flops directly connected to the original flop. This list includes flop types (FSM, COUNTER, or OTHER).
 - COI flops – list of flops in the COI of the original flop.
 - This command returns a list of space-separated counter names.
-

JasperGold Apps Command Reference Manual

General Commands

-property (Continued)

- `fsm`: identify FSMs in the vector of flops detected in the `-property` scope.
 - In addition to listing the names of flops, this command can print the following information when you use the `-verbosity high` switch:
 - Number of bits – flop width.
 - Size logic – number of gates inside the flop's COI.
 - Max depth – maximum number of flops from the original flop to a primary input.
 - Special values – values that are critical to the FSM
 - Direct flops – list of flops directly connected to the original flop. This list includes flop types (FSM, COUNTER, or OTHER).
 - COI flops – list of flops in the COI of the original flop.
 - This command returns a list of space-separated FSM names.
 - `array`: identify multidimensional arrays (including FIFOs) in the vector of flops detected in the `-property` scope.
 - Use `-no_aggregate` to get a bit-blasted list.
 - In addition to listing the names of flops, this command can print the following information when you use the `-verbosity high` switch:
 - Number of bits – flop width.
 - Size logic – number of gates inside the flop's COI.
 - Max depth – maximum number of flops from the original flop to a primary input.
 - Direct flops – list of flops directly connected to the original flop. This list includes flop types (FSM, COUNTER, or OTHER).
 - COI flops – list of flops in the COI of the original flop.
 - This command returns a list of space-separated array names..
-

JasperGold Apps Command Reference Manual

General Commands

-property (Continued)

- **register:** List all flops and latches in the -property scope, displaying their names and sizes (in bits).
This command returns a list of space-separated flop and latch names
- **flop:** List all flops in the -property scope, displaying their names and sizes (in bits).
 - The list includes flop numbers shown as x(y)(z property flop bits).
 - x is the total number of flops.
 - y is the sum of bits of the flops, not including property flop bits.
 - z is the sum of bits of property flops, which are generated as observers for SVA and PSL properties.
 - This command returns a list of space-separated flop names.
- **latch:** List all latches in the -property scope, displaying their names and sizes (in bits).
This command returns a list of space-separated latch names.
- **gate:** List the number of gates and gate bits for the top instance.
This command returns the number of gates.
- **input:** List the number of inputs followed by their names and sizes.
 - Inputs may be PIs, signals with hard or soft stopats, or output signals of instance stopats.
 - This command returns a list of space-separated input names.
- **undriven:** List the names and sizes of all undriven signals in the scope of the specified properties.
This command returns a list of space-separated names of undriven signals.
- **undriven_internal:** List the names and sizes of all undriven signals that are not primary inputs and that are in the scope of the specified properties.
This command returns a list of space-separated names of undriven internal signals.
- **connected_assumptions:** identify and list relevant assumptions that may affect the analysis region of the specified property.
This command returns a list of space-separated connected assumption names.

JasperGold Apps Command Reference Manual

General Commands

-property (Continued)

Note: This command does not include connected assumptions in statistics. To include them in computations, use the following command before running this command:

`set_design_exploration_include_connected_assumptions`

- **module:** List all modules in the -property scope, displaying their names.
This command returns a list of space-separated module names.
 - **instance:** List all instances in the -property scope, displaying their names.
This command returns a list of space-separated instance names. The list does not include black-boxed instances.
 - **signal:** List all signals in the -property scope, displaying their names and sizes in bits.
 - Signals are the set of all registers, primary inputs, primary outputs, and wires (signals = registers + PIs + POs + wires).
 - This command returns a list of space-separated signal names.
 - **wire:** List all wires in the -property scope, displaying their names and sizes in bits.
 - Wires are all the signals not classified as registers, primary inputs, or primary outputs (wires = signals - registers - PIs - POs).
 - This command returns a list of space-separated signal names.
 - **property**
assert
assume
cover
List one or more of the current task's embedded or Tcl properties of the specified type (all property types, assertions, assumptions, or cover directives).
 - This command supports wildcards. If you do not use wildcards, it returns the specified property.
 - It lists the number and names of properties and returns a list of space-separated property names.
-

JasperGold Apps Command Reference Manual

General Commands

-property (Continued)

- **stopat:** List all the stopats in the -property scope, displaying their names and sizes in bits.
This command returns a list of space-separated signal names.

Listing Design Info for the “-task” Scope

```
-task task_name+ [-region | -coi] [-compute_all]  
  [-list  
    ([basic] [counter] [fsm] [array [-no_aggregate]]  
     [register] [flop] [latch] [gate] [input]  
     [undriven] [undriven_internal]  
     [connected_assumptions]  
     [module] [instance] [signal] [wire]  
     [property] [assert] [assume] [cover]  
    )+  
  ]  
  [-verbosity (high | low | silent)]  
  [-typed_list] [-silent]  
  [-file file_name [-force]]  
  [-filter filter_expression [-regexp] [-case_sensitive]]  
  [-gui]
```

Get design information for one or more specified tasks.



- This analysis considers hard stopats, instance stopats, and soft stopats.
- Use the optional **-region** or **-coi** switch as follows:
 - Use **-region** to limit the analysis to the analysis region specified in the corresponding task. This is the default setting.
 - Use **-coi** to use the full cone of influence in the analysis.
- Use the optional **-compute_all** switch to request the computation of the list of registers for the specified task and for all properties within this task. The computation of registers for properties in a task can be done more efficiently if they are all done at the same time, and therefore, you will save time.

JasperGold Apps Command Reference Manual

General Commands

-task (Continued)

- This switch supports wildcards (*) and (?) in task names:
 - `get_design_info -task t*`
 - `get_design_info -task task1 {task2} t*`
 - `get_design_info -task task?`
- Information available for more than one task follows: number and list of flops, latches, registers and inputs (PIs and stopats).
- Below you will find definitions of the supported -list enums for the -task scope.
- To refine your list, for example to use it in scripting, refer to [“Additional Optional Switches for Customizing Lists”](#) on page 606.

Note: If you run the `reset` command prior to this command, the `reset` signal is listed as a task input signal. This is the behavior regardless of whether the task has properties.

Supported “-list” Enums for the “-task” Scope

- `basic`: List the default set of information for the -task scope.
 - The Tcl command returns the number of flops in the -task scope.
 - The command `get_design_info -list basic` has the same effect as `get_design_info` with no additional switches.
- `counter`: identify counters in the vector of flops detected in the -task scope.
 - In addition to listing the names of flops, this command can print the following information when you use the `-verbosity high` switch:
 - Number of bits – flop width.
 - Size logic – number of gates inside the flop's COI.
 - Max depth – maximum number of flops from the original flop to a primary input.
 - Special values – values that are critical to the counter.

JasperGold Apps Command Reference Manual

General Commands

-task (Continued)

- Direct flops – list of flops directly connected to the original flop. This list includes flop types (FSM, COUNTER, or OTHER).
 - COI flops – list of flops in the COI of the original flop.
 - This command returns a list of space-separated counter names.
-

JasperGold Apps Command Reference Manual

General Commands

-task (Continued)

- `fsm`: identify FSMs in the vector of flops detected in the -task scope.
 - In addition to listing the names of flops, this command can print the following information when you use the `-verbosity high` switch:
 - Number of bits – flop width.
 - Size logic – number of gates inside the flop's COI.
 - Max depth – maximum number of flops from the original flop to a primary input.
 - Special values – values that are critical to the FSM.
 - Direct flops – list of flops directly connected to the original flop. This list includes flop types (FSM, COUNTER, or OTHER).
 - COI flops – list of flops in the COI of the original flop.
 - This command returns a list of space-separated FSM names.
- `array`: identify multidimensional arrays (including FIFOs) in the vector of flops detected in the -task scope.
 - Use `-no_aggregate` to get a bit-blasted list.
 - In addition to listing the names of flops, this command can print the following information when you use the `-verbosity high` switch:
 - Number of bits – flop width.
 - Size logic – number of gates inside the flop's COI.
 - Max depth – maximum number of flops from the original flop to a primary input.
 - Direct flops – list of flops directly connected to the original flop. This list includes flop types (FSM, COUNTER, or OTHER).
 - COI flops – list of flops in the COI of the original flop.
 - This command returns a list of space-separated array names.
- `register`: List all flops and latches in the -task scope, displaying their names and sizes (in bits).
This command returns a list of space-separated flop and latch names.

JasperGold Apps Command Reference Manual

General Commands

-task (Continued)

- **flop:** List all flops in the -task scope, displaying their names and sizes (in bits).
 - The list includes flop numbers shown as x(y)(z property flop bits).
 - x is the total number of flops.
 - y is the sum of bits of the flops, not including property flop bits.
 - z is the sum of bits of property flops, which are generated as observers for SVA and PSL properties.
 - This command returns a list of space-separated flop names.
- **latch:** List all latches in the -task scope, displaying their names and sizes (in bits).
This command returns a list of space-separated latch names.
- **gate:** List the number of gates and gate bits for the top instance.
This command returns the number of gates.
- **input:** List the number of inputs followed by their names and sizes.
 - Inputs may be PIs, signals with hard or soft stopats, or output signals of instance stopats.
 - This command returns a list of space-separated input names.
- Note:** If you run the `reset` command prior to this command, the `reset` signal is listed as a task input signal. This is the behavior regardless of whether the task has properties.
- **undriven:** List the names and sizes of all undriven signals in the scope of the specified tasks.
This command returns a list of space-separated names of undriven signals.
- **undriven_internal:** List the names and sizes of all undriven signals that are not primary inputs and that are in the scope of the specified tasks.
This command returns a list of space-separated names of undriven internal signals.
- **connected_assumptions:** identify and list relevant assumptions that may affect the analysis region of the specified task.
This command returns a list of space-separated connected assumption names.

JasperGold Apps Command Reference Manual

General Commands

-task (Continued)

Note: This command does not include connected assumptions in statistics. To include them in computations, use the following command before running this command:

```
set_design_exploration_include_connected_assumptions
```

- **module:** List all modules in the -task scope, displaying their names. This command returns a list of space-separated module names.
- **instance:** List all instances in the -task scope, displaying their names. This command returns a list of space-separated instance names. The list does not include black-boxed instances.
- **signal:** List all signals in the -task scope, displaying their names and sizes in bits.
 - Signals are the set of all registers, primary inputs, primary outputs, and wires (signals = registers + PIs + POs + wires).
 - This command returns a list of space-separated signal names.
- **wire:** List all wires in the -task scope, displaying their names and sizes in bits.
 - Wires are all the signals not classified as registers, primary inputs, or primary outputs (wires = signals - registers - PIs - POs).
 - This command returns a list of space-separated signal names.
- **property**
assert
assume
cover
List embedded and Tcl properties of the specified type (all properties, assertions, assumptions, or cover directives) in the specified task. This command lists the number and names of properties and returns a list of space-separated property names.

Additional Optional Switches for Customizing Lists

Use the following switches with one or more of the enums listed above.

- **-verbosity (high | low | silent):** Use the specified level of verbosity to provide information.
 - **high** – Print all available information.
Example: when used with `-list flop`, this command prints:
 - The total number of flops and property flops (flops generated as observers for SVA and PSL properties).
 - A list of all the signal names with their sizes shown in parentheses.
 - The return value is the sum of bits of all flops.
 - **low** – Print minimal information.
Example: when used with `-list flop`, this command prints the space-separated flop names.
The `low` verbosity level is the default.
 - **silent** – Turn off output to the screen and return design info with a Tcl return value.
The `silent` option is helpful in Tcl scripts.
- **-typed_list:** Change the return format for the `-list` switch by adding a type element before each of the requested lists.

The list format is as follows:

```
{ input { <input_signal>+ } output { <output_signal>+ } }
```

Refer to [“Examples”](#) on page 610.

- **-silent:** Same effect as `-verbosity silent`.
-

JasperGold Apps Command Reference Manual

General Commands

Optional Switches (Continued)

- **-file *file_name*:** Print information to the specified *file_name*.
 Use **-force** to overwrite an existing file.
- **-filter *filter_expression* [-regexp] [-case_sensitive]:** Filters the returned list using case-insensitive wildcards (default).
 - Use **-regexp** to change the filter expression to a case-insensitive regular expression.
 - Use **-case_sensitive** to filter output using case-sensitive wildcards or regular expressions.
- **-gui:** Display the requested information in the Design Information GUI.
- **-include_hier_path:** Display signal names with the instance path.
 Use this switch with the following arguments:
 - **output**
 - **input**
 - **inout**

JasperGold Apps Command Reference Manual

General Commands

FSM and Counter Detection

```
-signal signal_name
  -is_counter | -is_fsm
  [-verbosity high]
```

Get design information for one or more specified signals.



- This analysis only considers hard stopats and instance stopats specified with `-env`.
- This command supports wildcards (`*` and `?`) in the signal name.
- `-is_counter` returns `true` if the signal is a counter or `false`, if not.
- `-is_fsm` returns `true` if the signal is a finite state machine (FSM) or `false`, if not.
- If you use `-verbosity high` and the signal is an FSM or a counter, the command prints special values, which are values that are critical to FSMs or counters.

Cached-Info Clearing

```
-clear
```

Remove all cached information for all `get_design_info` commands.

Use this command to free memory.

Symmetry Check

```
-symmetry -from from_bus -to to_bus  
[-full | -quick]
```

Check symmetry between buses specified by the `-from` and `-to` switches.

This check returns `true` (symmetric) or `false` (not symmetric). If the return is `true`, the check guarantees the result for the number of clock cycles checked. The number of cycles checked is proportional to the time spent to check symmetry.

- Use `-full` to request a complete check of symmetry. By default, the tool does a check that ensures symmetry for the first 20 cycles.
- Use `-quick` to request a two-cycle symmetry check.

With this check, you can use a subset of the bits in the buses, which makes properties more suited for formal verification.

Depth Calculation

```
-signal name+ | -property name+  
-list (flop | latch | register)  
-depth N  
[-boundary] [-silent]
```

List all the state elements that are within *N* layers of sequential logic from one or more target signals or properties.

Note: This switch supports wildcards (*) and (?) in signal and property names.

- `-list flop`: perform the depth calculation in terms of flop layers.
- `-list latch`: perform the depth calculation in terms of latch layers.
- `-list register`: perform the depth calculation in terms of both flops and latches.

If the target property or signal is a state element itself, it is considered to be at depth 0.

- Use `-boundary` to list only the state elements at the outer layer corresponding to depth *N*.
- Use `-silent` to get just the signal names list, not a printout of the output of each signal with bit-width annotations.

Default

Refer to the `get_design_info` syntax definitions for default values.

Return

`get_design_info` returns an integer, signal list, or true/false.

Examples

```
% get_design_info -instance ing2 -list stopat
% get_design_info -module pci_RST_int
% get_design_info -module pci_bridge32 -list output
% get_design_info -list counter

// Check if from v[0:3] to b[0][0:3] there is no operation in between that may
// change to value of b[0][0:3] on different entries of v[0:3].
% get_design_info -symmetry -from {v[0:3]} -to {b[0][0:3]}

// List FSMs and outputs for instance "arb", and return a typed list.
% get_design_info -instance arb -list fsm output -typed_list fsm
{arb_core.arb_state arb_core.pri_enc.curr_chan tag_mgr.tag_poolA
tag_mgr.tag_poolB tag_mgr.tag_poolC} output {dat_out dat_val full0 full1 full2
full3 wb0_ack_o wb0_data_o wb0_err_o wb1_ack_o wb1_data_o wb1_err_o wb2_ack_o
wb2_data_o wb2_err_o wb3_ack_o wb3_data_o wb3_err_o}

// State Element Calculations
// =====

// List the flops and latches that are within distance 1
// (in terms of layers of state elements) from the target signal.
% get_design_info -signal sigName1 -list register -depth 1

// List the flops on the second layer of flops from either one of the two
// target signals.
% get_design_info -signal sigName1 sigName2 -list flop -depth 2 -boundary

// List the flops that are on the 3rd layer of flops away from the target
// property.
% get_design_info -property propName1 -list flop -depth 3 -boundary

// List the state elements (flops or latches) that are 2 layers of state
```

JasperGold Apps Command Reference Manual

General Commands

```
// elements from the target properties.  
% get_design_info -property propName1 propName2 -list register -depth 2 -boundary  
  
// Compare metrics with and without connected assumptions  
% get_design_info -property <embedded>::my_prop  
% set_design_exploration_include_connected_assumptions off  
% get_design_info -property <embedded>::my_prop
```

See Also

[get_flop_info](#)
[get_latch_info](#)
[get_signal_info](#)
[set_design_exploration_include_clock_logic](#)
[set_design_exploration_include_connected_assumptions](#)

get_design_space_coverage

Description

Design Space Coverage is a feature that gives insight into the coverage information of properties, tasks, signals, and traces. Activate the Design Space Coverage feature through the GUI interface (for example, right-click on a property in a Property Table and choose *Design Space Coverage*) or the `get_design_space_coverage` command.

You can specify the scope for which this information is provided. For example, you might want to get the list of signals covered by all properties in a task or the list of uncovered signals.

Syntax

Scope Switches

```
get_design_space_coverage
  -signal <signal_name>+
  |-property <property_name>+ [-regexp]
  |-task <task_name>+
```

Trace Switches

```
get_design_space_coverage -trace
  [-add [-name <trace_name>] [-force]]
  [-save_data <file_name> [-name <trace_name>]]
  [-restore_data <file_name>]
  [-list_trace_names]
  [-clear [-name <trace_name>+]]
  [-clear_all]
  [-cumulative [-name <trace_name>+]]
  [-gui [-window <window_name>]]
```

Report Information

```
get_design_space_coverage (-list
  ( covered | potential_covered | not_covered | exercised
  | not_exercised
  | covered_line | potential_covered_line | not_covered_line
  | exercised_line | not_exercised_line
  | covered_percentage | potential_covered_percentage
  | not_covered_percentage
  | exercised_percentage | not_exercised_percentage
  | covered_line_percentage
```

JasperGold Apps Command Reference Manual

General Commands

```
| potential_covered_line_percentage  
| not_covered_line_percentage | exercised_line_percentage  
| not_exercised_line_percentage)+)*
```

Verbosity Switches

```
get_design_space_coverage [-verbosity ( high | low | silent | detail  
| detail_silent)]
```

Utility Switches

```
get_design_space_coverage [-silent]  
get_design_space_coverage [-gui]
```

Argument	Definition
Scope Switches	
Use the switches in this section to specify the following:	
<ul style="list-style-type: none">■ The scope of the information the <code>get_design_space_coverage</code> command returns: signal, property, task, or trace.■ The GUI display with the <code>get_design_space_coverage -gui</code> command.	
Signal Switch	
<code>-signal signal_name+</code>	<p>Get design space coverage information for one or more specified signals (<i>signal_name</i>) separated by spaces.</p> <p> <i>Important</i></p> <ul style="list-style-type: none">■ This analysis only considers hard stopats and instance stopats specified with <code>-env</code>.■ This switch supports wildcards (*) and (?).

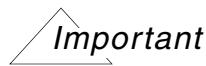
JasperGold Apps Command Reference Manual

General Commands

Property Switch

`-property property_name+ [-regexp]`

Get design space coverage information for one or more specified properties (*property_name*) separated by spaces.



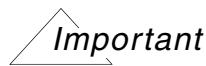
- This analysis considers hard stopats, instance stopats, and soft stopats in the corresponding task.
- This switch supports wildcards and regular expressions in property names. When using a regular expression, use the `-regexp` switch after the pattern:

```
get_design_space_coverage -property {pattern} -regexp
```

Task Switch

`-task task_name+`

Get design space coverage information for one or more specified tasks (*task_name*) separated by spaces.



- This analysis considers hard stopats, instance stopats, and soft stopats in the specified task.
- This switch supports wildcards and multiple task names.

Trace Switches

`-trace`

Get design space coverage information for the current opened trace target.

This is the only context that allows for the following:

- Generating `exercised` and `notExercised` coverage and percentages since this information requires a trace.
- Saving data to a trace database to generate requested information for more than one trace.

JasperGold Apps Command Reference Manual

General Commands

```
-add -name trace_name -force
```

Save the current opened trace data to the cumulative database.

- Use `-name` to tag the newly added trace with the specified *trace_name*. Tagged traces are used in related waveforms computation.
- Use `-force` to overwrite existing named traces.

```
-save_data file_name -name trace_name
```

Save all traces in the database to a persistent cache in the disk file *file_name*.

Use the `-name` switch to save just the data related to the specified trace to the specified disk file.

Note: The specified trace must be in the database to be cached.

```
-restore_data file_name
```

Load all traces saved in the specified disk file to the database.

```
-list_trace_names
```

List all traces saved in the trace database.

```
-clear -name trace_name...
```

Clear traces from the trace database.

- Use `-name` to delete the specified trace from the database. To specify multiple trace names, separate each name with a space (example: `-name t1 t2 ... tn`).

Note: This switch supports wildcards.

- Use `-clear` with no additional arguments to delete all unnamed traces from the database.

```
-clear_all
```

Clear the trace database completely, deleting all named and unnamed traces.

JasperGold Apps Command Reference Manual

General Commands

```
-cumulative -name trace_name...
```

Use the information in the trace database to generate requested information, such as a list of exercised signals or highlights for exercised code.

- Use the `-name` switch to generate data from one or more specified traces.
- To specify multiple trace names, separate each name with a space (example: `-name t1 t2 ... tn`).

Note: This switch supports wildcards.

Since the cumulative analysis uses traces saved in the trace database, it should be used with `-list exercised`, `-list exercised_percentage`,

`-list not_exercised`, or `-list not_exercised_percentage`. It has no effect when used with other `-list` switches.

When used with `-list exercised`, this command prints all signals already exercised by at least one of the saved traces.

When used with `-list not_exercised`, this command prints all signals not exercised by any trace (taking the complete design as reference).

```
-gui -window window_name
```

Highlight trace data code of the current opened trace.

- When used with the `-cumulative` switch, this command highlights trace data code of traces saved in trace database.
 - Use `-window` to highlight trace data code in the specified Visualize window.
-

Report Information

Use the switches in this section in combination with scope switches.

```
-list
    covered | potential_covered | not_covered
    exercised | not_exercised
    covered_line | potential_covered_line | not_covered_line
    exercised_line | not_exercised_line
    covered_percentage | potential_covered_percentage
    not_covered_percentage
    exercised_percentage | not_exercised_percentage
    covered_line_percentage | potential_covered_line_percentage
    not_covered_line_percentage | exercised_line_percentage
    not_exercised_line_percentage
```

Get the specified type of list.

- **covered:** List covered signals.

This comprises all signals in the analysis region of the given scope. The Tcl command returns the list of signals.

- **potential_covered:** List potentially covered signals.

This comprises all signals in the cone of influence of the given scope, but not in its analysis region. The Tcl command returns the list of signals.

- **not_covered:** List non-covered signals.

This comprises all signals in the design but not in the cone of influence of the given scope. The Tcl command returns the list of signals.

-list options (Continued)

- `exercised`: List signals exercised by the most recent trace.

This argument requires the `-trace` scope switch to retrieve information from the most recent trace. The Tcl command returns the list of signals.
 - `not_exercised`: List signals in the analysis region not exercised by the most recent trace.

This argument requires the `-trace` scope switch to retrieve information from the most recent trace. The Tcl command returns the list of signals.
 - `covered_line`: List covered lines.

This comprises all lines in the analysis region of the given scope. The Tcl command returns the list of triples:
`instance line file_name`.

When specified with verbosity level `detail`, this command displays the contents of each line. Verbosity level `detail_silent` is silent mode for verbosity level `detail`.
 - `potential_covered_line`: List potentially covered lines.

This comprises all lines in the cone of influence of the given scope, but not in its analysis region. The Tcl command returns the list of triples:
`instance line file_name`.

When specified with verbosity level `detail`, it displays the contents of each line. Verbosity level `detail_silent` is silent mode for verbosity level `detail`.
-

-list options (Continued)

- `not_covered_line`: List non-covered lines.

This comprises all lines in the design but not in the cone of influence of the given scope. The Tcl command returns the list of triples: `instance line file_name`.

When specified with verbosity level `detail`, it displays the contents of each line. Verbosity level `detail_silent` is silent mode for verbosity level `detail`.

- `exercised_line`: List lines exercised by the most recent trace.

For this argument, the scope switch `-trace` is required to show that it refers to the most recent trace. The Tcl command returns the list of triples: `instance line file_name`.

When specified with verbosity level `detail`, it displays the contents of each line. Verbosity level `detail_silent` is silent mode for verbosity level `detail`.

- `not_exercised_line`: List lines in the analysis region not exercised by the most recent trace.

For this argument, the scope switch `-trace` is required to show that it refers to the most recent trace. The Tcl command returns the list of triples: `instance line file_name`.

When specified with verbosity level `detail`, it displays the contents of each line. Verbosity level `detail_silent` is silent mode for verbosity level `detail`.

JasperGold Apps Command Reference Manual

General Commands

-list options (Continued)

- `covered_percentage`: Display the percentage of covered signals based on the complete design.

The Tcl command returns the percentage.
- `potential_covered_percentage`: Display the percentage of potentially covered signals based on the complete design.

The Tcl command returns the percentage.
- `not_covered_percentage`: Display the percentage of non-covered signals based on the complete design.

The Tcl command returns the percentage.
- `exercised_percentage`: Display the percentage of exercised signals based on the trace target region.

For this argument, the scope switch `-trace` is required to show that `exercised_percentage` refers to the most recent trace. The Tcl command returns the percentage.
- `not_exercised_percentage`: Display the percentage of non-exercised signals based on the trace target region.

For this argument, the scope switch `-trace` is required to show that `not_exercised_percentage` refers to the most recent trace. The Tcl command returns the percentage.
- `covered_line_percentage`: Display the percentage of covered lines.

The Tcl command returns the percentage.
- `potential_covered_line_percentage`: Display the percentage of potentially covered lines.

The Tcl command returns the percentage.

JasperGold Apps Command Reference Manual

General Commands

-list options (Continued)

- `not_covered_line_percentage`: Display the percentage of non-covered lines.
The Tcl command returns the percentage.
- `exercised_line_percentage`: Display the percentage of lines exercised by the most recent trace.
The Tcl command returns the percentage.
- `not_exercised_line_percentage`: Display the percentage of lines in the analysis region that have not been exercised by the most recent trace.
The Tcl command returns the percentage.

Verbosity Switches

`-verbosity high | low | silent | detail | detail_silent`

Use the specified level of verbosity when providing information.

- `high` – Print all available information.
This option is the default.
- `low` – Print minimal information.
- `silent` – Turn off output to the screen and return design information with a Tcl return value. The `silent` option is helpful in Tcl scripts.
- `detail` – Print the RTL line number and code.
- `detail_silent` – Print detailed information in `silent` mode.

Utility Switches

`-silent`

Turn off output to the screen and return design information with a Tcl return value. The `silent` option is helpful in Tcl scripts.

`-gui`

Open the Source Browser in the Design Space Coverage mode to display areas in the design that are covered or uncovered relative to the specified target (tasks, properties, and signals).

Default

None

Return

`get_design_space_coverage` returns a signal list.

Examples

```
// List covered signals in the s1 region.  
% get_design_space_coverage -signal s1 -list covered  
  
// Open Design Space Coverage GUI for s1 target signal.  
% get_design_space_coverage -signal s1 -gui  
  
// Add current opened trace to cumulative trace database.  
% get_design_space_coverage -trace -add  
  
// List signals exercised by all traces saved in cumulative trace database.  
% get_design_space_coverage -trace -list exercised -cumulative  
  
// Open Design Space Coverage GUI highlighting all traces saved in cumulative  
// trace database.  
% get_design_space_coverage -trace -cumulative -gui  
  
// Clear stored traces named t1 and t2.  
% get_design_space_coverage -trace -clear t1 t2  
  
// Clear all traces that begin with letter t (t1 and t2, for example).  
% get_design_space_coverage -trace -clear t*
```

See Also

[get_design_info](#)

get_fanin

Description

Use the `get_fanin` command to list the signals in the fanin (after skipping buffers) of the specified signal or property. You will typically use the *Add Fanin* GUI features instead of typing this command on the Tcl command line.

Syntax

```
get_fanin <signal_name> | <property_name>
    [-add_clock_logic] [-ignore_weak_driver]
    [-env]
    [-avoid_pin ( flop_async_reset | flop_clock
                  | latch_enable | mux_enable)+]

get_fanin -go_into_cell_define [-add_clock_logic]
    [-ignore_weak_driver] [-env]
    <signal_name> | <property_name>

get_fanin -transitive (<signal_name> | <property_name>)
    [-ignore_weak_driver]
    [-env |-task <task_name>]
    [-avoid_pin ( flop_async_reset | flop_clock
                  | latch_enable | mux_enable)+]
    [-filter_out ( boundary | non_boundary | user_reset
                  | data | control)+]
    [-barrier (<signal_name> | <signal_list> | <reg_expr>
               | <reg_expr_list>)
    [-silent]

get_fanin -show_expr <signal_name>
    [-no_constant_propagation]
    [-ignore_weak_driver]
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<pre>signal_name property_name [-add_clock_logic] [-ignore_weak_driver] [-env] [-avoid_pin (flop_async_reset flop_clock latch_enable mux_enable)+]</pre>	<p>Find the signals in the fanin of the specified signal or property.</p> <ul style="list-style-type: none">■ Use <code>-add_clock_logic</code> to traverse the clock pin logic.■ Use <code>-ignore_weak_driver</code> to specify that you want the tool to consider driver strengths and report only the possible active drivers of multiple-driven nets. <p>Note: The tool does not take into account constant values (propagated or not). Analysis is based entirely on the driver strengths declared in the RTL.</p> <ul style="list-style-type: none">■ Use <code>-env</code> to run analysis based on the entire design region and ignore the task context, such as assumptions and stopats.■ Use <code>-avoid_pin</code> with one or more of the following options to prevent the <code>get_fanin</code> traversal from going through the specified pins:<ul style="list-style-type: none">□ <code>flop_async_reset</code>□ <code>flop_clock</code>□ <code>latch_enable</code>□ <code>mux_enable</code>

JasperGold Apps Command Reference Manual

General Commands

```
-go_into_cell_define  
[-add_clock_logic] [-ignore_weak_driver] [-env]  
signal_name | property_name
```

Include signals in cell defines in the results.

- Use `-add_clock_logic` to traverse the clock pin logic.
 - Use `-ignore_weak_driver` to specify that you want the tool to consider driver strengths and report only the possible active drivers of multiple-driven nets.
- Note:** The tool does not take into account constant values (propagated or not). Analysis is based entirely on the driver strengths declared in the RTL.
- Use `-env` to run analysis based on the entire design region and ignore the task context, such as assumptions and stopats.



- If the target `signal` is in a cell define, the tool automatically includes fanin signals in that cell define.
- This switch can provide a considerable performance gain. Weigh this gain against ignoring task-based assumptions and stopats.

JasperGold Apps Command Reference Manual

General Commands

```
-transitive (signal_name | property_name)
[-ignore_weak_driver]
[-env |-task task_name]
[-avoid_pin ( flop_async_reset | flop_clock
              | latch_enable | mux_enable)+]
[-filter_out ( boundary | non_boundary | user_reset
                | data | control)+]
[-barrier ( signal_name | signal_list | reg_expr
              | reg_expr_list )
[-silent]
```

Run the command recursively until all results are primary inputs or stopats that are in the fanin cone of the specified signal or property. By default, the transitive `get_fanin` traverses clock pins and returns the full path.

- Use `-ignore_weak_driver` to specify that you want the tool to consider driver strengths and report only the possible active drivers of multiple-driven nets.
 - **Note:** The tool does not take into account constant values (propagated or not). Analysis is based entirely on the driver strengths declared in the RTL.
 - Use `-env` to run analysis based on the full design region ignoring the task context, such as assumptions and stopats.
 - Use `-task` to run analysis based on the task design region.
-

JasperGold Apps Command Reference Manual

General Commands

-transitive (Continued)

- Use `-avoid_pin` with one or more of the following options to prevent the `get_fanin` traversal from going through the specified pins:
 - `flop_async_reset`
 - `flop_clock`
 - `latch_enable`
 - `mux_enable`
 - Use `-filter_out` with one or more of the following options to avoid showing them in the results:
 - `boundary` – Filter out boundary signals from results.
 - `non_boundary` – Filter out non-boundary signals from results.
 - `user_reset` – Filter out user-specified reset signals.
 - `data` – Filter out data signals from results.
 - `control` – Filter out control (latch enable, mux enable and asynchronous reset condition) signals from results.
 - Use `-barrier` to allow the transitive `get_fanin` to stop at the specified signals as if they were boundaries.

Note: If you also use the switch `-filter_out`, the barrier signals will be seen as boundary signals.
 - Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.
-

JasperGold Apps Command Reference Manual

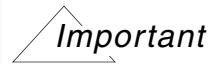
General Commands

```
-show_expr signal_name
[-no_constant_propagation]
[-ignore_weak_driver]
```

Return the combinational expression of the specified signal in terms of its immediate fanin signals rather than a simple list of the signals.

- Use `-no_constant_propagation` to avoid the expression simplification due to constant propagation.
- Use `-ignore_weak_driver` to specify that you want the tool to consider driver strengths and report only the possible active drivers of multiple-driven nets.

Note: The tool does not take into account constant values (propagated or not). Analysis is based entirely on the driver strengths declared in the RTL.



- This switch does not support `celldefine.
- This switch does not support the following, and in these cases, the command is not able to return the correct expression:
 - Hierarchical references
 - Escaped signals
 - Expression slicing, for example,

```
{1'b0, 1'b0, 1'b0, 1'b0, 1'b0, 1'b0, 1'b0,
o2p_queue_blk.pre_queue_valid} [o2p_queue_blk pci
_ping]
```
- `get_fanin -show_expr` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

Default

None

Return

Returns a string containing a list of the names of the fanin signals.

Examples

```
% get_fanin A  
% get_fanin -transitive arb_core.tag_req  
% get_fanin -show_expr signalB  
  
# The following command tells the tool not to traverse the clock pins and to  
# return only the boundary signals:  
% get_fanin -transitive signalA -avoid_pin flop_clock -filter_out non_boundary  
-silent
```

See Also

[get_fanout](#)
[get_signal_info](#)

get_fanout

Description

Use the `get_fanout` command to list the signals in the immediate fanout of the specified signal. You will typically use the *Add Fanout* GUI features instead of typing this command on the Tcl command line.

Syntax

```
get_fanout <signal_name>
[-env]

get_fanout -transitive <signal_name>
[-env |-task <task_name>]
[-avoid_pin ( flop_async_reset | flop_clock
              | latch_enable | mux_enable)+]
[-filter_out (boundary | non_boundary | user_reset)+]
[-barrier ( <signal_name> | <signal_list>
              | <reg_expr> | <reg_expr_list>) ]
[-silent]
```

Argument	Definition
<code>signal_name</code> [-env]	<p>Find the signals in the immediate fanin of the specified signal.</p> <p>Use <code>-env</code> to run analysis based on the entire design region and ignore the task context, such as assumptions and stopats.</p>

JasperGold Apps Command Reference Manual

General Commands

```
-transitive signal_name
[-env |-task task_name]
[-avoid_pin ( flop_async_reset | flop_clock
               | latch_enable | mux_enable)+]
[-filter_out (boundary | non_boundary | user_reset)+]
[-barrier ( signal_name | signal_list
               | reg_expr | reg_expr_list)]
[-silent]
```

Run recursively until all results are boundary signals that are in the fanout cone of the specified signal.

- Use `-env` to run analysis based on the full design region ignoring the task context, such as assumptions and stopats.
 - Use `-task` to run analysis based on the task design region.
 - Use `-avoid_pin` with one or more of the following options to prevent the `get_fanout` traversal from going through the specified pin types:
 - `flop_async_reset`
 - `flop_clock`
 - `latch_enable`
 - `mux_enable`
 - Use `-filter_out` with one or more of the following options to avoid showing them in the results:
 - `boundary` – Filter out boundary signals.
 - `non_boundary` – Filter out non-boundary signals.
 - `user_reset` – Filter out user-specified reset signals.
 - Use `-barrier` to allow the transitive `get_fanout` to stop at the specified signals as if they were boundary signals.
- Note:** If you also use the switch `-filter_out`, the barrier signals will be seen as boundary signals.

Default

None

Return

Returns a string containing the names of the fanout signals.

Examples

```
% get_fanout A  
  
# The following command tells the tool not to traverse the clock pins and to  
# return only the boundary signals:  
% get_fanout -transitive A -avoid_pin flop_clock -filter_out non_boundary -silent
```

See Also

[get_fanin](#)
[get_signal_info](#)

get_filename

Description

Use the `get_filename` command to display the file name for a specified module, instance, or entity.

Syntax

```
get_filename -module <module_name>
              |-inst <instance_name>
              |-entity <entity_name>
```

Argument	Definition
<code>-module <i>module_name</i></code>	Display the associated file name for the specified module.
<code>-inst <i>instance_name</i></code>	Display the associated file name for the specified instance.
<code>-entity <i>entity_name</i></code>	Display the associated file name for the specified VHDL entity.

Default

None

Return

Returns a string containing the name of the file associated with either the specified module or instance name.

Examples

```
% get_filename -module pci_core
```

See Also

No related commands

get_flop_info

Description

Use `get_flop_info` to get information about a flop, including reset information. When there is no asynchronous reset connected to the specified flop, the tool attempts to find a synchronous reset signal.

When the flop has both an asynchronous reset and a synchronous reset (according to JasperGold Apps heuristics) the command returns the asynchronous one unless you specify one of the `-sync_*` switches.

By default, `get_flop_info` returns one list for each flop gate driving the input signal. Each of these lists contains signals related to clock, data, power, reset, reset value pins, and reset type for the specified flop. In addition, the lists provide location (file name and line numbers) and indicate whether the reset signal found is asynchronous (`async`) or synchronous (`sync`). If there is no reset signal, the reset type is `none`. If the flop has an abstracted reset value (`abstract -reset_value`), the reset value pin includes the notation `abstracted`.

Syntax

```
get_flop_info [-clock]
               [-edge]
               [-data]
               [-power]
               [-reset_pin | -reset_type]
               [-sync_reset_pin]
               [-async_reset_pin]
               [-reset_value_pin]
               [-sync_reset_value_pin]
               [-async_reset_value_pin]
               [-file_name]
               [-line]
               [-bit_blast | -group_all]
               [-silent]
               [-typed_list | -compact]
               [-show_expr]
               [-defined_clock]
               <flop_name>
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
-clock	Include the clock signal that drives the specified flop in the result.
-edge	Include the sensitivity edge of the specified flop in the result.
-data	Include a list of signals related to the data pin in the result.
-power	Include the name of the primary supply net. Note: If the specified flop is not power aware, the report prints with an empty Power field and the return includes an empty Tcl list.
-reset_pin	Include a list of signals related to the reset pin in the result.
-reset_type	Include the type of the reset condition (synchronous or asynchronous).
-sync_reset_pin	Include the sync reset condition, if there is one, even if there is also an asynchronous reset.
-async_reset_pin	Include the asynchronous reset condition.
-reset_value_pin	Include a list of signals related to the reset value pin in the result, , or if the reset value is abstracted, report abstracted.
-sync_reset_value_pin	Include the synchronous reset value pin, if there is one, even if there is also an asynchronous reset value pin.
-async_reset_value_pin	Include the asynchronous reset value pin, if there is one.

JasperGold Apps Command Reference Manual

General Commands

-file_name

Include the name of the file where the flop is instantiated.

By default, `get_flop_info` includes the file name and one or more line numbers or ranges of numbers. Use this switch to include only the file location.

-line

Include the line number where the flop is instantiated.

By default, `get_flop_info` includes the file name and one or more line numbers or ranges of numbers. Use this switch to include only the line location.

-group_all

Group all requested information in a single list.

-bit_blast

Return the individual bits information of a flop.

-silent

Avoid printing user-friendly information on the screen.

-typed_list

Include the type information before each list in the result.

-compact

Exclude the flop name from the result.

JasperGold Apps Command Reference Manual

General Commands

-show_expr

Return the combinational expression of the specified signal in terms of its immediate fanin signals rather than a simple list of the signals.



- This switch does not support `celldefine.
- This switch does not support the following, and in these cases, the command is not able to return the correct expression:
 - Hierarchical references
 - Escaped signals
 - Expression slicing, for example, {1'b0, 1'b0, 1'b0, 1'b0, 1'b0, 1'b0, 1'b0, o2p_queue_blk.pre_queue_valid} [o2p_queue_blk pci_ping]
 - `get_flop_info -show_expr` is an initial release to gather feedback from early adopters and finalize implementation for formal release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

-defined_clock

Return the list of user-defined clocks connected to the specified flop.

Default

None

Return

Returns information about signals related to the input flop.

Examples

```
// Get fanin from data, clock, reset, and reset value pins.  
[<embedded>] % get_flop_info arb_core.arb_state  
Flop information:  
Flop: arb_core.arb_state
```

JasperGold Apps Command Reference Manual

General Commands

```
Clock: arb_core.clk
Data: arb_core.one_chan_rdy arb_core.one_more_chan_rdy
      arb_core.curr_chan_eof arb_core.cred_avail arb_core.curr_chan_vld
      arb_core.rst arb_core.arb_state

Reset pin: rst
Reset value pin:
Reset type: sync
Location: ./design/arb_core.v: 108-177,1-4

{arb_core.arb_state arb_core.clk {arb_core.one_chan_rdy
arb_core.one_more_chan_rdy arb_core.curr_chan_eof arb_core.cred_avail
arb_core.curr_chan_vld arb_core.rst arb_core.arb_state} rst {} sync
{./design/arb_core.v 108-177,1-4}}

// Return only clock and data pins fanin.
% get_flop_info -clock -data flop1

// Return clock information. (More than one list if multiFlop is related to
// more than one flop gate.)
% get_flop_info -clock multiFlop

// Return clock information. (Group all clocks in a single list.)
% get_flop_info -clock -group_all multiFlop

// Get info on a flop with an abstracted reset value
%get_flop_info q4
Flop information:
Flop: q4
Clock: clk
Data: q3
Reset pin: rst
Reset value pin:
Reset type: async
Location: floplatchline.v: 13-27,1-4

{q4 clk q3 rst {} async {floplatchline.v 13-27,1-4}}

abstract -reset_value q4
get_flop_info -show_expr q4
Flop information:
Flop: q4
Clock: clk
```

JasperGold Apps Command Reference Manual

General Commands

```
Data: q3
Reset pin: {rst, rst, rst}
Reset value pin: 3'b000 abstracted
Reset type: async
Location: floplatchline.v: 13-27,1-4
```

```
{q4 clk q3 {{{rst, rst, rst}}}} {3'b000 abstracted} async {floplatchline.v
13-27,1-4}
```

```
// Get info on a "mixed" signal. In the following example, the signal "q" is
// actually two different flops: q[0] and q[1], and one is sensitive to the
// rising edge of the clock, while the other is sensitive to the falling edge.
```

```
% get_flop_info q -edge -group_all
Flop information:
Flop: q
  Clock: clk1 clk2 mixed
  Data: d
  Reset pin: rst
  Reset value pin:
  Reset type: async
  Location: edge.v: 3-8,1-4 10-15,1-4
```

```
{q {mixed clk1 clk2} d rst {} async {edge.v {3-8,1-4 10-15,1-4}}}
```

See Also

[get_design_info](#)
[get_latch_info](#)
[get_signal_info](#)

get_latch_info

Description

Use the `get_latch_info` command to get information about a latch, excluding reset information.

By default, `get_latch_info` returns one list for each latch gate driving the input signal. Each of these lists contains signals related to enable and data pins of the latch. In addition, the lists provide location (file name and line numbers). Use the optional switches to refine the return information as shown in the examples.

Syntax

```
get_latch_info [-enable]
                [-data]
                [-power]
                [-file_name]
                [-line]
                [-bit_blast |-group_all]
                [-silent]
                [-typed_list]
                [-show_expr]
                <latch_name>
```

Argument	Definition
<code>-enable</code>	Include the enable signal that drives the specified latch in the result.
<code>-data</code>	Include a list of signals related to the data pin in the result.
<code>-power</code>	Include the name of the primary supply net. Note: If the specified latch is not power aware, the report prints with an empty <code>Power</code> field and the return includes an empty Tcl list.

JasperGold Apps Command Reference Manual

General Commands

-file_name

Include the name of the file where the latch is instantiated.

By default, `get_latch_info` includes the file name and one or more line numbers or ranges of numbers.

-line

Include the line number where the latch is instantiated.

By default, `get_latch_info` includes the file name and one or more line numbers or ranges of numbers.

-group_all

Group all requested information in a single list.

-bit_blast

Return the individual bits information of a latch.

-silent

Avoid printing user-friendly information on the screen.

-typed_list

Include the type information before each list in the result.

-show_expr

Return the combinational expression of the signals in terms of their immediate fanin signals rather than a simple list of the signals.

Default

By default, `get_latch_info` returns one list for each latch gate driving the input signal.

Return

`get_latch_info` returns lists of signal name lists.

Examples

```
// Get fanin from data and enable pins.  
[<embedded>] % get_latch_info latch1  
Latch information:  
Latch: latch1  
Data: rst1 in1
```

JasperGold Apps Command Reference Manual

General Commands

```
Enable: en1 rst1
Location: test_latch_info.v: 11-15,5-8

{latch1 {rst1 in1} {en1 rst1} {test_latch_info.v 11-15,5-8}}

// Return only enable and data pins' fanin.
%get_latch_info -enable -data latch1

// Return enable information. (More than one list if "multiLatch" is related to
// more than one latch gate.)
%get_latch_info -enable multiLatch

// Return enable information. (Group all enables in a single list.)
%get_latch_info -enable -group_all multiLatch
```

See Also

[get_design_info](#)
[get_flop_info](#)
[get_signal_info](#)

get_message

Description

Use the `get_message` command to get a list or number of occurrences of certain messages in a session.

Syntax

```
get_message -list (error | warning | info)  
get_message -number (error | warning | info)  
get_message -message message_id
```

Argument	Definition
<code>-list (error warning info)</code>	List all messages from the specified severity level that printed to the console or log during the current session. Note: The capacity of your message's storage limits the number of elements returned by this switch. See " set_message_history_limit " on page 1116 for more information.
<code>-number (error warning info)</code>	Returns the number of issued messages of the specified type.
<code>-message <i>message_id</i></code>	Get the number of occurrences of the specified message.

Default

None

Return

`get_message` returns a list of user messages or the number of occurrences.

JasperGold Apps Command Reference Manual

General Commands

Examples

```
% get_message -list warning
```

See Also

[set_message](#)

get_needed_assumptions

Description

The `get_needed_assumptions` command systematically disables and enables assumptions to eventually find a minimal set of assumptions needed to achieve the target prove status on the target property.

This command creates a task called `NDD_ASSUMES` and copies the current task's assumptions, stopats, initial value abstractions, and target property to the new task. Then, it iteratively uses bisection of the set of assumptions to find the needed assumptions.

Use this command to:

- Find conflicting assumptions when JasperGold reports an overconstraint.
- Find conflicting assumptions when a cover is unexpectedly unreachable.
- Find the minimal set of assumptions to prove a property, potentially speeding up the proof.

When trying to identify assumptions using this command, it is often useful to take note of which engine produced the result and ensure this engine is enabled by `set_engine_mode` before starting `get_needed_assumptions`. You may also want to note the status of the property. In particular, when the task is overconstrained and you have specified multiple engines with `set_engine_mode`, you may wish to either restrict the engine mode to the specific engine that found the result or give multiple statuses with the `-status` switch. For example, `get_needed_assumptions -status {proven error}`.

Engines are not guaranteed to detect an overconstrained situation. If you experience issues when using only the `error` status you may want to try `-status {error proven}` or `-status {error unreachable}`. In the overconstrained case, also consider using `check_assumptions` instead.

Note: By default, the tool displays the following in the console:

- Notification when the check begins and ends
- Incremental messages to report needed assumptions as they are discovered
- A final report of results

Use `-silent` to turn off these messages.

Syntax

```
get_needed_assumptions -property <property_name>
    [-status <target_status_tcl_list>]
    [-assumes <assumption_tcl_list>]
    [-include_env_assumes]
    [-include_liveness_assumes]
    [-include_clock]
    [-include_reset]
    [-required_assumes <assumption_tcl_list>]
    [-attempts <number_of_prove_attempts>]
    [-attempts_time_limit <time_limit>]
    [-stop_status <target_status_tcl_list>]
    [-verbose]
    [-silent]
```

Argument	Definition
-property <i>property_name</i>	Find needed assumptions for the specified property.
-status <i>target_status_tcl_list</i>	Try to achieve the specified status. Note: <ul style="list-style-type: none">■ If the property has been processed and has a proof result (proven, unreachable, or error) the tool uses that status. If it does not have a proof result, the default is proven for assertions and unreachable for covers.■ This switch accepts a Tcl list.
-assumes <i>assumption_tcl_list</i>	Use the specified assumptions only. <i>The default is to use all enabled assumptions.</i>
-include_env_assumes	Include assumptions in the global environment. Note: Using this will invalidate all proof results in all tasks.

JasperGold Apps Command Reference Manual

General Commands

`-include_liveness_assumes`

Consider liveness assumptions for safety properties. By default, the tool does not consider liveness assumptions when the specified property is a safety property.

`-include_clock`

Compute the needed set of clock constraints. For `clock -rate` constraints, these will not be minimized but applied iff the corresponding clock signal is part of the needed set of clock constraints.

Note: This switch can invalidate all proof results in all tasks.

`-include_reset`

Compute the needed set of reset expressions. `reset -vcd`, `reset -shm`, and `reset -fsdb` are not supported.

Note: This switch can invalidate all proof results in all tasks.

`-required_assumes assumption_tcl_list`

Include the specified assumptions in the set of needed assumptions. These will be assumed to be needed and included in the result.

`-attempts number_of_prove_attempts`

Stop after the specified number of proof attempts. *The default is 0 (unlimited).*

`-attempts_time_limit time_limit`

Time limit per iteration, measured in wall clock time.

The time limit is a positive integer that takes an optional trailing unit:

- `s` = seconds
- `m` = minutes
- `h` = hours
- `d` = days

If you do not specify a unit, the unit is seconds. *The default is 120s.*

`-stop_status target_status_tcl_list`

Stop immediately whenever one of the specified statuses is reached. The resulting list of assumptions are the assumptions used to reach that status, and they are not necessarily a minimal set.

JasperGold Apps Command Reference Manual

General Commands

-verbose

Output some progress information. *This is the default.*

-silent

Do not print messages to the log and do not report progress and results in the console.

Default

The defaults are as follows:

- -status: If the property has been processed and has a proof result (proven, unreachable, or error) the tool uses that status. If it does not have a proof result, the default is proven for assertions and unreachable for covers.
- -assumes: all enabled assumptions
- -attempts: 0
- -attempts_time_limit: 120s

Return

```
status prove_attempts assumptions [reset] [clock]
```

This command returns a tuple consisting of three to five elements. The three default elements are status, the number of prove attempts/iterations, and a set of needed assumptions. In addition, if you use -include_reset, the return includes a set of reset constraints. Likewise, if you use -include_clock, the return includes a set of clock constraints.

Interpret the status as follows:

Status	Definition
completed	A minimal set of assumptions has been found.
attempts_limit	The command was stopped because the attempts limit was reached. The set of assumptions returned is an overapproximation of the minimal needed set.
attempts_time_limit	The command was stopped because the attempts time limit was reached. In this case, none of the other elements is returned.

JasperGold Apps Command Reference Manual

General Commands

stopped_by_user	The command was stopped by the user. The set of assumptions returned is an overapproximation of the minimal needed set.
stopped_by_status	The command was stopped because the status specified by -stop_status was reached.
aborted	The command stopped due to an error. In this case, none of the other elements is returned.

Examples

```
% get_needed_assumptions -property cover_property -status unreachable
```

See Also

[assume](#)
[check_assumptions](#)
[prove](#)

get_proj_dir

Description

Use the `get_proj_dir` command to extract the location of the project directory.

Note: This command is helpful when you launch the tool with the `-proj` switch to change the default location.

Syntax

```
get_proj_dir
```

Argument	Definition
no arguments	

Default

None

Return

Returns the location of the project directory.

Examples

```
% get_proj_dir
```

See Also

No related commands

get_property_info

Description

Use the `get_property_info` command to get information about a property. You can run the command in either of two ways, with or without the `-list` switch.

Syntax

```
get_property_info <property_name>  
get_property_info -list  
  { [name] [task] [type] [clock] [command]  
    [expression] [precondition] [related_covers]  
    [liveness] [converted] [helper]  
    [app] [origin] [error] [disabled] [status]  
    [related_cover_status] [pseudo_constant]  
    [assume_bound] [time] [engine]  
    [min_length] [max_length] [trace_length]  
    [proof_effort] [engineL_trail] [eq_prop] }  
<property_name>
```

Argument	Definition
<code>property_name</code>	<p>Return information in the form of a list with an even number of elements. The first element in each pair is the name of an attribute and the second element in each pair is the value of the previous attribute.</p> <p>If an attribute has no value for the specified property, the tool omits it from the returned list.</p> <p>You can use the returned list to create a Tcl array with the help of the <code>array set</code> command. See “Examples” on page 658.</p> <p>Note: Do not rely on the order of the pairs in the list. New pairs might be inserted in the list at any position in a new version of JasperGold. We strongly advise that you access the returned value from this command through an array created with <code>array set</code> (see above).</p>

JasperGold Apps Command Reference Manual

General Commands

```
-list
{ [name] [task] [type] [clock] [command]
  [expression] [precondition] [related_covers]
  [liveness] [converted] [helper]
  [app] [origin] [error] [disabled] [status]
  [related_cover_status] [pseudo_constant]
  [assume_bound] [time] [engine]
  [min_length] [max_length] [trace_length]
  [proof_effort] [engineL_trail] [eq_prop] }
property_name
```

Return a Tcl list of specified values. Each element in the returned list is the value of the corresponding attribute in the attribute list. If you specify a property that has no value for the specified attribute, the command fails with an error.

Attributes Available for All Properties

- name: The name of the property
- task: The name of the task in which the property resides
- type: The type of the property
 - assert
 - assert(sec)
 - assert(spv)
 - assert(xprop)
 - assume
 - assume(soft)
 - assume(soft not to)
 - assume(soft to)
 - assume(xprop)
 - cover
 - cover(path)
- clock: The names of the clocks that trigger the property

Note: If the tool's fastest clock triggers the property, this clock is omitted from the Tcl return.
- command: The name of the Tcl command used on the property, for example, to disable it (possible values: assert, assume, cover).
- expression: The expression of the property

Note: For embedded properties, the tool lists the internal signal that represents the property inside the tool. You can use this signal to refer to the property, for example, in commands such as `get_fanin <signal>`.

JasperGold Apps Command Reference Manual

General Commands

Attributes (Continued)

- precondition: 1 if the property is a precondition, otherwise 0
- related_covers: A list of covers related to the property (see "[elaborate -create related covers](#)")
- liveness: 1 if the property is a liveness property, otherwise 0
- converted: 1 if the property was converted from another property, otherwise 0
- helper: 1 if the property is a helper assertion
- app: The short name of the app the property belongs to
- origin: The origin of the property

The value is a Tcl list and the number of elements depends on the first element. Possible values of the first element follow:

- file: Property from a source file

The returned list contains three additional elements:

- The name of the file
- The start line of the property
- The end line of the property

- pa: Property from a Proof Accelerator

The list contains no more elements.

- tcl: Property defined using a Tcl command

The list contains no more elements.

- error: 1 if the property failed to compile, otherwise 0

- disabled: 0 if the property is enabled, 1 if it is disabled

JasperGold Apps Command Reference Manual

General Commands

Attributes (Continued)

- status: The current status of the property
 - possible values for assertions
 - proven
 - marked_proven
 - cex
 - ar_cex
 - undetermined
 - unprocessed
 - processing
 - error
 - possible values for covers
 - unreachable
 - covered
 - ar_covered
 - undetermined
 - unprocessed
 - processing
 - error
 - possible values for assumptions:
 - temporary
 - approved

JasperGold Apps Command Reference Manual

General Commands

Attributes (Continued)

- `related_cover_status`: The aggregated proof status of the related covers
 - The returned value is one of the following:
 - ❑ `red` – At least one related cover has the status `unreachable`.
 - ❑ `yellow` – The condition for `red` is not fulfilled, and at least one related cover has the status `undetermined`, `unprocessed`, `processing`, or `error`.
 - ❑ `ar_green` – The condition for `yellow` is not fulfilled, and at least one related cover has the status `ar_covered`.
 - ❑ `green` – The condition for `ar_green` is not fulfilled, and there is at least one related cover. That is, all related covers have the status `covered`.
 - ❑ `white` – There are no related covers.
 - `pseudo_constant`: 1 if the property is a pseudo constant, otherwise 0
 - `assume_bound`: The length of the bound of the assumption, 0 for unbounded assumptions and other kinds of properties

Additional Attributes Available for Assertions and Covers

- `time`: The time spent on the property (in seconds)
- `engine`: The engine that delivered the best currently available information about the property. The engine can be one of the selected engines, see “[set_engine_mode](#)” on page 1009, or a special (virtual) engine:
 - ❑ `User` – See “[assert -mark_proven](#)” for additional information.
 - ❑ `PRE` – When the property was solved during preprocessing, see “[set_proof_simplification](#)” on page 1122 for additional information.
- `min_length`: The minimal length of a possible trace for the property in the form of a positive integer or the string `infinite`. This information tells you that shorter traces do not exist, which is also known as the `bounded proof`. The tool displays this number in the *Property Table* in the *Bound* column.

Additional Attributes Available for Assertions and Covers (Cont'd)

- `max_length`: The maximal length of a possible trace for the property in the form of a positive integer or the string `infinite`. The tool displays this number in the *Property Table* in the *Bound* column.
- `trace_length`: The length of the trace associated with the property, 0 if there is no trace.
- `proof_effort`: The Proof Effort is a rough indication of how much effort the engine has spent establishing a proven or unreachable result. Different engines use different scales. The value is also contingent on the full proof settings, which other properties were proven together with this, and on indeterministic ProofGrid timing variations. A very low 1 value indicates that this engine did not have to spend a lot of effort proving this property. A very low value does not necessarily mean that the property is low value or trivial.

Note: For liveness properties, the value is the maximum trace attempts performed on the property as can be seen in the ProofGrid Manager. You can think of this representation as “the tool has considered traces this long (stem + loop) for this property.” And you can use this information as a weak kind of engine-specific progress indicator.

- `engineL_trail`: The sequence of properties used to find a trace by engine L; {} if another engine found the trace or no trace has been found. See “[set_engineL_generate_trails](#)” on page 1062.
- `eq_prop`: If the property is known to semantically have the same status as other properties, the tool lists the next property in a circular chain of equivalent properties. Otherwise, it displays {}.

Note: If two properties are listed as equivalent here, it means that a bound, or a minimal trace, or a proof of the absence of traces (proven or unreachable) for one also applies to the other. The tool does not guarantee that all mentioned properties at all times will show the same actual status.

Default

No default values

Return

Returns a list of key/value pairs with information about the property.

The `-list` switch returns a Tcl list of specified values where each element in the returned list is the value of the corresponding attribute in the attribute list.

Examples

```
% get_property_info prop
name prop task <embedded> type assert clock clk command assert expression {a && b}
precondition 0 liveness 0 converted 0 helper 0 app fpv origin tcl error 0 disabled
0 status cex pseudo_constant 0 assume_bound 0 time 12.3456 engine L min_length 9
max_length 10 trace_length 10

% array set info [get_property_info <embedded>::prop]
% foreach attr [lsort [array names info]] { puts "$attr: ${info($attr)}" }
app: fpv
assume_bound: 0
command: assert
converted: 0
disabled: 0
engine: L
error: 0
expression: a && b
helper: 0
liveness: 0
max_length: 10
min_length: 9
name: propName
origin: tcl
precondition: 0
pseudo_constant: 0
status: cex
task: <embedded>
time: 12.3456
trace_length: 10
type: assert

% get_property_info -list {status engine time min_length trace_length} prop
cex L 12.3456 9 10
% [get_property_info -list command $propName] -disable $propName
```

JasperGold Apps Command Reference Manual

General Commands

```
# Listing related_covers with multiple attributes
# -----
% get_property_info -list {status engine time min_length trace_length
related_covers} prop
cex L 12.3456 9 10 {prop:precondition1 prop:precondition2 prop:precondition3}
...

# Listing related_covers as a single attribute
# -----
% get_property_info -list related_covers prop
{prop:precondition1 prop:precondition2 prop:precondition3}
```

See Also

[assert](#)
[assume](#)
[cover](#)
[elaborate](#)
[get_status](#)

get_reset_info

Description

Use the `get_reset_info` command to report reset information. If you do not supply any switches, the tool lists flops and latches with defined values (0 or 1). Use the optional switches to customize a return that you can then process to build a report. Use the `save_*` switches to save a file that can be used to specify the reset state.

Syntax

```
get_reset_info
  [[-all | -x_value]
   [-bit_blast]
   [-list
    ([type] [cycle] [module] [instance] [configuration])+
    [-sort_by (type | cycle | module | instance)]
    [-silent]
    |-get_value (signal_name [cycle | cycle:cycle])
    |-get_length
    |-save_reset_vcd file_name [-include_internal]
    |-save_reset_fsdb file_name
    |-save_reset_shm directory_name
    |-save_values file_name [-all | -x_value] [-bit_blast] [-comments]
      [-include_internal] [-include_encrypted]
    |-constant_registers
```

Argument	Definition
<code>-all</code>	List all design flops and latches and their values. If a register does not have a reset value, the tool prints value X.
<code>-x_value</code>	List only the uninitialized flops and latches (X).
<code>-bit_blast</code>	List each bit of a flop or latch separately. If all bits are 0, the tool aggregates them. For example, 10'b000000000000 prints as 10'b0.

JasperGold Apps Command Reference Manual

General Commands

```
-list ([type] [cycle] [module] [instance] [configuration])+
[-sort_by (type | cycle | module | instance)]
[-silent]
```

List the specified additional information for each register.

- Use `type` to include the type of register, for example, D flop.
- Use `cycle` to include the reset cycle.
Note: If you use this argument, and you specified reset with `reset -init_state`, the tool issues a warning because there is no reset cycle for this case.
- Use `module` to include the module where the register is instantiated.
- Use `instance` to include the name of the instance where the register is instantiated.
- Use `configuration` to include the reset configuration that was specified with the `reset` command.
- Use `-sort_by` with `type`, `cycle`, `module`, or `instance` to specify the sorting scheme.

Note: When sorting by `cycle`, the list is from highest to lowest.

- Use `-silent` to return the list as Tcl return values without printing the list on the screen.

```
-get_value (signal_name [cycle | cycle:cycle])
```

Return the value for a specified signal from a simulated reset trace.

Specifying cycles is optional.

- If you do not specify a cycle or cycle range, the tool returns the signal values of all cycles.
- If you specify a single cycle (for example, 7), you are asking the tool to get the value for the specified signal in cycle seven.
- You can specify a range of cycles; for example, 2:3.
- Use \$ to mean the last cycle. For example, 1:\$ is equivalent to always (which is the default when you do not specify cycles), and 4:\$ is equivalent to from cycle four until the end of the trace.

JasperGold Apps Command Reference Manual

General Commands

`-get_length`

Get the number of cycles reset analysis required for convergence.

Use `get_reset_info -get_length` without additional switches.

`-save_reset_vcd file_name [-include_internal]`

Create a VCD file from a simulated reset trace.

- Use `file_name` to specify the VCD file name.
- Use `-include_internal` to include internal signals in the VCD file.

Note: Loading files with the switch `-include_internal` carries a risk of a mismatch in internal signals that can lead to false proof results in some cases. Internal signals in reset files might not be the same as those generated by a different JasperGold release version or a different version of the design or a different design setup and configuration. (Design setup and configuration differences include the order in which files are passed to `analyze` and `elaborate` and the order in which assumptions and assertions are created.)

`-save_reset_fsdb file_name`

Create an FSDB file from a simulated reset trace.

Use `file_name` to specify the FSDB file name.

`-save_reset_shm directory_name`

Create an SHM directory from a simulated reset trace.

Use `directory_name` to specify the SHM directory name.

Note: `-save_reset_shm` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

JasperGold Apps Command Reference Manual

General Commands

```
-save_values file_name [-all |-x_value] [-bit_blast] [-comments]
[-include_internal] [-include_encrypted]
```

Generate a file that can be used with `reset -init_state`.

If you do not supply any additional switches, the tool only saves flops and latches with defined values.

- Use `-save_values` to do the following:
 - Save the registers and their values to a file that you can read with the `reset -init_state` command
 - Save the results of a complex `reset` command for use in another session
- Use `-all` to save values of all design flops and latches. If a register does not have a reset value, it saves as value X.
- Use `-x_value` to save only the uninitialized flops and latches (X).
- Use `-bit_blast` to list each bit of a register separately. If all bits are 0, the tool aggregates them. For example, 10'b0000000000 prints as 10'b0.
- Use `-comments` to list comments for each register listed in the output file.
- Use `-include_internal` to save values of internal flops.

Note: Loading files with the switch `-include_internal` carries a risk of a mismatch in internal signals that can lead to false proof results in some cases. Internal signals in reset files might not be the same as those generated by a different JasperGold release version or a different version of the design or a different design setup and configuration. (Design setup and configuration differences include the order in which files are passed to `analyze` and `elaborate` and the order in which assumptions and assertions are created.)

- Use `-include_encrypted` to capture reset values of encrypted flops in an encrypted file.

```
-constant_register
```

Return a list of flops and latches that have constant value at the proof stage.

Return

Returns the list of flops and latches and their values as Tcl return values.

The format is:

- **Without module or instance:**

```
{sig1 value1} {sig2 value2}
```

- **With module or instance:**

```
{sig1 value1 module1 instance1} {sig2 value2 module2 instance2}
```

- **With instance but not module:**

```
{sig1 value1 instance1} {sig2 value2 instance2}
```

- **With type and cycle but not instance nor module:**

```
{register_type_1 sig1 value1 cycle1} {register_type_2 sig2 value2 cycle2}
```

Examples

```
# Example for capturing information needed for customized reports:  
% analyze -sv my_design.v  
% elaborate  
% clock clk  
% reset {~resetn}  
% get_reset_info -bit_blast -list type cycle  
  
% get_reset_info -silent  
{sig1 value1} {sig2 value2}  
  
% get_reset_info -list type cycle module instance -silent  
{sig1 value1 type1 cycle1 module1 instance1} {sig2 value2 type2 cycle2 module2  
instance2}
```

See Also

No related commands

get_root_name

Description

Use the `get_root_name` command to get the root of the instance tree.

Syntax

```
get_root_name
```

Argument	Definition
no arguments	

Return

Returns a string (identifier name) for the root of the instance tree of the design.

Examples

```
% get_root_name
```

See Also

[elaborate](#)

[get_top_module](#)

get_signal_info

Description

Use the `get_signal_info` command to extract information about a signal.

Note: If the signal belongs to an instance of a module, you must include the hierarchy in the signal name. If you do not include the hierarchy, the command applies to the top-level module.

Syntax

```
get_signal_info
  [ -type | -logic | -port_type
  | -indexes [-show_unpacked]
  | -bit_blast | -width
  | -equiv [-forward]
  | -get_module | -get_instance [-split] | -get_signal [-full]
  ] <signal_name>

get_signal_info -declaration <signal_name>
  (-file_name | -line) +
get_signal_info -driver <signal_name>
  (-file_name | -line) +
```

Argument	Definition
<code>-type</code>	<p>Return the signal type.</p> <p>The signal type will be one of the following:</p> <ul style="list-style-type: none">■ <code>output</code> – signal is a top module output■ <code>input</code> – signal is a top module input■ <code>inout</code> – signal is both a top module input and output■ <code>undriven</code> – signal has no driving logic■ <code>MPRAM</code> – signal is a RAM net■ <code>logic</code> – signal is none of the above <p>The tool uses <code>-type</code> by default if you do not specify any switch.</p>

JasperGold Apps Command Reference Manual

General Commands

-logic

Return the signal logic type.

The return will be one of the following:

- latch
- flop
- wire
- MPRAM

-port_type *signal_name*

Return the port type for the specified signal.

The return will be one of the following:

- input – signal is a module input
- output – signal is a module output
- inout – signal is both a module input and output
- undef – signal is not a module port

-indexes [-show_unpacked]

Return details about the signal: MSB, LSB, fields (for structs and unions), and number of read and write ports (for multiport RAMs).

The command returns one of the following:

- Bit (in cases where the specified signal is not indexed)
- 1D_Array followed by the MSB and LSB indexes
- Struct followed by a list of fields
- Union followed by a list of fields
- MPRAM followed by the number of read, write, and asynchronous write ports
- MD_Array followed by each dimension's MSB and LSB indexes

Note: Use -show_unpacked to include the string `unpacked` in the return for any unpacked range in an MD_Array result.

- Enum followed by all enum values.

JasperGold Apps Command Reference Manual

General Commands

-bit_blast

Return the individual bits of a signal, most significant bit first.

Use this switch to print the individual bits of a signal within its specified range. For example, consider the following signal

```
A: wire [2:0] A;
```

If you run `get_signal_info -bit_blast A`, the result is the following:

```
A[2] A[1] A[0]
```

You can also specify a range within a signal, for example, `get_signal_info -bit_blast A[1:0]` returns the following:

```
A[1] A[0]
```

-width

Return the width of a signal.

For example, consider the following signal A:

```
wire [2:0] A;
```

If you run `get_signal_info -width A`, the result is the following: 3

JasperGold Apps Command Reference Manual

General Commands

-equiv [-forward]

Return a list of signals that are equivalent to the specified signal. The *signal_name* can be a signal or a bit.

- Use -equiv *signal_name* with no additional switches to print an ordered list of equivalent drivers.
- Use -equiv *signal_name* -forward to print a set of equivalent loads.

Note: The set that -forward returns is not ordered.

For example, consider the following signal C:

```
wire A, B, C;  
assign A=B;  
assign A=C;
```

If you run get_signal_info -equiv A, the result is the following:

A B C

If you run get_signal_info -equiv C -forward, the result is the following:

A B

-get_module

Return the name of the module in which the signal is declared.

For example, consider the following module:

```
module top ();  
    wire A;  
endmodule
```

If you run get_signal_info -get_module A, the result is the following:

top

JasperGold Apps Command Reference Manual

General Commands

```
-get_instance [-split]
```

Return the name of the instance in which the signal is instantiated.

Use `-split` to print the instance path list

For example, consider the following code:

```
module top();  
    mod1 inst();  
endmodule  
  
module mod1();  
    wire A;  
endmodule;
```

If you run `get_signal_info -get_instance inst.A`, the result is the following:

```
inst1
```

If the signal is in deeper hierarchy, the result can look like the following:

```
inst1.inst2.inst3
```

If you run `get_signal_info -get_instance inst1.inst2.inst3.A -split`, the result is the following:

```
inst1 inst2 inst3
```

JasperGold Apps Command Reference Manual

General Commands

-get_signal [-full]

Return the name of the signal within the instance hierarchy.

Use -full to get the same signal name, but without any slice operator.

For example, consider the following code:

```
module top();  
    mod1 inst();  
endmodule  
  
module mod1();  
    wire A[3:0];  
endmodule;
```

If you run `get_signal_info -get_signal inst.A`, the result is the following:

A

If you run `get_signal_info -get_signal inst.A[1:0]`, the result is the following:

A

If you run `get_signal_info -get_signal inst.A[1:0] -full`, the result is the following:

inst.A

signal_name

Get signal information for the specified *signal_name*.

JasperGold Apps Command Reference Manual

General Commands

```
-declaration signal_name (-file_name |-line) +
```

List the source location of the declaration of the signal.

- Use `-file_name` to get the name of the file where the signal is declared.
- Use `-line` to get the line number where the signal is declared.

Output:

Location: <left line num-right line num, left column num-right column num>

Return value:

Location: <left line num-right line num, left column num-right column num>

```
-driver signal_name (-file_name |-line) +
```

List the source location of the drivers of the signal.

- Use `-file_name` to get the name of the file where the signal is declared.
- Use `-line` to get the line number where the signal is declared.

Output:

Location: <left line num-right line num, left column num-right column num>

Return value:

Location: <left line num-right line num, left column num-right column num>

Default

None

Return

Returns the signal type, logic, port direction, indexes, expanded name into its bits, or width.

Examples

```
% get_signal_info -type s1 undriven  
% get_signal_info -indexes a -show_unpacked MD_Array unpacked 3 0 6 0
```

JasperGold Apps Command Reference Manual

General Commands

See Also

[get_design_info](#)

[get_fanin](#)

[get_fanout](#)

[get_flop_info](#)

[get_latch_info](#)

get_signal_list

Description

Use the `get_signal_list` command to get a list of signals.

Syntax

```
get_signal_list <signal_list>
  [-union | -intersection | -subtraction] <signal_list>
  [-pattern_filter_in <pattern> -regexp]
  [-pattern_filter_out <pattern> -regexp]
  [-bit_blast]
  [-case_sensitive]
  [-force]
  [-aggregate]
```

Argument	Definition
<code>-union signal_list</code>	<p>Return the union of signals in the first and second list.</p> <p>For example, consider the following signals A and B:</p> <pre>wire A; wire B;</pre> <p>If you run <code>get_signal_list {A} -union {B}</code>, the result is the following:</p> <p>A B</p>

JasperGold Apps Command Reference Manual

General Commands

-intersection *signal_list*

Return the intersection of signals in the first and second list.

For example, consider the following signals A and B:

```
wire A;
```

```
wire B;
```

If you run `get_signal_list {A} -intersection {A B}`, the result is the following:

```
A
```

-subtraction *signal_list*

Return the subtraction of signals in the first and second list.

For example, consider the following signals A and B:

```
wire A;
```

```
wire B;
```

If you run `get_signal_list {A B} -subtraction {A}`, the result is the following:

```
B
```

-pattern_filter_in *pattern* -regexp

Return the result list being filtered by the pattern. The results will be a list of signals that match the specified pattern.

For example, consider the following signals A and B:

```
wire A;
```

```
wire B;
```

If you run `get_signal_list {A B A} -pattern_filter_in A* -regexp`, the result is the following:

```
A
```

JasperGold Apps Command Reference Manual

General Commands

```
-pattern_filter_out pattern -regexp
```

Return the result list being filtered by the pattern. The result will be a list of signals that do not match the specified pattern.

For example, consider the following signals A and B:

```
wire A;
```

```
wire B;
```

If you run `get_signal_list {A B B} -pattern_filter_out A* -regexp`, the result is the following:

```
B
```

```
-bit_blast signal_name
```

Return the individual bits of a signal with the most significant bit first.

Use this switch to print the individual bits of a signal within its specified range. For example, consider the following signals A and B:

```
wire [2:0] A;
```

```
wire [2:0] B;
```

If you run `get_signal_list {A B} -bit_blast`, the result is the following:

```
A[2] A[1] A[0] B[2] B[1] B[0]
```

```
-aggregate
```

Return the list of the signals aggregated if they are part of the same structure.

For example, consider the following signals S.a, S.b:

```
struct {
    bit a;
    bit b;
} S;
```

If you run `get_signal_list {S.a S.b} -aggregate`, the result is the following:

```
S
```

JasperGold Apps Command Reference Manual

General Commands

Default

None

Return

Returns a list of signals.

Examples

```
% get_signal_list {A B} -union {C D}
```

See Also

No related commands

get_status

Description

Use the `get_status` command to see the proof status of an assertion or cover property.

Syntax

```
get_status <property_name_tcl_list> [-regexp]
```

Argument	Definition
<code>property_name_tcl_list</code> [-regexp]	<p>Display the aggregated proof status of a set of properties.</p> <p>The possible return values are listed below by property type:</p> <ul style="list-style-type: none">■ assertions<ul style="list-style-type: none">□ proven□ marked_proven□ cex□ ar_cex□ undetermined□ unprocessed□ processing□ error

JasperGold Apps Command Reference Manual

General Commands

Possible return values (Continued)

- covers
 - unreachable
 - covered
 - ar_covered
 - undetermined
 - unprocessed
 - processing
 - error
- mix of assertions and covers
 - determined
 - ar_determined
 - undetermined
 - unprocessed
 - processing
 - error
- no properties match the specified wildcard or regular expression
 - no_properties

Note:

- To get the status of properties not in the current task, prepend the property names with their task names.
- It is possible to combine properties from several tasks in one list.
- This command supports wildcards and regular expressions. The default is wildcards. Use the `-regexp` switch to use regular expressions instead.

Possible return values (Continued)

- If you run `get_status` on two or more properties that have a status of `error`, `unprocessed`, `undetermined`, or `processing`, the tool prioritizes the statuses in that order. That is, if there is one property that has status `error` and another one that has status `unprocessed`, the result of the command will be `error`.
 - If you specify a mix of assertions and covers, the return is `determined` if all assertions have status `proven` or `cex` and all covers have status `unreachable` or `covered`.
 - If you specify a mix of assertions and covers and the conditions for returning `determined` are not fulfilled, the return is `ar_determined` if all assertions have status `proven`, `cex`, or `ar_cex` and all covers have status `unreachable`, `covered`, or `ar_covered`.
-

Default

None

Return

Returns the aggregated proof status of a set of properties.

Examples

```
% get_status {p1}  
proven
```

```
% get_status {p2}  
covered
```

```
% get_status {p1 p2}  
determined
```

```
% get_status {p3}  
unprocessed
```

```
% get_status {*}  
unprocessed
```

JasperGold Apps Command Reference Manual

General Commands

See Also

[get_property_info](#)

[prove](#)

get_tool

Description

Use the `get_tool` command to determine which tool is running your script. This command can be useful for applying different settings depending on the JasperGold tool that is running the script.

Use the individual switches for this command in JasperGold Apps to access how the tool has been configured.

Syntax

```
get_tool [-batch | -app | -start_app | -list_apps]
```

Argument	Definition
<code>-batch</code>	Return true if the tool was launched in batch mode.
<code>-app</code>	Return the current app.
<code>-start_app</code>	Return the app that launched the tool.
<code>-list_app</code>	Return the list of available apps.

Default

None

Return

With no arguments, this command returns the abbreviation for the tool running your script. When you use arguments, the `get_tool` command returns one or more of the following: `true`, `false` or an abbreviation for an app.

JasperGold Apps Command Reference Manual

General Commands

Examples

```
% get_tool -app
```

See Also

[set_current_gui](#)

get_top_module

Description

Use the `get_top_module` command to display the name of the top module or entity that will be formally analyzed. The top module name contains any parameters used during elaboration, including their values.

This command is context-sensitive (it applies to the current task).

Syntax

```
get_top_module [-no_parameters]
```

Argument	Definition
<code>-no_parameters</code>	Return the top module name without any existing parameters.

Default

None

Return

Returns a string (identifier name) for the top hierarchical module or entity found in the design.

Examples

```
% get_top_module
```

See Also

[elaborate](#)

[get_root_name](#)

get_tx_attributes

Description

Use the `get_tx_attributes` command to get the signal attributes of a cover directive that represents a behavior.

Syntax

```
get_tx_attributes <cover_name>
```

Argument	Definition
<i>cover_name</i>	Display the signal attributes associated with the specified <i>cover_name</i> .

Default

No default values

Return

Returns a signal list.

Examples

```
% get_tx_attributes send_packet
```

See Also

[add_tx_attributes](#)
[remove_tx_attribute](#)

get_version

Description

Use the `get_version` command to display the current version release number.

Syntax

```
get_version
```

Argument	Definition
no arguments	

Default

None

Return

Returns a string containing the version release number.

Examples

```
% get_version
```

See Also

No related commands

help

Description

Use this command to display help information, including the following:

- List of all JasperGold Apps commands (use `help` with no arguments)
- Details of a specified command
- List of all message help (use `help -message` with no additional arguments)
- Details about a specified Error, Warning, or Info message

The `help` command supports tab completion, which behaves similar to a standard bash shell. That is, the first time you press the Tab key, nothing happens unless there is a unique completion for the command.

Example: If we have the commands `cmd1` and `cmd2` and you type `help c` and press Tab, the text will be expanded to `help cmd`. The second time you press Tab, the tool lists (in the log pane) all available commands that start with the characters to the left of the cursor. To distinguish this text from other output in this pane, the text is blue. In this example, the tool lists `cmd1` and `cmd2`. (You might need to scroll to the bottom of the log pane to see the return.)

Syntax

```
help [-usage] [<command_name>]  
help -message [<message_id>]  
help -apropos <keyword>  
help -gui [tcl | <command_name>]
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<code>[-usage] [command_name]</code>	<p>Display the command syntax and details for the specified command.</p> <p>If you do not specify a <i>command_name</i>, the tool lists all available command help.</p> <p>When you specify a non-existent command as an argument to the <code>help</code> command, the tool lists all commands that contain the argument as part of their names. For example, <code>help get</code> lists all commands that contain <code>get</code> in their names at any position.</p> <p>If you specify the switch <code>-usage</code>, the tool does not display the description and the example sections of the command help.</p>
<code>-message [message_id]</code>	<p>Display message help for the specified <i>message_id</i>.</p> <p>If you do not specify a <i>message_id</i>, the tool lists all available message help.</p>
<code>-apropos keyword</code>	<p>List available commands with a brief description containing the keyword.</p>
<code>-gui [tcl command_name]</code>	<p>Open the JasperGold Help Browser window to show the Application Guides and Tcl command descriptions.</p> <ul style="list-style-type: none">■ Use <code>tcl</code> to launch the window with the Tcl command descriptions in the foreground.■ Use the name of a command in JasperGold to directly open its help content.

Return

No value returned

Examples

```
% help
```

JasperGold Apps Command Reference Manual

General Commands

```
% help stopat  
% help -usage visualize  
% help -message  
% help -message ICD003  
% help -apropos counter  
% help -gui analyze
```

See Also

No related commands

include

Description

Use this command to read and execute a specified script file of Tcl commands or JasperGold Apps commands. From the Tcl interpreter's point of view, this command behaves exactly like the built-in `source` command. The difference between the two commands is that the `include` command echoes executed commands to the log file and to the console, which makes it easier to debug problems in script files. In the console, the echoed commands are preceded with %% and the font color is blue-green.

Note: For additional error information, check the content of the `$errorInfo` variable by running the following command:

```
% set $errorInfo
```

Syntax

```
include <file_name>
```

Argument	Definition
<i>file_name</i>	Read and execute the specified script.

Return

Same as Tcl's built-in `source` command.

Examples

```
% include script.tcl
```

See Also

[save](#)
[stopat](#)
[restore](#)

interrupt

Description

The command `interrupt -enable` makes a script interruptible. The command takes a script as an argument. It runs the script once and returns its result. While the script is running, interrupt checking is enabled, which enables the stop button in the GUI.

To use this capability, you must add explicit checks in the script with the command `interrupt -check`, which generates an error with the message “interrupted” error message if there is a pending request to interrupt. The error is normal except that the innermost `interrupt -enable` command catches it. To avoid catching the error, specify the `-rethrow` switch. When you use this switch, the error passes through like any other result from the script. This behavior is useful if you have nested `interrupt -enable` commands.

It is possible to run `interrupt -check` outside `interrupt -enable` to check whether the previous command was interrupted.

There are three sources of interrupts:

- The user pressed the stop button in the GUI.
- The command exited because of the command time limit.
- The user pressed `Ctrl+C` (batch mode only).

Note: `interrupt` without a switch is equivalent to `interrupt -check`.

Syntax

```
interrupt (-enable [-rethrow] <script>)
interrupt [-check]
```

Argument	Definition
<code>-enable script [-rethrow]</code>	<p>Make the specified script interruptible. Use <code>-rethrow</code> to avoid catching errors.</p>

JasperGold Apps Command Reference Manual

General Commands

[-check]

Return an error with the message “interrupted” if there is a pending request to interrupt.

Return

No return value.

Examples

```
// Count seconds until interrupted:  
% interrupt -enable {  
    set count 0  
    while {1} {  
        puts $count  
        incr count  
        after 1000  
        interrupt -check  
    }  
}
```

See Also

[get_cmd_time_limit](#)

ipk

Description

Use the `ipk` command to analyze an Intelligent Proof Kit (IPK) for use with formal verification, extract proof kit files for use with simulation, or load wrapper files.

Syntax

```
ipk -jdb <jdb>
  [ -analyze
    | -extract_pk_files
      (jaspergold | sim) [-dir <output_dir>]
    | -extract_wrapper [-dir <output_dir>]
    | -load_wrapper <wrapper_file_tcl_list>
  ]
ipk -list [-silent]
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<pre>-jdb jdb [-analyze -extract_pk_files (jaspergold sim) [-dir output_dir] -extract_wrapper [-dir output_dir] -load_wrapper wrapper_file_tcl_list]</pre>	<p>Load the specified IPK database.</p> <ul style="list-style-type: none">■ Use <code>-jdb</code> to specify the IPK you want to load. If you do not supply additional switches, the tool analyzes the specified IPK.■ Use <code>-analyze</code> to analyze the IPK for use with formal verification.■ Use <code>-extract_pk_files</code> to extract files suitable for verification with simulation and use <code>-dir</code> to specify the output directory for extracted files. The default directory is <code>pwd</code>. The supported arguments are:<ul style="list-style-type: none">□ <code>jaspergold</code>: JasperGold Apps□ <code>sim</code>: Simulation encrypted with IEEE P1735■ Use <code>-extract_wrapper</code> to extract a default wrapper for the IPK and use <code>-dir</code> to specify the output directory for extracted files. The default directory is <code>pwd</code>.■ Use <code>-load_wrapper</code> to analyze a proof kit using one or more specified wrappers.
<pre>-list [-silent]</pre>	<p>Show the list of IPK files installed in JasperGold Apps.</p> <p>Use <code>-silent</code> to return the list as Tcl return values without printing the list on the screen.</p>

Return

No return value.

Examples

```
# Load DFI 3.0 proof kit in JasperGold using a wrapper:
% ipk -jdb jasper_dfi3.jdb -load_wrapper jasper_dfi3_wrapper_1.sv
```

JasperGold Apps Command Reference Manual

General Commands

```
% ipk -jdb jasper_dfi3.jdb.gz -load_wrapper jasper_dfi3_wrapper_1.sv
```

See Also

[analyze](#)

[elaborate](#)

ipxact

Description

Use the `ipxact` command to get information from an IP-XACT specification or to request actions related to this specification.

Syntax

Action Switches

```
ipxact -load <xml_file_name.xml>+
[-lib_dir <ipxact_directory>]*
[-rtl_dir <rtl_directory>]*
[-recursive]

ipxact -validate <spirit:design_name>
(-top |-instance <instance_name>)

ipxact -clear
```

List Switches

```
ipxact -list
( file | design | component | interface
|-design <spirit:design_name>
( instance | component | interconnection | hier_connection
| ad_hoc_connection)
|-component <spirit:component_name>
( port | interface | design | memory_map)
|-component <spirit:component_name>
-interface <spirit:busInterface_name> (port_map | interface_type)
|-interface <spirit:busType_type> signal
)
```

Interface Connectivity Map

```
ipxact -generate_connectivity_map <csv_file_name.csv>
[<spirit:design_name>
 [-instance <instance_name>]
 [-module <module_name>]
]
[-interfaces]
```

JasperGold Apps Command Reference Manual

General Commands

```
[-recursive]
[-infer_interface_definitions]
[-no_debug]
```

[CSR Map](#)

```
ipxact -generate_csr_map <csv_file_name.csv>
[-component <spirit:component_name>]
[-maps <maps>]
[-no_debug]
```

[Setup Generation](#)

```
ipxact -generate_setup_script
-design <spirit:design_or_component_name>
-file <tcl_file_name.tcl> [-force]
[-analyze_opts <analyze_options_list>]
[-elaborate_opts <elaborate_options_list>]
[-alias <alias_name> <spirit:busType_type>] +
[-bind_file <bind_file.sv>]
```

Argument	Definition
Action Switches	
-load <i>xml_file_name.xml</i> + [-lib_dir <i>ipxact_directory</i>]* [-rtl_dir <i>rtl_directory</i>]* [-recursive]	<p>Load the IP-XACT specification contained in <i>xml_file_name.xml</i> files and the referenced IP-XACT elements.</p> <ul style="list-style-type: none">■ Use -lib_dir to search the specified directory for IP-XACT files for information referenced by the IP-XACT files loaded.■ Use -rtl_dir to search the specified directory for files specified by <spirit:component> or <spirit:fileSets>.■ Use -recursive to enable a recursive search in the specified directories.

JasperGold Apps Command Reference Manual

General Commands

```
-validate spirit:design_name [ (-top | -instance instance_name) ]
```

Analyze the specified <*spirit:design*> element and check for self-consistency within the IP-XACT content.

Use *-instance* to specify that *instance_name* should implement *spirit:design_name* when checked for consistency with the current elaborated RTL. Otherwise, *spirit:design_name* is assumed to be the specification of the top module.

Note: You must elaborate the design before using this command.

```
-clear
```

Clear all previously loaded IP-XACT elements from memory.

List Switches

```
-list (file | design | component | interface)
```

List the information that matches the specified criteria.

Note: The following switches are valid for the global IP-XACT scope.

- Use *file* to list the IP-XACT files that have been loaded.
 - Use *design* to list the <*spirit:design*> elements.
 - Use *component* to list the <*spirit:component*> elements.
 - Use *interface* to list all <*spirit:busInterface*> element names.
-

JasperGold Apps Command Reference Manual

General Commands

```
-design spirit:design_name
  -list ( instance | component | interconnection
         | hier_connection | ad_hoc_connection )
```

List the information that matches the specified criteria.

Note: The following switches are valid for the *spirit:design_name* scope.

- Use `instance` to list *spirit:component_name* instances.
- Use `component` to list components instantiated.
- Use `interconnection` to list `<spirit:interconnection>` elements. The result is a list of pairs. The first pair is formed by an instance name and the name of an interface implemented by that instance. The second element of the pair may also be a description of instance and interface, or a list of such descriptions. In the first case, the two interfaces are connected, while in the second case, the interfaces from the list monitor the interface described by the first element of the pair.
- Use `hier_connection` to list `<spirit:hierConnection>` elements. The result is a list of tuples. The first element of the tuple is a *spirit:busInterface_name* implemented by the component corresponding to *spirit:design_name*. The second and third elements of the tuple define an instance and a *spirit:busInterface_name*. The interface and the instance's bus interface form a connection.
- Use `ad_hoc_connection` to list `<spirit:adHocConnection>` elements. The result is a list of pairs. Each pair describes two ports. If a port is described by a single element, then that element is the port name and the port belongs to the component associated with *spirit:design_name*. If the description has two elements, then the first element is the name of the instance that contains the port and the second element is the port name.

JasperGold Apps Command Reference Manual

General Commands

```
-component spirit:component_name
    -list (port | interface | design | memory_map)
```

List the information that matches the specified criteria.

Note: The following switches are valid for the *spirit:component_name* scope.

- Use **port** to list ports present in the component.
- Use **interface** to list interfaces implemented by the specified *spirit:component_name*.
- Use **design** to get the <*spirit:design*> associated with the specified *spirit:component_name*.
- Use **memory_map** to list the address maps in the component.

```
-component spirit:component_name
    -interface spirit:busInterface_name
    -list (port_map | interface)
```

List the information that matches the specified criteria.

- Use **port_map** to list the maps between logical and physical ports in the *spirit:busInterface_name* implementation of the *spirit:component_name*. The result is a list of pairs with the logical and physical names.
- Use **interface_type** to list the *spirit:busType_type* and the *spirit:abstractionType_type* of the *spirit:busInterface_name* implementation of *spirit:component_name*.

```
-interface spirit:busType_type -list signal
```

List the signals of the <*spirit:abstractionDefinition*> element for *spirit:busType_type*.

Interface Connectivity Map

```
-generate_connectivity_map csv_file_name.csv
[sprit:design_name [-instance instance_name]
 [-module module_name]]
[-interfaces]
[-recursive]
[-infer_interface_definitions] [-no_debug]
```

Generate a Connectivity map based on connections found in *sprit:design_name*.

The resulting map has interface definitions and interface connections.

- Use `-instance` when *sprit:design_name* is implemented by *instance_name*. Otherwise, the tool assumes that the top module implements *sprit:design_name*.
- Use `-module` when *sprit:design_name* is implemented by *module_name*. Otherwise, if the instance name is provided, the tool searches for the module name in the current elaboration, or assumes that *sprit:design_name* is the module name if the instance is not specified.
- Use `-interfaces` to generate a map without connections. The map contains only interface definitions and port maps.
- Use `-recursive` to generate connectivity information for all IP-XACT elements found by recursively exploring the instance hierarchy rooted in *sprit:design_name*. By default, the command generates connectivity information only for *sprit:design_name*.
- Use `-infer_interface_definitions` to generate `INTERFACE_MAP` definitions for missing `<sprit:abstractionDefinition>` elements.
- Use `-no_debug` to generate the map without any reference to the IP-XACT file names and line numbers.

Note: You must elaborate the design before using this command.

CSR Map

```
-generate_csr_map csv_file_name.csv
[-component spirit:component_name]
[-maps maps] [-no_debug]
```

Generate a CSV file for `check_csr -load` based on the description of *spirit:component_name*.

- Use `-component` to specify the *spirit:component_name*. This switch is not required if only one *spirit:component* is loaded.
- Use `-maps` to limit the CSV file content to the specified maps.
- Use `-no_debug` to generate the map without any reference to the IP-XACT file names and line numbers.

Note: You must elaborate the design before using this command.

Setup Generation

```
ipxact -generate_setup_script  
  -design spirit:design_or_component_name  
  -file tcl_file_name.tcl [-force]  
  [-analyze_opts analyze_options_list]  
  [-elaborate_opts elaborate_options_list]  
  [-alias alias_name spirit:busType_type] +  
  [-bind_file bind_file.sv]
```

Generate a setup file that can be used to synthesize and read the netlist and bind IPK monitors.

The tool generates setup commands based on information extracted from the IP-XACT definition.

- Use `-design` to specify the top IP-XACT element.

Note:

- This element can be either a `spirit:design` or `spirit:component`.
 - Use the Tcl variable `ipk_directory` to point to the directory containing the IPK files.
- Use `-force` if the specified setup file or bind file already exists and you want to overwrite it.
- Use `-analyze_opts` to pass the specified arguments to the `analyze` command. If you do not use this switch, the tool uses the directories specified with `-rtl_dir` when loading the IP-XACT definition to generate `+incdir+` options.
- Use `-elaborate_opts` to pass the specified arguments to the `elaborate` command.
- Use `-alias` to create aliases for `spirit:busType_type` values. The tool uses these values to identify the IPK that will be used to verify the interface.
- Use `-bind_file` to generate a Verilog file instead of connect `-bind` commands to instantiate IPK monitors.

Note: The generated setup file might require manual editing before you can load it.

Return

A list of requested objects or the output of a requested action.

Examples

```
% ipxact -load main_ipxact_design.xml -lib_dir components -lib_dir interfaces
% ipxact -load main_ipxact_design.xml -rtl_dir design_rtl -recursive
% ipxact -list component
% ipxact -list port_map -component bridge -interface clock
% ipxact -list signals -interface ingress
% ipxact -list port -component bridge
% ipxact -list instance -design bridge

# Interface Connectivity Map Generator
# -----
# The following commands create a Connectivity map with the connections
# generated from an IP-XACT specification.
% ipxact -load bridge.xml -lib_dir ../ipxact_library -rtl_dir .
% ipxact -generate_connectivity_map interface_connections.csv bridge -instance
bridge0

# CSR Map Generator
# -----
# The following commands create a CSR map from the description of two
# memory maps of an IP-XACT component.
% ipxact -load data_banks.xml -lib_dir ../ipxact_library -rtl_dir .
% ipxact -generate_csr_map csr_map.csv -component data_bank -maps "data0 data1"
```

See Also

[check_conn](#)
[check_csr](#)

jasper_model_divider

Description

Use the `jasper_model_divider` command to instantiate and configure the `jasper_model_divider` Proof Accelerator. This Proof Accelerator provides a means of achieving full, unbounded proofs and visualizations on properties that involve logic containing dividers (properties that are otherwise generally intractable for formal tools).

To learn more about the `jasper_model_divider` Proof Accelerator, access the `jasper_model_divider` data sheet. To see the data sheet, do one of the following:

- Use the command `proof_accelerator -doc jasper_model_divider`.
- From the *Help* menu, choose *Proof Accelerator Datasheets – jasper_model_divider*.

Automatically extracted Proof Accelerator tasks can contain implicit assumptions that are not visible to the user, and constraining tasks with assumptions that are not visible makes it impossible to evaluate their impact.



If you are using Verification Component Proof Accelerators (for example, `jasper_scoreboard_2`) or Abstraction Component Proof Accelerators (for example, `jasper_model_fifo`), do not attempt to link a Proof Accelerator task to a non-Proof Accelerator task. Instead, you should do the following:

```
task -link my_task -to <pa_task>
```

This restriction does not apply to Verification Enabler Proof Accelerators (for example, `jasper_model_async_cdc_wire`).

Syntax

```
jasper_model_divider
  [-abstract <signal_name> [-parameter <param_name> <param_value>]*]
  [-connect <port_name> <signal_name>]*]
  [-auto]
  ]
  [-init [-preload <operand_1> <operand_2>]* <signal_name>]
  [-justify [-precise] <signal_name>]
  [-suggest]
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<code>-abstract signal_name</code>	<p>Replace the specified divider signal with an instance of the jasper_model_divider Proof Accelerator.</p> <p>Run this command after elaboration and before specifying the clock and reset conditions.</p>
<code>-parameter param_name param_value</code>	<p>Use the parameters specified by the <i>param_name</i> and <i>param_value</i> pairs.</p> <p> <i>Important</i></p> <ul style="list-style-type: none">■ Use this switch with the <code>-abstract</code> switch.■ If you do not supply parameters, the tool uses the default parameter values. (Refer to the jasper_model_divider datasheet.)
<code>-connect port_name signal_name</code>	<p>Connect the port (<i>port_name</i>) on the boundary of the jasper_model_divider Proof Accelerator to a specified signal (<i>signal_name</i>) in the design.</p> <p>Specify <i>signal_name</i> in one of two ways:</p> <ul style="list-style-type: none">■ As a port on the boundary of the module containing the divider being replaced■ As an absolute path to a signal anywhere in the verification environment <p>Use this switch with the <code>-abstract</code> switch.</p>

JasperGold Apps Command Reference Manual

General Commands

-auto

Automatically connect the input port of the jasper_model_divider Proof Accelerator to the design signal with the same name.



- Use this switch with the -abstract switch.
- If you also use the -connect switch, the explicit connections specified with that switch take precedence over the name-based matching specified by this -auto switch.
- The tool does not autoconnect signals of different widths.

Note: For proper functioning, take special care with the `clk` signal. Use one of the following three options:

- Ensure that the module containing the divider that is being abstracted contains a port named `clk`, and allow the connection of the jasper_model_divider `clk` signal to be handled with the -auto switch.
- Ensure that the module containing the divider that is being abstracted contains a clock port, and then use an explicit -connect switch to make the connection. If the clock port on the module containing the divider is called `clock`, then the syntax is as follows:

`-connect clk clock`

- Use an explicit `clock` command to define the clock behavior for the jasper_model_divider. If the path to the divider being abstracted is `hash_i.jasper_divider_i_quotient`, then the `clock` command is as follows:

`clock <clock> hash_i.jasper_divider_i_quotient.clk`

In this command, `<clock>` is a clock previously declared with the JasperGold Apps `clock` command.

-init *signal_name*

Set *signal_name* to the maximum level of abstraction.

Run this command after elaboration and before specifying the clock and reset conditions.

JasperGold Apps Command Reference Manual

General Commands

```
-preload operand_1 operand_2
```

Preload the abstraction for the specified signal with the results of a specific division.

```
-justify
```

Decrease the abstraction level of the specified instance by one degree.

```
-precise
```

As part of the `justify` operation, preload the abstraction for the specified signal with the first abstract division encountered in the trace.

```
-suggest
```

Analyze the current trace to determine whether justifying the specified divider will provide any benefit.

Return

When you use `-suggest`, the return value is a list of divider instances that fall into one of the following two categories:

- Instances that have been abstracted by `jasper_model_divider`
- Instances that are returning abstract values in the current trace (hence, these instances might benefit from the application of `jasper_model_divider -justify`).

See Also

[proof_accelerator](#)

jasper_model_mpram

Description

Use the `jasper_model_mpram` command to instantiate and configure the `jasper_model_mpram` Proof Accelerator. The `jasper_model_mpram` Proof Accelerator is an encrypted macro that, along with a supporting Tcl command, provides a means of achieving full, unbounded proofs on properties that involve logic containing memories (properties that are otherwise generally intractable for formal tools).

The `jasper_model_mpram` Proof Accelerator achieves this goal by providing a flexible abstraction that enables the formal engines to abstract the majority of the specified memory while preserving those parts of its behavior that are needed to achieve the proof.

To learn more about the `jasper_model_mpram` Proof Accelerator, access the `jasper_model_mpram` data sheet. To see the data sheet, do one of the following:

- Use the command `proof_accelerator -doc jasper_model_mpram`.
- From the *Help* menu, choose *Proof Accelerator Datasheets – jasper_model_mpram*.

Automatically extracted Proof Accelerator tasks can contain implicit assumptions that are not visible to the user, and constraining tasks with assumptions that are not visible makes it impossible to evaluate their impact.

If you are using Verification Component Proof Accelerators (for example, `jasper_scoreboard_2`) or Abstraction Component Proof Accelerators (for example, `jasper_model_fifo`), do not attempt to link a Proof Accelerator task to a non-Proof Accelerator task. Instead, you should do the following:

`task -link my_task -to <pa_task>`

Note: This restriction does not apply to Verification Enabler Proof Accelerators (for example, `jasper_model_async_cdc_wire`).

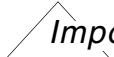
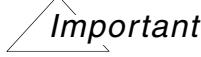
Syntax

```
jasper_model_mpram
  [-abstract <instance_name> [-parameter <param_name> <param_value>]*]
  [-connect <port_name> <signal_name>]*
  [-auto]
]
```

JasperGold Apps Command Reference Manual

General Commands

```
[-init [-preload <addr> [<data>]]* <instance_name>]
[-justify <instance_name>]
[-suggest]
[-clear]
```

Argument	Definition
<code>-abstract instance_name</code>	<p>Replace the specified memory instance and all other like instances with an instance of the <code>jasper_model_mpram</code> Proof Accelerator.</p> <p> <i>Important</i></p> <ul style="list-style-type: none">■ To replace a single memory instance, use one of the following workarounds:<ul style="list-style-type: none">□ Create a requirements file that manually instantiates <code>jasper_model_mpram</code>, then bind it to the specific instance with <code>connect -bind</code>.□ Elaborate the given memory instance with a unique parameter, then use the <code>jasper_model_mpram</code> command on the specific instance.■ Run this command after elaboration and before specifying the clock and reset conditions.
<code>-parameter param_name param_value</code>	<p>Use the parameters specified by the <code>param_name</code> and <code>param_value</code> pairs.</p> <p> <i>Important</i></p> <ul style="list-style-type: none">■ Use this switch with the <code>-abstract</code> switch.■ If you do not supply parameters, the tool uses the default parameter values. (Refer to the <code>jasper_model_mpram</code> datasheet.)

JasperGold Apps Command Reference Manual

General Commands

```
-connect port_name signal_name
```

Connect the port (*port_name*) on the boundary of the jasper_model_mpram Proof Accelerator to a specified signal (*signal_name*) in the design.

Specify *signal_name* in one of two ways:

- As a port on the boundary of the memory instance being replaced
- As an absolute path to a signal anywhere in the verification environment

Use this switch with the `-abstract` switch.

JasperGold Apps Command Reference Manual

General Commands

-auto

Automatically connect the input port of the jasper_model_mpram Proof Accelerator to the design signal with the same name.



- Use this switch with the -abstract switch.
- If you also use the -connect switch, the explicit connections specified with that switch take precedence over the name-based matching specified by this -auto switch.
- The tool does not autoconnect signals of different widths.

Note: For proper functioning, take special care with the `clk` signal. Use one of the following three options:

- Ensure that the RAM that is being abstracted contains a port named `clk`, and allow the connection of the jasper_model_mpram `clk` signal to be handled with the -auto switch.
- Ensure that the RAM that is being abstracted contains a clock port, and then use an explicit -connect switch to make the connection. If the clock port on the RAM is called `clock`, then the syntax is as follows:

-connect `clk` `clock`

- Use an explicit `clock` command to define the clock behavior for the jasper_model_mpram. If the path to the RAM being abstracted is `EXT_MEM`, then the `clock` command is as follows:

`clock <clock> EXT_MEM.jasper_mpram_i.clk`

In this command, `<clock>` is a clock previously declared with the JasperGold Apps `clock` command.

-init *instance_name*

Set *instance_name* to the maximum level of abstraction.

-preload *addr* *data*+

Treat the contents of the specified address (*addr*) as non-abstracted data and, when provided, use the optional data value (*data*) as the reset value for the specified memory location.

Use this switch with the -init switch.

JasperGold Apps Command Reference Manual

General Commands

-justify *instance_name*

Decrease the abstraction level of the specified instance by one memory location.

Note: This Proof Accelerator models the RAM being abstracted by selecting certain memory locations and treating them concretely. Generally, this is done on a first-come, first-served basis; however, in the event of multiple simultaneous accesses, ports are serviced in the following order:

- Write ports before read ports
- Lower numbered ports before higher numbered ports

-suggest

Analyze the current trace to determine whether justifying any of the instances of the jasper_model_mpram will provide any benefit.

-clear

Clear the jasper_model_mpram data structures.

Note: The command `clear -all` does not clear Proof Accelerator data structures.

Return

When you use `-suggest`, the return value is a list of memory instances that fall into one of the following two categories:

- Instances that have been abstracted by `jasper_model_mpram`
- Instances that are returning abstract values in the current trace (hence, these instances might benefit from the application of `jasper_model_mpram -justify`).

Examples

// Abstract the EXT_MEMORY:

```
% jasper_model_mpram -abstract EXT_MEMORY -auto \
    -connect write      {MEM_Req && !MEM_RnW} \
    -connect read       {MEM_Req && MEM_RnW} \
    -connect addr_rd   {MEM_Addr} \
    -connect addr_wr   {MEM_Addr} \
    -connect data_rd   {MEM_Read_data}
```

JasperGold Apps Command Reference Manual

General Commands

```
-connect    data_wr          {MEM_Write_data}           \
-connect    read_allocate   { (MEM_Addr == test_addr) } \
-connect    write_allocate  { (MEM_Addr == test_addr) } \
-connect    concrete        {concrete}                 \
-parameter ADDR_WIDTH 32                           \
-parameter DATA_WIDTH 3

// Constrain the MEM_Ready signal such that it cannot be asserted unless the
// MEM_Req signal is also asserted:
% assume {~MEM_Req |-> ~MEM_Ready}

// Initialize the EXT_MEMORY abstraction:
% jasper_model_mpram -init EXT_MEMORY

// Set the depth of the EXT_MEMORY abstraction (in this case, only one location is
// required):
% jasper_model_mpram -justify EXT_MEMORY
```

See Also

[proof_accelerator](#)

jasper_model_multiplier

Description

Use the `jasper_model_multiplier` command to instantiate and configure the `jasper_model_multiplier` Proof Accelerator. This Proof Accelerator provides a means of achieving full, unbounded proofs and visualizations on properties that involve logic containing multipliers (properties that are otherwise generally intractable for formal tools).

To learn more about the `jasper_model_multiplier` Proof Accelerator, access the `jasper_model_multiplier` data sheet. To see the data sheet, do one of the following:

- Use the command `proof_accelerator -doc jasper_model_multiplier`.
- From the *Help* menu, choose *Proof Accelerator Datasheets – jasper_model_multiplier*.

Automatically extracted Proof Accelerator tasks can contain implicit assumptions that are not visible to the user, and constraining tasks with assumptions that are not visible makes it impossible to evaluate their impact.



If you are using Verification Component Proof Accelerators (for example, `jasper_scoreboard_2`) or Abstraction Component Proof Accelerators (for example, `jasper_model_fifo`), do not attempt to link a Proof Accelerator task to a non-Proof Accelerator task. Instead, you should do the following:

```
task -link my_task -to <pa_task>
```

Note: This restriction does not apply to Verification Enabler Proof Accelerators (for example, `jasper_model_async_cdc_wire`).

Syntax

```
jasper_model_multiplier
  [-abstract <signal_name> [-parameter <param_name> <param_value>]*]
  [-connect <port_name> <signal_name>]*
  [-auto]
  [-init [-preload <operand_1> <operand_2>]* <signal_name>]
  [-justify [-precise] <signal_name>]
  [-suggest]
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<code>-abstract signal_name</code>	<p>Replace the specified multiplier signal with an instance of the jasper_model_multiplier Proof Accelerator.</p> <p>Run this command after elaboration and before specifying the clock and reset conditions.</p>
<code>-parameter param_name param_value</code>	<p>Use the parameters specified by the <i>param_name</i> and <i>param_value</i> pairs.</p> <p> <i>Important</i></p> <ul style="list-style-type: none">■ Use this switch with the <code>-abstract</code> switch.■ If you do not supply parameters, the tool uses the default parameter values. (Refer to the jasper_model_multiplier datasheet.)
<code>-connect port_name signal_name</code>	<p>Connect the port (<i>port_name</i>) on the boundary of the jasper_model_multiplier Proof Accelerator to a specified signal (<i>signal_name</i>) in the design.</p> <p>Specify <i>signal_name</i> in one of two ways:</p> <ul style="list-style-type: none">■ As a port on the boundary of the module containing the multiplier being replaced■ As an absolute path to a signal anywhere in the verification environment <p>Use this switch with the <code>-abstract</code> switch.</p>

JasperGold Apps Command Reference Manual

General Commands

-auto

Automatically connect the input port of the jasper_model_multiplier Proof Accelerator to the design signal with the same name.



- Use this switch with the -abstract switch.
- If you also use the -connect switch, the explicit connections specified with that switch take precedence over the name-based matching specified by this -auto switch.
- The tool does not autoconnect signals of different widths.

Note: For proper functioning, take special care with the `clk` signal. Use one of the following three options:

- Ensure that the module containing the multiplier that is being abstracted contains a port named `clk`, and allow the connection of the jasper_model_multiplier `clk` signal to be handled with the -auto switch.
- Ensure that the module containing the multiplier that is being abstracted contains a clock port, and then use an explicit -connect switch to make the connection. If the clock port on the module containing the multiplier is called `clock`, then the syntax is as follows:

-connect clk clock

- Use an explicit `clock` command to define the `clock` behavior for the jasper_model_multiplier. If the path to the multiplier being abstracted is `hash_i.jasper_multiplier_i_product`, then the `clock` command is as follows:

clock <clock> hash_i.jasper_multiplier_i_product.clk

In this command, `<clock>` is a clock previously declared with the JasperGold Apps `clock` command.

-init *signal_name*

Set *signal_name* to the maximum level of abstraction.

Run this command after elaboration and before specifying the clock and reset conditions.

JasperGold Apps Command Reference Manual

General Commands

-preload *operand_1* *operand_2*

Treat the product of the specified operands as non-abstracted data.

Use this switch with the -init switch.

Examples:

- To specify that the Proof Accelerator should start from reset knowing to track the result of 8'hff * 4'hB, use the following:
-preload 8'hFF 4'hB
 - To specify the same product, but with decimal notation, use the following:
-preload 8'd255 4'd11
-

-justify

Decrease the abstraction level of the specified instance by one multiply operation.

-precise

Specify that the first abstracted operation in the current trace should be preloaded into the abstraction as a part of the justify operation.

-suggest

Analyze the current trace to determine whether justifying any of the instances of the jasper_model_multiplier will provide any benefit.

Return

When you use -suggest, the return value is a list of multiplier instances that fall into one of the following two categories:

- Instances that have been abstracted by jasper_model_multiplier
- Instances that are returning abstract values in the current trace (hence, these instances might benefit from the application of jasper_model_multiplier -justify).

Examples

```
// Abstract the multiplier in a hash function:  
  
% jasper_model_multiplier -abstract hash_i.product \  
  -connect a  hash_i.address           \  
  -connect b  hash_i.m                \  
  -connect y  hash_i.product          \  
  -auto
```

JasperGold Apps Command Reference Manual

General Commands

```
-parameter A_WIDTH $cache_addr_width          \
-parameter B_WIDTH 6                         \
-parameter Y_WIDTH [expr $cache_addr_width + 6]

// Initialize the abstraction:
% jasper_model_multiplier -init hash_i.product

// Set the depth of the abstraction to one operation:
% jasper_model_multiplier -justify hash_i.product
```

See Also

[proof_accelerator](#)

jasper_model_ram

Description

Use the `jasper_model_ram` command to instantiate and configure the `jasper_model_ram` Proof Accelerator. This Proof Accelerator provides a means of achieving full, unbounded proofs and visualizations on properties that involve logic containing memories (properties that are otherwise generally intractable for formal tools).

To learn more about the `jasper_model_ram` Proof Accelerator, access the `jasper_model_ram` data sheet. To see the data sheet, do one of the following:

- Use the command `proof_accelerator -doc jasper_model_ram`.
- From the *Help* menu, choose *Proof Accelerator Datasheets – jasper_model_ram*.

Automatically extracted Proof Accelerator tasks can contain implicit assumptions that are not visible to the user, and constraining tasks with assumptions that are not visible makes it impossible to evaluate their impact.



If you are using Verification Component Proof Accelerators (for example, `jasper_scoreboard_2`) or Abstraction Component Proof Accelerators (for example, `jasper_model_fifo`), do not attempt to link a Proof Accelerator task to a non-Proof Accelerator task. Instead, you should do the following:

```
task -link my_task -to <pa_task>
```

This restriction does not apply to Verification Enabler Proof Accelerators (for example, `jasper_model_async_cdc_wire`).

Syntax

```
jasper_model_ram
  [-abstract <instance_name> [-parameter <param_name> <param_value>]*]
  [-connect <port_name> <signal_name>]*
  [-auto]
  [-init [-preload <addr> [<data>]]* <instance_name>]
  [-justify <instance_name>]
  [-suggest]
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<code>-abstract instance_name</code>	<p>Replace the specified memory instance and all other like instances with an instance of the jasper_model_ram Proof Accelerator.</p> <p> <i>Important</i></p> <ul style="list-style-type: none">■ To replace a single memory instance, use one of the following workarounds:<ul style="list-style-type: none">□ Create a requirements file that manually instantiates jasper_model_ram, then bind it to the specific instance with connect -bind.□ Elaborate the given memory instance with a unique parameter, then use the <code>jasper_model_ram</code> command on the specific instance.■ <i>Run this command after elaboration and before specifying the clock and reset conditions.</i>
<code>-parameter param_name param_value</code>	<p>Use the parameters specified by the <code>param_name</code> and <code>param_value</code> pairs.</p> <p> <i>Important</i></p> <ul style="list-style-type: none">■ Use this switch with the <code>-abstract</code> switch.■ If you do not supply parameters, the tool uses the default parameter values. (Refer to the jasper_model_ram datasheet.)

JasperGold Apps Command Reference Manual

General Commands

```
-connect port_name signal_name
```

Connect the port (*port_name*) on the boundary of the jasper_model_ram Proof Accelerator to a specified signal (*signal_name*) in the design.

Specify *signal_name* in one of two ways:

- As a port on the boundary of the memory instance being replaced.
- As an absolute path to a signal anywhere in the verification environment.

Use this switch with the `-abstract` switch.

Note: If you are connecting to a signal or port of an instance, you cannot specify *signal_name* signals that are in the top module.

JasperGold Apps Command Reference Manual

General Commands

-auto

Automatically connect the input port of the jasper_model_ram Proof Accelerator to the design signal with the same name.



- Use this switch with the -abstract switch.
- If you also use the -connect switch, the explicit connections specified with that switch take precedence over the name-based matching specified by this -auto switch.
- The tool does not autoconnect signals of different widths.

Note: For proper functioning, take special care with the `clk` signal. Use one of the following three options:

- Ensure that the RAM that is being abstracted contains a port named `clk`, and allow the connection of the jasper_model_ram `clk` signal to be handled with the -auto switch.
- Ensure that the RAM that is being abstracted contains a clock port, and then use an explicit -connect switch to make the connection. If the clock port on the RAM is called `clock`, then the syntax is as follows:

`-connect clk clock`

- Use an explicit `clock` command to define the clock behavior for the jasper_model_ram. If the path to the RAM being abstracted is `EXT_MEMORY`, then the `clock` command is as follows:

`clock <clock> EXT_MEMORY.jasper_ram_i.clk`

In this command, `<clock>` is a clock previously declared with the JasperGold Apps `clock` command.

JasperGold Apps Command Reference Manual

General Commands

```
-init instance_name
```

Set *instance_name* to the maximum level of abstraction.

Run this command after elaboration and before specifying the clock and reset conditions.

Note: “maximum level of abstraction” means that every access returns an abstract value (that is, nothing is justified). The default value is 32, which is also the maximum concrete values jasper_model_ram can store. Thus, there is no benefit to running `jasper_model_ram -justify <instance_name>` more than 32 times.

```
-preload addr data[+]
```

Treat the contents of the specified address (*addr*) as non-abstracted data, and when provided, use the optional [data](#) value as the reset value for the specified memory location.

Use this switch with the `-init` switch.

Examples:

- To specify that the Proof Accelerator should start from reset knowing to track address 8'bf, use the following:
`-preload 8'hFF`
- To specify the same address, but with decimal notation, use the following:
`-preload 8'd255`
- To specify the same address, but also supply the data value 32'b1, use the following:
`-preload 8'd255 32'b1`

In the first two examples, the tool uses the value of the `RESET_STYLE` parameter.

JasperGold Apps Command Reference Manual

General Commands

-justify

Decrease the abstraction level of the specified instance by one memory location.

Note:

- This Proof Accelerator models the RAM being abstracted by selecting certain memory locations and treating them concretely. Generally, this is done on a first-come, first-served basis; however, in the event of multiple simultaneous accesses, ports are serviced in the following order:
 - Write ports before read ports
 - Lower numbered ports before higher numbered ports
- The maximum number of concrete values jasper_model_ram can store is 32; therefore, there is no benefit to running jasper_model_ram -justify <instance_name> more than 32 times.

-suggest

Analyze the current trace to determine whether justifying any of the instances of the jasper_model_ram will provide any benefit.

Return

When you use -suggest, the return value is a list of memory instances that fall into one of the following two categories:

- Instances that have been abstracted by jasper_model_ram
- Instances that are returning abstract values in the current trace (hence, these instances might benefit from the application of jasper_model_ram -justify).

Examples

```
// Connect the ram PA (shown here for an example with two read ports and one
// write port):
%
% jasper_model_ram -abstract my_ram
          \
          -parameter ADDR_WIDTH           64 \
          -parameter DATA_WIDTH           16 \
          -parameter MEM_STYLE            0 \
          -parameter RESET_STYLE          1 \
          -connect clk                   clk \
          -connect rstN                  rstN \
```

JasperGold Apps Command Reference Manual

General Commands

```
-connect    write           myWrite    \
-connect    addr_wr        myWriteAddr \
-connect    data_wr        myWriteData \
-connect    read           myRead     \
-connect    addr_rd        myReadAddr \
-connect    data_rd        myReadData

// Initialize the abstraction (using the jasper_model_ram -init command):
% jasper_model_ram -init          \
    -preload   ...      \
    my_ram

// NOTE:
// a. If the PA is instantiated directly, the instance name used in the command
//     is the instance name of the PA
// b. If the PA is instantiated via Tcl, the instance name used in the command
//     is the instance name of the RAM being abstracted.

// Tune the abstraction (using the jasper_model_ram -justify command):
% jasper_model_ram -justify my_ram

// NOTE:
// a. If the PA is instantiated directly, the instance name used in the command
//     is the instance name of the PA
// b. If the PA is instantiated via Tcl, the instance name used in the command
//     is the instance name of the RAM being abstracted.
```

See Also

[proof_accelerator](#)

jasper_scoreboard_3

Description

Use the `jasper_scoreboard_3` command to load basic configuration definitions to be used with `jasper_scoreboard_3` instantiations. This PA provides a means to verify the integrity of data packets transferred through a design.

Note: `jasper_scoreboard_3` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

To learn more about the `jasper_scoreboard_3` PA, do one of the following to access the `jasper_scoreboard_3` data sheet:

- Use the command `proof_accelerator -doc jasper_scoreboard_3`.
- From the *Help* menu, choose *Proof Accelerator Datasheets – jasper_scoreboard_3*.

Syntax

```
jasper_scoreboard_3 -init
```

Argument	Definition
<code>-init</code>	<p>Load <code>jasper_scoreboard_3</code> parameter types and definition helpers. As shown in the <code>jasper_scoreboard_3</code> data sheet, all parameter types and values can be found within the <code>jasper_scoreboard_3_pkg</code> package or in the global scope with macros prefixed by the <code>JS3_</code> string.</p> <p>Run this command before analyzing your design files.</p> <p>Note: These definitions are stored together with <code>analyze</code> data; thus, invoking <code>analyze -clear</code> or <code>clear -all</code> unloads <code>jasper_scoreboard_3</code> definitions.</p>

Return

No return values

Examples

```
// Load parameter definitions  
% jasper_scoreboard_3 -init  
// Setup design  
% analyze -sv my_design.v  
% elaborate
```

See Also

[proof_accelerator](#)

justify

Description

Use this command to add a signal or signals to the analysis region, forcing them to take values consistent with their driving signals. You can only use this command when Design Tunneling is activated (`dst_mode` is on).

The recommended method is to use the GUI controls, (for example, in the Visualize windows and Complexity Manager windows) during an interactive JasperGold® Apps session instead of typing the `justify` commands in a script or at the Tcl command line. The `save` command typically generates the `justify` command.

A valid `justify -backTo*` includes any signal in the cone of influence. The following limitations apply to `justify` and `justify -conflict`.

- A valid `justify` includes at least one bit that is on the boundary of the analysis region and excludes signals that are not connected to the analysis region.
- An invalid `justify` includes bits that are in the interior of the region, in the cone of influence but outside the region, disconnected from the region, or outside the cone of influence.

Note: This command is context-sensitive (it applies to the current task).

Syntax

```
justify [-task <task_name>]
        [-name <name>]
        (<signal_name>+
        | [-conflict (<signal_name> <time_step>)+ ]
        )

justify -back_to_signal <signal_name>+
        [-task <task_name>] [-name <name>]
        [(-from_signal <signal_name>+)
        | (-from_property <property>+ [-regexp])]

justify -back_to_property <property>+ [-regexp]
        [-task <task_name>] [-name <name>]
        [(-from_signal <signal_name>+)
        | (-from_property <property>+ [-regexp])]

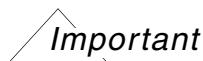
justify -back_to_instance <instance>+
        [-task <task_name>] [-name <name>]
        [(-from_signal <signal_name>+)
        | (-from_property <property>+ [-regexp])]
        [-until_inputs]
```

JasperGold Apps Command Reference Manual

General Commands

```
justify [-task <task_name>]
        (-list [-silent]
        |-show <name>
        |-remove <name>
        | -clear
        |-rename )<old_name> <new_name>
        )
```

Argument	Definition
<code>[-task task_name]</code>	Use the specified task instead of the current task for all operations. Replace <i>task_name</i> with a period (.) to specify the current task.
<code>[-name name]</code>	Label the current justify for future reference.
<code>signal_name</code>	Name of the signal you want to justify.
<code>[-conflict (signal_name time_step)+]</code>	Justify any of the signal-cycle inputs that would be included in the results of the suggest command and include relevant signals from performing Why on those signal-cycle pairs.



- Conflict analysis might include sequential logic.
- The tool bases the analysis on the value of the specified signal at the specified time.
- Use a signal name and trace time step to specify what you want to justify.
- Only use the *time_step* argument during `justify -conflict`. It has no effect on regular justifications.

JasperGold Apps Command Reference Manual

General Commands

Justifying Backwards to a Specified Signal, Property, or Instance

```
-back_to_signal signal_name+  
  [-task task_name] [-name name]  
  [-from_signal signal_name+ | -from_property property+ [-regexp]]
```

Expand the analysis region with logic connected to one or more specified signals.

The specified signals must be in the cone of influence of the task.

Switches:

- Use `-task` to specify a task other than the current task.
- Use `-name` to label the specified justify for future reference.
- Use `-from_signal` to specify one or more signals you want to justify.
- Use `-from_property` to specify one or more properties you want to justify. This switch supports wildcards and regular expressions; for example:

```
justify... -from_property {expression} -regexp
```

Note: If you do not use the `-from_signal` or `-from_property` switch, and there is no current trace target, the tool considers all the logic in the current or specified task.

JasperGold Apps Command Reference Manual

General Commands

```
-back_to_property property+ [-regexp]
[-task task_name] [-name name]
[-from_signal signal_name+ |-from_property property+ [-regexp]]
```

Expand the analysis region with logic connected to one or more specified properties.



- The specified properties must be in the cone of influence of the task.
- This switch supports wildcards and regular expressions; for example:

```
justify -back_to_property {expression} -regexp...
```

Switches:

- Use `-task` to specify a task other than the current task.
- Use `-name` to label the specified justify for future reference.
- Use `-from_signal` to specify one or more signals you want to justify.
- Use `-from_property` to specify one or more properties you want to justify. This switch supports wildcards and regular expressions; for example:

```
justify... -from_property {expression} -regexp
```

Note: If you do not use the `-from_signal` or `-from_property` switch, and there is no current trace target, the tool considers all the logic in the current or specified task.

JasperGold Apps Command Reference Manual

General Commands

```
-back_to_instance instance
[-task task_name] [-name name]
[-from_signal signal_name] | -from_property property [-regexp]
[-until_inputs]
```

Expand the analysis region with logic connected to the outputs of one or more specified instances.

The specified instances must be in the cone of influence of the task.

Switches:

- Use `-task` to specify a task other than the current task.
- Use `-name` to label the specified justify for future reference.
- Use `-from_signal` to specify one or more signals you want to justify.
- Use `-from_property` to specify one or more properties you want to justify. This switch supports wildcards and regular expressions; for example:

```
justify... -from_property {expression} -regexp
```

- Use `-until_inputs` to consider the inputs of the specified instance instead of the outputs.

Note: If you do not use the `-from_signal` or `-from_property` switch, and there is no current trace target, the tool considers all the logic in the current or specified task.

Utility Switches

```
[-task task_name]
```

Use the specified task instead of the current task for all operations.

Replace `task_name` with a period (.) to specify the current task.

```
-list
```

List the current task's justified signals and their justify names.

```
[-silent]
```

Turn off the output to the screen and return information with Tcl return values. (Use this option with any `-list*` option.)

Note: This option is helpful in Tcl scripts.

```
-show name
```

Return the specified justify's signals.

JasperGold Apps Command Reference Manual

General Commands

```
-remove name
```

Remove the specified justify.

```
-clear
```

Remove all justifies for the current task.

```
-rename old_name new_name
```

Change *old_name* justify to *new_name*.

Return

This command returns a string (identifier) for the justify.

Examples

```
% justify sigA
```

See Also

[set_dst_mode](#)

liberty

Description

Use the `liberty` command to load a Liberty™ library modeling standard file into JasperGold Apps to make `pg_pins` defined in the Liberty file accessible by the UPF command `connect_supply_net`.

Syntax

```
liberty -load <file_name>
```

Argument	Definition
<code>-load file_name</code>	Load the specified Liberty file.

Return

No return values

Examples

```
% liberty -load top.lib
```

See Also

No related commands

load_radix_file

Description

Use `load_radix_file` to load one or more radix files before a trace becomes available.

When you use this command, radix definitions are loaded and use clauses are only executed if a trace is available. When a trace becomes available, use clauses are applied automatically.

You can use this feature to read custom radix files and display predefined radix values (opcodes) instead of the original values on the trace. This feature is useful for debugging in a processor setting where you want to have the instruction type displayed instead of the bits that form the opcode. Refer to [Appendix H, “Custom Radix Files”](#) for radix file format and an example radix file.

Note: If a trace is available, this command is equivalent to `visualize -load_radix file_name`.

Syntax

```
load_radix_file <file_name>+
```

Argument	Definition
<i>file_name</i> +	<p>Load and apply radixes defined in the specified <i>file_name</i>. You may list one or more files. <i>file_name</i> file format consists of radix definition clauses and radix use clauses.</p>

Return

Returns an error message if a file cannot be loaded.

Examples

```
% load_radix_file radix_D.rdx
```

See Also

[visualize](#)

pretty_print

Description

Use `pretty_print` to display any Tcl list in a more easy-to-read format. With this command, the tool prints each element of a list on a separate line with no indentation. “{}” indicates null elements. Nested lists begin with “{”, end with “}”, and print with a two-space indentation.

Syntax

```
pretty_print
```

No arguments

Return

No return value.

Examples

```
// List without "pretty_print"  
% puts $someList  
a {} c {d.1 d.2} e {f.1 {} {f.3.1 f.3.2} f.4} g  
  
// List with "pretty_print"  
% pretty_print $someList  
a  
{ }  
c  
{  
    d.1  
    d.2  
}  
e  
{  
    f.1  
    {}  
    {  
        f.3.1  
        f.3.2  
    }  
}
```

JasperGold Apps Command Reference Manual

General Commands

```
}
```

```
f .4
```

```
}
```

```
g
```

See Also

No related commands

proof_accelerator

Description

Use this command to list Formal Scoreboard and Proof Accelerator modules info and to access documentation for a specific module.

To learn more about Proof Accelerators, access the following resources during your session:

- **Formal Scoreboard and Proof Accelerator data sheets** – To see the data sheets, do one of the following:
 - Use the command `proof_accelerator -doc <pa_name>`.
 - From the *Help* menu, choose *Proof Accelerator Datasheets* and the name of a Proof Accelerator.
- **Proof Accelerator tutorials** – On the Cadence Online Support website (support.cadence.com), go to the Tutorials page and download the Advanced Training presentation and the Formal Scoreboard Proof Accelerator lab kit.
- **Formal Scoreboard and Proof Accelerator examples** – In your build, access the examples in the `doc/example_jaspergold_apps/proof_accelerators` directory.

Syntax

```
proof_accelerator -list [-silent]
                     |-doc <pa_name>
```

Argument	Definition
<code>-list [-silent]</code>	List the set of available Proof Accelerator modules with a short description. Use the optional <code>-silent</code> switch to turn off output to the screen and return information with Tcl return values.
<code>-doc pa_name</code>	Open the PDF documentation file for the specified Proof Accelerator module.

JasperGold Apps Command Reference Manual

General Commands

Return

No value returned

Examples

```
% proof_accelerator -list  
% proof_accelerator -doc jasper_scoreboard_2
```

See Also

[jasper_model_mpram](#)
[jasper_model_multiplier](#)
[jasper_model_ram](#)

prove

Description

Use the `prove` command to determine if assertions are proven or cex (or ar_cex) and cover properties are unreachable or covered (or ar_covered).



- When a property result becomes cex, ar_cex, covered, or ar_covered, this command generates a trace, triggering the property. See “[visualize](#)” on page 864.
- For ar_cex and ar_covered, see “[set_dst_mode](#)” on page 1001.
- The main switches control which properties to prove. The tool accepts but ignores properties that are disabled. See `assert -disable` and `cover -disable`.
- A number of commands, most notably `set_engine_mode` and `set_proofgrid_mode` control how property results are determined.
- By default, all engines leverage proven properties as assumptions for the proof of other properties proven together.
- You can monitor proof progress by watching the *Properties* pane, *Console* tab, *Proof Messages* tab, and ProofGrid Manager window.
- Also use this command to control a `prove` command running in the background (see `prove -bg`).

Note:

- Determining property results can consume considerable time and memory.
- ProofGrid proves multiple properties in parallel using multiple licenses. Refer to “[set_proofgrid_per_engine_max_jobs](#)” on page 1141 for additional information.

Syntax

```
prove ( -all  
        |-task <task_name_tcl_list>  
        |-property <property_name_tcl_list> [-regexp]  
        |-instance <instance_name> [-except <subinstance_name>+]  
        |<expression>  
      )  
      [-auto]
```

JasperGold Apps Command Reference Manual

General Commands

```
[-with_helpers]
[-with_proven]
[-bg]
[-force]
[-asserts] [-covers]
[-quiet]
[-soft_constraints <signal_name_tcl_list> [-extend_region]]
[-soft_low_constraints <signal_name_tcl_list> [-extend_region]]
[-engine_mode <engine_mode_tcl_list>]
[-iter <N> ]
[-first_trace_attempt <N>]
[-per_property_time_max_limit <time_limit>]
[-per_property_time_limit <time_limit>]
[-per_property_time_limit_factor <N>]
[-time_limit <time_limit>]
[-verbosity <N> |-silent]
[-per_engine_max_jobs <N>]
[-no_traces]
[-no_cover_traces]
[-prefer_shortest]
[-dump_trace [-dump_trace_type (shm | vcd | fsdb)]
[-dump_trace_dir <dir_name>]]
```

When a prove command is running in the background:

```
prove ( -stop [[<index.>]<engine_mode>]
| -kill index.engine_mode
| -wait
| -status [-silent]
| -per_engine_max_jobs <N>
| -add <engine_mode>
| -restart [<index.>engine_mode]
| -ignore_property <property_name_tcl_list>
)
[-thread <thread_id>]
```

Argument	Definition
-all	Prove all assertions and cover properties in all tasks.

JasperGold Apps Command Reference Manual

General Commands

```
-task task_name_tcl_list
```

Prove all assertions and cover properties in the specified tasks.

Note:

- If you do not specify a *task_name_list*, the tool proves all assertions and covers in the current task.
- When specifying a list of task names, enclose the list in curly braces.
- This switch supports wildcards (*) and (?).

```
-property property_name_tcl_list [-regexp]
```

Prove one or more specified properties in the current task.

Note:

- To get maximal control over which proven properties are used as assumptions, use the default `set_proven_directive true`, do not use `-with_proven` (see definition below), and do include the proven properties you want to use in the list of properties. In this case, the tool uses only the proven properties that are on your list.
- To prove properties not in the current task, prepend the property names with `<task>::`.
- All listed properties must belong to the same task.
- This switch supports wildcards and regular expressions; for example:

```
prove -property {expression} -regexp
```

```
-instance instance_name [-except subinstance_name+]
```

Prove all assertions and cover properties in the specified module instance and its associated subinstances except the specified subinstances. The module subinstance names are relative to the module instance name, that is, you will not specify a full path.

Note:

- This option only works with assertions that belong to the `<embedded>` task, so the current task must be the `<embedded>` task (`task -set <embedded>`).
- This switch supports wildcards (*) and (?) in instance names.

JasperGold Apps Command Reference Manual

General Commands

expression

Enter the specified expression as a new assertion in the current task, and then prove this new assertion.

Note: JasperGold Apps supports SVA expressions on the command line. Refer to [Appendix D, “Tcl Support for SVA Expressions”](#) for guidance.

-auto

Dynamically stop the proof to add engines from `engine_mode` to optimize the use of available computing resources and scan the properties to optimize the use of `prove_per_property_time_limit` and `prove_per_property_time_limit_factor`.

If you use `prove -auto` with `proofgrid_mode local`, `proofgrid_max_local_jobs` is limited to the number of execution threads in the local machine.

Note: `prove -auto` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

-with_helpers

Prove with helper assertions.

You can prioritize helper assertions over regular assertions by submitting them first to the engines with `assert -helper`....

-with_proven

Use all proven properties in the same task as assumptions.

This switch offers a convenient way to specify that you want the tool to use all proven properties regardless of the properties you list and the current `proven_directive` setting.

-bg

Run the `prove` command in the background and continue to accept and run new commands.

Note: By default, the `prove` command runs in the foreground. Use this option to override the default and run the proof in the background.

-force

Clear existing proof results for the selected properties before beginning.

See [“clear”](#) on page 453.

JasperGold Apps Command Reference Manual

General Commands

-asserts

Prove only assertions, not covers.

-covers

Prove only covers, not assertions.

-quiet

Use all undriven signals as extra soft low constraints. This is a complement to assume -soft, prove -soft_constraints , and prove -soft_low_constraints that offers more convenience but less control.

-soft_constraints *signal_name_tcl_list* [-extend_region]

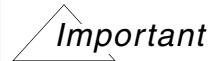
Use the selected signals (those matching any of the list of wildcards) as extra soft constraints.



- Use this switch for signals you prefer to be high in traces. See assume -soft in [“Creating an Assumption”](#) on page 189.
- Use -extend_region to prevent the tool from ignoring out-of-region signals and issuing a warning.
- This switch supports wildcards (*) and (?) in signal names.
- This switch supports Tcl lists as arguments.

-soft_low_constraints *signal_name_tcl_list* [-extend_region]

Use the selected signals (those matching any of the list of wildcards) as extra soft low constraints.



- Use this switch for signals you prefer to be low in traces. See assume -soft in [“Creating an Assumption”](#) on page 189.
- Use -extend_region to prevent the tool from ignoring out-of-region signals and issuing a warning.
- This switch supports wildcards (*) and (?) in signal names.
- This switch supports Tcl lists as arguments.

JasperGold Apps Command Reference Manual

General Commands

-engine_mode *engine_mode_tcl_list*

Override the current engine mode for this prove command.

See “[set_engine_mode](#)” on page 1009.

-iter *N*

Override the current max_trace_length for this prove command.

See “[set_max_trace_length](#)” on page 1114.

Note: You can also use this command to specify a bounded proof, that is, a proof that terminates after reaching the specified iteration.

-first_trace_attempt *N*

Override the current first_trace_attempt for this prove command.

See “[set_first_trace_attempt](#)” on page 1106.

-per_property_max_time_limit *time_limit*

Override the current prove_per_property_max_time_limit for this prove command.

See “[set_prove_per_property_max_time_limit](#)” on page 1173.

-per_property_time_limit *time_limit*

Override the current prove_per_property_time_limit for this prove command.

See “[set_prove_per_property_time_limit](#)” on page 1176.

-per_property_time_limit_factor *N*

Override the current prove_per_property_time_limit_factor for this prove command.

See “[set_prove_per_property_time_limit_factor](#)” on page 1179.

-time_limit *time_limit*

Override the current prove_time_limit for this prove command.

See “[set_prove_time_limit](#)” on page 1189.

JasperGold Apps Command Reference Manual

General Commands

-verbosity *N* | -silent

Override the current prove_verbosity for this prove command.

Note:

- Supported verbosity levels are 0-11. The default verbosity level is 6.
 - See “[set_prove_verbosity](#)” on page 1192.
-

-per_engine_max_jobs *N*

Run the specified number of engine jobs per engine.

Note: This switch overrides proofgrid_per_engine_max_jobs and proofgrid_per_engine_max_local_jobs. See “[set_proofgrid_per_engine_max_jobs](#)” on page 1141 and “[set_proofgrid_per_engine_max_local_jobs](#)” on page 1144.

-no_traces

-no_cover_traces

Do not generate traces or cover traces specifically.

Use this switch to avoid the overhead of generating and storing traces.



- The properties will show cex and covered statuses as usual, but there will be no traces to inspect.
 - You must re-do the proof to get traces.
 - Use this switch when the trace data is not important.
-

-prefer_shortest

Override the current prove_prefer_shortest for this prove command. See “[set_prove_prefer_shortest](#)” on page 1181.

JasperGold Apps Command Reference Manual

General Commands

```
-dump_trace [-dump_trace_type (shm | vcd | fsdb)]  
[-dump_trace_dir dir_name]
```

Dump traces to the disk as soon as the engines find them. The tool uses the FSDB format as the default and writes files in the project directory.

- Use `-dump_trace_type` with `shm`, `vcd`, or `fsdb` to specify the file dump type. VCD files are dumped as compressed files to save disk space.
- Use `-dump_trace_dir` to specify the dump directory where the files will be written.

Switches for a “prove” Command Running in the Background

```
-stop [engine_mode | index.engine_mode]  
[-thread thread_id]
```

Stop the entire `prove` command running in the background or a single `engine_mode` or `engine_job` (`index.engine_mode`).

If the `parallel_proof_mode` variable is on, use `-thread` to stop a proof thread or an `engine_mode` or `engine_job` that is associated with a specified thread.

Note:

- `-stop` is not blocking and may return before the proof has actually stopped. If you want blocking behavior, use `prove -stop; prove -wait`. See the description for [-wait](#) below.
- If the `parallel_proof_mode` variable is on, `prove -stop` stops all running proof threads.

```
-kill index.engine_mode  
[-thread thread_id]
```

Kill the `proofgrid_shell` process for the specified engine job.

If the `parallel_proof_mode` variable is on, use `-thread` to kill a proof thread or an `engine_mode` or `engine_job` that is associated with a specified thread.

Note: Depending on the nature of the `proofgrid_shell` you are using (see [“set_proofgrid_shell”](#) on page 1152), killing a job can have drawbacks. Therefore, you should only use this command as a last resort when `-stop index.engine_mode` does not appear to be working.

JasperGold Apps Command Reference Manual

General Commands

```
-wait
```

```
[-thread thread_id]
```

Wait for the currently running `prove` command to complete before running additional commands.

This option is useful when `prove` is running in the background (`-bg`). In effect, it moves `prove` back to the foreground, and the tool accepts no additional commands until the proof completes.

`prove -wait` returns the value the `prove` command would have returned if it had run without `-bg` (see above). It returns this value even if the previously started proof has already stopped or never ran in the background.

If the `parallel_proof_mode` variable is on, use `-thread` to tell the tool to wait for the `prove` command running under the specified thread to complete before running additional commands.

JasperGold Apps Command Reference Manual

General Commands

```
-status  
[-thread thread_id]  
[-silent]
```

List the running engine jobs and their statuses.

- If the `parallel_proof_mode` variable is on, use `-thread` to list the status of a specified thread's running engine jobs.
- Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

The following list shows prove statuses.

- `NeverStarted`: Nothing has run in this session.
- `Pending`: There is currently a prove about to start.
- `Simplify`: A prove is currently performing proof simplification.
- `Running engineJob+N`: A prove is running with the listed number of `engineJobs` currently running.
- `Completed`: No job is currently running. The last prove has completed.
- `licenseStarved`: A job stopped because, for example, it surrendered a nice license.

Note:

- Refer to “[-nice](#)” on page 72 for additional information.
- Refer to “[set_proofgrid_per_engine_privileged_jobs](#)” on page 1146 to learn how you can specify the number of “non-nice” licenses used by ProofGrid.

Note: If the `parallel_proof_mode` variable is on, use `prove -status` to list the status for all running engine jobs for all proof threads. The tool prefixes statuses with the thread ID.

```
-per_engine_max_jobs N  
[-thread thread_id]
```

Dynamically change the `per_engine_max_jobs` setting for the running `prove` command.

If the `parallel_proof_mode` variable is on, use `-thread` to change the setting for the `prove` that is associated with a specified thread.

JasperGold Apps Command Reference Manual

General Commands

```
-add engine_mode
```

```
[-thread thread_id]
```

Add the specified *engine_mode* to the running `prove` command.

If the `parallel_proof_mode` variable is on, use `-thread` to add the specified *engine_mode* to the `prove` that is associated with a specified thread.

```
-restart engine_mode | index.engine_mode
```

```
[-thread thread_id]
```

Stop and restart the specified engine or engine job.

Use `-thread` to stop and restart the proof thread or an *engine_mode* or *engine_job* that is associated with a specified thread.

```
-ignore_property property_name_tcl_list
```

```
[-thread thread_id]
```

Instruct single-property engines in the running `prove` command to stop processing the specified properties.

If the `parallel_proof_mode` variable is on, use `-thread` to ignore properties only in the specified thread.

Note: If the `parallel_proof_mode` variable is on, `prove -ignore` affects all running proof threads.

Return

When the `prove` command is not running in the background, the tool returns the values listed in [Table 4-4](#) on page 751 below.

Table 4-4 Prove Command Return Values

Return Values	Definition
proven	All properties have the same determined status. The return is one of the values listed.
cex	
unreachable	
covered	
ar_covered	
ar_cex	

Table 4-4 Prove Command Return Values

determined	All properties are fully determined, but they have different statuses (proven, cex, unreachable, covered).
determined_or_skipped	Some properties were skipped (are ignored by the selected proof setting). All other properties are fully determined.
ar_determined	All properties are determined or ar_covered or ar_cex, and at least one property is ar_covered or ar_cex.
overconstrained	At least one task has inconsistent assumptions or clock and reset.
per_property_time_limit time_limit	time_limit or per_property_time_limit has expired.
max_trace_length	max_trace_length (iter) is reached.
stopped_by_user	The prove command was stopped by the user.
out_of_memory	At least one engine ran out of memory.
engineJ_attempts	Engine J failed to satisfy all constraints (see “ set_engineJ_attempts ” on page 1048, “ set_engineJ_restarts ” on page 1054).
internal_limit	Internal limit exceeded.
no_properties	The given selection contained no properties.
background	The prove command is running in the background.
background (thread <i>thread_id</i>)	The prove command is running in the background and is associated with thread <i>thread_id</i> . (Applies only if the parallel_proof_mode variable is on.)
aborted	The prove command stopped due to a ProofGrid error.
failed	The prove command stopped due to an error.

Examples

```
% prove -all
% prove -task taskA -engine D
% prove -task { taskA taskB taskC }
% prove { mon.master_error == 1'b0 }
% prove -instance instA -except subinstB
```

JasperGold Apps Command Reference Manual

General Commands

```
// Specify multiple engines for parallel processing for faster run time.  
% prove -task taskA -engine { d g h }  
  
// Instruct all running single-property engines to stop processing propertyA  
% prove -ignore_property { propertyA }
```

See Also

[get_status](#)
[set_dst_mode](#)
[set_engine_mode](#)
[set_max_trace_length](#)
[set_parallel_proof_mode](#)
[set_proof_simplification](#)
[set_proofgrid_per_engine_max_jobs](#)
[set_proofgrid_per_engine_max_local_jobs](#)
[set_proofgrid_mode](#)
[set_prove_per_property_time_limit](#)
[set_prove_time_limit](#)
[set_proven_directive](#)
[visualize](#)

redirect

Description

Use this command to capture output written to `stdout` by a Tcl command. You can redirect the output into either a file or a Tcl variable.

Note: Output written to `stdout` by background jobs is also redirected. Therefore, avoid using this command while background jobs are running, for example, prove `-bg` and visualize `-bg`.

Syntax

```
redirect (-file <file_name> | -variable <var_name>)
         [-force | -append] [-tee] <command>
```

Argument	Definition
<code>-file file_name [-force -append]</code>	<p>Direct the output to the specified file.</p> <ul style="list-style-type: none">■ Use <code>-force</code> to overwrite an existing file.■ Use <code>-append</code> to append to an existing file.
<code>-variable var_name [-append]</code>	<p>Direct the output to the specified Tcl variable.</p> <p>Use <code>-append</code> to append output to an existing Tcl variable. If the variable already exists and you do not use <code>-append</code>, the tool replaces the existing value with the new value.</p>
<code>[-tee]</code>	In addition to redirecting output to a file or Tcl variable, write the output to the JasperGold Apps console.

Return

The value returned by the specified *command*.

Examples

```
% redirect -file prove-log.txt -force { prove -all }  
% redirect -file visualize-log.txt -force { visualize -violation -property a }
```

See Also

No related commands

remove_tx_attribute

Description

Use the `remove_tx_attribute` command to remove signals from the attribute list of a cover directive that is used to capture a behavior.

Note: This command supports regular expressions in cover names.

Syntax

```
remove_tx_attribute (<cover_name> [-regexp]) <signal_name>
```

Argument	Definition
<code>cover_name</code> [-regexp]	Remove the specified attribute from the specified <code>cover_name</code> .
<code>signal_name</code>	Remove the specified <code>signal_name</code> from the list of attributes associated with the specified <code>cover_name</code> .

Default

No default values

Return

No return value.

Examples

```
% remove_tx_attribute send_packet data
```

See Also

[add_tx_attribute](#)
[get_tx_attributes](#)

report

Description

Use this command to report verification results. These reports contain lists of all properties and their proof status (pass, fail, unproven, and so forth). If you do not include any options with this command, the tool prints out a header and a table of results.

Syntax

```
report [-all [-include_type] [-info] [-summary]
        [-results] [-detailed]
        [-file <file_name>] [-force | -append]]
| [-info] [-summary]
  [-include_type] [-results] [-detailed]
  [-file <file_name>] [-force | -append]]
| [-csv] [-include_type] [-results]
  [-file <file_name> [-force | -append]]

[-task <task_name_tcl_list>]
```

Argument	Definition
<code>[-all [-include_type] [-info] [-summary] [-results] [-detailed] [-file <i>file_name</i> [-force -append]]</code>	<p>Generate a report with all sections, that is, summary, info, results, and details.</p> <ul style="list-style-type: none">■ Use <code>-include_type</code> to include the property type in the report table in a <i>Type</i> column.■ Use <code>-info</code> to include design info: hard stopats, global assumptions, resets, and clocks.■ Use <code>-summary</code> to generate the results summary.■ Use <code>-results</code> to generate the table of results (the default option if you do not provide other switches).

JasperGold Apps Command Reference Manual

General Commands

-all (Continued)

- Use `-detailed` to generate the table of detailed results, including all proof details for each property: associated assumptions, proof bound, and proof engine used.
- Use `-file` to save the report to a specified file instead of displaying it on the screen.
 - Use `-force` if the specified report file already exists and you want to overwrite it.
 - Use `-append` to add the report to the end of the specified file.

```
[-info] [-summary]
[-include_type] [-results] [-detailed]
[-file file_name [-force | -append]]
```

Generate a report with design info: hard stopats, global assumptions, resets, and clocks.

- Use `-summary` to generate the results summary.
- Use `-include_type` to include the property type in the report table in a *Type* column.
- Use `-results` to generate the table of results (the default option if you do not provide other switches).
- Use `-detailed` to generate the table of detailed results, including all proof details for each property: associated assumptions, proof bound, and proof engine used.
- Use `-file` to save the report to a specified file instead of displaying it on the screen.
 - Use `-force` if the specified report file already exists and you want to overwrite it.
 - Use `-append` to add the report to the end of the specified file.

JasperGold Apps Command Reference Manual

General Commands

```
[-csv] [-include_type] [-results]
[-file file_name [-force |-append]]
```

Generate a table of results.

- Use `-csv` to generate the report as comma-separated values (CSV).
- Use `-include_type` to include the property type in the report table in a *Type* column.
- Use `-file` to save the report to a specified file instead of displaying it on the screen.
 - Use `-force` if the specified report file already exists and you want to overwrite it.
 - Use `-append` to add the report to the end of the specified file.

```
-task task_name_tcl_list
```

Generate a report for the specified tasks only.

Note:

- Use this switch with any of the groups of switches above.
- Replace `task_name_list` with a period (.) to specify the current task.
- When specifying a list of task names, enclose the list in curly braces.
- Without this switch, the report includes all tasks.

Return

No value returned

Examples

```
% report
% report -summary -task {taskA taskB}
% report -file report.txt -force -detailed

# generate a report for the current task in CSV format:
% report -task . -csv -file report_[task -set .].csv
```

JasperGold Apps Command Reference Manual

General Commands

See Also

No related commands

reset

Description

Use the `reset` command to specify the reset condition for the design under verification. Reset conditions are always *globally* applied and noncumulative. The `reset` command only considers global stopats and assumptions (specified with `-env`). Task-specific information (including assumption and stopat information) does not affect reset values. The tool fully propagates assumptions that directly force signals to constant values but does not guarantee that more complex assumptions are enforced during reset analysis. The tool runs an implicit clear for each `reset` command.



- Embedded SVA and PSL assumptions are not used during reset analysis unless you elaborate with `-global_embedded_assumes`. When using this switch, reset analysis considers these assumptions provided they are written on primary inputs (including the clock).
- Before you can run reset analysis, including the reset analysis that runs for `prove` and `visualize`, you must specify `reset`. Use one of the following commands:
 - `reset signalName` to declare a reset pin
 - `reset -expression` to use a well-defined expression
 - `reset -sequence` to set a reset sequence file
 - `reset -init_state` to set a reset snapshot file
 - `reset -none` to specify a free-running reset environment

Note:

- If you use `reset -none`, the tool runs reset analysis, so if you do not want to run the analysis, use an empty `reset -init_state` file.
- If you are not ready to run one of the specified commands, use `reset -analyze` to help set up the reset environment properly.
- During reset analysis simulation, the tool prints a message after advancing 10 percent of the iterations, and it continues to print messages at every successive 10 percent increment. For example, if the maximum number of iterations in simulation is 100, a

message prints every 10 iterations until the iteration of convergence (that is, less than or equal to the maximum number of iterations).

There are five primary ways to specify reset conditions:

1. A set of pin constraint expressions
2. A reset sequence in a file
3. A VCD file (typically from the results of simulation) with the reset values for registers
4. An FSDB file
5. An SHM file
6. An initial state file with the reset values for a specific set of registers

You may also adjust the values for specific registers by providing the following commands:

- `reset -non_resettable_regs`: all undefined initial values (but not those from `abstract -init_value`) will assume the specified initial value
- `abstract -init_value`: force the registers to have undefined initial values

With two exceptions, you can use these options in any combination, but at least one of the options from 1 to 6 must be present.

Exceptions:

- Options 1 and 2 above are mutually exclusive; however, you can postfix expressions to the sequence file. (Refer to the `-sequence` definition below.)
- The options `-fsdb`, `-vcd`, and `-shm` are mutually exclusive.

To display the reset sequence as a prefix in the Visualize window (with negative cycle numbers), click *View – Show Reset Cycles*.

Syntax

```
reset [[ -expression <pin_constraint>+)
        [-max_iterations <N>] [-clock <clock_signal>]
        |-sequence (-vcd [-include_internal] | -fsdb | -shm) <file_name>
          [-start_time <time>] [-time <time>]
          [-time_scale (ms | us | ns | ps)]
          [-hier_path <path_difference>
            |-hier_map -from <source_hier> -to <destination_hier>]
          |-sequence <seq_file_name>] [-disable_wildcard] [-include_internal]
        [(-vcd [-include_internal] | -fsdb | -shm) <file_name> [-time <time>
          [-time_scale (ms | us | ns | ps)]]
```

JasperGold Apps Command Reference Manual

General Commands

```
[ -hier_path <path_difference>
  |-hier_map -from <source_hier> -to <destination_hier>]]
[-init_state <file_name>] [-disable_wildcard]]+
[-non_resettable_regs <value>]

reset -analyze [-synchronous] [-list flop] [-silent]
reset -none
reset -list [signal] [-typed_list] [-silent]
reset -clear
```

Argument	Definition
<code>-expression pin_constraint+ [-max_iterations N] [-clock clock_signal]</code>	<p>A set of pins (input or global stopat net) that are asserted when the design is being reset by the tool and de-asserted when the design is being formally verified.</p> <p>Reset analysis runs until flop values stop changing or until it reaches 100 iterations.</p> <p>Use <code>-max_iterations</code> with an integer to change the default. And use <code>-clock</code> to specify that you want the number of iterations based on the specified clock.</p> <p>Note: The specified clock must be a declared clock or a propagated clock signal.</p> <p>Note:</p> <ul style="list-style-type: none">■ The tool currently only supports pin constraints that force an input to have a constant value, for example, <code>rst</code>, <code>~rst</code>, <code>a==1'b1</code>, <code>a==1'b0</code>, and so forth.■ <code>-expression</code> and <code>-sequence</code> are mutually exclusive; however, you can postfix expressions to the sequence file.■ Without the <code>-expression</code> switch, all signals behave as free inputs during the proof or visualization.

JasperGold Apps Command Reference Manual

General Commands

```
-sequence (-vcd [-include_internal] |-fsm |-shm) file_name  
[-start_time time] [-time time] [-time_scale (ms | us | ns | ps)]  
[-hier_path path_difference]  
[-hier_map -from source_hier -to destination_hier]]
```

Get reset values of registers from the specified sequence file.

All the reset values will be defined by simulating this sequence for the specified interval, or for the entire trace length. This is done by reading values for all PIs, undriven signals and initial values for registers, and applying these as inputs to the full design simulation.

- Use `-vcd`, `-fsdb`, or `-shm` to specify the type of trace to be interpreted. And use `-vcd -include_internal` to load internal signals from the a VCD file.

Note:

- `reset -sequence -vcd -include_internal` carries a risk of a mismatch in internal signals that can lead to false proof results in some cases. Internal signals in reset files might not be the same as those generated by a different JasperGold release version or a different version of the design or a different design setup and configuration. (Design setup and configuration differences include the order in which files are passed to `analyze` and `elaborate` and the order in which assumptions and assertions are created.)
- `reset... -shm` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.
- Use `-start_time` to specify an integer representing the time point where the tool should start reading the trace. The time is proportional to the timescale defined with `timescale in the testbench. If you do not specify a time, the tool starts from time 0.
- Use `-time` to specify an integer representing the time point where the tool should stop reading the trace. The time is proportional to the timescale defined with `timescale in the testbench. If you do not specify a time, the tool will read the file to the end of the trace.
- Use `-time_scale` with the unit of measure (ms, us, ns, or ps) to specify the time scale. If you do not specify a time scale, the tool uses the default from the trace file.

-sequence (Continued)

- Trace files are usually generated during simulation where the testbench module is the top module for the verification run, and consequently, for the trace file too. When using a file generated by simulation to reset your design in JasperGold Apps, use the optional `-hier_path` switch to specify the hierarchical path difference from the testbench module to the design top module.

For example, if the design top module is `dut_top`, which is instantiated as the top module in the testbench `testbench_top` as `i_dut_top`, use the following command to reset the design with the trace file (replace `file_name` with actual value):

```
reset -sequence -vcd <file_name> -hier_path  
testbench_top.i_dut_top  
reset -sequence -fsdb <file_name> -hier_path  
testbench_top.i_dut_top  
reset -sequence -shm <file_name> -hier_path  
testbench_top.i_dut_top
```

- Use `-hier_map` for trace files generated during simulation if you have a wrapper module.
 - Use `-from` to specify the source (simulation environment) hierarchy.
 - Use `-to` to specify the destination (formal environment) hierarchy.

JasperGold Apps Command Reference Manual

General Commands

```
-sequence seq_file_name [-disable_wildcard] [-include_internal]
```

Get reset values of registers from the specified sequence file.

- Use `-disable_wildcard` to turn off the wildcard matching feature when reading sequence files to (potentially) read files faster.
- Use `-include_internal` to load internal signals from the reset file.

All the reset values will be defined by simulating this sequence for the specified number of iterations.

Note:

- `reset -sequence -vcd -include_internal` carries a risk of a mismatch in internal signals that can lead to false proof results in some cases. Internal signals in reset files might not be the same as those generated by a different JasperGold release version or a different version of the design or a different design setup and configuration. (Design setup and configuration differences include the order in which files are passed to `analyze` and `elaborate` and the order in which assumptions and assertions are created.)
- `-expression` and `-sequence` are mutually exclusive; however, you can add expressions to the sequence file as “postfix” information. This feature makes it possible to specify the inputs that should be forced to a certain value (to “de-assert” the reset signal) during `prove` and `Visualize` analysis.

If you use the `$past/prev` operators, this simulation will be used as the pre-verification history to define initial values.

Refer to [“Sequence File Format, Guidance, and Examples”](#) on page 773.

JasperGold Apps Command Reference Manual

General Commands

```
(-vcd [-include_internal] |-fsdb |-shm) file_name -time time
-time_scale (ms | us | ns | ps)
-hier_path path_difference
-hier_map -from source_hier -to destination_hier
```

Get reset values of registers from the specified file.

- Use `-vcd`, `-fsdb`, or `-shm` to specify the type trace to be interpreted. And use `-vcd -include_internal` to load internal signals from the a reset file.

Note:

- `reset -vcd -include_internal` carries a risk of a mismatch in internal signals that can lead to false proof results in some cases. Internal signals in reset files might not be the same as those generated by a different JasperGold release version or a different version of the design or a different design setup and configuration. (Design setup and configuration differences include the order in which files are passed to `analyze` and `elaborate` and the order in which assumptions and assertions are created.)
- `reset... -shm` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments.
- Use `-time` to specify an integer representing the time point where the signal values are sampled. The time is proportional to the timescale defined with `timescale in the testbench.

If you do not specify a time, the tool uses the last time in the file.

- Use `-time_scale` with the unit of measure (ms, us, ns, or ps) to specify the time scale. If you do not specify a time scale, the tool uses the default from the trace file.
- Use `-hier_path` to specify the hierarchical path difference from the testbench module to the design top module.

Trace files are usually generated during simulation where the testbench module is the top module for the verification run, and consequently, for the trace file too. This switch is helpful when you are using a file generated by simulation to reset your design in JasperGold Apps.

JasperGold Apps Command Reference Manual

General Commands

-vcd | -fsdb... (Continued)

For example, if the design top module is `dut_top`, which is instantiated as the top module in the testbench `testbench_top` as `i_dut_top`, use the following JasperGold Apps command to reset the design with the trace file (replace `file_name` and `time` with actual values):

```
reset -vcd <file_name> -time <time> -hier_path  
testbench_top.i_dut_top  
reset -fsdb <file_name> -time <time> -hier_path  
testbench_top.i_dut_top  
reset -shm <file_name> -time <time> -hier_path  
testbench_top.i_dut_top
```

- Use `-hier_map` for trace files generated during simulation if you have a wrapper module.
 - Use `-from` to specify the source (simulation environment) hierarchy.
 - Use `-to` to specify the destination (formal environment) hierarchy.
 - Use `reset -sequence (-vcd | -fsdb | -shm) trace_filename` to use the inputs from the specified trace file for reset simulation.
-

JasperGold Apps Command Reference Manual

General Commands

```
-init_state file_name [-disable_wildcard]
```

Get reset values of registers from the specified file.

Use `-disable_wildcard` to turn off the wildcard matching feature when reading sequence files to (potentially) read files faster.



- Any undefined value not resulting from abstract `-init_value` will be filled with the provided value. If there is inconsistency, you will be warned and the value in the file will be discarded.
- The tool supports both Verilog and VHDL constant values except for VHDL decimal.
- The tool supports encrypted files saved with `reset -save_values -include_encrypted` and `visualize -save -init_state -include_encrypted`.
- The tool supports the IEEE definition of variable names, including escaped names.
- The tool supports wildcards (*) and (?) in `reset -init_state` files (see “[Examples](#)” on page 773 section below). However, do not attempt to use wildcards for escaped names. The tool interprets them as literals.

Note: Use `-disable_wildcard` to disable this feature.

- Value can be either a numeric constant or an enum.
- Prefix your comments with #. The tool recognizes the end of the line as the end of the comment.
- Signals with an unspecified bus size are interpreted as decimal, for example:

```
signal_A[3:0]
10
signal_A has 4 bits, 10 is interpreted as 4'd10 (4'b1010).
```

JasperGold Apps Command Reference Manual

General Commands

-init_state (Continued)

- Signals with an incorrect width are truncated or extended, for example:

```
signal_A[1:0]  
4'b1001  
signal_B[3:0]  
1'b1  
signal_A takes the value 2'b01  
signal_B takes the value 4'b0001
```

Refer to [“Reset from Initial State File – Format and Examples”](#) on page 776.

-non_resettable_regs *value*

Set the reset values of registers with the specified *value*.

Any undefined value that did not result from using abstract -init_value will be filled with the specified *value*. (*value* must be 0 or 1.)

-analyze

```
[-synchronous]  
[-list flop]  
[-silent]
```

Analyze the design’s reset logic and return information about simple asynchronous reset signals (driven by signal) and complex asynchronous reset signals if they are driven by an AND-gated tree.

- Use -synchronous to also return information about synchronous signals.
 - Use -list flop to list the design flops that are connected to a reset signal.
 - Use -silent to return the list as Tcl return values without printing the list on the screen.
-

JasperGold Apps Command Reference Manual

General Commands

-analyze (Continued)

The return value is a list of the reset signals (negated if they are active-low). If you use `-list flop`, the return inserts a list containing the design flops connected to each reset signal with the number of bits, for example:

```
{~rst Async Simple 6 {{a[1:0]} {a[3]} b} 0}
```

In this example, 6 is the total number of bits (3 from `a` and 3 from `b`).

The reset report includes the following information:

- Signal name – if the inferred reset is driven by an expression and represented by an internal signal, the report displays the internal signal's fanin instead of the internal signal's name.
- Type
 - Simple reset – driven by signal.
 - Complex reset – driven by an AND-gated tree.
- Design flops connected – Number of flops in the design that are affected by this signal. For example:

```
Design flops connected 2 (6) : a[1:0] a[3] b
```

In this example, 2 is the number of nets (`a` and `b`) connected to the reset signal, and 6 is the total number of bits (3 from `a` and 3 from `b`).

- Number of PSL/SVA flops connected – Number of flops generated as observers for SVA and PSL properties that are affected by this signal.

JasperGold Apps Command Reference Manual

General Commands

-none

Run the reset flow with an empty list of reset signals. This means:

- Run constant propagation for design constants and environment assumptions.
- Run a simulation until it converges (at most 100 iterations).

Note:

- This command can be helpful when you intend to Visualize and debug reset logic.
- Be aware that if you use `reset -none...`
 - The tool runs a basic reset simulation and combinational constant propagation (which takes into account the global assumptions, global stopats, and design constants). And as a result, there will be at least one clock iteration of reset prior to the formal analysis, and any non-reset registers that are tied to a constant will take on that constant value.
 - Reset simulation considers all user-defined clocks *and* uses the propagated constants derived from the global environment and design constants. If you do not define clocks, the simulation will be clocked by only the internal fastest clock.
 - Defining the clocks can impact reset analysis since they can trigger many registers during reset simulation.

Utility Switches

`-list [signal]
[-typed_list]
[-silent]`

Show the current list of reset conditions on the screen.

- Use `signal` to list only reset signal names associated with the reset expression(s) in the Tcl return.
- Use `-typed_list` to add a type element before each of the requested lists.

Refer to [“List File Format and Example”](#) on page 776.
- Use `-silent` to turn off output to the screen and return information with Tcl return values. This option is helpful in Tcl scripts.

JasperGold Apps Command Reference Manual

General Commands

```
-clear
```

Remove all previously specified global reset configurations.

Return

No value returned

Examples

```
% reset -expression ~rst1N rst2
% reset -expression ~rst1N rst2 -max_iterations 150

// Using wildcards in init_state files. Consider the following init_state
// declaration:
    sigA 4'b0
    sigB 4'b0
    sigC 4'b0

// You can declare the following:
    sig* 4'b0

// And if you want to set a value for every signal/input in the design, do the
// following:
    * 0
```

Sequence File Format, Guidance, and Examples

The following model shows the format for a sequence file, including postfix information.

```
Line 1: Signal_Name Value
Line 2: Signal_Name Value
Line 3: Number of clock ticks
Line 4: Signal_Name Value
...
Line n: Number of clock ticks
#### (Optional) Postfix information
#### Postfix information is the section that contains a list of signal
#### names and values between the last clock ticks specified and $.
Line m: Signal_Name
```

JasperGold Apps Command Reference Manual

General Commands

```
Line m+1: Value
...
Line m+2: $
```

The following information is implementation guidance for using a sequence file:

- The initial definition of signals (before any clock tick definition) may contain both input values (primary input or hard stopats) and flop reset values.
- Clock ticks are relative to the fastest clock.
- If the signal appears in one iteration and not in the following iterations, its value will be maintained in the interval.
- The simulation will iterate until the final iteration number. If there is no such definition, it will iterate until the last iteration where signals were defined.
- The format of a line that contains a signal and a value must be strictly `<Signal_Name><Value>`.
- A `Value` can be either a numeric constant or an enum value.
- The maximum size of a line in a reset file should be 64k characters.
- The tool supports wildcards (*) and (?) in `reset -sequence` files. However, do not attempt to use wildcards for escaped names. The tool interprets them as literals.

Note: Use `-disable_wildcard` to disable this feature.

- The postfix signal values are used as global assumptions to be observed during the proof phase. The tool supports single-bit and wide signals. The following is an example of wide signals in the postfix section:

```
WReset 2'b10
2
WReset 2'b11
2
WReset 2'b00
$
```

- Prefix your comments with #. The tool recognizes the end of the line as the end of the comment.
- The tool supports binary, octal, hexadecimal, decimal, and signed decimal values. It is possible to include more than one radix in a file, as shown below.

```
in_9bit 9'b001101001
# 9'b1101001 is interpreted as 9'b001101001
in_9bit 9'b1101001
in_9bit 9'h6_9
rstN 3'o0 # octal number
in_9bit 9'o151
in_9bit 9'd105 # decimal number
in_9bit 9'b001x01001
# 9'bx1010x1 is interpreted as 9'bxxx1010x1
in_9bit 9'bx1010x1
```

- You can also mix radices in one signal:

```
# signal_16bits[0:15]
signal_16bits[0:7] 8'b010101xx
signal_16bits[8:15] 8'd5
```

The parser only accepts unsigned numbers and it will not accept definitions where:

- A constant is bigger than the signal (signal_16bits 17'd0)
- A signal is in different order from the design definition:

```
# signal_16bits[0:15]
signal_16bits[12:5] 8'd0
```

The following is an example of a sequence file.

```
counter 2'b00
resetN 1'b1
input_a 2'b10
1
input_a 8'b00
2
resetN 1'b0
2
#postfix information
resetN 1'b1
$
```

JasperGold Apps Command Reference Manual

General Commands

The simulation for the example above takes five iterations, and the values are:

Iteration	1	2	3	4	5
counter	00	(simulation --> this is a flop)			
resetN	1	1	1	0	0
input_a	10	00	00	00	00

Reset from Initial State File – Format and Examples

Format your file as follows:

```
Line 1: <Signal_A>
Line 2: <Value>
Line 3: <Signal_B>
Line 4: <Value>
...

```

Example 1:

```
wb_dat_o
8'b11000001
prer
16'b10101010_10101010
...

```

Or format your file as follows:

```
Line 1: <Signal_A> <Value>
Line 2: <Signal_B> <Value>
...

```

Example 2:

```
wb_dat_o 8'b11000001
prer 16'b10101010_10101010
...

```

List File Format and Example

The following model shows the format for a typed list file.

JasperGold Apps Command Reference Manual

General Commands

```
$resetConfig(none) {true/false <is reset none>}  
$resetConfig(expression) {<list of reset expressions>}  
$resetConfig(file_type) {<file type>}  
$resetConfig(sequence) {true/false <is simulated reset trace from file>}  
$resetConfig(file_name) {<file name>}  
$resetConfig(time) {<vcd/fsdb time>}  
$resetConfig(start_time) {<simulated vcd/fsdb start_time>}  
$resetConfig(end_time) {<simulated vcd/fsdb end_time>}  
$resetConfig(hier_path) {<vcd/fsdb hier path>}  
$resetConfig(time_scale) {<vcd/fsdb time scale>}  
$resetConfig(max_iterations) {<max iteration config>}  
$resetConfig(non_resettable_regs) {<resettable regs value>}  
$resetConfig(abstract_value) {<abstract bits>}  
$resetConfig(resetDeclared) {true/false <any reset declared>}
```

Note: Each one of these fields is empty by default ({}) except for \$resetConfig(*none*) and \$resetConfig(*reset_declared*), which are false or true, depending on the reset configuration.

Example:

```
% reset -list -typed_list  
reset expressions:  
-----  
0: rst  
-----  
  
expression rst max_iterations {} file_type {} sequence {} file_name {} hier_path  
{} time {} start_time {} end_time {} time_scale {} non_resettable_regs {}  
abstract_value {} none false reset_declared true
```

Reset from a VCD File Generated by Simulation with a Wrapper Module

```
# Simulation environment:  
# SimulationEnv.TestBench.<Instances_I_want_to_verify>  
  
# Formal environment:  
# Formal_wrapper.<Instances_I_want_to_verify>  
  
# Use the following command:
```

JasperGold Apps Command Reference Manual

General Commands

```
% reset -vcd my_vcd -hier_map -from SimulationEnv.TestBench -to Formal_wrapper  
% reset -fsdb my_fsdb -hier_map -from SimulationEnv.TestBench -to Formal_wrapper  
% reset -shm my_shm -hier_map -from SimulationEnv.TestBench -to Formal_wrapper
```

See Also

[assume](#)
[clock](#)
[get_reset_info](#)
[prove](#)
[sanity_check](#)
[set_trace_show_reset](#)
[stopat](#)

restore

Description

Use `restore` to automatically read all design and requirement files to restore a previously saved session. If the files are not present in the original directory, the tool prints an error message and ignores the command.

Note:

- If you use the `-db` option, the tool only loads the database.
- Do not attempt to restore a database while a proof is in progress.

JasperGold Apps Session Management Tip. You can run the command `save -dir` while a proof is running in the background (not when one is running in the foreground). You can then run `restore -dir` to restore the database in a different JasperGold Apps session and start debugging while the proof is still running in the original session.

Syntax

```
restore [-db] [-dir <database_directory>] [-force]
restore -jdb <jasper_database>
    [(-define <tcl_variable> <tcl_variable_value>) *
     [-update_all_sessions]]
restore -elaborated_design <file_name>
```

Argument	Definition
no arguments	<p>If you do not specify arguments, the tool attempts to restore <code>[get_proj_dir]/auto_saved_db</code>.</p> <p>For tools that do not have an automatically saved session (for example, JasperGold), you must use the <code>-dir</code> switch.</p>

JasperGold Apps Command Reference Manual

General Commands

```
[-db] [-dir database_directory] [-force]
```

Restore the specified session that was saved by the `save` command.

- Use `-db` to upload the database information without reading the RTL and requirement files again. If the current RTL or requirement setup is inconsistent with the database, the tool prints an error message and ignores the command.
- Use `-dir` to restore a previous session from the specified directory.
- Use `-force` if you do not want to perform the checksum comparison before restoring the specified database.

Note:

- When you use the `save -dir` command, the tool performs a checksum of all opened files. When you attempt to restore a database, the tool checks open files and files in the checksum file against the saved checksum.
- Understand that if you use the `-force` switch, there is no guarantee of a successful outcome.

```
-jdb jasper_database
```

```
[(-define tcl_variable tcl_variable_value)+]  
[-update_all_sessions]]
```

Load the database saved by `save -jdb`.

This is the primary mode of saving and restoring work in progress with JasperGold Apps.

- Use `-define` to specify variables and their values, which the tool adds to the setup script of the current session.
- Use `-update_all_sessions` to propagate the change to all sessions. The tool then refreshes the sessions.

```
-elaborated_design file_name
```

Load the specified elaborated design.

Note: `restore -elaborated_design` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

Return

No value returned

Examples

The following are context examples.

```
% jaspergold      # Start a new session  
...  
% save -dir db_dir  
% exit  
% jaspergold      # Start a new session  
% restore -dir db_dir  
  
% mkdir session_123b # Before you begin your session, make a directory  
% jaspergold      # Start a new session  
...  
% save -dir session_123b  
% exit  
% jaspergold      # Start a new session  
% restore -dir session_123b  
...  
% restore -jdb mydb.jdb -define var "value" -update_all_sessions  
% restore -elaborated_design design.jnl
```

See Also

[get_proj_dir](#)
[save](#)

sanity_check

Description

Use the `sanity_check` command to help set up the global environment. `sanity_check` is an interactive command, that is, you will potentially run several iterations of it. Use each run to clean up certain issues in clock or reset setup. When the command no longer returns problematic signals and suggestions, you have successfully set up your environment.

This command covers simple clock, unintended constant clock, gated clock, complex clock, D-flop clocks, and simple reset configurations. You can also use this command to waive intended constant clocks, gated clocks, D-flop clocks, and undeclared clocks and resets.

Syntax

```
sanity_check -analyze
  [ simple_clk | inactive_clk | gated_clk | complex_clk | -dflop_clk
  | simple_reset | all]
  [-ignore_latches]
  [-task task_name]
  [-verbose | -silent]

sanity_check -waiver
  ( ( -add | -remove) <signal_name>
    (-type ( simple_clk | inactive_clk | gated_clk | -dflop_clk
              | simple_reset | all))
    |-list
    |-clear
  )

sanity_check -auto
  (simple_clk | inactive_clk | gated_clk | complex_clk | dflop_clk | all)
  -set_severity_level (ignore | info | warning | error)

sanity_check -auto -disable
  (simple_clk | inactive_clk | gated_clk | complex_clk | dflop_clk | all)

sanity_check -auto
  (simple_clk | inactive_clk | gated_clk | complex_clk | dflop_clk | all)
  -list
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<pre>-analyze (simple_clk inactive_clk gated_clk complex_clk dflop_clk simple_reset all) [-ignore_latches] [-task task_name] [-verbose -silent]</pre>	<p>Return a list of problematic items that were not appropriately treated during the environment setup.</p> <ul style="list-style-type: none">■ Use <code>simple_clk</code> to direct the analysis to consider undeclared simple clock signals. These include primary inputs, undriven signals due to stopats, inconsistent clock edge declarations, and clocking signals that come from a black-box output or registers (possibly clock divider).■ Use <code>inactive_clk</code> to direct the analysis to consider unintended constant clocks.■ Use <code>gated_clk</code> to consider the following:<ul style="list-style-type: none">□ Undefined gated clocks, that is, AND/OR gated clocks without a declared clock□ Possible glitches detected in gated clocks with problematic enable clock configurations■ Use <code>complex_clk</code> to consider complex clocks.■ Use <code>dflop_clk</code> to detect flops whose input pin logic involves the flop clocking signal.■ Use <code>simple_reset</code> to consider undefined simple reset signals, undefined simple AND reset expressions, and undefined reset polarities, including asynchronous and synchronous resets.■ Use <code>all</code> to consider all of the types listed above except <code>dflop_clock</code>.

Continued below.

JasperGold Apps Command Reference Manual

General Commands

-analyze (Continued)

- Note:** This command returns the result for each clock analysis type separately, in an iterative way, in the following sequence: simple reset, simple clock, inactive clock, gated clock, and complex clock.
- Use `-ignore_latches` to exclude all latches from the sanity check analysis.
 - Note:** This switch is only allowed with `sanity_check -analyze gated_clk` and `sanity_check -analyze all`.
 - Use `-task` to run the analysis for the specified task. If you do not specify a task, `sanity_check` runs the analysis for the global environment configuration.
 - Use `-verbose` to report the problematic signals followed by suggested actions.
 - Use `-silent` to hide the reporting information and issue only the Tcl return that includes the problematic signals.

-waiver

```
( (-add |-remove) signal_name
  (-type
    ( simple_clk | inactive_clk | gated_clk
      | dflop_clk | simple_reset | all)
  )
  |-list
  |-clear
)
```

Perform the specified action on waivers or get the specified information.

- Use `-add` to add the specified signal with the specified waiver type to the waiver list.
- Use `-remove` to remove the specified signal with the specified waiver type from the waiver list.
- Use `-list` to return a list of all waived signals and their types.
- Use `-clear` to remove all waivers.

JasperGold Apps Command Reference Manual

General Commands

```
-auto
( simple_clk | inactive_clk | gated_clk | complex_clk
| dflop_clk | all)
-set_severity_level (ignore | info | warning | error)
```

Run the clock sanity check analysis during the proof flow (that is, while running `prove` and `visualize` commands) and set the message severity level for the issues found.

- Use one of the following to specify the type of clock analysis you want `sanity_check` to run:
 - `simple_clk`
 - `inactive_clk`
 - `gated_clk`
 - `complex_clk`
 - `dflop_clk`
 - `all`
- Use `-set_severity_level` to set the severity level for messages generated by the sanity check.

Note: You can also use the command `set_message` to change the severity levels for the related messages.

```
-auto -disable
( simple_clk | inactive_clk | gated_clk | complex_clk
| dflop_clk | all)
```

Disable a specified sanity check analysis.

```
-auto
( simple_clk | inactive_clk | gated_clk | complex_clk
| dflop_clk | all)
-list
```

List the severity level set for the specified type of clock analysis.

Note: Use `all` to list all levels.

Return

This command returns the following:

- For `-analyze`, this command returns the problematic signals followed by suggested actions.
- For `-waiver`, this command returns the user-specified waiver type.
- For `-auto`, there is no return value.

Examples

```
% sanity_check -analyze inactive_clk  
  
// Adding an intended constant clock to the waivers list:  
% sanity_check -waiver -add sigConst -type inactive_clk  
% sanity_check -waiver -remove sigConst -type inactive_clk  
% sanity_check -waiver -list
```

See Also

[clock](#)
[reset](#)

save

Description

Use `save` to save the current session or generate a Tcl script for regression proofs.

When you use the `save -dir` command, the tool performs a checksum of all opened files. When you attempt to restore a database, the tool checks opened files and files in the checksum file against the saved checksum. Refer to the documentation for `restore -force` for situations when you might not want the tool to perform the checksum comparison before restoring the specified database.

JasperGold Apps Session Management Tip. You can run the command `save -dir` while a proof is running in the background (not when one is running in the foreground). You can then run `restore -dir` to restore the database in a different JasperGold Apps session and start debugging while the proof is still running in the original session.

Syntax

```
save ( (-jdb <database>
        [-capture_setup |-import_setup <script>]
        [-clear_session_data |-capture_session_data]
        [-clear_dsc_cache])
      | (-dir <database_directory>
        [-elaborated_design <file_name> [-force]])
      | (-script <file_name>
        [-force]
        [-check]
        [-task <task_name>]
        [-ar]
        [-prove |-prove_task <task_name> |-prove_property <property_name>]
        [-sst]
        [-elaborated_design <file_name> [-force]])
      | (-elaborated_design <file_name> [-force]))
    )
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<pre>-jdb database [-capture_setup -import_setup <i>script</i>] [-clear_session_data -capture_session_data] [-clear_dsc_cache]</pre>	<p>Save the current JasperGold Apps database as a single tar file.</p> <p>This is the primary mode of saving and restoring your work with JasperGold Apps.</p> <ul style="list-style-type: none">■ Use <code>-capture_setup</code> to extract the setup script from the analysis session into the database as the default setup script before saving the JasperGold Apps database.■ Use <code>-import_setup</code> to copy a specified script into the database as the default setup script before saving the JasperGold Apps database.■ Use <code>-clear_session_data</code> to clear the session data (the previous <code>-capture_session_data</code> operation) in the database before saving the JasperGold Apps database.■ Use <code>-capture_session_data</code> to capture the data, including proof results, into the database.■ Use <code>-clear_dsc_cache</code> to clear the trace-based Design Space Coverage cache in the database before saving the JasperGold Apps database.

JasperGold Apps Command Reference Manual

General Commands

```
-dir database_directory  
[-elaborated_design file_name [-force]]
```

Save the current database in the specified directory.

Use `-elaborated_design` to copy the elaborated design to the specified file in the database. And use `-force` to overwrite any existing file that has the specified name.

`-elaborated_design` makes it possible to restore an elaborated design without re-elaborating it.

CAUTION: For this switch to work, you must set the variable `capture_elaborated_design` to `on` before elaboration. For additional details, see [“capture_elaborated_design”](#) on page 964.

Note: `save -elaborated_design` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

JasperGold Apps Command Reference Manual

General Commands

```
-script file_name [-force] [-check] [-task task_name] [-ar]
[-prove | -prove_task task_name | -prove_property property_name]
[-sst] [-elaborated_design file_name [-force]]
```

Create a Tcl file for restoring the analysis state and checking proofs.

Note: You can read in the Tcl file with the `Tcl source` or `include` command. The script also expands the `prove` command into separate justifications, if applicable, for regression testing. The tool writes the file to the directory in which you invoked it.

- Use `-force` to overwrite any existing file that has the specified name.
- Use `-check` when using one of the following options, `-prove`, `-prove_task`, or `-prove_property`, to add a `check_code` command for each property that has a valid result.
- Use `-task` to only generate Tcl commands to recreate the specified task.

Note:

- If you use this option with `-prove`, this command only generates `prove` commands for the specified *task_name*.
- Replace *task_name* with a period (.) to specify the current task.
- Use `-ar` to only generate Tcl commands to set up analysis region information.

Note: If you use this option, the tool does not save commands for reading in HDL files.

- Use `-prove` to add a `prove` command that proves all properties.
- Use `-prove_task` to add a `prove` command that proves the properties in the specified task.

-script (Continued)

- Use -prove_property to add a prove command that proves the specified property.
- Use -sst to save an SST script that will reproduce the current state with helper assertions and so forth.
- Use -elaborated_design to copy the elaborated design to the specified file in the database. And use -force to overwrite any existing file that has the specified name.
-elaborated_design makes it possible to restore an elaborated design without re-elaborating it.

CAUTION: For this switch to work, you must set the variable capture_elaborated_design to on before elaboration. For additional details, see “[capture_elaborated_design](#)” on page 964.

Note: save -elaborated_design is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

-elaborated_design *file_name* [-force]

Copy the elaborated design to the specified file. This command makes it possible to restore an elaborated design without re-elaborating it.

Use -force to overwrite any existing file that has the specified name.

CAUTION: For this switch to work, you must set the variable capture_elaborated_design to on before elaboration. For additional details, see “[capture_elaborated_design](#)” on page 964.

Note: save -elaborated_design is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

Return

No value returned

Examples

% save

JasperGold Apps Command Reference Manual

General Commands

```
% save -script my_proof  
% save -force -script my_proof  
% save -jdb my_db.jdb -capture_setup  
% save -elaborated_design design.jnl
```

The following are context examples.

```
> jaspergold      # Start a new session  
...  
% save -dir db_dir  
% exit  
> jaspergold      # Start a new session  
% restore -dir db_dir  
  
% mkdir session_123b # Before you begin your session, make a directory  
> jaspergold      # Start a new session  
...  
% save -dir session_123b  
% exit  
> jaspergold      # Start a new session  
% restore -dir session_123b
```

See Also

[get_proj_dir](#)
[restore](#)
[set_capture_elaborated_design](#)

scan

Description

Use `scan` to scan and extract property candidates from simulation traces.

Note: If you use `scan -trace` with no additional switches, the tool scans traces previously specified with `waveform -import`.

Syntax

```
scan -trace
  [-fsdb | -vcd [-vcd_lang (vhdl | verilog)]] (<file>)+
  | [-shm (<directory>)+]
  [-hier_path <path>]
  [-start_time <integer>]
  [-end_time <integer>]
  [-scope <id_tcl_list>]
  [-seek <seek_mode>]
  [-per_scope_time_limit <time_limit>]
  [-verbosity <verbosity_level>]
  [-silent]

scan -trace
  [-ddk ] (<file>)+
  [-scope <id_tcl_list>]
  [-seek <seek_mode>]
  [-per_scope_time_limit <time_limit>]z
  [-verbosity <verbosity_level>]
  [-silent]

scan -no_sim
  [-task <task_name>]
  [-from_fsms]
  [-from_property <property_list>]
  [-scope <id_tcl_list>]
  [-seek <seek_mode>]
  [-time_limit <time_limit>]
  [-dump_vcd <path>]

scan -no_sim -from_formal_search
  [-task <task_name>]
  [-depth <N>]
  [-seek <seek_mode_2>]
  [-time_limit <time_limit>]

scan -merge <jdb_file_list>
  [-min_confirm <N>]

scan -clear
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<pre>-trace [-fsdb -vcd [-vcd_lang (vhdl verilog)] file+ [-shm directory+] [-hier_path path] [-start_time integer] [-end_time integer] [-scope id_tcl_list] [-seek seek_mode] [-per_scope_time_limit time_limit] [-verbosity verbosity_level] [-silent]</pre>	<p>Scan VCD, FSDB, or SHM trace files produced by a previous simulation run.</p> <p>This command accepts one or more switches followed by one or more file names. You can also use one or more file names without the <code>-fsdb</code> or <code>-vcd</code> switch as long as the file name includes either the <code>.fsdb</code> or <code>.vcd</code> extension.</p> <ul style="list-style-type: none">■ Use <code>-shm</code> to import SHM directories that contain SST2 databases.■ Use <code>-vcd_lang</code> followed by either <code>vhdl</code> or <code>verilog</code> to specify the language used in a scanned VCD file.■ Use <code>-hier_path</code> to capture the hierarchical difference between the module that was simulated and the module that is under test.■ Use <code>-start_time</code> to skip the initial chunk of simulation data. Typically, you will use this switch to skip the reset sequence and so forth.■ Use <code>-end_time</code> to skip the final chunk of simulation data.■ Use <code>-scope</code> to target the specified POIs.■ Use <code>-seek</code> to configure the property synthesis algorithm. This switch accepts a Tcl list.

JasperGold Apps Command Reference Manual

General Commands

-trace (Continued)

- Use `-per_scope_time_limit` to set a time limit for the scan operation.
- Use `-verbosity` to control the amount of information printed to the log and console. And use `-silent` to turn off output to the screen.

Note: See “[set_scan_verbosity](#)” on page 1219 for additional information.

Note: To further manipulate waveforms imported from trace files or PLI simulation runs, see “[waveform](#)” on page 190.

-trace

```
[-ddk] file+
[-scope id_tcl_list]
[-seek seek_mode]
[-per_scope_time_limit time_limit]
[-verbosity verbosity_level] [-silent]
```

Scan .ddk trace files produced by a previous simulation run.

Note: This command accepts one switch followed by one or more file names. You can also use one or more file names without the `-ddk` switch as long as the file name includes the `.ddk` extension.

- Use `-scope` to target the specified POIs.
- Use `-seek` to configure the property synthesis algorithm.
- Use `-per_scope_time_limit` to set a time limit for the scan operation.
- Use `-verbosity` to control the amount of information printed to the log and console. And use `-silent` to turn off output to the screen.

Note: See “[set_scan_verbosity](#)” on page 1219 for additional information.

JasperGold Apps Command Reference Manual

General Commands

```
-no_sim
[-task task_name]
[-from_fsms]
[-from_property property_list]
[-scope id_tcl_list]
[-seek seek_mode]
[-time_limit time_limit]
[-dump_vcd path]
```

Perform RTL analysis for extracting property candidates.

Note: Use the [“set scan no_sim max_covers”](#) on page 1206 command to specify the maximum number of covers the BPS App will use for property synthesis when you use the `-from_property` or `-from_fsms` switches.

- Use `-task` to apply the command to a specified task instead of the current task
- Use `-from_fsms` to direct the tool to perform the RTL analysis on the FSM scopes previously extracted.

Note: If no FSM scopes were extracted, BPS will extract them automatically.

- Use `-from_property` to direct the tool to perform the RTL analysis on a specified list of properties.
 - Use `-scope` to target the specified POIs.
 - Use `-seek` to configure the property synthesis algorithm.
 - Use `-time_limit` to set a time limit for the RTL analysis.
 - Use `-dump_vcd` to create VCD files for debugging and reuse.
-

JasperGold Apps Command Reference Manual

General Commands

```
-no_sim -from_formal_search
[-task task_name]
[-depth N]
[-seek seek_mode_2]
[-time_limit time_limit]
```

Perform RTL analysis for extracting property candidates using formal technology. Using the `-from_formal_search` switch will create a scope called `formal_search`.

- Use `-task` to apply the command to a specified task instead of the current task
 - Use `-depth` to limit searching to a certain bound. The default depth is 20.
 - Use `-seek` to configure the property mining algorithm.
- Note:** This controls whether to search for onehot properties.
- Use `-time_limit` to set a time limit for the RTL analysis.

```
-merge jdb_file_list [-min_confirm N]
```

Merge a list of .jdb files into one.

Use `-min_confirm` to require that each invariant be confirmed by at least `N` traces.

Note: The `scan -merge` utility analyzes a large number of simulation traces. All .jdb files must contain the same POI configuration.

```
-clear
```

Clear previous scan data so that future property candidate extraction starts from scratch and does not evolve from previously generated property candidates.

Return

Context-sensitive return values.

Examples

```
% scan -trace -fsdb first.fsdb second.fsdb
% scan -no_sim -from_fsms
```

JasperGold Apps Command Reference Manual

General Commands

See Also

[check_bps](#)
[set_scan_per_scope_time_limit](#)
[set_scan_seek_depth](#)
[set_scan_seek_mode](#)
[waveform](#)

schematic_viewer

Description

Use the `schematic_viewer` command to open a schematic viewer window and draw the top module schematic, its instances, or specific signals.

Note: `schematic_viewer` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

Syntax

```
schematic_viewer -clear [-window <window_name>]  
schematic_viewer -draw [-instance <instance_name>]  
    [-level <uint_range 1:$>]  
    [-task <task_name>]  
    [-window <window_name>]  
schematic_viewer -open [-task <task_name>]  
    [-window <window_name>]  
schematic_viewer -delete [-window <window_name>]  
schematic_viewer -list  
schematic_viewer -signal <signal_list> -all  
    [-window <window_name>]  
schematic_viewer -signal <signal_list> -driver  
    [-window <window_name>]  
schematic_viewer -path  
    -from <from_signal_list>  
    -to <to_signal_list>  
    [-window <window_name>]
```

Argument	Definition
<code>-clear [-window window_name]</code>	<p>Clear the contents of the specified schematic viewer window. Use <code>-window</code> to clear a window other than the default window.</p>

JasperGold Apps Command Reference Manual

General Commands

```
-draw [-instance instance_name]  
[-level uint_range 1:$]  
[-task task_name]  
[-window window_name]
```

Draw the schematic of the top module on the schematic viewer window.

- Use `-instance` to draw a specified instance instead of the top module on the schematic viewer window.
- Use `-level uint_range 1:$` to draw the schematic of an entire module hierarchically up to the specified number of levels on the schematic viewer window. The default level value is 1.
- Use `-task` to draw the schematic of the specified task on the schematic viewer window.
- By default, the tool applies your `schematic_viewer` command to the current window. Use `-window` to apply your command to a window other than the current window.

```
-open [-task task_name] [-window window_name]
```

Open the specified schematic viewer window.

- Use `-task` to open a task other than the current task.
- Use `-window` to specify a window other than the default window.

```
-delete [-window window_name]
```

Close the specified schematic viewer window.

Use `-window` to close a window other than the default window.

```
-list
```

List the names of all schematic viewer windows. The order of the elements in the list is not defined.

```
-signal signal_list -all [-window window_name]
```

Draw all gates connected to the signals specified by the signal list on the schematic viewer window.

```
-signal signal_list -driver [-window window_name]
```

Draw the gates that drive the signals specified by the signal list on the schematic viewer window.

JasperGold Apps Command Reference Manual

General Commands

```
-path -from from_signal_list -to to_signal_list [-window window_name]
```

Show all paths between two sets of signals.

If there is no path between the sets of signals, the viewer displays the signals themselves and their driving logic.

Return

No return values

See Also

No related commands

scope

Description

Use this command to specify the scope for property candidate extraction. A scope is a collection of points of interest (POIs). The POIs can be any or all of the following:

- Counter
- FIFO
- FSM
- Onehot
- User-defined
- Custom FSM
- Module interfaces (mi)
- Module

During POI extraction, the tool also captures relevant RTL information, such as the clock that the tool inferred and the critical values a counter has been compared to.

Syntax

```
scope -extract (counter | fsm | fifo | onehot)+  
    [ -instances <instance_list> |-hierarchies <instance_list>  
     | -exclude_hierarchies <instance_list> |-top] [-clock <signal>]  
     [-clock_edge (posedge | negedge | both_edges | auto_detect)]  
     [-silent]  
  
scope -extract (module_interface | mi)  
    [ -instances <instance_list> |-hierarchies <instance_list>  
     | -exclude_hierarchies <instance_list> |-top | -bbox]  
     [-min_ports_threshold <threshold>]  
  
scope -extract module  
    [ -instances <instance_list> |-hierarchies <instance_list>  
     | -exclude_hierarchies <instance_list> |-top]  
     [-min_ports_threshold <threshold>]  
  
scope -extract all  
    [-clock <signal>]  
    [-clock_edge (posedge | negedge | both_edges | auto_detect)]
```

JasperGold Apps Command Reference Manual

General Commands

```
scope -add (<signal>)+  
    [-clock <signal>]  
    [-clock_edge (posedge | negedge | both_edges | auto_detect)]  
    [-silent]  
  
scope -add (<signal>)+ -custom_fsm  
    [-clock <signal>]  
    [-clock_edge (posedge | negedge | both_edges | auto_detect)]  
    [-silent]  
  
scope -add -from_file <xml_file> [-silent]  
  
scope <id_tcl_list>  
    (-cv | -add_critical_values) (<critical_value>)  
    [-silent]  
  
scope <id_tcl_list>  
    (-ce | -add_critical_expressions) (<critical_expr>)  
    [-silent]  
  
scope <id_tcl_list>  
    (-ve | -add_valid_expression) (<valid_expr>)  
    [-silent]  
  
scope [<id_tcl_list>] -export [-file <file>]  
    [ -format (pli_configuration | shm_dump_configuration  
    | fsdb_dump_configuration)]  
    [-hier_path <path>] [-no_reset]  
    [-silent]  
  
scope <id_tcl_list> (-enable | -disable)  
    [-silent]  
  
scope (-enable | -disable)  
    -size_threshold <size>  
    [-silent]  
  
scope (-enable | -disable)  
    (-pattern <pattern>)+  
    [-silent]  
  
scope -show <id_tcl_list>  
    [-baseline]  
    [-silent]  
  
scope <id_tcl_list> -get  
    ( type | clock | clock_edge | | clock_edge_per_signal  
    | enabled | signals | user_defined | roles | instance | template  
    | critical_values | cv  
    | critical_expressions | ce  
    | valid_expression | ve  
    | input | output | internal | free_net)  
    [-baseline]  
    [-silent]  
    scope -list
```

JasperGold Apps Command Reference Manual

General Commands

```
(-type ( counter | fsm | fifo | onehot
         | user_defined | custom_fsm
         | module_interface | mi))+

[-obsolete]
[-silent]

scope -fold ([<id_tcl_list>] -all
            |-list_module_candidates
            |-list_folding_candidates
            |-list_folded_pois
            |(-type ( counter | fsm | fifo
                      | onehot | user_defined | custom_fsm
                      | module_interface | mi))+
            | (-module <module_name>))

scope -unfold ([<id_tcl_list>] -all
              |(-type ( counter | fsm | fifo
                        | onehot | user_defined | custom_fsm
                        | module_interface | mi))+
              | (-module <module_name>))

scope -highlight <id>

scope -set_active_high <signal_tcl_list>
scope -set_active_low <signal_tcl_list>
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<pre>-extract (counter fsm fifo onehot)+ [-instances <i>instance_list</i> -hierarchies <i>instance_list</i> -exclude_hierarchies <i>instance_list</i> -top] [-clock <i>signal</i>] [-clock_edge (posedge negedge both_edges auto_detect)] [-silent]</pre>	<p>Analyze the RTL to extract POIs that will be the basis for capturing property candidates from simulation.</p> <ul style="list-style-type: none">■ Use one of the following enum elements to specify the type of POIs you want to extract:<ul style="list-style-type: none">□ counter□ fsm□ fifo□ onehot■ Use <code>-instances</code> to limit extraction to the specified instances.■ Use <code>-hierarchies</code> to limit extraction to the specified hierarchies.■ Use <code>-exclude_hierarchies</code> to exclude the specified hierarchies.■ Use <code>-top</code> to limit extraction to the top instance.■ Use <code>-clock</code> to force the extracted POIs to ignore clocks specified with the <code>clock</code> command and use this clock instead.■ Use <code>-clock_edge</code> to specify the triggering edge.<ul style="list-style-type: none">□ Use <code>posedge</code> to specify the rising edge.□ Use <code>negedge</code> to specify the falling edge.□ Use <code>both_edges</code> to specify both edges.□ Use <code>auto_detect</code> to automatically detect the triggering edge for each signal in the POI based on waveform analysis.■ Use <code>-silent</code> to suppress messages and generate only Tcl return.

JasperGold Apps Command Reference Manual

General Commands

```
-extract (module_interface | mi)
[ -instances instance_list |-hierarchies instance_list
| -exclude_hierarchies instance_list |-top |-bbox]
[-min_ports_threshold threshold]
```

Analyze the RTL to extract module interface POIs that will be the basis for capturing property candidates from simulation.

- Use `-instances` to limit extraction to the specified instances.
 - Use `-hierarchies` to limit extraction to the specified hierarchies.
 - Use `-exclude_hierarchies` to exclude the specified hierarchies.
 - Use `-top` to limit extraction to the top instance.
 - Use `-bbox` to limit extraction to black-boxed instances.
 - Use `-min_ports_threshold` to extract only those modules with the specified minimum port count.
-

```
-extract module
[-instances instance_list |-hierarchies instance_list
| -exclude_hierarchies instance_list |-top]
[-min_ports_threshold threshold]
```

Analyze the RTL to extract module POIs that will be the basis for capturing property candidates from simulation.

- Use `-instances` to limit extraction to the specified instances.
 - Use `-hierarchies` to limit extraction to the specified hierarchies.
 - Use `-exclude_hierarchies` to exclude the specified hierarchies.
 - Use `-top` to limit extraction to the top instance.
 - Use `-min_ports_threshold` to extract only those modules with the specified minimum port count.
-

JasperGold Apps Command Reference Manual

General Commands

```
-extract all  
[-clock signal]  
[-clock_edge (posedge | negedge | both_edges | auto_detect)]
```

Extract POIs of all types.

This command is equivalent to `-extract counter fsm onehot is mi`. It does not extract structural check types.

- Use `-clock` to force the extracted POIs to ignore clocks specified with the `clock` command and use this clock instead.
- Use `-clock_edge` to specify the triggering edge.
 - Use `posedge` to specify the rising edge.
 - Use `negedge` to specify the falling edge.
 - Use `both_edges` to specify both edges.
 - Use `auto_detect` to automatically detect the triggering edge for each signal in the POI based on waveform analysis.

```
-add signal...  
[-clock signal]  
[-clock_edge (posedge | negedge | both_edges | auto_detect)]  
[-silent]
```

Capture one or more specified signals as user-defined POIs.

- Use `-clock` to specify the clock for the group of signals.
- Use `-clock_edge` to specify the triggering edge.
 - Use `posedge` to specify the rising edge.
 - Use `negedge` to specify the falling edge.
 - Use `both_edges` to specify both edges.
 - Use `auto_detect` to automatically detect the triggering edge for each signal in the POI based on waveform analysis.
- Use `-silent` to suppress messages and generate only Tcl return values.

JasperGold Apps Command Reference Manual

General Commands

```
-add signal+ -custom_fsm
[-clock signal]
[-clock_edge (posedge | negedge | both_edges | auto_detect)
[-silent]
```

Create a custom FSM from a concatenation of the specified signals.

- Use `-clock` to specify the clock for the `custom_fsm`.
- Use `-clock_edge` to specify the triggering edge.
 - Use `posedge` to specify the rising edge.
 - Use `negedge` to specify the falling edge.
 - Use `both_edges` to specify both edges.
 - Use `auto_detect` to automatically detect the triggering edge for each signal in the POI based on waveform analysis.
- Use `-silent` to suppress messages and generate only Tcl return values.

```
-add -from_file xml_file
[-silent]
```

Create all POIs as defined in the specified XML file.

Use `-silent` to suppress messages and generate only Tcl return values.

Note: For a sample XML file, see the Appendix, “User-Defined POIs from XML,” in the *Behavioral Property Synthesis App User’s Guide*. This guide is available from Cadence Online Support (support.cadence.com).

```
id_tcl_list
(-cv |-add_critical_values) critical_value
[-silent]
```

In addition to any completed RTL analysis for the specified POIs (`id_tcl_list`), add the specified critical values.

Use `-silent` to suppress messages and generate only Tcl return values.

```
id_tcl_list
(-ce |-add_critical_expressions) critical_expr
[-silent]
```

In addition to any completed RTL analysis for the specified POIs (`id_tcl_list`), add the specified critical expressions.

Use `-silent` to suppress messages and generate only Tcl return values.

JasperGold Apps Command Reference Manual

General Commands

```
id_tcl_list
(-ve | -add_valid_expressions) valid_expr
[-silent]
```

Restrict analysis for specified POI or POIs to cycles where *valid_expr* is true. Synthesized properties will have the general form *valid_expr imply property*.

Use *-silent* to suppress messages and generate only Tcl return values.

```
id_tcl_list -export [-file file]
[ -format (pli_configuration | shm_dump_configuration
| fsdb_dump_configuration)
[-hier_path path]
[-no_reset]
[-silent]
```

Export POI(s) configuration to file. Use the exported file to configure simulation trace dump with JasperGold Apps PLI, SHM, or FSDB formats. If you do not specify a POI ID, the default is all enabled POIs.

- Use *-file* to specify a file name.
- Use *-format pli_configuration*, which is the default, to export POIs in JasperGold Apps PLI configuration format. Use this file to run simulation(s) with JasperGold Apps PLI module.
- Use *-format shm_dump_configuration* to export POIs in *irun -bps_cfg* command text file format. Use this file to dump relevant signals from simulation(s). For example, the following dumps the specified signals and scopes from *configuration_filename* to *bps_dump.shm*:

irun -bps_cfg configuration_filename
- Use *-format fsdb_dump_configuration* to export POIs in Novas FSDB dump command text file format. Use this file to dump relevant signals from simulation(s) to FSDB with Novas PLI *fsdbDumpvarsByFile* method. For example, when specified in a Verilog design, both of the following will dump the specified signals and scopes from *configuration_filename* to *test.fsdb*:

- *\$fsdbDumpfile("test.fsdb");*
\$fsdbDumpvarsByFile("configuration_filename");
 - *\$fsdbDumpvarsByFile("configuration_filename" , "+fsdbfile+test.fsdb");*

JasperGold Apps Command Reference Manual

General Commands

id_tcl_list -export (Continued)

Note:

- ❑ For VHDL designs, omit the dollar sign.
- ❑ From the simulator command line, type the following:

Cadence:

```
call fsdbDumpvarsByFile "configuration_filename"  
"
```

ModelSim:

```
fsdbDumpvarsByFile "Configuration_filename"
```

Synopsys:

```
$fsdbDumpvarsByFile("configuration_filename");
```

- Use **-hier_path** to specify a hierarchical path.
- Use **-no_reset** with PLI to dump the whole simulation run rather than just the non-reset portions.

Note: This switch is irrelevant for FSDB partial dump.
- Use **-silent** to suppress messages and generate only Tcl return values.

*id_tcl_list (-enable | -disable)
[-silent]*

Enable or disable the POIs with the specified IDs.

- Use **-enable** to enable `scan` to extract properties from the specified POIs.
- Use **-disable** to prevent `scan` from extracting properties from the specified POIs.
- Use **-silent** to suppress messages and generate only Tcl return values.

*-enable | -disable) -size_threshold size
[-silent]*

Enable POIs with fewer than the number of bits specified or disable POIs with greater than the number of bits specified.

Use **-silent** to suppress messages and generate only Tcl return values.

JasperGold Apps Command Reference Manual

General Commands

```
-enable | -disable) -pattern pattern+
[-silent]
```

Enable or disable POIs that contain signals matching the specified pattern.

Use `-silent` to suppress messages and generate only Tcl return values.

```
-show id_tcl_list [-baseline]
[-silent]
```

Show the details of the specified POIs.

- Use `-baseline` to see the details of the POIs in the baseline snapshot.
- Use `-silent` to suppress messages and generate only Tcl return values.

```
id_tcl_list -get
( type | clock | clock_edge | clock_edge_per_signal | enabled
  signals | user_defined | roles | instance | template
  critical_values | cv | critical_expressions | ce
  valid_expression | ve | input | output | internal
  free_net)
[-baseline]
[-silent]
```

Get the specified attribute of the specified POIs.

- Use `-baseline` to see the details of the POIs in the baseline snapshot.
- Use `-silent` to suppress messages and generate only Tcl return values.

```
-list
(-type ( counter | fsm | fifo | onehot | user-defined
  | custom_fsm | module_interface | mi) )+
[-obsolete]
[-silent]
```

List the IDs for the POIs that match one or more specified types.

- Use one `-type` switch for each type you specify.
- Use `-obsolete` to include obsolete POIs.
- Use `-silent` to suppress messages and generate only Tcl return values.

JasperGold Apps Command Reference Manual

General Commands

```
-fold (id_tcl_list -all  
      | -list_module_candidates  
      | -list_folding_candidates  
      | -list_folded_pois  
      | (-type ( counter | fsm | fifo | onehot | user-defined  
                | custom_fsm | module_interface | mi ) )+  
      | (-module module_name) )
```

Fold POIs instantiated more than once on the same module into one POI. To fold only specified POIs, list the POI IDs.

Note: The tool refers to POIs instantiated more than once as folding candidates.

- Use `-all` to fold all POIs that can be folded, that is, candidate POIs.
- Use `-list_module_candidates` to list all modules with more than one instantiation in the design. This switch helps you identify folding candidates.
- Use `-list_folding_candidates` to list all POIs that can be folded.
- Use `-list_folded_pois` to list all folded POIs.
- Use `-type` to fold all candidates of the specified type.
- Use `-module` to fold all candidates of the specified module.

```
-unfold (id_tcl_list -all  
        | (-type ( counter | fsm | fifo | onehot | user-defined  
                  | custom_fsm | module_interface | mi ) )+  
        | (-module module_name) )
```

Unfold previously folded POIs. To unfold only specified POIs, list the POI IDs.

- Use `-all` to unfold all previously folded POIs.
- Use `-type` to unfold POIs of the specified type.
- Use `-module` to unfold POIs of the specified module.

```
-highlight id
```

Highlight the POI with the specified ID in the *Scope* pane. (The tool takes no action if the ID is invalid or it is currently invisible due to filtering.)

JasperGold Apps Command Reference Manual

General Commands

```
-set_active_high signal_tcl_list
```

Mark the matching signals as active high.

The Behavioral Property Synthesis App assigns an active phase to every scalar signal. The active phase determines how the signal will appear in synthesized properties. By default, the signal's active phase is analyzed from the trace. However, you can use this command to set the phase for specified signals, that is, you can overwrite the active phase analysis. This command has no impact on vector signals.

Note: This command accepts signal or regular expression lists.

```
-set_active_low signal_tcl_list
```

Mark the matching signals as active low.

The Behavioral Property Synthesis App assigns an active phase to every scalar signal. The active phase determines how the signal will appear in synthesized properties. By default, the signal's active phase is analyzed from the trace. However, you can use this command to set the phase for specified signals, that is, you can overwrite the active phase analysis. This command has no impact on vector signals.

Note: This command accepts signal or regular expression lists.

Return

Context-sensitive return values.

Examples

```
% scope -extract all
% scope -add ready0 ready1 ready2 ready3 -clock clk
% scope 1 -get clock_edge_per_signal
% scope -set_active_low read_stall *_1
% scope -add mi -instances { inst1 inst2 inst3 }

// Example command for a FIFO with two pointers. Each of the pointers is
controlled by a clock. This command forces a specific clock to be the POI
// clock.

% scope -extract is -clock myclk -clock_edge posedge
```

JasperGold Apps Command Reference Manual

General Commands

See Also

[check_bps](#)

[scan](#)

[set_scope extract min_ports threshold](#)

[waveform](#)

session

Description

Use the `session` command to manipulate analysis sessions.

Note: The tool uses session identifiers in the form `session_N` and stores their associated log and command files in the project directory, for example, `jgproject/sessionLogs/session_0`. For convenience, you can specify a session name for the current session (`session -set_name mySession`) and launch a new session with a specified name (`session -new myOtherSession`), but the session identifiers in the project directory remain unchanged.

Syntax

[General Session Manipulation](#)

```
session -refresh

session -new [<session_name>] [-remote]
    (-define <tcl_variable> <tcl_variable_value>)*
    [-callback_on_close <commands>]
    [-dont_echo_callback_on_close]
    [-reuse_elaborated_design]

session -close [<session_name>]

session -run_cmds <session_name> <commands>
    [ -use_hier_path
        |-add_hier_path <path>
        |-remove_hier_path <path>]
    [-callback <commands>]
    [-dont_echo]
    [-dont_echo_callback]

session -run_script <tcl_script>+
    [ -use_hier_path
        |-add_hier_path <path>
        |-remove_hier_path <path>]

session -clear_msg
```

[Support for Exporting Session Configuration](#)

```
session -set_name <session_name>
session -get_name
```

JasperGold Apps Command Reference Manual

General Commands

```
session -get_setup  
session -get_proj_dir  
session (-export_sva |-export_psl)  
  [(-task <task_name>) |-env] (-file <file_name> [-force])
```

[Support for Setting Up the Database for a Different Design Hierarchy](#)

```
session -clear_hier_path  
session -set_hier_path [-add |-remove] <path>
```

[Support for Configuring How a Remote Session Should be Launched](#)

```
session -set_remote_mode (local | shell | lsf | oge)  
session -set_remote_shell <full_path_to_shell_plus_args>  
session -get_remote_mode  
session -get_remote_shell
```

[Plugin Configuration](#)

```
session -load_plugin <plugin_list> -plugin_dir <plugin_dir>
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
General Session Manipulation	
-refresh	<p>Refresh the analysis session with the default setup in the database and reload the RTL.</p>
-new [session_name] [-remote] -define tcl_variable tcl_variable_value [-callback_on_close commands] [-dont_echo_callback_on_close] [-reuse_elaborated_design]	<p>Launch a new analysis session with the specified name.</p> <p>If you do not specify a name, the tool names the new session <code>session_n</code>, where <code>n</code> is a number greater than zero. This command returns the name of the new session.</p> <ul style="list-style-type: none">■ Use <code>-remote</code> to start a new session with the user-specified launcher that you specified with the <code>-remote_mode</code> switch at startup or the <code>session -set_remote_mode</code> command. If you use mode <code>lsf</code> or <code>oge</code>, or have a shell script that calls your server farm, the tool can support multiple analysis sessions starting on different machines. <p>Note: When using <code>-remote</code> with <code>lsf</code> or <code>oge</code>, refer to "LSF_BINDIR" on page 76 or "SGE_ROOT" on page 78.</p> <ul style="list-style-type: none">■ Use <code>-define</code> to start a new session and use the specified values for the specified Tcl variables. <p>Implementation guidance:</p> <ul style="list-style-type: none">□ The <code>-define</code> switch is especially useful for launching two analysis sessions, each with a different version of the RTL so you can bring up the corresponding waveforms for comparison.

new (Continued)

- Define multiple variables by using one `-define` switch for each definition.
 - Use `-callback_on_close` to register commands to be executed in the current analysis session when the launched session is closed (or crashed).
 - Use `-dont_echo_callback_on_close` to suppress command echoing for the commands registered with `-callback_on_close`. Without this switch, the tool echoes the commands on the console of the analysis session that executed the session `-new` command.

Note: `-callback_on_close` is a prototype and has limitations.

- Use `-reuse_elaborated_design` to automatically load the design that was elaborated in the current session without re-elaborating it.

Note:

- For this switch to work, you must set the `capture_elaborated_design` variable to `on` before elaboration. For additional details, see [“set_capture_elaborated_design”](#) on page 964.
 - `session -reuse_elaborated_design` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

`-close [session_name]`

Close the session with the user-specified name.

Important things to know about this command:

- If multiple sessions with the same name are present, this command closes the one launched earliest.
- If you do not provide a name, the session that executed this command closes.

JasperGold Apps Command Reference Manual

General Commands

```
-run_cmds session_name commands
```

```
[-use_hier_path |-add_hier_path path | -remove_hier_path path]  
[-callback commands] [-dont_echo] [-dont_echo_callback]
```

Run the specified commands in the session with the specified name.

- Return `true` if the specified session exists.
- Return `false` if the specified session does not exist.

If multiple sessions with the same name are present, this command uses the one launched earliest.

- Use `-use_hier_path`, `-add_hier_path`, or `-remove_hier_path` to execute scripts generated for RTL with a different hierarchy. See `session -set_hier_path` for determining the value of the path argument.
- Use `-dont_echo` to suppress command echoing on the console of the specified analysis session.
- Use `-callback` to register commands to be executed in the current analysis session when the session with the matching name has completed the specified commands.
- Use `-dont_echo_callback` to suppress command echoing for the callback commands. Without this switch, the tool echoes the commands on the console of the current analysis session.

Note: `-callback` is a prototype and has limitations.

```
-run_script tcl_script+
```

```
[-use_hier_path |-add_hier_path path | -remove_hier_path path]
```

Run the Tcl script in batch so that `assume`, `assert`, and `cover` commands do not attempt to populate the database.

Note: Run a structural analysis after the script if you want to populate the database with the `assume`, `assert`, and `cover` elements.

- Use `-use_hier_path`, `-add_hier_path`, or `-remove_hier_path` to execute scripts generated for RTL with a different hierarchy.
- See `session -set_hier_path` for determining the value of the path argument.

```
-clear_msg
```

Clear all message log panes (all tabs) for the active session.

JasperGold Apps Command Reference Manual

General Commands

Support for Exporting Session Configuration

`-set_name session_name`

Set the name of the current analysis session.

`-get_name`

Return the name of the current analysis session.

`-get_setup`

Return the content of the setup script used to set up the current session.

`-get_proj_dir`

Return the project directory specific to the current session.

Compare this command to the `get_proj_dir` command, which returns the project directory specific to the tool.

`-export_sva | -export_psl`

`([-task task_name | -env]
(-file file_name [-force]))`

Translate the session's Tcl temporal properties into SVA or PSL and save them in the specified file.

Important things to know about this command:

- Tcl temporal properties are created with JasperGold Apps `assume`, `assert`, and `cover` commands.
 - If you specify `-export_sva`, bind the resulting SVA to the RTL file with SVA bind.
 - If you specify `-export_psl`, analyze the new PSL file along with the design to see the extracted properties in their environment.
 - By default, this command applies to the current task.
Use `-task` to translate properties that belong to a task other than the current task.
 - Use `-env` to extract only the assumptions created with `assume -env`.
- Note:** Environmental (global) assumptions are *not* included by default.
- Use `-force` to overwrite any existing file that has the specified name.

Support for Setting Up the Database for a Different Design Hierarchy

`-clear_hier_path`

Clear the hierarchical path for this session.

The tool automatically runs this command right before running the default setup script in the database.

`-set_hier_path [-add path | -remove path]`

Set the hierarchical path for this session.

This path represents the difference between the data captured in the Behavioral Indexing database and the hierarchy of the RTL in the current session. For example,

- If an embedded property has `task::top.a.b.c`, and the current session only uses `task::top2.c`, use the following command:
`session -set_hier_path -remove top.a.b`
- If an embedded property has `task::top.c`, and the current session only uses `task::top2.a.b.c`, use the following command:
`session -set_hier_path -add top2.a.b`



- Add this command to the default setup script of the database depending on how the RTL is elaborated according to the Tcl variables that can be configured by the `session -define` commands.
- If you do not specify either `-add` or `-remove`, the default is `-add`.

JasperGold Apps Command Reference Manual

General Commands

Support for Configuring How a Remote Session Should be Launched

-set_remote_mode local | shell | oge | lsf

Set up the configuration for session -remote -new.

This is similar to `set_proofgrid_mode`, but it applies to session management rather than ProofGrid management.

- local – Launch remote analysis session on your local machine (the machine running the JasperGold Apps session).
- shell – Use a custom command or script to spawn the jobs, see `session -set_remote_shell`.
- lsf – Launch remote analysis sessions on the Platform LSF cluster that is expected to be available using the `LSF_BINDIR` environment. See “[LSF BINDIR](#)” on page 76 for additional information.
- oge – Launch remote analysis session on the Oracle Grid Engine that is expected to be available using the `SGE_ROOT` environment variable. See “[SGE_ROOT](#)” on page 78 for additional information.

Note: This option was previously named `sge`, which stands for Sun Grid Engine.

-set_remote_shell arg

The argument must be a string beginning with the full path to an executable. As an option, that string can continue with a space-separated set of extra arguments that will be given to the executable. You can use double and single quotes to form an argument that contains spaces. The executable must take the job to start and its arguments as further arguments. See “[proofgrid_socket_communication](#)” on page 1161 for additional information.

-get_remote_mode

Return the current setting from the `-set_remote_mode` switch.

-get_remote_shell

Return the current setting from the `-set_remote_shell` switch.

Plugin Configuration

```
-load_plugin plugin_list -plugin_dir plugin_dir
```

Load the plugins (provided by JasperGold Apps) from the plugin directory, including but not limited to “new proof accelerators” to be instantiated and connected to the RTL and “new tcl commands” to be called from the Tcl interface.

Return

Various

Examples

```
% session -new session2
% session -run_cmds session2 "source my.tcl"

// Running a new session in remote mode with Platform LSF:
% session -set_remote_mode lsf
% session -remote -new session2

// Running a new session in remote mode with a shell script:
% session -set_remote_mode shell
% session -set_remote_shell "my.script -arg abcd"
% session -remote -new session2
```

See Also

[database](#)

set_annotation

Description

Use the `set_annotation` command to annotate a property or signal or to modify an existing annotation. View annotations in the Visualize window *Waveforms* pane by using the mouse to hover over the signal name and reveal a tooltip.

Syntax

```
set_annotation ((-property <property_name> [-regexp])
               |-signal <signal_name>) '{'<annotation>'}'
```

Argument	Definition
<code>-property property_name [-regexp]</code>	Annotate the specified property. Note: <ul style="list-style-type: none">■ This switch supports wildcards and regular expressions; for example: <code>set_annotation -property {expression} -regexp {annotation}</code>■ When the tool finds a wildcard or regular expression match for the property names, it applies the corresponding annotations to all matching properties.
<code>-signal signal_name</code>	Annotate the specified signal.
<code>{annotation}</code>	Annotate the specified property or signal with the specified description that is enclosed by curly braces. To remove an annotation, use empty curly braces.

Default

None

Return

No return values

Examples

```
% set_annotation -signal clk {Global clock}
```

See Also

[get_annotation](#)

set_current_gui

Description

Use the `set_current_gui` command to change the current GUI view. When an app has already been launched, this command is essentially the same as clicking its tab in the main window. If the specified app has not been launched, this command launches it.

Syntax

```
set_current_gui ( arch | bps | conn | cov | csr | dld | fpv | lpv  
                  | rtld | sec | spec | sps | spv | unr | xprop)
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
(arch bps conn cov csr dld fpv lpv rtld sec spec sps spv unr xprop)	<p>Launch or change the GUI to the specified app:</p> <ul style="list-style-type: none">■ arch – Architectural Modeling App■ bps – Behavioral Property Synthesis App■ conn – Connectivity Verification App■ cov – Coverage App■ csr – CSR Verification App■ dld – Deadlock Detection App■ fpv – Formal Property Verification App■ lpv – Low Power Verification App■ rtld – RTL Development App■ sec – Sequential Equivalence Checking App■ spec – Executable Specification App■ sps – Structural Property Synthesis App■ spv – Security Path Verification App■ unr – Coverage Unreachability App■ xprop – X-Propagation Verification App

Default

None

Return

No return value.

JasperGold Apps Command Reference Manual

General Commands

Examples

```
% set_current_gui fpv
```

See Also

[get_tool](#)

set_message

Description

Use the `set_message` command to manipulate and change severity for JasperGold Apps and frontend messages. This command is helpful when you want to filter out specific info and warning messages that are not meaningful to you or change the severity of messages to suit your needs.

Command Guidelines

- Specify a single message ID with the command. (Use the command multiple times if you want to manipulate multiple messages.)
- Message IDs are case sensitive.

Syntax

```
set_message -disable <message_id>
set_message -enable <message_id>
set_message -error <message_id>
set_message -warning <message_id>
set_message -info <message_id>
set_message -default (<message_id> | -all)
set_message -unlimit_repeated_message <message_id>
    |-limit_repeated_message <message_id>
    |-limit_all_repeated_message <message_id>
set_message -limit_beta_warning_message
    |-unlimit_beta_warning_message
set_message -limit_initial_release_message
    |-unlimit_initial_release_message
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<code>-disable message_id</code>	<p>Disable the specified info or warning message.</p> <p> <i>Important</i></p> <ul style="list-style-type: none">■ You cannot specify an error message ID.■ This command issues a special info message that appears in the “Warnings/Errors” message tab in the console.
<code>-enable message_id</code>	<p>Enable the specified info or warning message.</p> <p>Note: This command issues a special info message that appears in the “Warnings/Errors” message tab in the console.</p>
<code>-error message_id</code>	<p>Increase the severity of the specified message to ERROR or activate the specified error message.</p> <p> <i>Important</i></p> <ul style="list-style-type: none">■ This switch applies broadly to messages beginning with VERI-, VHDL-, or VDB-.■ This switch applies to JasperGold Apps messages ICK010, ICK011, ICK012, ICK013, WFL012, WLPV030, WLPV031, WLPV036, WLPV050, WLPV051, WNL015, WNL021, WNL023, WNL038, WPM018, WPM022, WRS014, and WSW008. The tool generates an error message when the specified message ID is not supported.■ Changing message severity is not recommended because it can produce unexpected results and adversely affect debugging.

JasperGold Apps Command Reference Manual

General Commands

`-warning message_id`

Decrease the severity of the specified error message to WARNING or increase the specified info message to WARNING.



- This switch applies to all JasperGold Apps and frontend info messages.
- This switch applies to a limited set of frontend error messages, including VERI-1010, VERI-1084, VERI-1139, VERI-1148, VERI-1158, VERI-1216, VERI-1245, VERI-1264, VERI-1273, VERI-1289, VERI-1348, VERI-1691, VERI-1906, VERI-1928, VERI-1991, VERI-2072, VERI-2079, VERI-2094, VERI-9008, VHDL-1160, and VHDL-1454.
- This switch applies to a limited set of JasperGold Apps error messages, including ELPV055, ELPV056, ELPV057, ELPV102, ENL098, EPF020, EPM016, ERS034, ESEC093, ETR056, and all EOBS* messages.
- The tool generates an error message when the specified message ID is not supported.
- This switch enables a limited set of frontend warning messages that the tool ignores by default:
 - VERI-9002 (warning for LHS width of an assignment larger than RHS width)
 - VERI-9004 (comparison between unequal length operands of size 1 and 2)
 - VERI-9030 (warning for implicitly declared nets)
 - VERI-9032 (warning for negative range limits)
- Downgrading message severity is not recommended because it can produce unexpected results and adversely affect debugging.

JasperGold Apps Command Reference Manual

General Commands

```
-info message_id
```

Decrease the severity of the specified warning message to INFO.

Note:

- This switch applies to all JasperGold Apps and frontend warning messages.
- This switch applies to a limited set of frontend error messages, including VERI-1010, VERI-1084, VERI-1148, VERI-1216, VERI-1264, VERI-1348, VERI-2094, VERI-2138, VERI-2139, VERI-2140, VHDL-1591, VHDL-1669, and VHDL-1704.
- This switch applies to a limited set of JasperGold error messages, including ELPV055, ELPV056, ELPV057, ELPV102, ENL098, EOBS001, EOBS002, EOBS003, EOBS004, EOBS005, EOBS006, EOBS007, EOBS008, EOBS009, EOBS010, EOBS011, EOBS012, EOBS013, EOBS014, EOBS015, and EOBS016.

```
-default message_id | -all
```

Re-instate the default message severity level for the specified message or for all messages.

```
-unlimit_repeated_message message_id  
| -limit_repeated_message message_id  
| -limit_all_repeated_message message_id)
```

Turn on or off specified repetitive frontend messages.

Note:

- By default, the tool does not display repetitive frontend messages.
- Specify your preferences for Verific messages that have the same source location before elaborating the design.
- Use `-unlimit_repeated_message` if you want the tool to print all Verific messages that have the same source location.
- Use `-limit_repeated_message` to specify a message you do not want the tool to repeat.
- Use `-limit_all_repeated_message` to specify the default handling. With this switch, the tool does not repeat Verific messages that have the same source location.

JasperGold Apps Command Reference Manual

General Commands

-limit_beta_warning_message | -unlimit_beta_warning_message

Turn off repetitive beta warning messages or reinstate the default.

Note: By default, the tool prints a beta warning message each time you run a beta command or use a beta feature.

- Use `-limit_beta_warning_message` to print the warning only the first time you run a beta command or use a beta feature.
 - Use `-unlimit_beta_warning_message` to print the warning every time you run a beta command or use a beta feature. This is the default.
-

-limit_initial_release_message | -unlimit_initial_release_message

Turn off repetitive initial release messages or reinstate the default.

Note: By default, the tool prints an initial release message each time you run an initial release command or use an initial release feature.

- Use `-limit_initial_release_message` to print the warning only the first time you run an initial release command or use an initial release feature.
 - Use `-unlimit_initial_release_message` to print the warning every time you run an initial release command or use an initial release feature. This is the default.
-

Default

None

Return

For warning and info messages (`-disable`, `-enable`, `-error`), this command triggers an error if the specified ID is not valid.

For `-error` with frontend messages, the tool does not validate IDs, and this command only triggers an error if the ID does not start with VERI-, VHDL-, or VDB-.

For `-warning`, this command triggers an error if the specified ID is not valid.

Examples

```
% set_message -disable ICD003  
% set_message -error VERI-1173
```

JasperGold Apps Command Reference Manual

General Commands

```
% set_message -error WNL038  
% set_message -warning VERI-1158
```

See Also

[get_message](#)
[set_log_timestamp_format](#)
[set_log_timestamp_mode](#)
[set_message_history_limit](#)

show_list

Description

Use this command to format the list of values returned by a Tcl or JasperGold Apps command with a carriage return after each item in the list. This command is useful for formatting a list of values that is going to be printed to a log file.

Syntax

```
show_list '[' <command> ']' [-unsorted]
```

Argument	Definition
<i>command</i> [-unsorted]	<p>Format the list of values returned by the specified command. Use -unsorted to return the list in its original order.</p>

Return

No value returned

Examples

```
% show_list [ dglob ]
```

See Also

No related commands

show_schematic

Description

Use this command to show a flat or hierarchical schematic view of the design.

Mandatory requirement: Before starting your session, add VerdiTM to your PATH.

If necessary, this command will try to:

1. Launch a schematic viewer (Synopsys Verdi is supported)
2. Establish a connection with the viewer
3. Import the design into the viewer
4. Launch a schematic window

JasperGold Apps can manage two schematic windows, one flat and one hierarchical. If there is a JasperGold trace available, the signals in the schematic will be color coded with respect to that trace.

Syntax

```
show_schematic -instance <instance>  
show_schematic -clock [-ar] [-coi] [-full]
```

Argument	Definition
-instance <i>instance</i>	Load the specified instance in the hierarchical schematic window.
-clock	Add clock signals to the flat schematic window. Note: Use this option alone or with -ar, -coi, or -full.

JasperGold Apps Command Reference Manual

General Commands

[-ar]

Clear the flat schematic window and then add all signals in the analysis region with respect to the trace shown in the Visualize window.

The signals will be color coded. Click *Help – Mini Guides – JasperGold Visualize GUI Features* to learn about the significance of signal waveform line colors.

[-coi]

Clear the flat schematic window and then add all signals in the cone of influence with respect to the trace shown in the Visualize window.

The signals will be color coded. Click *Help – Mini Guides – JasperGold Visualize GUI Features* to learn about the significance of signal waveform line colors.

[-full]

Add all signals in the design to the flat schematic window.

The signals will be color coded. Click *Help – Mini Guides – JasperGold Visualize GUI Features* to learn about the significance of signal waveform line colors.

Return

No value returned

Examples

```
% show_schematic -clock -ar  
% show_schematic -instance my_design
```

See Also

[set_schematic_viewer_command](#)
[set_schematic_viewer_connection_timeout](#)
[set_schematic_viewer_design_script](#)
[show_waveform](#)

show_source_browser

Description

Use `show_source_browser` to show the Source Browser window. If you specify no arguments, the window appears in the foreground in its current state. If you specify an instance or property, the window appears in the foreground, and it displays the source for the instance or property. This command has no effect in batch mode.

Syntax

```
show_source_browser [(-instance <instance_name>  
| -property <property_name>) ]
```

Argument	Definition
<code>[-instance instance_name -property property_name]</code>	<p>Show the source for the specified instance or property in the Source Browser window.</p> <p>If you do not specify an instance or property, the tool opens the Source Browser window in its current state.</p>

Return

No return value.

Examples

```
% show_source_browser  
% show_source_browser -instance instanceA  
% show_source_browser -property propertyA
```

See Also

No related commands

show_waveform

Description

Use this command to show the current trace or a VCD file in Verdi nWave viewer. If you specify a VCD file, the tool establishes a connection to the viewer and imports the VCD data into the viewer, making the necessary conversions.

Mandatory requirement: Before starting your session, add Verdi to your PATH.

Note: JasperGold Apps only supports Verdi standalone waveform viewer.

Syntax

```
show_waveform -file <file_name>
```

Argument	Definition
-file <i>file_name</i>	<p>Load the specified VCD file in the configured viewer window. If you do not specify a file, the tool generates a VCD for the current trace and loads this new VCD file in the viewer.</p> <p>Note: The signals will be region colored.</p>

Return

No value returned

Examples

```
% show_waveform -file my_file.vcd  
% show_waveform
```

See Also

[set_schematic_viewer_command](#)
[set_schematic_viewer_connection_timeout](#)
[set_schematic_viewer_design_script](#)
[show_schematic](#)

stopat

Description

Use the `stopat` command to specify that the tool should treat an internal design signal as a free input. In other words, it can have any possible value at any clock cycle.

The `stopat` command forces the tool to stop traversing a netlist at a specified signal or instance name. As a result, the logic driving the signal or instance is not included in the current analysis.

The specified signals are essentially disconnected from their driving logic. You can effectively use this command when the logic driving a signal is very large (for example, a 64-bit address comparison), and it is known that this signal can take any possible value (for example, address matches certain values or it doesn't).

Stopats are either hard (default) or soft. You can justify a soft stopat. You cannot justify hard stopat values, but you can remove them manually. All instance stopats are hard. Hard stopats are placed precisely at the specified signal. Soft stopats at signals are placed at their buffer-equivalent sources.

This command is context-sensitive, that is, it applies to the current task unless you use the `-env` switch to apply it globally or the `-task` switch to specify a task other than the current task.

Note:

- The `stopat` command supports wildcards (*) and (?) in signal names.
- The `stopat` command supports escaped names, including those that contain * and ?.
- The `reset` command only considers global (-env) stopats.

Syntax

```
stopat [-env] (<signal_name_tcl_list> | <instance_name_tcl_list>)
stopat [-task <task_name>] [-soft]
      (<signal_name_tcl_list> | <instance_name_tcl_list>)
stopat [-env | (-task <task_name>)]
      (-list
       |-list_user
       |-list_all
      )
      [-silent]
```

JasperGold Apps Command Reference Manual

General Commands

```
stopat [-env | (-task <task_name>)]  
        ( -remove [-silent] <signal_name>  
        | -clear)  
stopat -reset <signal_list> [-remove]  
stopat -reset (-list | -clear)
```

Argument	Definition
<code>[-env] signal_name_tcl_list instance_name_tcl_list</code>	<p>Apply the stopat on one or more specified signal or instance name(s).</p> <p>Use <code>-env</code> to apply the stopat globally. If you do not use <code>-env</code>, this command applies to the current task.</p>



- Do not use `-env` and `-soft` in the same command.
- If you specify more than one signal or instance, use curly braces for each name. Refer to the example section below.
- This switch supports Tcl lists as arguments.
- If you add a stopat to an instance, its undriven outputs do not receive a stopat. (Stopats for undriven outputs are unnecessary once the stopat is in place for the instance.)

JasperGold Apps Command Reference Manual

General Commands

```
[-task task_name] [-soft] signal_name_tcl_list | instance_name_tcl_list
```

Apply the stopat on one or more specified signal or instance name(s).

- Use `-task` to add the stopat to a task other than the current task.
- Use `-soft` to add a soft stopat at one or more specified signals.



- Do not use `-env` and `-soft` in the same command.
- If you specify more than one signal or instance, use curly braces for each name. Refer to the example section below.
- This switch supports Tcl lists as arguments.
- If you add a stopat to an instance, its undriven outputs do not receive a stopat. (Stopats for undriven outputs are unnecessary once the stopat is in place for the instance.)

Note: Replace `task_name` with a period (.) to specify the current task.

```
[-env | -task task_name] -list |-list_user |-list_all  
[-silent]
```

List the specified set of stopats.

- Use `-env` to list the global stopats.
- Use `-task` to list stopats that impact the specified task, that is, task and global stopats.
- Use `-list` with no additional switches to list all stopats with driven nets in the current task plus the global stopats.
- Use `-list_user` for a list of user-defined stopats.
- Use `-list_all` for a list of all stopats, including undriven signals.
- Use `-silent` to return the list as Tcl return values without printing the list on the screen.

Note: Replace `task_name` with a period (.) to specify the current task.

JasperGold Apps Command Reference Manual

General Commands

```
[-env | -task task_name] -remove  
[-silent] signal_name
```

Remove the stopat from the specified signal.

- Use `-env` to remove a global stopat.
- Use `-task` to remove a stopat in the specified task.
- If you do not use `-env` or `-task`, this command applies to the current task.
- Use `-silent` to return Tcl return values without printing to the screen.

Note: The tool returns `true` if it removed the stopat and `false` if it did not.

```
[-env | -task task_name] -clear
```

Remove the specified set of stopats.

- Use `-env` to remove global stopats.
- Use `-task` to remove stopats from the specified task.
- If you do not use `-env` or `-task`, this command applies to the current task.

```
-reset signal_list [-remove]
```

Create and manage stopats that are only valid during reset analysis.

- Use `-reset <signal_list>` to create a reset-specific stopat.
- Use `-reset <signal_list> -remove` to remove a reset-specific stopat.

Note: `stopat -reset` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

```
-reset (-list | -clear)
```

List or remove all declared reset-specific stopats.

Return

`remove` returns `true` or `false`.

JasperGold Apps Command Reference Manual

General Commands

Examples

```
% stopat signalA  
% stopat {ing0} {ing1} {ing2} {ing3}
```

See Also

[justify](#)

suggest

Description

Use this command to analyze the current analysis region and suggest choices for potential justification or assumptions.

This command considers the last proof attempt and examines the design to make recommendations on boundary nets that you should examine further. The most likely candidates are those for which the tool finds a potential conflict in a signal that contributes to the previous `cex` or `ar_cex` proof. The JasperGold Visualize™ window highlights these nets.

You can use `justify` and `prove` to ask the engine to justify the questionable signal values for these choices and modify the analysis region accordingly. As an alternative, you can add assumptions about these signals (instead of modifying the analysis region). Typically, you only want to select a small subset of the provided choices to continue the analysis.

This command is context-sensitive (it applies to the current task).

Syntax

```
suggest (<signal_name> <time_step>) +
```

Argument	Definition
<code>(signal time_step) +</code>	Generate a list of action commands for justifying the selected <code>signal-time_step</code> combination.

Return

No value returned

Examples

```
% suggest
```

See Also

[assume](#)
[justify](#)

JasperGold Apps Command Reference Manual

General Commands

[prove](#)

[visualize](#)

task

Description

Use this command to create, link, copy, and get information about tasks or to set a task as the current task.

Syntax

[Getting the Current Task's Name](#)

```
task (no arguments)
```

[Task Creation Switches](#)

```
task -create [-set] <new_task_name>
    [-source_task <source_task_name>]
    [-set]
    [-copy_all]
    [-copy_abstractions (all | counters | cdc | init_value)+]
    [-copy Assumes]
    [-copy Asserts]
    [-copy Covers]
    [-copy Related_Covers]
    [-copy Stopats]
    [-copy <property_name>+ [-regexp]]
```

[Task Inheritance Switches](#)

```
task -link (<task_name>)+ [-to <inheriting_task_name>] [-silent]
task -unlink (<task_name>)+ [-to <inheriting_task_name>] [-silent]
```

[Task Set/Remove Switches](#)

```
task -set <task_name>
task -remove [<task_name>]
```

JasperGold Apps Command Reference Manual

General Commands

Utility Switches

```
task -list [-silent]
|-show [<task_name>] [-silent]
|-num_asserts [<task_name>]
|-num_covers [<task_name>]
|-num_assumes [<task_name>]
```

Argument Definition

Getting the Current Task's Name

no arguments

Return the name of the current task.

Task Creation Switches

```
-create new_task_name
[-source_task source_task_name] [-set] [-copy_all]
[-copy_abstractions (all | counters | cdc | init_value)+]
[-copy_assumes] [-copy_asserts] [-copy_covers] [-copy_related_covers]
[-copy_stopats] [-copy property_name+ [-regexp]]
```

Create a new interactive task with the specified name.

Note: With one exception, you can specify the *new_task_name* at any point after the `-create` switch. You cannot specify the *new_task_name* after `-copy`.

- Use `-set` to set the newly created task as the current task.
- Use `-source_task` to copy contents from the specified task instead of the current task.
- Use `-copy_all` to copy all assertions, assumptions, covers, hard stopats, and soft stopats from the current or specified task.
- Use `-copy_abstractions` to copy one or more abstraction types from the current or specified task:
 - `all` – counter, cdc, and `init_value`.
 - `counter` – counter abstractions. See “[Abstracting Counters](#)” on page 146.
 - `cdb` – clock domain crossing abstractions. See “[Abstracting CDC Flops](#)” on page 147.
 - `init_value` – initial value abstractions.

JasperGold Apps Command Reference Manual

General Commands

-create (Continued)

- Use -copy_assumes to copy all assumptions from the current or specified task.
- Use -copy_asserts to copy all assertions from the current or specified task.
- Use -copy_covers to copy all cover directives from the current or specified task.
- Use -copy_related_covers to copy all automatically extracted related covers for the properties that are being copied.

Note: For additional information about related covers, see elaborate -create_related_covers and the command reference manual appendix “Vacuity Checking with Automatically Extracted Covers”.

- Use -copy_stopats to copy all hard and soft stopats from the current or specified task.
- Use -copy with one or more arguments to copy the specified assertions, assumptions, or covers from the current or specified task.

Note:

- The -copy switch supports wild cards (*) and (?) and regular expressions for property names.
- When using a regular expression, use the -regexp switch after the pattern. Refer to “Examples” section below.

Task Inheritance Switches

Task inheritance promotes hierarchical building of verification tasks to assist “divide and conquer” strategies in task development by creating live links between tasks. A task “inherits” the contents of one or more supporting tasks, and the inherited contents are treated as part of the inheriting task. The contents in question are: assumptions, assertions, cover directives, and hard stopats.

Note: When you remove an assumption in a supporting task, it is removed in the inheriting task that is linked to that supporting task.

```
-link (task_name) +  
  [-to inheriting_task_name]  
  [-silent]
```

Link the specified tasks to the current task.

Use the optional `-to` switch to create a link to an inheriting task that is not the current task.

Use `-silent` to turn off info messages. This switch is helpful for scripting.



Automatically extracted Proof Accelerator tasks can contain implicit assumptions that are not visible to the user, and constraining tasks with assumptions that are not visible makes it impossible to evaluate their impact.

If you are using Verification Component Proof Accelerators (for example, `jasper_scoreboard_2`) or Abstraction Component Proof Accelerators (for example, `jasper_model_fifo`), do not attempt to link a Proof Accelerator task to a non-Proof Accelerator task. Instead, you should do the following:

```
task -link my_task -to <pa_task>
```

Note: This restriction does not apply to Verification Enabler Proof Accelerators (for example, `jasper_model_async_cdc_wire`).

JasperGold Apps Command Reference Manual

General Commands

```
-unlink (task_name)+
[-to inheriting_task_name]
[-silent]
```

Remove an existing link between the specified tasks and the current task.

Use the optional `-to` switch to remove a link to an inheriting task that is not the current task.

Use `-silent` to turn off info messages. This switch is helpful for scripting.

Task Set/Remove Switches

```
-set task_name
```

Set the specified task as the current task.

```
-remove [task_name]
```

Remove the specified task (or the current task if you do not specify a `task_name`).

Note: You can only remove user-created tasks.

Utility Switches

The tool accepts the following commands when a `prove` or `visualize` process is running in the background.

```
-list [-silent]
```

Return all tasks.

Use `-silent` to turn off output to the screen and return information with Tcl return values. This option is helpful in Tcl scripts.

```
-show [task_name] [-silent]
```

Print property types and names and return property names for the current or specified task.

Use `-silent` to turn off output to the screen and return information with Tcl return values. This option is helpful in Tcl scripts.

```
-num_asserts [task_name]
```

Return the number of assertions in the current or specified task.

```
-num_assumes [task_name]
```

Return the number of assumptions in the current or specified task.

JasperGold Apps Command Reference Manual

General Commands

```
-num_covers [task_name]
```

Return the number of covers in the current or specified task.

Return

Various. With no arguments, returns current task name

Examples

```
% task
% task -create -set new_task assert:0
% task -show
% task -list
% task -set taskB

# Pattern-matching with regular expressions:
# -----
# Use the "-regexp" switch after the pattern.
% task -create "gui_copy_3" -set -source_task "<embedded>" -copy
"SPS_overflow_i.SPS_arithmetic_overflow_poi_98_prop_98" -regexp

# Copy all assertions and related covers from the current task to the new
# task named "my_task".
% task -create my_task -copy_asserts -copy_related_covers
```

See Also

No related commands

template_property

Description

Use the `template_property` command to load and manipulate property templates. The BPS App uses property templates during property synthesis. Property templates reside in libraries and may be enabled or disabled (per property or library).

Note: To scan property templates, you must enable the `template` seeker using one of the following commands:

- `scan -trace -seek template`
- `set_scan_seek_mode template`

Syntax

```
template_property -load <file_name>
  [-library <lib_name>]
  [-append]
  [-silent]

template_property -list
  [-disabled | -enabled]
  [-silent]

template_property -clear

template_property -enable <lib_or_template_name>
template_property -disable <lib_or_template_name>
template_property -show <template_name>
  [-silent]
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<code>-load file_name [-library lib_name] [-append] [-silent]</code>	<p>Load template properties from the specification file.</p> <ul style="list-style-type: none">■ Use <code>-library</code> to load properties into a user-specified library. The default library name is the specification file name.■ Use <code>-append</code> to add content from the specification file to an existing library. By default, this switch overwrites the existing content.■ Use <code>-silent</code> to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.
<code>-list [-disabled -enabled] [-silent]</code>	<p>List template properties. By default, the <code>-list</code> switch returns all template properties regardless of whether they are disabled or enabled.</p> <ul style="list-style-type: none">■ Use <code>-disabled</code> to list only disabled template properties.■ Use <code>-enabled</code> to list only enabled template properties.■ Use <code>-silent</code> to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.
<code>-clear</code>	<p>Clear (remove) all template properties.</p> <p>Note: This operation has no impact on properties synthesized by the BPS App. It only affects future <code>scan</code> command results.</p>
<code>-enable lib_or_template_name</code>	<p>Enable the specified library or template for property synthesis.</p> <p>Note:</p> <ul style="list-style-type: none">■ Templates are enabled by default.■ An <code>-enable</code> or <code>-disable</code> on a library enables or disables all templates in that library (including child libraries).

JasperGold Apps Command Reference Manual

General Commands

```
-disable lib_or_template_name
```

Disable the specified library or template for property synthesis.

```
-show template_name  
[-silent]
```

Show the details of the template property.

Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

Return

Various

Examples

```
% template_property -load my_templates.jst  
% template_property -load my_other_templates.jst -library my_templates -append  
% template_property -list  
% template_property -disable my_templates.read_align
```

See Also

[check_bps](#)
[scan](#)
[scope](#)
[set_scan_seek_mode](#)

trace

Description

Use this command to display a signal trace (waveform) illustrating a scenario considered by the last prove step. When the tool generates a trace, the next step is to determine whether this trace illustrates a real design bug. If you determine that it does not, check your `clock` and `reset` settings, property and assumptions, or if you are using manual tunneling (see “[set dst mode](#)” on page 1001), use `justify` and `prove` to continue the proof.

If there is no active Visualize window when you run this command, JasperGold Apps launches a new window to display the signal trace. Otherwise, the JasperGold Apps Visualize window displays this trace in the current (active) window.

When there are many traces, JasperGold Apps saves them to your disk in the project directory. These traces are deleted at exit time. If the tool exits unexpectedly, the saved traces might remain in the project directory.

To display the reset sequence as a prefix in the waveforms pane of the Visualize window (with negative cycle numbers), click *View – Show Reset Cycles*.

Syntax

```
trace [-name <name>]
      [-property <property_name>]
      [-sig_order <file_name>]
      |-add_sig (<signal_name> <expression>) [<row>]
      |-add_sig -file <file_name>
      |-save_vcd <vcd_file_name> [-sig_order <sig_order_file_name>]
      |-get_type
      |-get_value <signal_name> [<cycle>|<cycle:cycle>]
          [-radix (2 | 8 | 10 | 16 | bin | oct | dec | signed_dec | hex)]
trace -get_length
trace -get_stem_length
trace -get_loop_length
trace -is_looping
trace -get_looping_flops
trace -load_radix <file_name>
trace [-property <property_name>]
      (-list [-silent]
       |-show <name>)
```

JasperGold Apps Command Reference Manual

General Commands

```
| -remove <name>
| -clear
|
trace -define_color_annotation '{' '{'<signal_name> <time_step> <color>'}'+' '}'
[<annotation_name>]
trace -activate_color_annotation <annotation_name>
```

Argument	Definition
-name <i>name</i>	Assign a name (an alpha-numeric identifier) to the current trace so it can be reviewed or referenced in the future. (See <code>-show</code> .) Note: The named traces are not recoverable when they are not applicable to the latest analysis result, for example, after you add a new assumption.
-property <i>property_name</i>	Display the archived counterexample trace for the specified property.
-sig_order <i>file_name</i>	Display signals from the specified file on the current trace.
-add_sig <i>signal_name expression [row]</i> -add_sig -file <i>file_name</i>	Add the specified signal or expression at the specified row or the signals or expressions in the specified file to the trace. The top of the waveform is row 0. Note: JasperGold Apps supports SVA expressions on the command line. Refer to Appendix D, “Tcl Support for SVA Expressions” for guidance.

JasperGold Apps Command Reference Manual

General Commands

```
-save_vcd vcd_file_name  
[-sig_order sig_order_file_name]
```

Save the current trace values in VCD format to the specified *vcd_file_name*.

The VCD file preserves design hierarchy.

Use the option `-sig_order sig_order_file_name` to save only the values of signals that are included in *sig_order_file_name*.



With regard to saving a trace to a VCD file...

- If you have also used the `set_trace_optimization` command, the VCD file reflects those settings, for example:
 - If trace optimization is `ar`, then only the analysis region is saved into the VCD file. *This option is the default.*
 - If trace optimization is `standard`, then the entire design is dumped into the VCD. However, you should be aware that this setting can increase the time it takes to generate a trace.
- `-save_vcd` is not supported in conjunction with other trace switches.

```
-get_type
```

Show the current trace type (for example, violation).

```
-get_value signal_name [cycle | cycle:cycle]  
[-radix (2 | 8 | 10 | 16 | bin | oct | dec | signed_dec | hex)]
```

Display the value of the specified signal.

- Specifying cycles is optional.
 - If you specify a single cycle (for example, 7), you are asking the tool to get the value for the specified signal in cycle seven.
 - You can specify a range of cycles; for example, 2:3.
 - Use \$ to mean “the last cycle.” For example, 1:\$ is equivalent to “always” (which is the default when you do not specify cycles) and 4:\$ is equivalent to “from cycle four until the end of the trace.”
- Use `-radix` to return values in the specified radix.

JasperGold Apps Command Reference Manual

General Commands

-get_length

Display the length of the current trace.

Use `trace -get_length` without additional switches.

-get_stem_length

Get the length of the stem of the current trace. The stem is the part of the trace that is not part of the loop.

This option is related to liveness properties.

Use `trace -get_stem_length` without additional switches.

-get_loop_length

Get the length of the loop of the current trace.

For looping traces (counterexamples to liveness assertions) this value is greater than 0; for other traces, it is 0.

Use `trace -get_loop_length` without additional switches.

-is_looping

Return true if the current trace has a loop.

Counterexamples to liveness assertions have loops, other traces do not.

Use `trace -is_looping` without additional switches.

-get_looping_flops

Return a list of all the flops with looping behavior in the current liveness trace and return an empty list if there is no liveness trace.

Use `trace -get_looping_flops` without additional switches.

-load_radix *file_name*

Load and apply radices defined in the specified *file_name*.

file_name format consists of radix definition clauses and radix use clauses.

When you have an active Visualize window, you can use this feature to read custom radix files and display predefined radix values (opcodes) instead of the original values on the trace. This feature is useful for debugging in a processor setting where you want to have the instruction type displayed instead of the bits that form the opcode. Refer to [Appendix H, “Custom Radix Files.”](#)

JasperGold Apps Command Reference Manual

General Commands

Utility Switches

-list

[-silent]

Display a list of all traces applicable to the current property.

Use the optional -silent switch to turn off output to the screen and return information with Tcl return values. This option is helpful in Tcl scripts.

-show *name*

Display the specified trace for the current property.

-remove *name*

Remove the specified trace for the current property.

-clear

Remove all trace information for the current property.

Color-Annotation Switches

Use the following switches to define, activate, and name a color annotation for the Visualize window.

- Each color annotation consists of a set of one or more triplets.
- Each triplet consists of a *signal_name-time_step-color*.

```
# Define a new color annotation
trace -define_color_annotation {{a 1 red} {b 2 blue}} red_and_blue

# Activate the new color annotation
trace -activate_color_annotation red_and_blue

# Delete the color annotation
trace -define_color_annotation {} red_and_blue
```

```
-define_color_annotation signal_name
    time_step
    color...
    annotation_name
```

Define the color annotation for the specified *signal_name* at the specified *time_step* point.

The tool lists annotation names in the Visualize window toolbar *Trace Explanation* combo box.

- **signal_name time_step color** – specify this first-argument triplet as follows:
 - **signal_name** – The name of the signal to which the color annotation applies.
 - **time_step** – The time point when the color annotation applies.
 - **color** – The annotation color. Possible values are gray, blue, green, yellow, orange, and red.

To delete an annotation, specify an empty argument by including the {} with nothing enclosed.

JasperGold Apps Command Reference Manual

General Commands

-define_color_annotation (Continued)

- **annotation_name** – The name of the color annotation. If the specified name already exists, this command redefines the annotation. You cannot redefine predefined annotations, such as those from ATD and Visualize. The default name is “User Defined.”

The tool returns the name of the annotation to show you have successfully specified it.

This command is helpful for highlighting interesting parts of the trace.

-activate_color_annotation *annotation_name*

Apply the specified color annotation.

annotation_name – Any of the annotations in the *Trace Explanation* combo box (both user-defined and predefined). Use the default annotation named Off to clear the current annotation from the trace.

Return

Returns a string (identifier name) for the trace.

Examples

```
% trace  
% trace -name req_ack_bug  
% trace -add_sig pic_core.arb1 0  
% trace -load_radix radix_2b.rdx
```

See Also

[load radix file](#)
[prove](#)
[visualize](#)

view

Description

Use this command to view the contents of a specified file.

This command is particularly useful when viewing encrypted requirement files in pre-built kits designed for JasperGold Apps.

Syntax

```
view <file_name>
```

Argument	Definition
<i>file_name</i>	View the contents of the specified file.

Return

No value returned

Examples

```
% view requirement.v
```

See Also

No related commands

visualize

Description

This command supports the RTL Exploration components of JasperGold Apps. Use the `visualize` command to Visualize a property or design scenario in the form of a violation or sanity trace. You can also Visualize how the requirement description might reach a certain coverage point in the form of a cover trace. You will typically use the GUI Visualize window and access additional switches from a menu.



You can run the `visualize` command without arguments or options. If there is an active Visualize window, the command is equivalent to `visualize -replot`.

Visualize has the same engine selection and `dst_mode` capabilities as the `prove` command, for example, you can do the following:

- Use the `set_engine_mode` command at any point prior to using Visualize to specify the engine the command will use. For additional information about the default engine mode set, see [“set_engine_mode”](#) on page 1009.
- Use parallel-engine processing.
- By default, the `visualize` command uses the entire cone of influence during analysis. Use `set_dst_mode` to change the analysis region.

Recommended engines:

- Engine Ht
- Engine B

Using expressions:

The tool supports Visualize configuration for an *expression* before or after you use `-cover`, `-violation`, or `-sanity`. (See [“Examples”](#) on page 935.) In addition, JasperGold Apps supports SVA expressions on the command line. Expressions can be added incrementally to the Visualize window, as long as their inputs are inside the analysis region. Refer to [Appendix D, “Tcl Support for SVA Expressions”](#) for guidance.

Displaying reset sequence:

To display the reset sequence as a prefix in the waveforms pane of the Visualize window (with negative cycle numbers), click *View – Show Reset Cycles*.

Support for multiple Visualize windows:

The tool supports multiple Visualize windows. By default, the tool applies your `visualize` command to the current window. Many `visualize` commands support the `-window` switch to apply your command to a window other than the current window.

When generating an initial waveform, consider the following guidance for the `-new_window`, `-window`, and `-reset_window` switches:

- For the `-new_window` switch:
 - This switch generates the waveform in a new window with a tool-generated Visualize name.
 - Use `-new_window` with the optional `new_window_name` argument to specify a name for the new window.
- For the `-window` switch:
 - If you specify a name that does not yet exist, this switch renames the current window with the specified name.
 - If there are no Visualize windows, a new window opens with the specified name.
- Use the `-reset_window` switch to apply your command to the Reset Analysis window.
- `-window` and `-new_window` are mutually exclusive.

Consult the syntax in the definition section to see when these switches are supported and to learn about any exceptions to these guidelines.

Understanding Visualize semantics:

To gain a better understanding of the meaning of various `visualize` commands, refer to [Appendix K, “Understanding Visualize Semantics with SVA Equivalents.”](#)

Note:

- This command is context-sensitive (it applies to the current task).
- This command supports regular expressions for property names.

Syntax

[Generating Initial Visualize Waveforms](#)

```
visualize [-cover]
    (<expression> | (-property <property_name>) [-regexp])
    [-new_window [<new_window_name>]] [-window <window_name>]
    [-engine (<engine_mode> | {<engine_mode>+}))]
    [-proof_time <time_limit>] [-annotation '{'<annotation>'}']
    [-batch] [-silent] [-bg]
    [(-sig_order <sig_order_file_name>) |-no_signals]
    [-task <task_name>]

visualize -violation
    (<expression> | (-property <property_name>) [-regexp])
    [-new_window [<new_window_name>]] [-window <window_name>]
    [-engine (<engine_mode> | {<engine_mode>+}))]
    [-proof_time <time_limit>] [-annotation '{'<annotation>'}']
    [-batch] [-silent] [-bg]
    [(-sig_order <sig_order_file_name>) |-no_signals]
    [-task <task_name>]

visualize -sanity
    (<expression> | (-property <property_name>) [-regexp])
    [-new_window [<new_window_name>]] [-window <window_name>]
    [-engine (<engine_mode> | {<engine_mode>+}))]
    [-proof_time <time_limit>] [-annotation '{'<annotation>'}']
    [-batch] [-silent] [-bg]
    [(-sig_order <sig_order_file_name>) |-no_signals]
    [-task <task_name>]

visualize -consistency [-bg]
    [-new_window [<new_window_name>]] [-window <window_name>]
    [(-sig_order <sig_order_file_name>) |-no_signals]
    [-task <task_name>]

visualize -state_removal] [-bg]
    [-new_window [<new_window_name>]] [-window <window_name>]
    [(-sig_order <sig_order_file_name>) |-no_signals]
    [-task <task_name>]
```

[Setting a New Target](#)

```
visualize -set_target [-cover |-violation |-sanity]
    [-window <window_name>]
    (<expression> | (-property <property_name> [-regexp]))
```

JasperGold Apps Command Reference Manual

General Commands

[Configuring Waveforms](#)

```
visualize -force -soft <expression> [-name <name>]
    [-group_name <group_name>] [-annotation '{'<annotation>''}']
    [<cycle> | <cycle:cycle>]
    [-window <window_name>]

visualize -force (<expression> | (-property <property_name> [-regexp]))
    [<cycle>|<cycle:cycle>] [-name <name>]
    [ (-after <expression_1>
        | (-after -after_property <property_name> [-regexp])
        | (-before <expression_1>)]
       [-group_name <group_name>]
       [-annotation '{'<annotation>''}'] [-loop]
       [-window <window_name>]

visualize -at_least_once (<expression> | (-property <property_name> [-regexp]))
    [(-after <expression_1>
        | (-after -after_property <property_name> [-regexp])]
    [-before <expression_1>]
    [<cycle>|<cycle:cycle>] [-name <name>] [-group_name <group_name>]
    [-annotation '{'<annotation>''}'] [-loop]
    [-window <window_name>]

visualize -not -soft <expression> [-name <name>]
    [-group_name <group_name>]
    [-annotation '{'<annotation>''}'] [<cycle> | <cycle:cycle>]
    [-window <window_name>]

visualize -not (<expression> | (-property <property_name> [-regexp]))
    [<cycle>|<cycle:cycle>] [-name <name>]
    [ (-after <expression_1>
        | (-after -after_property <property_name> [-regexp])
        | (-before <expression_1>)]
       [-group_name <group_name>]
       [-annotation '{'<annotation>''}']
       [-window <window_name>]

visualize -min_length <length> [-window <window_name>]
visualize -max_length <length> [-window <window_name>]
visualize -freeze <cycle> [-window <window_name>]

visualize -replot [-new_window [<new_window_name>]] [-force]
    [-engine (<engine_mode> | {<engine_mode+>})) [-proof_time <time_limit>]
    [-quiet]
    [-min_variation] [-batch] [-silent] [-bg]
    [(-sig_order <sig_order_file_name>) |-no_signals]
    [-window <window_name>]
```

JasperGold Apps Command Reference Manual

General Commands

```
visualize -go_backward [-new_window [<new_window_name>]] [-force]
    [(-sig_order <sig_order_file_name>) |-no_signals]
    [-proof_time <time_limit>]
    [-min_variation] [-batch] [-silent] [-bg]
    [-window <window_name>]

visualize -go_forward [-new_window [<new_window_name>]] [-force]
    [(-sig_order <sig_order_file_name>) |-no_signals]
    [-proof_time <time_limit>]
    [-min_variation] [-batch] [-silent] [-bg]
    [-window <window_name>]
```

[Utility Switches for Configurations](#)

```
visualize -rename_conf <old_conf_name> <new_conf_name> [-window <window_name>]
    |-remove_conf <conf_name> [-window <window_name>]
    |-disable_conf <conf_name> [-window <window_name>]
    |+enable_conf <conf_name> [-window <window_name>]
    |-offset_conf <conf_name> <offset_value> [-window <window_name>]
    |-clear_conf [-window <window_name>]

visualize -replace_conf (-force -soft |-not -soft) <expression>
    [<cycle>|<cycle:cycle>] -name <conf_name> [-window <window_name>]

visualize -replace_conf (-force |-not |-at_least_once)
    (<expression> |-property <property_name>)
    [(-after <expression_1>) | (-after -after_property <property_name>
    [-regexp]) [-before <expression_1>]
    [<cycle>|<cycle:cycle>]
    -name <conf_name> [-loop] [-window <window_name>]

visualize -replace_conf -signal <signal_name> -sequence
    ('{}<value>'['{}<cycles>'']) [ <value>['{}<cycles>'']]*)' [<cycle>])
    [-name <conf_name>] [-window <window_name>]
```

[Utility Switches – General](#)

```
visualize -list [-silent] [-all | <group_name>] [-window <window_name>]

visualize -list -all_windows [-silent]

visualize -save [-script] [-force |-append]
    [-setup_only [-window <window_name>]
    |-config_only [-window <window_name>]
    [-add_hier_path <path_name> |-remove_hier_path <path_name>]]
    <file_name>

visualize -save -init_state <file_name> [-cycle <N>] [-force]
    [-include_internal] [-include_encrypted] [-bit_blast]
    [-window <window_name>]
```

JasperGold Apps Command Reference Manual

General Commands

```
visualize -save -reset_sequence [-force] <file_name>
    [-include_internal] [-include_encrypted] [-bit_blast]
    [-window <window_name>]

visualize -save [-vcd] [-imp |-spec] <vcd_file_name> [-force]
    [-sig_order <sig_order_file_name>]
    [-reset_window |-window <window_name>]

visualize -save -fsdb <fsdb_file_name> [-force]
    [-time_scale [<N>] [time_unit]]
    [-sig_order <sig_order_file_name>]
    [-window window_name |-reset_window]

visualize -save -shm <shm_directory_name> [-force]
    [-time_scale [N] [time_unit]]
    [-sig_order sig_order_file_name]
    [-reset_window |-window window_name]

visualize -save_recipe <waveform_name> [-task <task_name>] [-save_signal_order]
    [-save_behavior] [-description <description>] [-batch] [-png]

visualize -run_recipe <element_id>

visualize -load (-vcd |-fsdb |-shm) <trace_file_name>
    [-time_scale [N][time_unit]]
    [-hier_path <path_difference>]
    [-start_time <N>] [-end_time <N>]
    [-hier_map -from <source_hier> -to <destination_hier>]
    [-window <window_name>]
    [-batch]

visualize -confirm (-vcd |-fsdb |-shm) <trace_file_name>
    [-time_scale [N][time_unit]]
    [-hier_path <path_difference>]
    [-start_time <N>] [-end_time <N>]
    [-hier_map -from <source_hier> -to <destination_hier>]
    [-window <window_name>]
    [-batch]

visualize -reset

visualize -show [-property property_name] [-silent]

visualize -save_sig_order <file_name>
    [(-window <window_name>) |-reset_window] [-save_highlight]

visualize -set_silent_mode (true | false)

visualize -get_silent_mode

visualize -undo |-redo [-window <window_name>]

visualize -highlight -diff <other_window_name>
    [-window <window_name>]

visualize -highlight -diff (-prev_plot |-next_plot)
    [-window <window_name>]
```

JasperGold Apps Command Reference Manual

General Commands

```
visualize -relevant_logic <signal> <cycle>
    [-configuration ([control_path | data_path] [undriven_only]
                    [filter_async_reset] [show_clock] [show_equal]
                    [show_twins_diff] [instance_ports_only]) +
     [ -iteration_depth <N> |-num_whys <N>]
     [(-window <window_name>) |-reset_window]]

visualize -clear_all [-window <window_name>]

visualize -sig_order <sig_order_file_name>
    [(-window <window_name>) |-reset_window]

visualize -no_signals [-window <window_name>]

visualize -new_window [<new_window_name>]
    [-copy [-window <old_window_name>]]

visualize -rename_window <old_name> <new_name>

visualize -remove_window (<window_name> |-reset_window)

visualize -set_current_window <window_name>

visualize -get_current_window

visualize -remove_current_window

visualize -add_sig (<signal_name> | <expression>) [<row>]
    |-file <file_name>
    [(-window <window_name>) |-reset_window]

visualize (-get_type
           |-get_value <signal_name> [<cycle>|<cycle:cycle>]
           |-radix (2 | 8 | 10 | 16 | bin | oct | dec | signed_dec | hex)
           |-get_length
           |-get_stem_length
           |-get_loop_length
           |-get_looping_flops)
    [-window <window_name>]

visualize -is_looking [-window <window_name>]

visualize (-load_radix <file_name>) [-window <window_name>]

visualize -get_marker_location (current | primary | secondary)
    [-window <window_name>]
```

Color-Annotation Switches

```
visualize (-define_color_annotation
           '{' ''{'<signal_name> <time_step> <color>'}' '+' '}'
           annotation_name
           |-activate_color_annotation <annotation_name>
           [(-window <window_name>) |-reset_window]
```

JasperGold Apps Command Reference Manual

General Commands

Signal Highlight Switches

```
visualize -define_signal_highlight '{}' '{{<signal_name> <color>}}'+ '}'
[(-window <window_name>) [-reset_window] [-add |-remove]
```

Interactive Constraint Extraction

```
visualize -extract_conf [-reason [-depth <depth_value>] [-transitive]] [-value]
'{} '{'<signal_name> [<cycle> | <cycle>:<cycle>] '+' '}'
[<cycle>| <cycle>:<cycle>] [-name <conf_name>]
[-window <window_name>]

visualize -concretize_conf '{'<conf_name>'}'
[-window <window_name>]
```

Signal and Expression Exploration

```
visualize -explore (<signal> |-property <property_name>) <window> <iterations>
[-new_window <new_window_name>] [-batch] [-silent] [-bg]
[-wide_signal_threshold <threshold>]
[-window <window_name>]
```

Waveform Analysis

```
visualize -check_props
[-cover |-violation]
[-update_gui]
[-names_only]
[-filter <property_name_filter_list> [-regexp]]
[-silent]
[-window <window_name>]

visualize -compare <signalA_name> <signalB_name>
[-shift_cycles <cycles>]
[-ignore_x] [-ignore_values <value>+] [-window <window_name>]
```

State-Space Tunneling Visualize

```
visualize -sst
[-property <property_name> [-with_proven]] | [-task <task_name>]
[-window <window_name>]
```

Design-Space Tunneling Visualize

```
visualize -dst (true | false) [-window <window_name>]
```

JasperGold Apps Command Reference Manual

General Commands

[Quiet Visualize](#)

```
visualize -quiet ((true [<cycle:cycle>]) | false) [-window <window_name>]
```

[Configuration Grouping](#)

```
visualize -group
  ('{}<conf_name+'>' [-name <group_name>]
   |-expand <group_name>
   |-remove <group_name>
   |-rename <old_group_name> <new_group_name>
   |-replace '{}<conf_name>+'>' -name <group_name>
   |-add_conf <conf_name> <group_name>
   |-remove_conf <conf_name>
   |-merge <group_name_1> <group_name_2> [-name <final_group_name>]
  )
  [-window <window_name>]
```

[Annotating Targets and Configurations](#)

```
visualize -set_annotation
  (-target |-conf <conf_name>) {<annotation>}
  [-window <window_name>]

visualize -get_annotation
  (-target |-conf <conf_name>) [-window <window_name>]
```

[Specifying a Waveform Sequence in a Single Command](#)

```
visualize -signal <signal_name>
  -sequence
    ('{}<value>['{}<cycles>'']) [<value>['{}<cycles>'']]*'>'
    [<cycle>]
  )
  [-name <name>] [-group_name <group_name>]
  [-window <window_name>]
```

[Showing Relevant Behaviors](#)

```
visualize -relevant_props <signal_name> <cycle> [<depth_value>] [-silent]
  [-window <window_name>]
```

[RAM Abstraction](#)

```
visualize -init_PA [<ram_instance>] [-depth <depth_value>]
  [-window <window_name>]
```

JasperGold Apps Command Reference Manual

General Commands

```
visualize -suggest_PA  
    [-window <window_name>]  
  
visualize -justify_PA [<ram_instance>]  
    [-window <window_name>]  
  
visualize -auto_justify_PA [<ram_instance>]  
    [-window <window_name>]
```

Visualize Task Manipulation

```
visualize -set_task <task_name>  
    [-window <window_name>]  
  
visualize -get_task [-window <window_name>]
```

Switches for Background Visualize

```
visualize -stop [-thread <thread_id>]  
visualize -wait  
visualize -status [-silent]
```

Switches for Task Context Visualize

```
visualize -assume ( [-bound <N>] <expression> | -constant <signal_name> )  
    [-name <name>]  
    [-annotation <'{'annotation'}'>]  
    [-replace] [-window <window_name>]  
  
visualize -justify  
    ( <signal_name>+  
        [-conflict (<signal_name> <time_step>)+]  
    |-back_to_signal <signal_name>+  
        [-from_signal <signal_name>+ | -from_property <property>+]  
    |-back_to_property <property>+  
        [-from_signal <signal_name>+ | -from_property <property>+]  
    |-back_to_instance <instance>+  
        [-from_signal <signal_name>+ | -from_property <property>+]  
        [-until_inputs]  
    )  
    [-name <name>]  
    [-annotation <'{'annotation'}'>]  
    [-replace] [-window <window_name>]  
  
visualize -stopat <signal_or_instance_name>+  
    [-name <name>]  
    [-annotation <'{'annotation'}'>]  
    [-replace] [-window <window_name>]
```

JasperGold Apps Command Reference Manual

General Commands

```
visualize -abstract -counter <signal_name>+
    [-name <name>]
    [-annotation <'{'annotation'}'>]
    [-replace] [-remove] [-window <window_name>]

visualize -set_as_AR [-window <window_name>]
```

Why

```
visualize -why -target <signal_name> -cycle <cycle>
    [-transitive]
    [-ignore_connected_assumptions]
    [-simple_buffer_transitive]
    [-stop_at_hierarchy_change]
    [-window <visualize_window>] | [-reset_window]
```

Argument	Definition
Generating Initial Visualize Waveforms	
<pre>-cover expression -property property_name -regexp -new_window new_window_name -window window_name -engine engine_mode {engine_mode+} -proof_time time_limit -annotation {annotation} -batch -silent -bg -sig_order sig_order_file_name -no_signals -task task_name</pre>	<p>Generate an initial waveform for the specified expression or property to Visualize how a design might cause the requirement description to enter a certain state. (Implicitly, the tool also uses a <code>-clear_all</code> on previous Visualize data if the target differs from the previous target.)</p> <p>Note: By default Visualize analyzes the cone of influence. Use the <code>set_dst_mode</code> command to change the analysis region.</p> <p>Implementation guidance:</p> <ul style="list-style-type: none">■ Replace <code><expression></code> with an expression for a cover point or use <code>-property</code> to Visualize a property in the current task. (You cannot use the <code>-cover</code> switch with an enabled assumption.) <p>Note: The <code>-property</code> switch supports wildcards and regular expressions. An error results unless there is exactly one matching property. When using a regular expression, use the <code>-regexp</code> switch after the pattern:</p> <pre>visualize... -property {pattern} -regexp</pre>

-cover (Continued)

- Use `-engine` to specify one or more engine modes when performing `Visualize`. Use curly braces to specify multiple engine modes. See “[set_engine_mode](#)” on page 1009 for available arguments and restrictions.

- Use `-proof_time` to set a limit on the time for this command. See “[set_prove_time_limit](#)” on page 1189.

- Use `-annotation` to annotate the target with the description enclosed in the curly braces.

View annotations in the `Visualize` window by using the mouse to hover over the target name and reveal a tooltip. You can print annotations to the screen with the `-get_annotation` switch.

- Use `-batch` to prevent the tool from displaying the waveform it generates, which improves performance.

- Use `-silent` to turn off the output to the screen and return information with a Tcl return value.

Note: `-silent` does not stop the trace from being shown in the `Visualize` window. You must use `-batch` to suppress the trace.

- Use `-bg` to run the command in the background and continue to accept and run new commands.

- Use `-sig_order` to generate a trace that displays only the signals in the specified signal order file.

- Use `-no_signals` to load a trace without displaying signals.

- Use `-task` to apply the command to the specified task instead of the current task. Do not use the `-task` and `-property` switches in the same command. If you need to specify both a `property_name` and `task_name`, use the `-property` switch and prepend the property name with `<task>::`. (Replace `<task>` with an actual task name.)

JasperGold Apps Command Reference Manual

General Commands

```
-violation expression | -property property_name -regexp  
-new_window new_window_name  
-window window_name  
-engine engine_mode | {engine_mode+}  
-proof_time time_limit  
-annotation {annotation}  
-batch  
-silent  
-bg  
-sig_order sig_order_file_name | -no_signals  
-task task_name
```

Generate an initial waveform for the specified expression or property to Visualize how a design with bugs might fail the requirement. (Implicitly, the tool uses a `-clear_all` on previous Visualize data if the target differs from the previous target.)

Note: By default Visualize analyzes the cone of influence. Use the `set_dst_mode` command to change the analysis region.

Implementation guidance:

- Replace `<expression>` with an expression for a cover point or use `-property` to Visualize a property in the current task. (You cannot use the `-violation` switch with an enabled assumption.)

Note: The `-property` switch supports wildcards and regular expressions. An error results unless there is exactly one matching property. When using a regular expression, use the `-regexp` switch after the pattern:

```
visualize... -property {pattern} -regexp
```

- Use `-engine` to specify one or more engine modes when performing Visualize. Use curly braces to specify multiple engines. See “[set_engine_mode](#)” on page 1009 for available arguments and restrictions.
- Use `-proof_time` to set a limit on the time for this command. See “[set_prove_time_limit](#)” on page 1189.

-violation (Continued)

- Use `-annotation` to annotate the target with the description enclosed in the curly braces.

View annotations in the Visualize window by using the mouse to hover over the target name and reveal a tooltip. You can print annotations to the screen with the `-get_annotation` switch.
- Use `-batch` to prevent the tool from displaying the trace it generates, which improves performance.
- Use `-silent` to turn off the output to the screen and return information with a Tcl return value.

Note: `-silent` does not stop the trace from being shown in the Visualize window. You must use `-batch` to suppress the trace.
- Use `-bg` to run the command in the background and continue to accept and run new commands.
- Use `-sig_order` to generate a trace that displays only the signals in the specified signal order file.
- Use `-no_signals` to load a trace without displaying signals.
- Use `-task` to apply the command to the specified task instead of the current task. Do not use the `-task` and `-property` switches in the same command. If you need to specify both a `property_name` and `task_name`, use the `-property` switch and prepend the property name with `<task>::` (Replace `<task>` with an actual task name.)

JasperGold Apps Command Reference Manual

General Commands

```
-sanity expression | -property property_name -regexp  
-new_window new_window_name  
-window window_name  
-engine engine_mode | {engine_mode+}  
-proof_time time_limit  
-annotation {annotation}  
-batch  
-silent  
-bg  
-sig_order sig_order_file_name | -no_signals  
-task task_name
```

Generate an initial waveform for the specified expression or property to Visualize how a design might operate under the obligation imposed by the specified requirement or design functionality. (Implicitly, the tool also uses a `-clear_all` on previous Visualize data if the target differs from the previous target.)

Note: By default Visualize analyzes the cone of influence. Use the `set_dst_mode` command to change the analysis region.

Implementation guidance:

- Replace `<expression>` with an expression for a cover point or use `-property` to Visualize a property in the current task.

Note: The `-property` switch supports wildcards and regular expressions. An error results unless there is exactly one matching property. When using a regular expression, use the `-regexp` switch after the pattern:

```
visualize... -property {pattern} -regexp
```

- Use `-engine` to specify one or more engine modes when performing Visualize. Specify a single engine or use curly braces to specify multiple engines. See “[set_engine_mode](#)” on page 1009 for available arguments and restrictions.
- Use `-proof_time` to set a limit on the time for this command. See “[set_prove_time_limit](#)” on page 1189.

JasperGold Apps Command Reference Manual

General Commands

-sanity (Continued)

- Use `-annotation` to annotate the target with the description enclosed in the curly braces.

View annotations in the Visualize window by using the mouse to hover over the target name and reveal a tooltip. You can print annotations to the screen with the `-get_annotation` switch.
- Use `-batch` to prevent the tool from displaying the trace it generates, which improves performance.
- Use `-silent` to turn off the output to the screen and return information with a Tcl return value.

Note: `-silent` does not stop the trace from being shown in the Visualize window. You must use `-batch` to suppress the trace.
- Use `-bg` to run the command in the background and continue to accept and run new commands.
- Use `-sig_order` to generate a trace that displays only the signals in the specified signal order file.
- Use `-no_signals` to load a trace without displaying signals.
- Use `-task` to apply the command to the specified task instead of the current task. Do not use the `-task` and `-property` switches in the same command. If you need to specify both a `property_name` and `task_name`, use the `-property` switch and prepend the property name with `<task>::`. (Replace `<task>` with an actual task name.)

Do not attempt to use `-sanity` with covers. (Instead, use `visualize -cover`.)

JasperGold Apps Command Reference Manual

General Commands

```
-consistency -bg  
-new_window new_window_name  
-window window_name  
-sig_order sig_order_file_name | -no_signals  
-task task_name
```

Generate an initial waveform to Visualize how assumptions might be inconsistent with the design and each other.

- Use `-bg` to run the command in the background and continue to accept and run new commands.
- Use `-sig_order` to generate a trace that displays only the signals in the specified signal order file.
- Use `-no_signals` to load a trace without displaying signals.
- Use `-task` to apply the command to the specified task instead of the current task.

Note:

- This command automatically sets the `maxlength` to 32 unless you specify a non-infinite `maxlength` before running this command. Use `visualize -max_length` to specify a non-infinite `maxlength`.
- To learn more about checking assumptions for consistency, see [“check_assumptions”](#) on page 224.

```
-state_removal -bg  
-new_window new_window_name  
-window window_name  
-sig_order sig_order_file_name | -no_signals  
-task task_name
```

Generate an initial waveform to Visualize how assumptions might remove states.

- Use `-bg` to run the command in the background and continue to accept and run new commands.
- Use `-sig_order` to generate a trace that displays only the signals in the specified signal order file.
- Use `-no_signals` to load a trace without displaying signals.
- Use `-task` to apply the command to the specified task instead of the current task.

Note: To learn more about checking assumptions for state removal, see [“check_assumptions”](#) on page 224.

Setting a New Target

```
-set_target [-cover |-violation |-sanity]
(expression | (-property property_name [-regexp])) 
[-window window_name]
```

Replace the target for the current or specified Visualize window (or set an initial target if no Visualize window exists), but do not replot.

Note: If some of the trace is frozen and you attempt to target an X-Prop property, the tool generates an error.

Implementation guidance:

- The default is `-cover` if you do not specify a switch (`-cover`, `-violation`, or `-sanity`).
- Replace `<expression>` with an expression for a cover point.
- The `-property` switch supports wildcards and regular expressions. An error results unless there is exactly one matching property. When using a regular expression, use the `-regexp` switch after the pattern:

```
visualize... -property {pattern} -regexp
```

This command preserves the following:

- All previously added configurations (`-force`, `-at_least_once`, `-min_length`, `-max_length`).
- The Visualize stack (modified and replotted versions of the current trace).

Configuring Waveforms

```
-force -soft expression  
  -name name -group_name group_name  
  -annotation {annotation} cycle | cycle:cycle  
  -window window_name
```

Give preference to generating waveforms where the specified expression is satisfied in as many cycles as possible.

Note: When there are multiple `-force -soft` configurations, they may conflict with each other. The generated trace presents a local maxima satisfying as many as possible in as many cycles possible; however, soft constraints are optional, and will never affect (extend) trace length.

Implementation guidance:

- Use the `-name` switch to specify a *name* for this configuration.
- Use the `-group_name` switch to specify a *group_name* for this configuration.
- Use `-annotation` to annotate the configuration with the description enclosed in the curly braces.

View annotations in the Visualize window by using the mouse to hover over the configuration name and reveal a tooltip. You can print annotations to the screen with the `-get_annotation` switch.
- Specifying cycles is optional. If you do not specify an cycle, that is equivalent to “always.”
 - If you specify a single cycle (for example, 7), you are asking the tool to prefer that the expression generated by `-force` is true in cycle seven.
 - Use \$ to mean “the last cycle” and \$-N to mean “N cycles before the last cycle.”
 - When `cycle_1=1`, the tool will give preference to showing the expression being covered between the first cycle of the trace (the first cycle after reset) and `cycle_2`; when `cycle_2=$`, the tool gives preference to showing the expression being covered between `cycle_1` and the last cycle of the trace.

Note: If you use `-force -soft` to create a soft constraint, QuietTrace will not attempt to force the specified signal to constant low.

JasperGold Apps Command Reference Manual

General Commands

```
-force (expression | (-property property_name [-regexp]))  
[cycle | cycle_1:cycle_2] [-name name]  
[-after expression_1)  
| (-after -after_property property_name [-regexp])  
| (-before expression_1)  
[-group_name group_name]  
[-annotation {annotation}]  
[-loop] [-window window_name]
```

Ensure the specified expression or property is true from *cycle_1* to *cycle_2* of the trace.

The generated trace will have at least as many cycles as *cycle_1*. If you do not specify cycles, that is equivalent to “always.”

Implementation guidance:

- Specify an *expression* or *-property property_name* for this configuration.
- Use *-after* or *-after -after_property* to configure a trace that shows one scenario occurring after another has occurred. When using these after switches, the cycle count begins after *expression_1* (or *property_name*) instead of after reset.
- The *-property* and *-after -after_property* switches support wildcards and regular expressions. An error results unless there is exactly one matching property. When using a regular expression, use the *-regexp* switch after the pattern:

```
visualize... -property {pattern} -regexp
```

- Use *-before* to configure a trace that shows one scenario occurring before another has occurred. When using *-before*, the cycle count begins before *expression_1* instead of after reset.

-force (Continued)

- When *cycle_1*=1, the expression will be true starting in the first cycle of the trace (the first cycle after reset); when *cycle_2*=\$, the expression will be true until the last cycle of the trace.
- Use the cycle range \$:\$ to force the expression or property at the last cycle of the generated trace. This range is not supported with -after or -before.
- Use negative cycle numbers to refer to a range that counts back from the end of the trace, for example:
 - -7:-5 refers to a range from 7 cycles before the end of the trace through the cycle that occurs 5 cycles before the end of the trace.
 - 1:-5 refers to a range from cycle 1 through the cycle that occurs 5 cycles before the end of the trace.
 - -5:\$ refers to a range from 5 cycles before the end of the trace to the end of the trace.
- For liveness properties, do not specify cycles or ranges. You must specify the full range {1:\$} or omit cycle specification.
- Use -name to specify a *name* for this configuration.
- Use -group_name to specify a *group_name* for this configuration.
- Use -annotation to annotate the configuration with the description enclosed in the curly braces.

View annotations in the Visualize window by using the mouse to hover over the configuration name and reveal a tooltip. You can print annotations to the screen with the -get_annotation switch.
- Use -loop when adding the constraint to a liveness trace.

JasperGold Apps Command Reference Manual

General Commands

```
-at_least_once (expression | (-property property_name [-regexp]))  
  [ (-after expression_1)  
    | (-after -after_property property_name [-regexp]) ]  
  [-before expression_1]  
  [cycle | cycle_1:cycle_2]  
  [-name name]  
  [-group_name group_name]  
  [-annotation {annotation}]  
  [-loop] [-window window_name]
```

Ensure the expression or property is true at least once between *cycle_1:cycle_2* of the trace.

The generated trace will have at least as many cycles as *cycle_1*.

Implementation guidance:

- Specify an *expression* or -property *property_name* for this configuration.
- Use -after or -after -after_property to configure a trace that shows one scenario occurring after another has occurred. When using these after switches, the cycle count described below begins after *expression_1* (or *property_name*) instead of after reset.
- The -property and -after -after_property switches support wildcards and regular expressions. An error results unless there is exactly one matching property. When using a regular expression, use the -regexp switch after the pattern:

visualize... -property {pattern} -regexp.

- Use -before to configure a trace that shows one scenario occurring before another has occurred. When using -before, the cycle count begins before *expression_1* instead of after reset.
- Use -after together with -before to configure a trace that shows one scenario occurring between two others. When using both -after and -before, the cycle count described below actually determines the number of times the primary behavior will occur.

-at_least_once (Continued)

- When *cycle_1*=1, the expression will be covered between the first cycle of the trace (the first cycle after reset) and *cycle_2*; when *cycle_2*=\$, the expression will be covered between *cycle_1* and the last cycle of the trace.
- Use the cycle range \$: \$ to see the expression or property covered at the last cycle of the generated trace. This range is not supported with -after or -before.
- Use negative cycle numbers to refer to a range that counts back from the end of the trace, for example:
 - -7:-5 refers to a range from 7 cycles before the end of the trace through the cycle that occurs 5 cycles before the end of the trace.
 - 1:-5 refers to a range from cycle 1 through the cycle that occurs 5 cycles before the end of the trace.
 - -5:\$ refers to a range from 5 cycles before the end of the trace to the end of the trace.
- For liveness properties, do not specify cycles or ranges. You must specify the full range {1:\$} or omit cycle specification.
- Use the -name switch to specify a *name* for this configuration.
- Use the -group_name switch to specify a *group_name* for this configuration.
- Use -annotation to annotate the configuration with the description enclosed in the curly braces.

View annotations in the Visualize window by using the mouse to hover over the configuration name and reveal a tooltip. You can print annotations to the screen with the -get_annotation switch.

- Use -loop when adding the constraint to a liveness trace.
-

JasperGold Apps Command Reference Manual

General Commands

```
-not -soft expression
  -name name
  -group_name group_name
  -annotation {annotation}
    cycle | cycle_1:cycle_1
  -window window_name
```

Prefer traces where the opposite of the specified expression is satisfied in as many cycles as possible.

Note: When there are multiple `-not -soft` configurations, they may conflict with each other. The trace given is guaranteed to present a local maxima satisfying as many as possible in as many cycles as possible; however, soft constraints are optional, and will never affect (extend) trace length.

Implementation guidance:

- Use the `-name` switch to specify a *name* for this configuration.
- Use the `-group_name` switch to specify a *group_name* for this configuration.
- Use `-annotation` to annotate the configuration with the description enclosed in the curly braces.

View annotations in the Visualize window by using the mouse to hover over the configuration name and reveal a tooltip. You can print annotations to the screen with the `-get_annotation` switch.
- Specifying cycles is optional. If you do not specify an cycle, that is equivalent to “always.”
 - If you specify a single cycle (for example, `7`), you are asking the tool to prefer the opposite of the expression in cycle seven.
 - Use `$` to mean “the last cycle” and `$-N` to mean “N cycles before the last cycle.”
 - When `cycle_1=1`, the tool will give preference to showing the opposite of the expression being covered between the first cycle of the trace (the first cycle after reset) and `cycle_2`; when `cycle_2=$`, the tool gives preference to showing the opposite of the expression being covered between `cycle_1` and the last cycle of the trace.

Note: If you use `-not -soft` to create a soft constraint, QuietTrace will not artificially display the specified signal as constant low.

JasperGold Apps Command Reference Manual

General Commands

```
-not (expression | (-property property_name [-regexp]))  
[cycle | cycle_1:cycle_2] [-name name]  
[(-after expression_1)  
| (-after -after_property property_name [-regexp])  
| (-before expression_1)  
[-group_name group_name]  
[-annotation {annotation}]  
[-window window_name]
```

Ensure the opposite of the expression or property is true for the specified *cycle* or range of cycles (*cycle_1:cycle_2*) of the trace.

The generated trace will have at least as many cycles as *cycle_1*. If you do not specify cycles, that is equivalent to “always.”

Implementation guidance:

- Specify an *expression* or *-property property_name* for this configuration.
- Use *-after* or *-after -after_property* to configure a trace that shows one scenario occurring after another has occurred. When using these after switches, the cycle count begins after *expression_1* (or *property_name*) instead of after reset.
- The *-property* and *-after -after_property* switches support wildcards and regular expressions. An error results unless there is exactly one matching property. When using a regular expression, use the *-regexp* switch after the pattern:

visualize... -property {pattern} -regexp.

- Use *-before* to configure a trace that shows one scenario occurring before another has occurred. When using *-before*, the cycle count begins before *expression_1* instead of after reset.

JasperGold Apps Command Reference Manual

General Commands

-not (Continued)

- When *cycle_1*=1, the opposite of the expression will be covered between the first cycle of the trace (the first cycle after reset) and *cycle_1*; when *cycle_2*=\$, the opposite of the expression will be covered between *cycle_1* and the last cycle of the trace.
- Use the cycle range \$:\$ to force the opposite of the expression or property at the last cycle of the generated trace. This range is not supported with -after or -before.
- Use the -name switch to specify a *name* for this configuration.
- Use the -group_name switch to specify a *group_name* for this configuration.
- Use -annotation to annotate the configuration with the description enclosed in the curly braces.

View annotations in the Visualize window by using the mouse to hover over the configuration name and reveal a tooltip. You can print annotations to the screen with the -get_annotation switch.

-min_length *length*
-window *window_name*

Extend the trace to have the specified minimum length.

The tool clears this specification if you use visualize -clear_all or visualize -clear_conf.

-max_length *length*
-window *window_name*

Set the specified length as the maximum trace length the Visualize engine will attempt to generate (default \$, the last cycle).

To remove the upper bound on the length of the trace, use \$ for *length*.

The tool clears this specification if you use visualize -clear_all or visualize -clear_conf.

JasperGold Apps Command Reference Manual

General Commands

```
-freeze cycle
    -window window_name
```

Lock the values of all the inputs and the initial values of the registers (even those that are not shown in the Visualize window) from the beginning of the trace through the specified *cycle*.

The `-freeze` switch accepts negative numbers for the *cycle* argument. Using a negative number indicates that you want to count back from the end of the trace and freeze all cycles up to that point. The following examples use a 10-cycle trace:

- `-4` freezes the first six cycles of the trace.
- `4` freezes the first four cycles of the trace.
- `$` freezes all 10 cycles.

Note:

- Freezing is not supported with X-Prop properties or the SST flow.
 - If an assertion can be violated, different engines can generate different traces, but no engine will fail to generate a trace except when using the Visualize freeze-and-extend mode of analysis. In these cases, it is possible that, using the same command set that successfully generated a trace with one engine will fail to generate a trace with another engine because the frozen portion can be different for different engines.
-

JasperGold Apps Command Reference Manual

General Commands

```
-replot [-new_window [new_window_name]] [-force]
[-engine {engine_mode | {engine_mode+}}]
[-proof_time time_limit]
[-quiet] [-min_variation]
[-batch] [-silent] [-bg]
[(-sig_order sig_order_file_name) | -no_signals]
[-window window_name]
```

Regenerate a trace that satisfies the current configuration (that is, update the trace based on reconfiguration).

Implementation guidance:

- Use `-new_window` to copy the current window and replot it in a new window.
 - To specify the new window's name, use the optional `new_window_name` argument. If you do not specify a name, then the name is the name of the current window with `_copy` appended.
 - To copy a window other than the current window, use the `-window` switch.
- Use `-force` to ignore any cached trace core and call the engine to generate a new full trace.
- Use `-force` in conjunction with the `-engine` switch to replot using a different engine. Using `-engine` without `-force` only works if there is no cached trace core or you use the `-new_window` switch. In addition, `-engine` only works for the current `-replot` command. For future `-replot` commands, the default engine setting applies unless you include `-engine`.
- Use `-proof_time` to set a limit on the time for this command. See ["set_prove_time_limit"](#) on page 1189.
- Use `-quiet` to enable QuietTrace.
- Use `-min_variation` to replot with as little change as possible in the trace. This switch suppresses differences that are unrelated to changes you made prior to replotting.

JasperGold Apps Command Reference Manual

General Commands

-replot (Continued)

- Use `-batch` to prevent the tool from displaying the traces it generates, which improves performance.
 - Use `-silent` to turn off the output to the screen and return information with a Tcl return value.
Note: `-silent` does not stop the trace from being shown in the Visualize window. You must use `-batch` to suppress the trace.
 - Use `-bg` to run the command in the background and continue to accept and run new commands.
 - Use `-sig_order` to generate a trace that displays only the signals in the specified signal order file.
 - Use `-no_signals` to load a trace without displaying signals.
-

JasperGold Apps Command Reference Manual

General Commands

```
-go_backward -new_window new_window_name -force  
-min_variation -batch -silent -bg  
-proof_time time_limit  
-sig_order sig_order_file_name | -no_signals  
-window window_name
```

If there are new configurations, remove them and keep the current trace. Otherwise, go back to the previous trace.

Returns `false` if it is not possible to go backward because the displayed trace is the first trace or because the set of configurations reached is one where no trace is possible; otherwise, it returns `true`.

Implementation guidance:

- Use `-new_window` to copy the current window and apply `-go_backward` to a new window.
 - To specify the new window's name, use the optional `new_window_name` argument. If you do not specify a name, then the name is the name of the current window with `_copy` appended.
 - To copy a window other than the current window, use the `-window` switch.
- Use `-force` to ignore any cached trace core and call the engine to generate a new full trace.
- Use `-min_variation` to replot with as little change as possible in the trace. This switch helps you understand the new trace by suppressing differences that are unrelated to changes you made prior to replotting.

JasperGold Apps Command Reference Manual

General Commands

-go_backward (Continued)

- Use `-batch` to prevent the tool from displaying the trace it generates, which improves performance.
- Use `-silent` to turn off the output to the screen and return information with a Tcl return value.
- Use `-bg` to run the command in the background and continue to accept and run new commands.
- Use `-proof_time` to set a limit on the time for this command. See "[set_probe_time_limit](#)" on page 1189.
- Use `-sig_order` to generate a trace that displays only the signals in the specified signal order file.
- Use `-no_signals` to load a trace without displaying signals.

```
-go_forward -new_window new_window_name -force  
-min_variation -batch -silent -bg  
-proof_time time_limit  
-sig_order sig_order_file_name | -no_signals  
-window window_name
```

Go forward to the next trace. (Use this option after using the `-go_backward` option.)

Returns `false` if it is not possible to go forward because the displayed trace is the last trace or because the set of configurations reached is one where no trace is possible; otherwise, it returns `true`.

Implementation guidance:

- Use `-new_window` to copy the most recently created window and replot in that window. You can specify the new window's name using the optional `new_window_name` argument. If you do not specify a name, then the name is the name of the copied window with `_copy` appended.
- Use `-force` to ignore any cached trace core and call the engine to generate a new full trace.

JasperGold Apps Command Reference Manual

General Commands

-go_forward (Continued)

- Use `-min_variation` to replot with as little change as possible in the trace. This switch helps you understand the new trace by suppressing differences that are unrelated to changes you made prior to replotting.
- Use `-silent` to turn off the output to the screen and return information with a Tcl return value.
- Use `-bg` to run the command in the background and continue to accept and run new commands.
- Use `-proof_time` to set a limit on the time for this command. See "[set_prove_time_limit](#)" on page 1189.
- Use `-sig_order` to generate a trace that displays only the signals in the specified signal order file.
- Use `-no_signals` to load a trace without displaying signals.

Utility Switches for Configurations

`-rename_conf old_conf_name new_conf_name`
 `-window window_name`

Rename the specified configuration (-force | -at_least_once).

`-remove_conf conf_name`
 `-window window_name`

Remove the specified configuration (-force | -at_least_once).

This switch supports wildcards (*) and (?) in the name.

`-disable conf_name`
 `-window window_name`

Keep the configuration but do not pass the information to the engines or include it in any replot.

This switch supports wildcards (*) and (?) in the name.

`-enable conf_name`
 `-window window_name`

Re-enable the specified (previously disabled) configuration.

This switch supports wildcards (*) and (?) in the name.

JasperGold Apps Command Reference Manual

General Commands

```
-offset_conf conf_name offset_value
           -window window_name
```

Add *offset_value* to both cycles of the configuration with the given name.

This switch supports wildcards (*) and (?) in the name. *offset_value* must be a positive integer.

Refer to "[Examples](#)" on page 935.

```
-clear_conf
      -window window_name
```

Remove all existing configurations for the next trace without clearing previously generated traces.

```
-replace_conf -force -soft | -not -soft expression
               cycle | cycle:cycle -name conf_name -window window_name
```

Replace the existing configuration (-name *conf_name*) with a configuration specified by the remaining -replace_conf parameters.

Note:

- These switches mirror the GUI procedure for editing an existing configuration. You will typically use the GUI for this purpose.
 - -replace_conf does not support -group_name.
-

JasperGold Apps Command Reference Manual

General Commands

```
-replace_conf -force | -not | -at_least_once  
    expression | -property property_name  
    -after expression_1 | -after -after_property property_name  
    -regexp  
    -before expression_1  
    cycle | cycle:cycle  
    -name conf_name  
    -loop -window window_name
```

Replace the existing configuration (-name *conf_name*) with a configuration specified by the remaining -replace_conf parameters.

Use -loop when replacing a force or at-least-once constraint in a liveness trace.

Note:

- These switches mirror the GUI procedure for editing an existing configuration. You will typically use the GUI for this purpose.
- The -property switch supports wildcards and regular expressions. An error results unless there is exactly one matching property. When using a regular expression, use the -regexp switch after the pattern:

```
visualize... -property {pattern} -regexp
```

- -replace_conf does not support -group_name.

```
-replace_conf -signal signal_name -sequence  
    {value{cycles} value{cycles}}... cycle  
    -name conf_name -window window_name
```

Replace the specified configuration (-name *conf_name*) with a configuration specified by the remaining -replace_conf parameters.

Note:

- These switches mirror the GUI procedure for editing an existing configuration. You will typically use the GUI for this purpose.
- -replace_conf does not support -group_name.

Utility Switches – General

```
-list -silent -all | group_name  
-window window_name
```

List the configuration that is currently specified.

The tool lists the Visualize configurations you specified thus far for ungrouped (top-level) configurations.

- Use `-silent` to turn off output to the screen and return the list as a Tcl return value.
- Use `-all` to view all configurations, grouped or ungrouped.
- Provide a `group_name` to view configurations in the specified group.

```
-list -all_windows -silent
```

List the names of all Visualize windows.

Use `-silent` to turn off the output to the screen and return information with a Tcl return value.

JasperGold Apps Command Reference Manual

General Commands

```
-save -script  
  -force | -append  
  -setup_only | -config_only (-add_hier_path path_name  
    |-remove_hier_path path_name)  
  -window window_name  
  file_name
```

Save the configurations for the current trace into the specified script file.

Pending (blue) configurations added since the last replot are saved as pending configurations.

If the specified *file_name* already exists,

- Use `-force` to overwrite the existing file.
- Use `-append` to append the script to the end of the file.
- If you do not use `-force` or `-append`, the tool returns an error message.

The script contains both setup (analyze, elaborate, and so forth) and Visualize configurations. To limit the contents of the script:

- Use `-setup_only` to save only the setup portion of the script, not the actual Visualize configurations.
- Use `-config_only` to save only the Visualize configuration portion of the script, not the setup.
 - Use `-add_hier_path` or `-remove_hier_path` to run scripts generated for RTL with a different hierarchy.
 - See `session -set_hier_path` for determining the value of the path argument.

Note: Specifying `visualize -save file_name` is the same as `visualize -save -script file_name`.

JasperGold Apps Command Reference Manual

General Commands

```
-save -init_state file_name [-cycle N] [-force]
[-include_internal] [-include_encrypted] [-bit_blast] [-window window_name]
```

Save the signal values from the last or specified cycle of the trace in a format suitable for use with `reset -init_state`.

This command only saves values for registers that are part of the AR/COI of the current Visualize window. Run `set_trace_optimization standard` before the initial trace generation to dump values for all design registers instead.

The saved file reflects the configurations you specified prior to the last time you used `visualize -replot`. Pending configurations added since the last `replot` are not used.

- Use `-cycle` to specify a cycle other than the last cycle.
- If the specified `file_name` already exists, use `-force` to overwrite the existing file. If you do not use `-force`, the tool returns an error message.
- Use `-include_internal` to save the internal state based on assumptions, assertions, temporal operators, and so forth.

Note: Loading files with the switch `-include_internal` carries a risk of a mismatch in internal signals that can lead to false proof results in some cases. Internal signals in reset files might not be the same as those generated by a different JasperGold release version or a different version of the design or a different design setup and configuration. (Design setup and configuration differences include the order in which files are passed to `analyze` and `elaborate` and the order in which assumptions and assertions are created.)

- Use `-include_encrypted` to capture the values of encrypted flops and the internal state of Proof Accelerators in an encrypted file.
 - Use `-bit_blast` to save each bit of a signal separately.
-

JasperGold Apps Command Reference Manual

General Commands

```
-save -reset_sequence file_name [-force]
[-include_internal] [-include_encrypted] [-bit_blast] [-window window_name]
```

Save the register values from the first cycle of the trace and the input values from all cycles of the trace in a format suitable for use with `reset -sequence`.

This command only saves values for registers that are part of the AR/COI of the current Visualize window. Run `set_trace_optimization standard` before the initial trace generation to dump values for all design registers instead.

The saved file reflects the configurations you specified prior to the last time you used `visualize -replot`. Pending configurations added since the last `replot` are not used.

- If the specified *file_name* already exists, use `-force` to overwrite the existing file. If you do not use `-force` the tool returns an error message.
- Use `-include_internal` to save the internal state based on assumptions, assertions, temporal operators, and so forth.

Note: Loading files with the switch `-include_internal` carries a risk of a mismatch in internal signals that can lead to false proof results in some cases. Internal signals in reset files might not be the same as those generated by a different JasperGold release version or a different version of the design or a different design setup and configuration. (Design setup and configuration differences include the order in which files are passed to `analyze` and `elaborate` and the order in which assumptions and assertions are created.)

- Use `-include_encrypted` to capture the values of encrypted signals and the internal state of Proof Accelerators in an encrypted file.
- Use `-bit_blast` to save each bit of a signal separately.

JasperGold Apps Command Reference Manual

General Commands

```
-save -vcd [-imp | -spec] vcd_file_name [-force]
[-sig_order sig_order_file_name]
[-reset_window | -window window_name]
```

Save the current trace values in VCD format to the specified *vcd_file_name*.

The VCD file preserves design hierarchy.

- Use `-imp` to save only SEC imp signals.
- Use `-spec` to save only SEC spec signals.

Note: `-imp` and `-spec` are mutually exclusive. The default, if you do not use `-imp` or `-spec`, is to save both imp and spec signals.

- Use `-force` if the specified VCD file already exists and you want to overwrite it.
- Use `-sig_order sig_order_file_name` to save only the values of signals that are included in *sig_order_file_name*.

Note:

- If you have also used the `set_trace_optimization` command, the VCD file reflects those settings, for example:
 - If trace optimization is `ar`, then only the analysis region is saved into the VCD file. *This option is the default*
 - If you use `set_trace_optimization standard`, then the entire design is dumped into the VCD. However, you should be aware that this setting can increase the time it takes to generate a trace.
- Run the `set_trace_optimization` command before generating a trace.

JasperGold Apps Command Reference Manual

General Commands

```
-save -fsdb fsdb_file_name [-force]
[-time_scale [N] [time_unit]]
[-sig_order sig_order_file_name]
[-window window_name | -reset_window]
```

Save the current trace values in FSDB format to the specified file.

- Use `-force` if the specified FSDB file already exists and you want to overwrite it.
- Use `-time_scale` to specify the time scale:
 - *N* is mandatory if you do not specify the *time_unit*.
 - If you use *N* and do not specify a *time_unit*, the default is nanoseconds (ns).
 - *N* is a natural non-zero number.
 - *time_unit* is mandatory if you do not specify *N*.
 - If you use *time_unit* and do not specify N, the default is 1.
 - *time_unit* is s, ms, us, ns, ps, or fs.
- Use `-sig_order` and specify a *sig_order_file_name* to save only the values of signals that are included in *sig_order_file_name*.

The FSDB file preserves design hierarchy.

Note:

- If you have also used the `set_trace_optimization` command, the FSDB file reflects those settings, for example:
 - If trace optimization is `ar`, then only the analysis region is saved into the FSDB file. (This is the default setting.)
 - If trace optimization is `standard`, then the entire design is dumped into the FSDB. However, you should be aware that this setting can increase the time it takes to generate a trace.
- Run the `set_trace_optimization` command before generating a trace
- `-save -fsdb` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

JasperGold Apps Command Reference Manual

General Commands

```
-save -shm shm_directory_name [-force]
[-time_scale [N] [time_unit]]
[-sig_order sig_order_file_name]
[-window window_name | -reset_window]
```

Save the current trace values in SST2/SHM format.

- Use `-force` if the specified SHM directory already exists and you want to overwrite it.
- Use `-time_scale` to specify the time scale:
 - *N* is mandatory if you do not specify the *time_unit*.
 - If you use *N* and do not specify a *time_unit*, the default is nanoseconds (ns).
 - *N* is a natural non-zero number.
 - *time_unit* is mandatory if you do not specify *N*.
 - If you use *time_unit* and do not specify N, the default is 1.
 - *time_unit* is s, ms, us, ns, ps, or fs.
- Use `-sig_order` and specify a *sig_order_file_name* to save only the values of signals that are included in *sig_order_file_name*.

The SHM file preserves design hierarchy.

Note:

- If you have also used the `set_trace_optimization` command, the SHM directory reflects those settings, for example:
 - If trace optimization is `ar`, then only the analysis region is saved into the SHM file. (This is the default setting.)
 - If trace optimization is `standard`, then the entire design is dumped into the SHM. However, you should be aware that this setting can increase the time it takes to generate a trace.
- Run the `set_trace_optimization` command before generating a trace
- `-save -shm` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

JasperGold Apps Command Reference Manual

General Commands

```
-save_recipe waveform_name [-task task_name]
[-save_signal_order][-save_behavior]
[-description description] [-batch] [-png]
```

Search for the specified waveform in the database; if not present, add a new waveform element.

- Use `-task` to specify the context of the recipe.
 - Use `-save_signal_order` to save the signal order for the specified waveform.
 - Use `-save_behavior` to convert the recipe's at-least-once constraints into behaviors and save them as elements. They will become children of the recipe element.
 - Use `-description` to add a description.
 - Use `-batch` to add the new element without asking for confirmation.
 - Use `-png` to capture the waveform image and display it in the document pane.
-

```
-run_recipe element_id
```

Run the Visualize recipe that has the specified database element ID.

JasperGold Apps Command Reference Manual

General Commands

```
-load (-vcd |-fsdb |-shm) trace_file_name
[-time_scale [N] [time_unit]]
[-hier_path path_difference]
[-start_time N] [-end_time N]
[-hier_map -from source_hier -to destination_hier]
[-window window_name] [-batch]
```

Load the trace from the specified file as a Visualize trace.

Visualize loads the trace in read-only mode, which means it loads values for all design signals in memory without performing any simulation. This method is usually much faster than using `visualize -confirm`, but it does not yet support `visualize -check_props`.

Note: `-load` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

- Use `-vcd`, `-fsdb`, or `-shm` to specify the type of the trace to be interpreted.
- Use `-time_scale` with the unit of measure (ms, us, ns, or ps) to specify the time scale. If you do not specify a time scale, the tool uses the default from the trace file.
- Use `-hier_path` when the trace file contains a hierarchy that is different from the one in the design loaded in JasperGold. For example, if the design top module is `dut_top`, which is instantiated as the top module in the testbench `testbench_top` as `i_dut_top`, use the following command (replace `file_name` with an actual value):

```
visualize -load (-vcd |-fsdb |-shm) <file_name> -hier_path
testbench_top.i_dut_top
```
- Use `-start_time` to skip the initial chunk of simulation data. Typically, you will use this switch to skip the reset sequence and so forth.
- Use `-end_time` to ignore the last chunk of simulation data.
- Use `-hier_map` for trace files generated during simulation if you have a wrapper module.
 - Use `-from` to specify the source (simulation environment) hierarchy.
 - Use `-to` to specify the destination (formal environment) hierarchy.
- Use `-batch` to prevent the tool from displaying the trace it generates.

JasperGold Apps Command Reference Manual

General Commands

```
-confirm (-vcd | -fsdb | -shm) trace_file_name
[-time_scale [N] [time_unit]]
[-hier_path path_difference] [-start_time N] [-end_time N]
[-hier_map -from source_hier -to destination_hier]
[-window window_name] [-batch]
```

Load the trace from the specified file as a Visualize trace and confirm it against the design.

Visualize loads the trace by running a simulation and propagating values from inputs to outputs. This command respects the `set_trace_optimization` setting when running the simulation. At the end of the simulation, Visualize checks for inconsistencies between simulated values and the original trace values and reports them as warnings or errors.

Note: `-confirm` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

- Use `-vcd`, `-fsdb`, or `-shm` to specify the type of the trace to interpret.
- Use `-time_scale` with the unit of measure (`ms`, `us`, `ns`, or `ps`) to specify the time scale. If you do not specify a time scale, the tool uses the default from the trace file.
- Use `-hier_path` when the trace file contains a hierarchy that is different from the one in the design loaded in JasperGold. For example, if the design top module is `dut_top`, which is instantiated as the top module in the testbench `testbench_top` as `i_dut_top`, use the following command (replace `file_name` with an actual value):

```
visualize -load (-vcd | -fsdb | -shm) <file_name> -hier_path
testbench_top.i_dut_top
```
- Use `-start_time` to skip the initial chunk of simulation data. Typically, you will use this switch to skip the reset sequence.
- Use `-end_time` to ignore the last chunk of simulation data.
- Use `-hier_map` for trace files generated during simulation if you have a wrapper module.
 - Use `-from` to specify the source (simulation environment) hierarchy.
 - Use `-to` to specify the destination (formal environment) hierarchy.
- Use `-batch` to prevent the tool from displaying the trace it generates.

JasperGold Apps Command Reference Manual

General Commands

-reset

Load the trace from reset as a Visualize trace.



- Run this command after you use `analyze`, `elaborate`, and commands that set up your environment (for example, `clock`, `reset`, `assume`, and so forth).
- This command opens a new window with the reset trace. The window name is Reset Analysis.

-show [-property *property_name*] [-silent]

Report the settings used for the most recent Visualize and the status.

- Use `-property` to specify a property other than the one used in the most recent Visualize.
- Use `-silent` to print the status to the screen and return all other information with a Tcl return value.

-save_sig_order *file_name*
[(-window *window_name*) | -reset_window] [-save_highlight]

Save the current signal list to the specified file.

Note: If the file name already exists, this command replaces the file's contents with the current signal list.

Use `-save_highlight` if you want to save the row highlight status into the file also.

-set_silent_mode true | false

Turn off the output to the screen and return information with a Tcl return value.

Note: This setting persists for `visualize` until you change it with another `-set_silent_mode` command. When silent mode is set, the tool suppresses all info messages and proof messages that would normally be generated by successive `visualize` commands, and the tool does not generate an error when `Visualize` fails to find a trace.

-get_silent_mode

Report whether silent mode is enabled for `visualize` commands.

JasperGold Apps Command Reference Manual

General Commands

```
-undo -window window_name
```

Undo the previous configuration.



- Run the command multiple times to step back through configurations.
 - This command has no effect after `-replot`.
-

```
-redo -window window_name
```

Redo the previous configuration.



- Run the command multiple times to reinstate multiple configurations you previously removed.
 - This command has no effect after `-replot`.
-

```
-highlight -diff other_window_name [-window window_name]
```

Apply highlights to show the differences between the current Visualize window and the specified Visualize window.

Use `-window` to specify a window other than the current window.

Note: The tool applies yellow highlights to signals that have different values. If the current trace has more cycles than the one you are comparing it to, the tool applies an orange highlight to the additional cycles.

```
-highlight -diff (-prev_plot | -next_plot) [-window window_name]
```

Apply highlights to show the differences between two contiguous plots in the current Visualize stack.

- Use `-prev_plot` to create highlights that compare the current window's trace and the previous trace in the stack.
- Use `-next_plot` to create highlights that compare the current window's trace and the next trace in the stack.

Note: The tool applies yellow highlights to signals that have different values. If the current trace has more cycles than the one you are comparing it to, the tool applies an orange highlight to the additional cycles.

JasperGold Apps Command Reference Manual

General Commands

```
-relevant_logic signal iteration
[-configuration ([control_path | data_path] [undriven_only]
               [filter_async_reset] [show_clock] [show_equal]
               [show_twins_diff] [instance_ports_only]) +
 [-iteration_depth N |-num_whys N]
 [(-window window_name) |-reset_window]
```

Highlight the transitive “Why” of the specified signal/cycle.

Use this command to check the logic that generated a certain value in the trace.

- Use `-configuration` with one or more of the following arguments to configure the relevant logic trace:

- `data_path`: Highlight only the data path of the design.
 - `control_path`: Highlight only the control path of the design.
 - `undriven_only`: Highlight only undriven signals.
 - `filter_async_reset`: Do not highlight reset-related signals.
 - `show_clock`: Highlight clock-related signals.
 - `show_equal`: Highlight equivalent (buffer) signals.
 - `show_twins_diff`: Highlight SEC twin signals that have different values.
 - `instance_ports_only`: Highlight only signals that are instance ports.

- Use `-iteration_depth` to specify the trace length (number of cycles). The default is a 5-cycle trace. Use 0 to specify unlimited cycles.

Note: Specifying a large number of cycles can degrade performance in terms of run time.

- Use `-num_whys` to specify the number of Whys you want to call.

Note: This behavior is consistent with the pre-2014 behavior for `-depth`. The `-depth` switch is being deprecated.

```
-clear_all
  -window window_name
```

Clear all configurations and all previously generated traces.

Use `-window` to specify a window other than the current window.

JasperGold Apps Command Reference Manual

General Commands

```
-sig_order sig_order_file_name  
[(-window window_name) | -reset_window]
```

Launch a Visualize window containing the signals in the specified signal order file.

Use `-window` to specify a window other than the current window.

```
-no_signals  
-window window_name
```

Clear all signals from the current trace.

```
-new_window new_window_name  
-copy -window old_window_name
```

Launch a new Visualize window.

If you do not use additional switches, the new window is empty.

- Use the optional argument to `-new_window` to specify a name for the new window.
- Use `-copy` to clone the current Visualize window. The default name is the name of the copied window with `_copy` appended.
- Use `-window` to specify an original Visualize window other than the current one.

```
-rename_window old_name new_name
```

Rename the specified Visualize window.

```
-set_current_window window_name
```

Set the specified Visualize name (window) as the current Visualize name.

```
-get_current_window
```

Report the current Visualize name (window).

```
-remove_window (window_name | -reset_window)
```

Remove the specified Visualize name (window or Reset Analysis window).

```
-remove_current_window
```

Remove the current Visualize name (window).

JasperGold Apps Command Reference Manual

General Commands

```
-add_sig (signal_name | expression) row
  [-file file_name
  [(-window window_name) |-reset_window]
```

Add the specified signal or expression at the specified row or the signals or expressions in the specified file to the trace. The top of the waveform is row 0.

```
-get_type
  [-window window_name]
```

Show the current Visualize type (violation, sanity, cover, consistency, or state-removal) for the current or specified window.

```
-get_value signal_name cycle | cycle:cycle
  -radix (2 | 8 | 10 | 16 | bin | oct | dec | signed_dec | hex)
  -window window_name
```

Get the value of the specified signal.

- Specifying cycles is optional.
 - If you specify a single cycle (for example, 7), you are asking the tool to get the value for the specified signal in cycle seven.
 - You can specify a range of cycles; for example, 2:3.
 - Use \$ to mean “the last cycle” and \$-N to mean “N cycles before the last cycle”. For example:
 - 1:\$ is equivalent to always (which is the default when you do not specify cycles)
 - 4:\$ is equivalent to from cycle four until the end of the trace.
 - \$-3 is equivalent to 3 cycles before the last cycle.
- Use **-radix** to return values in the specified radix.

```
-get_length
  [(-window window_name) |-reset_window]
```

Get the length of the current trace.

```
-get_stem_length
  -window window_name
```

Get the length of the stem of the current trace. The stem is the part of the trace that is not part of the loop.

JasperGold Apps Command Reference Manual

General Commands

```
-get_loop_length  
    -window window_name
```

Get the length of the loop of the current trace.

For looping traces (counterexamples to liveness assertions) this value is greater than 0, for other traces it is 0.

```
-get_looping_flops  
    -window window_name
```

Return a list of all the flops with looping behavior in the current liveness trace and return an empty list if there is no liveness trace.

```
-is_looping  
    -window window_name
```

Return `true` if the current trace has a loop. Counterexamples to liveness assertions have loops, other traces do not.

```
-load_radix file_name  
    -window window_name
```

Load and apply radices defined in the specified file.

`file_name` format consists of radix definition clauses and radix use clauses.

See [“Custom Radix Files”](#) on page 1407.

```
-get_marker_location (current | primary | secondary)  
    [-window window_name]
```

Return the cycle number of the marker location.

- Use `current` to specify that you want the location of the current (red) vertical marker.
- Use `primary` to specify that you want the location of the primary (blue) vertical marker.
- Use `secondary` to specify that you want the location of the secondary (dark green) vertical marker.

Color-Annotation Switches

Use the following switches and arguments to define, activate, and name a color annotation for the Visualize window.

- Each color annotation consists of a set of one or more triplets.
- Each triplet consists of a `signal_name-time_step-color`.

Refer to the example section below.

```
-define_color_annotation  
{ {signal_name time_step color}+ } annotation_name  
-window window_name
```

Define the color annotation for the specified “`signal_name`” at the specified `time_step` point.

The tool lists annotation names in the *Trace Exploration* combo box located on the Visualize window toolbar.

- `signal_name time_step color` – Specify this first-argument triplet as follows:
 - `signal_name` – The name of the signal to which the color annotation applies.
 - `time_step` – The time point when the color annotation applies.
 - `color` – The annotation color. Possible values are gray, blue, green, yellow, orange, and red.

To delete an annotation, specify an empty argument by including the `{ }` with nothing enclosed.

- (Optional) `annotation_name` – The name of the color annotation. If the specified name already exists, this command redefines the annotation. You cannot redefine predefined annotations, such as those from ATD and Visualize. If you do not specify a name, the default name is “User Defined.”

The tool returns the name of the annotation to show you have successfully specified it.

JasperGold Apps Command Reference Manual

General Commands

```
-activate_color_annotation annotation_name
[ (-window window_name) | -reset_window]
```

Apply the specified color *annotation_name*.

annotation_name is any of the annotations in the *Trace Exploration* combo box (both user-defined and predefined).

Use the default annotation named “Off” to clear the current annotation from the trace.

Signal Highlight Switches

```
-define_signal_highlight
{ {signal_name color}+ }
[ (-window window_name) | -reset_window] [add | -remove]
```

Define the row highlighting for the specified *signal_name* using the specified color.

- **signal_name color:** Specify this first-argument pair as follows:
 - **signal_name:** The name of the signal to which the highlight applies.
 - **color:** The highlight color. Possible values are blue, green, yellow, orange, and red.
- Use **-add** to add highlight signals incrementally.
- Use **-remove** to remove highlights from specified signals.
- If you do not specify either **-add** or **-remove**, the specified highlight list replaces the current highlight list.
- To delete all signal highlights, specify an empty argument by including the **{ }** with nothing enclosed.

Interactive Constraint Extraction

```
-extract_conf -reason -depth depth_value -transitive -value
{{signal_name cycle | cycle:cycle}...}
-name conf_name
-window window_name
```

Find a new trace that shows one or more of the specified signals showing:

- A value that was seen on one of the cycles specified for that signal, but with different values on the immediate fanin (-reason). *This option is the default*
- A value that is different from those seen on any of the cycles specified* for that signal (-value).
- Either different values on the immediate fanin or a different value on the signal (-reason -value).

Switches:

- Use -depth with -reason to tell how many levels of registers to look through for the reason for a signal. *N* must be a positive integer.
- Use -transitive with -reason if one of the signals returned as part of the reason is an internal wire. (The tool then recursively does *Why* on that signal until all the signals reported are either inputs or registers.)
- Specifying cycles is optional. If you do not specify cycles, that is equivalent to always.
 - If you specify a single cycle (for example, 7), you are asking the tool to ensure the expression generated by -extract_conf is true in cycle seven.
 - You can specify a range of cycles, for example, 2 : 3.
 - Use \$ to mean the last cycle. For example, 1 : \$ is equivalent to “always” and 4 : \$ is equivalent to “from cycle four until the end of the trace.”
- Use -name to specify a *conf_name* for this configuration.

Implementation guidance: Use curly braces around each signal-cycle set.

Refer to “[Examples](#)” on page 935.

JasperGold Apps Command Reference Manual

General Commands

```
-concretize_conf {conf_name}
    -window window_name
```

Make the specified, extracted configuration concrete.

Use this command to replace the specified extracted configuration with a configuration that captures how the extracted configuration is satisfied in the current trace.

Signal and Expression Exploration

```
-explore (signal | -property property_name) window iterations
    -new_window new_window_name -batch -silent -bg -wide_signal_threshold
    threshold -window window_name
```

Generate a series of traces to assign different values to the target signal or expression and to exercise different paths from the inputs to the target.

Note: X-Prop and SPV traces do not support this command.

- If you use `-property`, the tool attempts to generate a series with different ways to exercise the property. This feature supports the following types of properties:

- Assertion of the form seqA | -> seqB
 - Assertion of the form seqA | => seqB
 - Cover of the form seqA

It does not support properties with temporal preconditions.

This command is currently intended for on-the-fly properties with Verilog SVA syntax. The feature has known limitations for VHDL designs.

- `window` is the number of cycles starting from the beginning of the current trace that will be explored. *The default value is \$.*
- `iterations` is the number of times the tool should repeat the exploration.

JasperGold Apps Command Reference Manual

General Commands

-explore (Continued)

- Use `-new_window` to generate the waveform in a new window. You can specify the new window's name using the optional `new_window_name` argument. If you do not specify a name, then the name is the name of the copied window with `_copy` appended.
- Use `-batch` to prevent the tool from displaying the traces it generates, which improves performance.
- Use `-silent` to turn off the output to the screen and return information with a Tcl return value.

Note: `-silent` does not stop the trace from being shown in the Visualize window. You must use `-batch` to suppress the trace.
- Use `-bg` to run the command in the background and continue to accept and run new commands.
- Use `-wide_signal_threshold` to specify the number of distinct values a signal needs to exhibit in the trace to be considered wide. *The default value is 4.*

Waveform Analysis

-check_props

```
[-cover | -violation] [-update_gui] [-names_only]
[-filter property_name_filter_list [-regexp]] [-silent]
[-window window_name]
```

Analyze exercised covers and violated assertions in the current waveform.

- Use `-cover` or `-violation` to check only for covers or violations.
- Use `-update_gui` to refresh the Visualize window with the current analysis results.
- Use `-names_only` to only show the names of properties. By default, the tool also shows the properties' type and the cycles in which they are covered or violated.
- Use `-filter` to specify one or more properties to be checked. And use `-regexp` to enable regular expressions.

Note: This switch supports wildcards and Tcl lists.

- Use `-silent` to turn off output to the screen.

Note: This operation is typically conducted with the Indexed Behaviors table *Analyze* button.

JasperGold Apps Command Reference Manual

General Commands

```
-compare signalA_name signalB_name  
[-shift_cycles cycles] [-ignore_x]  
[-ignore_values value+]  
[-window window_name]
```

Compare the value of two signals cycle-by-cycle.

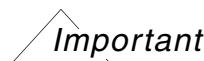
This command generates a comparison results row in the Visualize window to show cycles that match and cycles that do not match.

- Use `-shift_cycles` to offset the cycles on compared signals.
- Use `ignore_x` if you do not care about x value. If you do not use this switch, any x value shows as mismatch in the comparison results.
- Use `ignore_values` and specify a list of values that should not be considered mismatches. The values in the list produce a match result when doing the value comparison.

State-Space Tunneling Visualize

```
-sst
  -property property_name -with_proven
  |-task task_name
  -window window_name
```

Ignore reset, and Visualize how the target assertion might be satisfied by starting from an uninitialized state.



- The trace does not have to start from reset state.
- The trace will satisfy all assumptions and proven helper assertions in the task.
- This command ignores bounded assumptions.
- The trace will be a CEX of the property `sst_property`, which is defined by the `sst_undetermined_helpers` flag.
- The trace will violate the `sst_property` only at the last cycle.
- The trace length is set by the `sst_default_trace_length` flag. You can override this setting with `visualize -max_length` and `visualize -min_length`.
- The tool considers undetermined helper assertions as defined by the command `set_sst_undetermined_helpers`.
- Use `-with_proven` to force the generated trace to satisfy all proven assertions.
- Use `visualize -sst` with no additional arguments to generate an SST trace that violates the logical AND of all the helper assertions in the current task. Use `-task` to specify a task other than the current task. This command only works with SST conjunct mode (see [“set_sst_undetermined_helpers” on page 1281](#)).

Note: Freeze is not supported.

Design-Space Tunneling Visualize

```
-dst true | false  
-window window_name
```

Set the tunneling mode of the Visualize window and session to manual. When starting a new Visualize session, the value of `dst` defaults to the tunnel mode of the Visualize task.

Quiet Visualize

```
-quiet  
((true [cycle:cycle]) | false)  
[-window window_name]
```

Generate quieted waveforms to decrease the unnecessary distraction that commonly comes from looking at waveforms with many signals toggling at the boundary and in the internal logic that are of no consequence to the behavior being illustrated.

The QuietTrace goal is to provide a faster and better understanding of the waveforms. The *Visualize Configuration* pane in the Visualize window includes a tree node for Quiet data. When you enable the feature, it displays figures that show, out of the total number of soft constraints or preferences that were possible, how many were satisfied and how many were not satisfied in the final waveform.

- Use `true` to turn QuietTrace on, and if you want to apply QuietTrace to a subset of the trace, specify a cycle range. Refer to “[Examples](#)” on page 935.
 - Use `false` to turn off QuietTrace.
-

Configuration Grouping

```
-group
  {conf_name+} -name group_name
  |-expand group_name
  |-remove group_name
  |-rename old_group_name new_group_name
  |-replace {conf_name+} -name group_name
  |-add_conf {conf_name+} group_name
  |-remove_conf {conf_name+}
  |-merge group_name_1 group_name_2 -name final_group_name
  -window window_name
```

Add or edit configuration groups.

- Use `-name` to place one or more specified configurations in a group with the specified `group_name`. Use curly braces around the set of configuration names.
- Use `-expand` to ungroup the specified `group_name`.
- Use `-remove` to remove the specified `group_name` and all its contents.
- Use `-rename` to rename the specified group.
- Use `-replace` with `-name` to edit the specified group by replacing its current configurations with one or more specified configurations. Use curly braces around the set of configuration names
- Use `-add_conf` to edit the specified `group_name` by adding the specified configuration to the current configurations. Use curly braces around the set of configuration names.
- Use `-remove_conf` to edit a group by removing the specified configuration from the group. Use curly braces around the set of configuration names. If you remove all configurations from a group, the empty group remains.
- Use `-merge` to merge the configurations from `group_name_1` and `group_name_2` into a new group. Use `-name` to specify a preferred name.

JasperGold Apps Command Reference Manual

General Commands

Annotating Targets and Configurations

```
-set_annotation (-target | -conf conf_name) {annotation}  
[-window window_name]
```

Annotate the Visualize target or specified configuration or modify the existing annotation with the specified description that is enclosed by curly braces.

To remove an annotation, use empty curly braces.

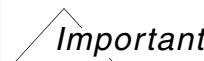
```
-get_annotation (-target | -conf conf_name)  
[-window window_name]
```

Print the annotation for the target or specified configuration to the screen.

Specifying a Waveform Sequence in a Single Command

```
-signal signal_name  
-sequence {value{cycles} value{cycles}}... cycle  
-name name  
-group_name group_name  
-window window_name
```

Display a waveform sequence for the specified signal that shows the specified values.



- Use one blank space as the delimiter between values.
- Use x or X to show a don't care value that does not constrain the signal.
- Use the optional *{cycles}* argument after a value, with no space between the value and curly braces, to specify the number of times the value is repeated.
- Use the optional *cycle* argument to specify the cycle in which the first specified *value* applies. If you do not specify an *cycle*, the first *value* applies in cycle 1.
- Use the optional *-name* switch to specify a *name* for this configuration.
- Use *-group_name* to specify a *group_name* for this configuration.

Refer to “[Examples](#)” on page 935.

Showing Relevant Behaviors

```
-relevant_props signal_name cycle depth_value -silent  
-window window_name
```

Show properties that affect the value of the specified *signal_name* at the specified *cycle* by placing an asterisk in the *Indexed Behaviors* table.

Important things to know about this command:

- Use *depth_value* to specify how many levels of registers to look through for the relevant properties. The default value is 6.
- Use *-silent* to print the list of relevant properties in the log pane instead of showing them in the *Indexed Behaviors* table.

RAM Abstraction

Use `elaborate -abstract_ram` to make this RAM abstraction command effective.

```
-init_PA ram_instance -depth N  
-window window_name
```

Reset the depth of the specified RAM abstraction to *N*.



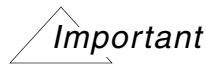
- If you do not use the *-depth* switch, the default setting is 0.
- If you do not specify a *ram_instance*, this command resets the depth of all RAM abstractions to *N*.

JasperGold Apps Command Reference Manual

General Commands

```
-suggest_PA  
    -window window_name
```

Return a list showing which RAMs are returning abstract results that are relevant to the current trace.



- “Relevant to the current trace” means it is in the transitive why of any of the following:
 - The trace target
 - A force configuration
 - An at-least-once configuration
 - A signal sequence configuration
- The elements of the list will be of the form {*<ram_instance>* *<address>* *<cycle>*}
 - *<ram_instance>* is the full path to the instance of the RAM abstraction.
 - *<address>* is the address of the first access to the specified RAM that is returning a relevant abstract value.
 - *<cycle>* is the cycle in the trace in which the relevant abstract value is being returned.
- If no trace is available, the tool returns an error.

JasperGold Apps Command Reference Manual

General Commands

```
-justify_PA ram_instance  
-window window_name
```

Increase the depth of the specified RAM instance by one.



- If you do not specify a RAM instance, the command uses the RAM instances returned by -suggest_PA.
- If the RAM instance you provide is not justifiable (that is, it would not have been returned by -suggest_PA) or if you do not specify a RAM instance and -suggest_PA provides nothing to justify, the tool returns an error.

```
-auto_justify_PA ram_instance  
-window window_name
```

Implement the following loop:

```
while {[llength [visualize -suggest_PA] > 0] {  
    visualize -justify_PA  
    visualize -replot  
}
```



- If you provide a specific *ram_instance*, the tool uses it as the argument to the -justify_PA command in the loop.
- If you do not provide a specific *ram_instance*, the tool calls the -justify_PA command in the loop with no arguments (that is, it will use whatever -suggest_PA provides at that cycle).
- If -suggest_PA provides nothing to justify, the tool returns an error.

Visualize Task Manipulation

```
-set_task task_name  
-window window_name
```

Set the specified task as the current Visualize task.

This command is not supported when the trace is frozen (see -freeze).

JasperGold Apps Command Reference Manual

General Commands

```
-get_task  
    -window window_name
```

Report the name of the current Visualize task.

Switches for Background Visualize

```
-stop [-thread thread_id]
```

Stop the `visualize` command running in the background or a single thread of parallel jobs.

Note:

- If `parallel_proof_mode` is off, the command `visualize -bg` returns the string `background`.
 - If `parallel_proof_mode` is on, the command returns the string `background (<thread_id>)`.
 - Use `visualize -stop` to stop the currently running background Visualize.
 - Use `visualize -thread thread_id` to stop a specified Visualize thread.
-

```
-wait
```

Wait for the running `visualize -bg` command.

This option is useful when `visualize` is running in the background (-`bg`). In effect, it moves Visualize back to the foreground, and the tool accepts no additional commands until Visualize completes.

JasperGold Apps Command Reference Manual

General Commands

```
-status -silent
```

List the running engine jobs and their statuses.

Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

The following list shows Visualize statuses.

- `NeverStarted`: Nothing has run in this session.
 - `Pending`: There is currently a Visualize about to start.
 - `Simplify`: A Visualize is currently performing proof simplification.
 - `Running engineJob+:` A Visualize is running with the listed number of `engineJobs` currently running.
 - `Processing results`: The proof has been completed and Visualize is processing the proof results.
 - `Completed`: No job is currently running. The last Visualize has completed.
-

JasperGold Apps Command Reference Manual

General Commands

Switches for Task Context Visualize

```
-assume -bound N expression | -constant signal_name  
    -name name  
    -annotation {annotation}  
    -replace -window window_name
```

Create a task context assumption for the specified Boolean expression or signal.

- Use `-bound` with an expression to apply the specified assumption for *N* number of cycles after reset. By default an assumption is applicable for all cycles.
- Use `-constant` and a signal name to assume that the signal is pseudo constant, that is, the signal has a fixed but unknown value throughout the analysis.
- Use `-name` to label the newly created assumption so it can be reviewed or referenced in the future.
- Use `-annotation` to annotate the specified property with the description enclosed by the curly braces.
- Use `-replace` to replace the expression for the specified assumption or to change the task context constraint type.
- Use `-name` to specify the assumption you are editing.
- Use `-annotation` to add or change the assumption's annotation.

JasperGold Apps Command Reference Manual

General Commands

```
-justify signal_name+ -conflict signal_name time_step+
| -back_to_signal signal_name+
|   -from_signal signal_name+ |-from_property property+
| -back_to_property property+
|   -from_signal signal_name+ |-from_property property+
| -back_to_instance instance+
|   -from_signal signal_name+ |-from_property property+
-until_inputs
-name name
-annotation {'annotation'}
-replace
-window window_name
```

Add one or more specified signals to the AR, forcing them to take values consistent with their driving signals.

- Use `-conflict` with a `signal_name` and `time_step` to justify any of the signal-cycle inputs that would be included in the results of the `suggest` command and include relevant signals from performing “why” on those signal-cycle pairs.
- Use `-back_to_signal` to justify backwards from the AR frontier to one or more specified signals. The specified signals must be in the COI of the task.
 - Use `-from_signal` to specify one or more signals you want to justify.
 - Use `-from_property` to specify one or more properties you want to justify.
- Use `-back_to_property` to justify backwards from the AR frontier to one or more specified properties. The specified properties must be in the COI of the task.
 - Use `-from_signal` to specify one or more signals you want to justify.
 - Use `-from_property` to specify one or more properties you want to justify.

JasperGold Apps Command Reference Manual

General Commands

-justify (Continued)

- Use `-back_to_instance` to justify backwards from the AR frontier to the outputs of one or more specified instances. The specified instances must be in the COI of the task.
 - Use `-from_signal` to specify one or more signals you want to justify.
 - Use `-from_property` to specify one or more properties you want to justify.
 - Use `-until_inputs` to consider the inputs of the specified instance instead of the outputs.
- Use `-name` to label the newly created justify so it can be reviewed or referenced in the future.
- Use `-annotation` to annotate the specified justify with the description enclosed by the curly braces.
- Use `-replace` to replace the signal or task context constraint type for the specified justify; for example:
`visualize -justify {out0} -name stopat0 -replace`
 - Use `-name` to specify the justify you are editing.
 - Use `-annotation` to add or change the justify's annotation.

Note: `visualize -justify` triggers an error if you attempt to justify a signal that has already been justified.

JasperGold Apps Command Reference Manual

General Commands

```
-stopat signal_or_instance_name+  
  -name name  
  -annotation {annotation}  
  -replace  
  -window window_name
```

Stop traversing a netlist at the specified signals or instances.

- Use `-name` to label the newly created stopat so it can be reviewed or referenced in the future.
- Use `-annotation` to annotate the specified stopat with the description enclosed by the curly braces.
- Use `-replace` to replace the signal or instance name or the task context constraint type for the specified stopat; for example:
`visualize -stopat {out0} -name justify0 -replace`
 - Use `-name` to specify the stopat you are editing.
 - Use `-annotation` to add or change the stopat's annotation.

```
-abstract -counter signal_name+  
  -name name  
  -annotation {annotation}  
  -replace  
  -remove  
  -window window_name
```

Create a counter abstraction for one or more specified signals.

- Use `-name` to label the newly created abstraction so it can be reviewed or referenced in the future.
- Use `-annotation` to annotate the specified abstraction with the description enclosed by the curly braces.
- Use `-replace` to replace the signal or instance for the specified abstraction or to change the task context constraint type.
 - Use `-name` to specify the stopat you are editing.
 - Use `-annotation` to add or change the stopat's annotation.
- Use `-remove` to remove the specified counter abstraction.
 - Use `-name` to specify the abstraction you are removing.
 - Use `-annotation` to remove the abstraction's annotation.

JasperGold Apps Command Reference Manual

General Commands

```
-set_as_AR  
    -window window_name
```

Modify the AR to reflect the changes made with task context constraints.

Why

```
-why -target signal_name -cycle cycle+  
    [-transitive]  
    [-ignore_connected_assumptions]  
    [-simple_buffer_transitive]  
    [-stop_at_hierarchy_change]  
    [-window window_name] | [-reset_window]
```

Find signals in the immediate fanin of the specified signal that affect its value at the specified cycle.

By default, the tool applies your `visualize` command to the current window.

- If you have already launched a Reset Analysis window, use `-reset_window` to apply your command to the Reset Analysis window.
- Or use `-window` to apply your command to a window other than the current window.
- Use `-transitive` to perform transitive Why analysis and find the root cause of the signal value.
- Use `-ignore_connected_assumptions` to prevent the search for assumptions that might force values of primary input targets (Why on boundary).
- Use `-simple_buffer_transitive` to enable transitivity through *simple* buffers (inverters and netmixes are NOT considered *simple* buffers).
- Use `-stop_at_hierarchy_change` to disable *default* transitivity through the *first* hierarchy change (instance port crossing).

Return

When visualizing properties without `-bg` these are the return values:

- `visualize -violation` returns `cex`, `ar_cex`, or `proven`.
- `visualize -cover` and `visualize -sanity` return `covered`, `ar_covered`, or `unreachable`.

JasperGold Apps Command Reference Manual

General Commands

- `visualize -violation`, `visualize -cover`, and `visualize -sanity` can return the following:
 - `undetermined` – No trace was found because the tool reached the maximum trace length.
 - `error` – The tool detected overconstraint.

Note: Not all engines can detect overconstraint; therefore, it is possible for the same configuration to return a value of `unreachable/proven` or a value of `error`, depending on the engine.

Examples

```
% analyze requirement.v
% elaborate
% reset rst
% visualize -violation -property requirement

// req must occur in the first 5 cycles
% visualize -at_least_once req 1:5
// req must be followed by a grant within 5 cycles
% visualize -at_least_once gnt 1:5 -after req
```

The following is the expected use model.

```
% # possibly some preconfigurations, for example:
% visualize -min_length 5
% # now the main command:
% visualize -cover {out}
% # at this point one trace has appeared
% # now we may add more configurations, for example:
% visualize -at_least_once in 2:3
% # then we can replot to see what changed
% visualize -replot
% # and repeat the sequence of adding configurations, then replotting, as many
% # times as we want
```

Example for color annotation:

```
// Define a new color annotation
% visualize -define_color_annotation {{a 1 red} {b 2 blue}} red_and_blue
```

JasperGold Apps Command Reference Manual

General Commands

```
// Activate the new color annotation  
% visualize -activate_color_annotation red_and_blue  
  
// Delete the color annotation  
% visualize -define_color_annotation {} red_and_blue
```

Example for -replace_conf:

```
// Specifying a configuration  
% visualize -at_least_once {(foo == 1'b01)} 1:1 -name myConf  
  
// Modifying the myConf configuration  
% visualize -replace_conf -force {(foo == 1'b01)} 1:2 -name myConf
```

Example for -offset_conf:

In the following example, we specify conf1. We then use `-offset_conf` to change the cycle range from 3:5 to 5:7.

```
// Specify conf1  
% visualize -force exprA 3:5 -name conf1  
  
// Offset the range for conf1  
% visualize -offset_conf conf1 2  
  
// The resulting conf1 is the equivalent of the following command:  
% visualize -force exprA 5:7 -name conf1
```

Example for -extract_conf:

In the following example, if we have a trace where `in == 2'b00` for the first two cycles and then `2'b01` for the third and fourth cycles, we can say:

```
% visualize -extract_conf {my_input {1} my_input {3:4}} -value -name myConfName  
5:$
```

This is the equivalent of:

```
% visualize -at_least_once {my_input != 2'b00 && my_input != 2'b01} -name  
myConfName 5:$
```

Example for specifying a waveform sequence:

```
% visualize -signal A -sequence {0 1 1 0 0 1 1 1 0}
```

JasperGold Apps Command Reference Manual

General Commands

```
# The {3} shows the number of times the value is repeated
% visualize -signal B -sequence {0 1{3} 0 1 0}
% visualize -signal C -sequence {0 1 1 1 0 0 x x 0}
% visualize -signal D -sequence {0 1{3} 0{2} x{3} 0}
% visualize -signal *busA -sequence {8'b0001_0000 8'hffff 8'h0 8'd0}
% visualize -signal busB -sequence {8'b0001_0000 8'hffff 8'h0{2}}
```

Example for applying QuietTrace™ to a cycle range:

In the following example, the second command overrides (and erases) the first one. That is, the commands are not cumulative.

```
% visualize -quiet true 4:9
% visualize -quiet true 12:100
```

Example for loading a VCD file generated by simulation when you have a wrapper module:

```
// Simulation environment:
// SimulationEnv.TestBench.<Instances_I_want_to_verify>
//
// Formal environment:
// Formal_wrapper.<Instances_I_want_to_verify>
//
// Use the following command:
% visualize -load -vcd my_vcd -hier_map -from SimulationEnv.TestBench -to
Formal_wrapper
% visualize -load -fsdb my_fsdb -hier_map -from SimulationEnv.TestBench -to
Formal_wrapper
% visualize -load -shm my_shm -hier_map -from SimulationEnv.TestBench -to
Formal_wrapper
```

See Also

[set_engine_mode](#)
[set_proofgrid_mode](#)
[set_prove_time_limit](#)
[set_trace_optimization](#)

waveform

Description

Use the `-waveform` command to manipulate waveforms imported from trace files or PLI simulation runs. Each waveform has a source name and path and a start and end simulation time.

Syntax

```
waveform -import
  [-fsdb | -vcd [-vcf_lang (vhdl | verilog)]] <file>+
  | [-shm <directory>+]
  [-hier_path <path>]
  [-start_time <integer>]
  [-end_time <integer>]

waveform -import
  [-ddk <file>+]

waveform -list
  [-baseline] [-silent]

waveform -reset <expr>

waveform -show <id_tcl_list>
  [-baseline] [-silent]
```

JasperGold Apps Command Reference Manual

General Commands

Argument	Definition
<pre>-import [-fsdb -vcd [-vcd_lang (vhdl verilog)] file+ [-shm directory] [-hier_path path] [-start_time integer] [-end_time integer]</pre>	<p>Scan VCD, FSDB, or SHM trace files produced by a previous simulation run.</p> <ul style="list-style-type: none">■ Use <code>-shm</code> to import SHM directories which contain SST2 databases.■ To specify the language used in an imported VCD file, use <code>-vcd_lang</code> followed by either <code>vhdl</code> or <code>verilog</code>.■ Use <code>-hier_path</code> to capture the hierarchical difference between the module that was simulated and the module that is under test.■ Use <code>-start_time</code> to skip the initial chunk of simulation data. Typically, you will use this switch to skip the reset sequence and so forth.■ Use <code>-end_time</code> to ignore the last chunk of simulation data.
<pre>-import [-ddk file+]</pre>	<p>Import one or more specified DDK files produced by a previous simulation run using PLI.</p>
<pre>-list [-baseline] [-silent]</pre>	<p>List the IDs for the waveforms that match the specified criteria.</p> <ul style="list-style-type: none">■ Use <code>-baseline</code> to use the baseline waveforms.■ Use <code>-silent</code> to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

JasperGold Apps Command Reference Manual

General Commands

```
-reset expr
```

Set the reset condition used for waveform analysis.

By default, the tool analyzes one or more consecutive segments inside a trace (waveform) to determine the reset condition. These segments may be the whole trace or a portion of the trace. To specify a portion, use one or both of the switches `-start_time` and `-end_time` with the commands `waveform -import` and `scan -trace`. When using PLI, use `$jasper_ddk_init`, `$jasper_ddk_dump_on()`, and `$jasper_ddk_dump_off()`.

In addition or as an alternative, you can specify a reset condition. Specify `expr` as `signal_name==0` or `signal_name==1`. If you define a reset condition with the command `waveform -reset`, the tool analyzes a single segment where the reset condition evaluates to false. If the trace has multiple consecutive segments where the reset condition evaluates to false, the tool may analyze all segments or just the final segment. See [“set_waveform_import_multiple_segments”](#) on page 1307 for additional information.

Note: You can also add a testbench signal as the reset condition. To do so, specify the full path to the testbench signal. For example,

```
waveform -reset "testbench.rst==0"  
scan -trace -vcd -hier_path testbench.pattern_i
```

If you do not specify a reset condition, `waveform -reset` returns the current value.

```
-show id_tcl_list
```

```
[-baseline]  
[-silent]
```

Show the details of the waveforms.

- Use `-baseline` to show the baseline waveforms.
- Use `-silent` to turn off the output to the screen and return information with a Tcl return value, which is helpful in Tcl scripts.

Return

No value returned

Examples

```
% waveform -import sys.fsdb -hier_path sys.core0.exe -start_time 20000  
% waveform -import -fsdb fsdb.99 fsbd.100 -hier_path top.io_port.dram04  
% waveform -import -shm test.shm  
% waveform -list  
% waveform -reset "bus_rst_l==0"
```

See Also

[check_bps](#)
[scan](#)
[scope](#)
[set_waveform_import_multiple_segments](#)

JasperGold Apps Command Reference Manual

General Commands

Configuration Commands

This chapter includes documentation for the configuration commands. It includes one section for each root command name (for example, engine_mode for the commands get_engine_mode and set_engine_mode). Each section describes the command variants, lists the syntax with definitions, and provides the defaults and return values.

Note: The syntax for some of the longer commands is organized by category. Click on the category heading to go directly to the relevant descriptions.

analyze_libnonamehide

Commands

`set_analyze_libnonamehide`
`get_analyze_libnonamehide`

Description

Use the command `set_analyze_libnonamehide` to control the `+libnonamehide` switch in the `analyze` command. When you enable this feature, the tool keeps all read modules in a temporary library during compilation.



- For the default global file compilation unit (GFCU) mode of `analyze`, this setting is on by default.
- For multi- and single-file compilation unit (MFCU and SFCU) mode, this setting is off by default.
- Use `set_analyze_libnonamehide default` to return the tool to the GFCU, MFCU, and SFCU default settings.

Use the command `get_analyze_libnonamehide` to report whether the `analyze` command keeps all read modules in a temporary library during compilation.

Syntax

`set_analyze_libnonamehide (default | on | off)`

<code>default</code>	Use the default setting: <ul style="list-style-type: none">■ GFCU – on■ MFCU and SFCU – off
<code>on</code>	Keep all read modules in a temporary library during compilation. This option is the default when in GFCU mode.
<code>off</code>	Disable storing read modules in a temporary directory. This option is the default when in MFCU or SFCU mode.

JasperGold Apps Command Reference Manual

Configuration Commands

get_analyze_libnonamehide

No arguments

Default

The default depends on the file compilation mode:

- For the default GFCU mode of analyze, this setting is on by default, the `analyze` command does store read modules in a temporary directory.
- For MFCU and SFCU mode, this setting is off by default, the `analyze` command does not store read modules in a temporary directory.

Return

`get_analyze_libnonamehide` returns the value of the variable `analyze_libnonamehide`, on, off, or default.

Examples

```
% set_analyze_libnonamehide on  
% get_analyze_libnonamehide
```

See Also

[analyze](#)

[set_analyze_librescan](#)

analyze_librescan

Commands

set_analyze_librescan
get_analyze_librescan

Description

Use the command `set_analyze_librescan` to control the `+librescan` switch in the `analyze` command. When you enable this feature, the tool always scans all Verilog® directories in the directory list when looking for a new module.

Use the command `get_analyze_librescan` to report whether the `analyze` command always scans all Verilog directories in the directory list when looking for a new module.

Syntax

set_analyze_librescan (on off)
on Scan all Verilog directories in the directory list when looking for a new module.
off Disable library re-scanning. <i>This option is the default.</i>
get_analyze_librescan
No arguments

Default

By default, the `analyze` command does not scan all Verilog directories in the directory list when looking for a new module.

Return

`get_analyze_librescan` returns the value of the variable `analyze_librescan`, on or off.

Examples

```
% set_analyze_librescan on  
% get_analyze_librescan
```

See Also

[analyze](#)
[set_analyze_libnonamehide](#)

analyze_libunboundsearch

Commands

`set_analyze_libunboundsearch`
`get_analyze_libunboundsearch`

Description

Use the `set_analyze_libunboundsearch` command to control the `+libunboundsearch` switch in the `analyze` command. When you enable this feature, the tool does not clear `-L` options when analyzing in multi- and single-file compilation unit (MFCU and SFCU) mode.

Use the command `get_analyze_libunboundsearch` to report whether the `analyze` command will clear `-L` options when in MFCU or SFCU mode.

Syntax

<code>set_analyze_libunboundsearch (on off)</code>
<code>on</code> Enable cross-file compilation unit <code>-L</code> option.
<code>off</code> Disable cross-file compilation unit <code>-L</code> option. <i>This option is the default.</i>
<code>get_analyze_libunboundsearch</code>
No arguments

Default

By default, the `analyze` command disables the cross-file compilation unit `-L` option.

Return

`get_analyze_libunboundsearch` returns the value of the variable `analyze_libunboundsearch`, `on` or `off`.

Examples

```
% set_analyze_libunboundsearch on
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_analyze_libunboundsearch
```

See Also

[analyze](#)

[set_analyze_librescan](#)

arch_prll_property_consolidation

Commands

`set_arch_prll_property_consolidation`
`get_arch_prll_property_consolidation`

Description

Use the command `set_arch_prll_property_consolidation` to generate one parallel property instead of $n \choose 2$ parallel properties, which is helpful when dealing with tables that have a large number of rows. With this feature, you can check parallel properties on large tables with a single generated property that is both easy to understand and easy to debug. The default value is 50; thus, if you do not change the default value, and if a table has fewer than 50 rows, the tool generates $n \choose 2$ properties, and each property covers unique row pairs in the table. If your table has 50 or more rows, the tool creates one property that covers all the cases.

Use the `get_arch_prll_property_consolidation` command to get the value of the variable `arch_prll_property_consolidation`.

Syntax

`set_arch_prll_property_consolidation <N>`

`N` If a table has more than the specified number of rows, generate one parallel property instead of $n \choose 2$ parallel properties.

`get_arch_prll_property_consolidation`

No arguments

Default

The default value is 50.

Return

`get_arch_prll_property_consolidation` returns the value of the variable `arch_prll_property_consolidation`.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_arch_prll_property_consolidation 40  
% get_arch_prll_property_consolidation
```

See Also

[check_arch](#)

auto_disable_related_covers

Commands

set_auto_disable_related_covers
get_auto_disable_related_covers

Description

By default, the tool automatically disables all enabled related covers when you disable their main properties. Use `set_auto_disable_related_covers` to turn this feature on or off.

Use the `get_auto_disable_related_covers` to determine whether related covers will be disabled along with their main properties.

Syntax

set_auto_disable_related_covers (true false)	
true	Disable all enabled related covers along with their main properties. <i>This option is the default.</i>
false	Do not disable enabled related covers along with their main properties.
get_auto_disable_related_covers	
No arguments	

Default

By default, the tool automatically disables all enabled related covers along with their main properties (`set_auto_disable_related_covers=true`).

Return

`get_auto_disable_related_covers` returns the value of the variable `auto_disable_related_covers`.

Examples

```
% set_auto_disable_related_covers false
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_auto_disable_related_covers
```

See Also

No related commands

auto_syn_first_trace_attempt

Commands

set_auto_syn_first_trace_attempt
get_auto_syn_first_trace_attempt

Description

Use `set_auto_syn_first_trace_attempt` to control the length of traces `auto_syn -flow` generates when you specify a target property with the `-target_property` switch.

Note: If you do not specify a target property, this command has no affect.

Use the `get_auto_syn_first_trace_attempt` command to determine the length of the traces that `auto_syn -flow` will generate when you specify a target property.

Syntax

```
set_auto_syn_first_trace_attempt <N>
```

`N` Specify the length of the traces in number of cycles that `auto_syn -flow` will generate when the `-target_property` switch is used.

```
get_auto_syn_first_trace_attempt
```

No arguments

Default

The default value is 30.

Return

`get_auto_syn_first_trace_attempt` returns the value of the variable `auto_syn_first_trace_attempt`.

Examples

```
% set_auto_syn_first_trace_attempt 150
% get_auto_syn_first_trace_attempt
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[auto_syn](#)
[scan](#)

auto_syn_structural_trace_generator

Commands

`set_auto_syn_structural_trace_generator`
`get_auto_syn_structural_trace_generator`

Description

By default, when you use the `auto_syn -flow... -target_property <property_name>` command, the tool generates traces based on the specified target property. If you prefer to generate traces based on more general structural analysis even if you specify a target property, set the `auto_syn_structural_trace_generator` variable to `true`.

NOTE: If you do not use the `-target_property` switch, `auto_syn` generates structural traces regardless of the `auto_syn_structural_trace_generator` variable setting.

Use `get_auto_syn_structural_trace_generator` to determine whether traces will be based on a specified target property or on structural analysis.

Syntax

`set_auto_syn_structural_trace_generator (true | false)`

<code>true</code>	Generate general traces from RTL structures like FSMs and counters even if a target property is specified.
<code>false</code>	Generate traces based on the specified target property. <i>This option is the default</i> if a target property is specified.

`get_auto_syn_structural_trace_generator`

`No arguments`

Default

By default, `auto_syn` generates traces based on the specified target property if you use the `-target_property` switch. Otherwise, `auto_syn` generates traces based on structural analysis.

Return

`get_auto_syn_structural_trace_generator` returns the value of the variable `auto_syn_structural_trace_generator`.

Examples

```
% set_auto_syn_structural_trace_generator true  
% get_auto_syn_structural_trace_generator
```

See Also

[auto_syn](#)
[scan](#)

automatic_library_search

Commands

set_automatic_library_search
get_automatic_library_search

Description

By default, analyze only searches for modules in the default library work. You can explicitly specify additional user-declared Verilog packages or a VHDL or Verilog library with the analyze -L switch.

Use the set_automatic_library_search command to enable future analyze commands to automatically search for modules in user-defined packages or libraries. This behavior is equivalent to adding an -L switch to analyze commands for each user-declared package or library.

Note: If the same name exists for multiple definitions of a module in different libraries, set_automatic_library_search on chooses any of these definitions during elaborate, and it does not generate a warning message for this situation.

Use the get_automatic_library_search command to determine whether searches for modules in user-declared libraries for analyze are enabled.

Syntax

set_automatic_library_search (on off)	
on	Enable searches for modules in user-declared libraries for analyze.
off	Disable searches for modules in user-declared libraries for analyze. <i>This option is the default.</i>
get_automatic_library_search	
No arguments	

Default

By default, searches for modules in user-declared libraries for analyze are disabled.

Return

`get_automatic_library_search` returns the value of the configuration variable `automatic_library_search`, on or off.

Examples

```
% set_automatic_library_search on

# Analyzes dut module inside my_lib library.
% analyze -sv -lib my_lib dut.sv

# Analyzes top module, which instantiates dut.
# "set_automatic_library_search" implicitly adds
# "-L my_lib" switch to the following command:
% analyze -sv top.sv
% elaborate -top top
```

See Also

[analyze](#)
[elaborate](#)

cache_proof_simplification

Commands

`set_cache_proof_simplification`
`get_cache_proof_simplification`

Description

`set_cache_proof_simplification` enables or disables caching of the result of the simplification of the verification task, which is controlled by `set_proof_simplification`. Use this command for a faster proof startup when running `prove` or `visualize` on the same task multiple times. Proof simplification caching is off by default.



- The tool stores the cache in the project directory, and these files store the entire netlist, which can be some megabytes per cached result on larger designs.
- You can clear the cache with `clear -cache_proof_simplification`.
- Proof simplification normally uses proven properties to simplify the netlist, but if you use caching, the tool cannot take advantage of newly proven properties. As a consequence, proof performance can be affected.
- To take advantage of the cache, a task must be identical to a previous prove or Visualize run except for property status. Additional proven properties should not affect the cache hit rate.
- You will know the tool is using a cached result by checking the log for a message such as the following:

```
Performing Proof Simplification...
0.Hp: Loaded cached simplification result
Proof Simplification completed in 1.07 s
```

Use `get_cache_proof_simplification` to determine if proof simplification caching is enabled.

Syntax

set_cache_proof_simplification (on off)
on Enable proof simplification caching.
off Disable proof simplification caching. <i>This option is the default.</i>
get_cache_proof_simplification
No arguments

Default

By default, proof simplification caching is disabled.

Return

get_cache_proof_simplification returns on or off.

Examples

```
% set_cache_proof_simplification on  
% get_cache_proof_simplification
```

See Also

[prove](#)
[set_proof_simplification](#)
[visualize](#)

cache_reset_values

Commands

set_cache_reset_values
get_cache_reset_values

Description

Use the `set_cache_reset_values` command to specify whether the tool should cache reset values from VCD files. Because VCD dumps can be large and parsing these files is time-consuming, caching reset values speeds up successive uses of the `reset -vcd` command. It works by resetting the circuit to a known state if the registers, VCD dump, and time slot you are interested in have not changed.

Use the `get_cache_reset_values` command to find out if reset state caching is enabled for VCDs.

Syntax

set_cache_reset_values (true false)
true Enable reset value caching from VCDs. <i>This option is the default.</i>
false Disable reset value caching from VCDs.
get_cache_reset_values
No arguments

Default

By default, caching is enabled.

Return

`get_cache_reset_values` returns true or false.

Examples

```
% set_cache_reset_values false
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_cache_reset_values
```

See Also

[reset](#)

capture_elaborated_design

Commands

`set_capture_elaborated_design`
`get_capture_elaborated_design`

Description

By default, the tool does not keep the elaborated JasperGold Apps netlist after you run the `elaborate` command. Use the `set_capture_elaborated_design` command to specify that you want to keep this file so you can use it with commands such as `save -elaborated_design` and `session -new -reuse_elaborated_design`.

CAUTION: `capture_elaborated_design` must be on before you run the `elaborate` command. The tool saves the elaborated design in `/tmp`, and depending on design size, the size of the file can be quite large.

Note: `set_capture_elaborated_design` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

Use the `get_capture_elaborated_design` command to find out if the tool will capture the elaborated JasperGold Apps netlist.

Syntax

<code>set_capture_elaborated_design (on off)</code>
<code>on</code> Capture the elaborated design.
<code>off</code> Do not capture the elaborated design. <i>This option is the default.</i>
<code>get_capture_elaborated_design</code>
No arguments

Default

By default, the tool does not keep the elaborated JasperGold Apps netlist.

Return

`get_capture_elaborated_design` returns the value of the configuration variable `capture_elaborated_design` on or off.

Examples

```
% set_capture_elaborated_design on  
% get_capture_elaborated_design
```

See Also

[get_capture_elaborated_design](#)
[restore](#)
[save](#)
[session](#)

cell_define_debug_visibility

Commands

`set_cell_define_debug_visibility`
`get_cell_define_debug_visibility`

Description

Use the `set_cell_define_debug_visibility` command to specify that you want the tool to go inside cell define blocks during design traversals, that is, fanin, fanout, driver, and load. If you set this configuration variable to `on`, both the Tcl and GUI design traversals consider results that are inside cell defines as valid results.

Use `get_cell_define_debug_visibility` to get the value of the variable `cell_define_debug_visibility`.

Syntax

```
set_cell_define_debug_visibility (on | off)
```

`on` Consider results that are inside cell defines as valid results.

`off` Do not go inside cell define blocks during design traversals, that is, fanin, fanout, driver, and load. *This option is the default.*

```
get_cell_define_debug_visibility
```

No arguments

Default

By default, design traversals skip cell define blocks (`off`).

Return

`get_cell_define_debug_visibility` returns the value of the variable `cell_define_debug_visibility`, `on` or `off`.

Examples

```
% set_cell_define_debug_visibility on
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_cell_define_debug_visibility
```

See Also

[get_fanin](#)
[get_fanout](#)
[set_function_debug_visibility](#)

clock_auto_stabilize

Commands

set_clock_auto_stabilize
get_clock_auto_stabilize

Description

The `set_clock_auto_stabilize` command enables support for RTL with potential race hazards due to looping clocking signals.

WARNING: With this command turned on, functionality in the design that depends on glitches might not be modeled correctly. This command can produce unexpected results. Contact support@cadence.com for any issues related to this command.

To learn more about situations that might trigger an error with a recommendation to use this command, refer to the *JasperGold Clock Handling* mini manual. To access the manual, go to the JasperGold Apps COS page (http://support.cadence.com/jasergold_apps) and click on the Product Manuals link.

Use `get_clock_auto_stabilize` to determine whether support for potential race hazards is enabled.

Syntax

```
set_clock_auto_stabilize (on | off)
```

on	Enable support for potential race hazards. The tool will issue a warning message WCK113 for each stabilized clocking signal.
off	Disable support. The tool will fail with the error message ECK113 on potential race hazards. <i>This option is the default.</i>

```
get_clock_auto_stabilize
```

No arguments

Default

By default, support for potential race hazards is disabled (`off`).

Return

`get_clock_auto_stabilize` returns the value of the configuration variable `set_clock_auto_stabilize`, `on` or `off`.

Examples

```
% visualize -violation -property {<embedded>::arb.arb_fifo1.no_fifo_underflow}
ERROR (ECK113): Potential critical race hazard detected...
% set_clock_auto_stabilize on
% visualize -violation -property {<embedded>::arb.arb_fifo1.no_fifo_underflow}
% get_clock_auto_stabilize
```

See Also

No related commands

clock_handle_self_disabling_registers

Commands

`set_clock_handle_self_disabling_registers`
`get_clock_handle_self_disabling_registers`

Description

Use the command `set_clock_handle_self_disabling_registers` to disable (or re-enable) support for flops that combinatorially gate their own clock. These flops are allowed to disable themselves causing a glitch on their clock pin. See below when d is 0:

```
reg q;  
assign gclk = q & clk;  
always @ (posedge gclk) q <= d;
```

Note:

- The glitch will not be visible in the Visualize™ window.
- To identify this type of flop, the tool issues a single message for each relevant proof, which summarizes dependency on disabling clocks and provides example signals.

Use the command `get_clock_handle_self_disabling_registers` to find out if support is enabled.

Syntax

```
set_clock_handle_self_disabling_registers (on | off)
```

on	Enable support for flops that gate their own clock. <i>This option is the default.</i>
----	--

off	Disable support for flops that gate their own clock.
-----	--

```
get_clock_handle_self_disabling_registers
```

```
No arguments
```

Default

By default, support is enabled (on).

Return

`get_clock_handle_self_disabling_registers` returns the value of the configuration variable `clock_handle_self_disabling_registers` (on or off).

Examples

```
% get_clock_handle_self_disabling_registers  
% set_clock_handle_self_disabling_registers off  
% prove -all
```

See Also

[set_clock_auto_stabilize](#)
[set_clock_process_reconvergent_logic](#)

clock_process_reconvergent_logic

Commands

`set_clock_process_reconvergent_logic`
`get_clock_process_reconvergent_logic`

Description

Reconvergent logic in clock pins can cause glitches in design logic. Therefore, if the tool encounters reconvergent logic, it errors out, which prevents verification. Use the command `set_clock_process_reconvergent_logic` to process and remove reconvergent logic to avoid erroring out.

Note:

- This setting can cause state removal; therefore, we recommend that you carefully review the glitch behavior in the design.
- The trace might not always show the glitches that the engines detected due to the low signal resolution with the internal fast clock.

Use the command `get_clock_process_reconvergent_logic` to find out if reconvergent logic processing and removal is enabled.

Syntax

`set_clock_process_reconvergent_logic (on | off)`

`on` Process and remove reconvergent logic.

`off` Error out if reconvergent logic is present in clock pins. *This option is the default.*

`get_clock_process_reconvergent_logic`

No arguments

Default

By default, `clock_process_reconvergent_logic` is off. The tool errors out if it encounters reconvergent logic in clock pins.

Return

`get_clock_process_reconvergent_logic` returns the value of the `clock_process_reconvergent_logic` variable, on or off.

Examples

```
% set_clock_process_reconvergent_logic on  
% get_clock_process_reconvergent_logic
```

See Also

No related commands

clock_structural_glitch_analysis

Commands

```
set_clock_structural_glitch_analysis  
get_clock_structural_glitch_analysis
```

Description

Use the `set_clock_structural_glitch_analysis` command to enable or disable the clock structural glitch analysis that runs before proving a property. This analysis is responsible for issuing the WCK010 glitch-related warning messages. Turning this analysis on may help you catch an incomplete environment configuration. Turning this analysis off results in a faster proof flow start.

Use the command `get_clock_structural_glitch_analysis` to find out if structural glitch analysis is enabled.

Syntax

set_clock_structural_glitch_analysis (on off)
on Enable clock structural glitch analysis.
off Disable clock structural glitch analysis. <i>This option is the default.</i>
get_clock_structural_glitch_analysis
No arguments

Default

By default, `clock_structural_glitch_analysis` is off.

Return

`get_clock_structural_glitch_analysis` returns the value of the `clock_structural_glitch_analysis` variable, on or off.

Examples

```
% set_clock_structural_glitch_analysis on
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

No related commands

cmd_time_limit

Commands

`set_cmd_time_limit`
`get_cmd_time_limit`

Description

Use the `set_cmd_time_limit` command to change the time limit for all commands except those listed below.

- `check_arch -prove`
- `check_conn -prove`
- `check_csr -prove`
- `check_sec -prove`
- `check_spv -prove`
- `check_xprop -prove`
- `prove`
- `visualize`

Note: These commands honor `prove_time_limit`. Refer to [“See Also”](#) on page 978.

Use the `get_cmd_time_limit` command to display the time limit for all commands except those listed above. This command is context-sensitive (it applies to the current task).

JasperGold Apps Command Reference Manual

Configuration Commands

Syntax

```
set_cmd_time_limit <time_limit>
```

time_limit Use the specified time limit for all commands except those listed below.

- check_arch -prove
- check_conn -prove
- check_csr -prove
- check_sec -prove
- check_spv -prove
- check_xprop -prove
- prove
- visualize

Specify the time limit as an integer followed by one of the following (no spaces):

- s = seconds
- m = minutes
- h = hours

To remove the limit, use 0s.

```
get_cmd_time_limit
```

No arguments

Default

By default, the command time limit is 7200s.

Return

`get_cmd_time_limit` returns the value of the `cmd_time_limit` variable.

Examples

```
% set_cmd_time_limit 2m  
% get_cmd_time_limit
```

See Also

[configure](#)
[set_elab_time_limit](#)
[set_prove_per_property_time_limit](#)
[set_prove_time_limit](#)

command_license_wait_interval

Commands

set_command_license_wait_interval
get_command_license_wait_interval

Description

Use the `set_command_license_wait_interval` to set the time interval between license checkout attempts.

Note: This setting only affects checkouts requested by Tcl commands.

Use the `get_command_license_wait_interval` command to return the current time interval between license checkout attempts.

Syntax

<code>set_command_license_wait_interval <wait_interval></code>
<code>wait_interval</code> Set the time interval between license checkout attempts.
<code>get_command_license_wait_interval</code>
No arguments

Default

By default, the command wait interval is 1s.

Return

`get_command_license_wait_interval` returns the value of the `command_license_wait_interval` variable.

Examples

```
% set_command_license_wait_interval 2s
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[set_command_license_wait_retries](#)

command_license_wait_retries

Commands

`set_command_license_wait_retries`
`get_command_license_wait_retries`

Description

Use the `set_command_license_wait_retries` to set the number of checkout retries the tool should attempt before failing.

Note: This setting only affects license checkouts requested by Tcl commands.

Use the `get_command_license_wait_retries` command to return the number of checkout retries the tool will attempt before failing.

Syntax

```
set_command_license_wait_retries <N>
```

`N` Set the number of checkout retries the tool should attempt before failing.

```
get_command_license_wait_retries
```

No arguments

Default

The default is 0.

Return

`get_command_license_wait_retries` returns the value of the `command_license_wait_retries` variable.

Examples

```
% set_command_license_wait_retries 1  
% get_command_license_wait_retries
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[set_command_license_wait_interval](#)

complexity_manager_allow_link_expansion

Commands

`set_complexity_manager_allow_link_expansion`
`get_complexity_manager_allow_link_expansion`

Description

By default, Complexity Manager allows link (underlined) node expansion. Use the command `set_complexity_manager_allow_link_expansion` to disable link node expansion and to re-enable the default setting.

Use the command `get_complexity_manager_allow_link_expansion` to learn whether Complexity Manager is set to allow link (underlined) node expansion.

Syntax

<code>set_complexity_manager_allow_link_expansion (on off)</code>
<code>on</code> Enable link (<u>underlined</u>) node expansion. <i>This option is the default.</i>
<code>off</code> Disable link expansion.
<code>get_complexity_manager_allow_link_expansion</code>
No arguments

Default

By default, `set_complexity_manager_allow_link_expansion` is on.

Return

`get_complexity_manager_allow_link_expansion` returns the value of `set_complexity_manager_allow_link_expansion`, on or off.

Examples

```
% set_complexity_manager_allow_link_expansion on  
% get_complexity_manager_allow_link_expansion
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

No related commands

complexity_manager_compute_statistics

Commands

`set_complexity_manager_compute_statistics`
`get_complexity_manager_compute_statistics`

Description

By default, Complexity Manager only computes design statistics when you request it. Use the command `set_complexity_manager_compute_statistics` to specify that Complexity Manager will compute design information for all nodes as soon as they are added to the tree and to re-enable the default setting.

Note: In the GUI, you can use the *View* menu to request design statistics.

Use the command `get_complexity_manager_compute_statistics` to learn whether Complexity Manager is set to compute design information for all nodes as soon as they are added to the tree.

Syntax

```
set_complexity_manager_compute_statistics (on | off)
```

on Compute design statistics for all nodes.

off Do not compute design statistics. *This option is the default.*

```
get_complexity_manager_compute_statistics
```

No arguments

Default

By default, `set_complexity_manager_compute_statistics` is off.

Return

`get_complexity_manager_compute_statistics` returns the value of `set_complexity_manager_compute_statistics`, on or off.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_complexity_manager_compute_statistics on  
% get_complexity_manager_compute_statistics
```

See Also

No related commands

complexity_manager_design_information_mode

Commands

`set_complexity_manager_design_information_mode`
`get_complexity_manager_design_information_mode`

Description

By default, Complexity Manager computes statistics in the same way Design Information does, which results in faster response times. Use the command

`set_complexity_manager_design_information_mode` to change the computation mode.

When you set `complexity_manager_design_information_mode` to `off`, node statistics do not include statistics for child link nodes.

Use the command `get_complexity_manager_design_information_mode` to learn whether Complexity Manager is set to compute statistics in the same way Design Information does.

Syntax

`set_complexity_manager_design_information_mode (on | off)`

`on` Compute Complexity Manager statistics in the same way Design Information does, including in the node statistics the statistics for the child link nodes as well. *This option is the default.*

`off` Construct the entire tree before computing any statistics and then compute statistics for each signal or property without including link node statistics.

`get_complexity_manager_design_information_mode`

No arguments

Default

By default, `set_complexity_manager_design_information_mode` is `on`.

JasperGold Apps Command Reference Manual

Configuration Commands

Return

`get_complexity_manager_design_information_mode` returns the value of `set_complexity_manager_design_information_mode`, on or off.

Examples

```
% set_complexity_manager_design_information_mode on  
% get_complexity_manager_design_information_mode
```

See Also

No related commands

complexity_manager_show_connected_assumptions

Commands

```
set_complexity_manager_show_connected_assumptions  
get_complexity_manager_show_connected_assumptions
```

Description

By default, the Complexity Manager tree displays connected assumptions. Use the command `set_complexity_manager_show_connected_assumptions` to hide connected assumptions and to re-enable the default setting.

Use the command `get_complexity_manager_show_connected_assumptions` to learn whether the Complexity Manager tree is set to display connected assumptions.

Syntax

set_complexity_manager_show_connected_assumptions (on off)	
on	Show connected assumptions in the Complexity Manager <i>Tree</i> pane. <i>This option is the default.</i>
off	Hide connected assumptions in the Complexity Manager tree.
get_complexity_manager_show_connected_assumptions	
No arguments	

Default

By default, `set_complexity_manager_show_connected_assumptions` is on.

Return

`get_complexity_manager_show_connected_assumptions` returns the value of `set_complexity_manager_show_connected_assumptions`, on or off.

Examples

```
% set_complexity_manager_show_connected_assumptions on
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_complexity_manager_show_connected_assumptions
```

See Also

No related commands

complexity_manager_show_inputs

Commands

`set_complexity_manager_show_inputs`
`get_complexity_manager_show_inputs`

Description

By default, the Complexity Manager tree displays inputs. Use the command `set_complexity_manager_show_inputs` to hide inputs and to re-enable the default setting.

Use the command `get_complexity_manager_show_inputs` to learn whether the Complexity Manager tree is set to display inputs.

Syntax

<code>set_complexity_manager_show_inputs (on off)</code>	
<code>on</code>	Show inputs in Complexity Manager <i>tree</i> pane. <i>This option is the default.</i>
<code>off</code>	Hide inputs in Complexity Manager tree.
<code>get_complexity_manager_show_inputs</code>	
No arguments	

Default

By default `set_complexity_manager_show_inputs` is on.

Return

`get_complexity_manager_show_inputs` returns the value of `set_complexity_manager_show_inputs`, on or off.

Examples

```
% set_complexity_manager_show_inputs on
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_complexity_manager_show_inputs
```

See Also

No related commands

complexity_manager_show_link_statistics

Commands

`set_complexity_manager_show_link_statistics`
`get_complexity_manager_show_link_statistics`

Description

By default, Complexity Manager shows the design information for link (underlined) nodes. Use the command `set_complexity_manager_show_link_statistics` to hide design information for link nodes and to re-enable the default setting.

Use the command `get_complexity_manager_show_link_statistics` to learn whether Complexity Manager is set to display design information for link (underlined) nodes.

Syntax

`set_complexity_manager_show_link_statistics (on | off)`

`on` Show design information for link (underlined) nodes. *This option is the default.*

`off` Hide design information for link (underlined) nodes.

`get_complexity_manager_show_link_statistics`

No arguments

Default

By default `set_complexity_manager_show_link_statistics` is on.

Return

`get_complexity_manager_show_link_statistics` returns the value of `set_complexity_manager_show_link_statistics`, on or off.

Examples

```
% set_complexity_manager_show_link_statistics on
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_complexity_manager_show_link_statistics
```

See Also

No related commands

counter_speculative_detection

Commands

`set_counter_speculative_detection`
`get_counter_speculative_detection`

Description

Use the `set_counter_speculative_detection` to enable the speculative counter detection algorithm. With this algorithm, the tool employs more permissive rules that allow it to detect more counters than the default algorithm. This new algorithm is useful when counter signals or signals within the path back to the counter flop are sliced.

Use the command `get_counter_speculative_detection` to report whether the tool has been set to use more permissive counter detection rules..

Syntax

<code>set_counter_speculative_detection (on off)</code>	
<code>on</code>	Enable the speculative counter detection algorithm.
<code>off</code>	Disable the speculative counter detection algorithm. <i>This option is the default.</i>
<code>get_counter_speculative_detection</code>	
No arguments	

Default

By default `set_counter_speculative_detection` is off.

Return

`get_counter_speculative_detection` returns the value of the variable `counter_speculative_detection`, on or off.

Examples

```
% set_counter_speculative_detection on
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_counter_speculative_detection
```

See Also

No related commands

design_exploration_include_clock_logic

Commands

```
set_design_exploration_include_clock_logic  
get_design_exploration_include_clock_logic
```

Description

When `set_design_exploration_include_clock_logic` is on, Design Information, Design Space Coverage, and Complexity Manager include clock logic in their computations.

Note: In the case of Design Space Coverage, if you intend to include clock logic in computations, you must run this command before running the `get_design_space_coverage` command or opening the Design Space Coverage GUI.

Use the `get_design_exploration_include_clock_logic` command to report whether Design Information, Design Space Coverage, and Complexity Manager include clock logic in their computations.

Syntax

```
set_design_exploration_include_clock_logic (on | off)
```

on	Include clock logic in Design Information, Design Space Coverage, and Complexity Manager computations.
----	--

off	Do not include clock logic in Design Information, Design Space Coverage, and Complexity Manager computations. <i>This option is the default.</i>
-----	--

```
get_design_exploration_include_clock_logic
```

```
No arguments
```

Default

By default, clock logic is not included in computations (`off`).

Return

`get_design_exploration_include_clock_logic` returns the current setting (`on` or `off`).

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_design_exploration_include_clock_logic on  
% get_design_exploration_include_clock_logic
```

See Also

[get_design_info](#)
[get_design_space_coverage](#)

design_exploration_include_connected_assumptions

Commands

`set_design_exploration_include_connected_assumptions`
`get_design_exploration_include_connected_assumptions`

Description

By default, Design Information, Design Space Coverage, and Complexity Manager exclude connected assumptions in their computations. Use

`set_design_exploration_include_connected_assumptions` to include connected assumptions in design exploration computations and to re-enable the default setting.

Use the `get_design_exploration_include_connected_assumptions` command to report whether Design Information, Design Space Coverage, and Complexity Manager include connected assumptions in their computations.

Syntax

`set_design_exploration_include_connected_assumptions (on | off)`

<code>on</code>	Include connected assumptions in Design Information, Design Space Coverage, and Complexity Manager computations.
-----------------	--

<code>off</code>	Do not include connected assumptions in Design Information, Design Space Coverage, and Complexity Manager computations. <i>This option is the default.</i>
------------------	--

`get_design_exploration_include_connected_assumptions`

No arguments

Default

By default, connected assumptions are excluded (`off`).

Return

`get_design_exploration_include_connected_assumptions` returns the current setting (`on` or `off`).

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_design_exploration_include_connected_assumptions on  
% get_design_exploration_include_connected_assumptions
```

See Also

[get_design_info](#)
[get_design_space_coverage](#)

dst_mode

Commands

set_dst_mode
get_dst_mode

Description

Use `set_dst_mode` to enable or disable Design Tunneling. This setting affects how the *Visualize* and *Prove* features modify the analysis region, but does not affect the *Justify* or *Justify Conflict* features.

This command is context-sensitive. It applies to the current task. Use the `-env` switch to apply it globally.

Important

Differential analysis properties (X-Prop and path) are not supported with tunneling. The tool ignores these properties when verifying tasks that contain them.

Note: Use this command in an interactive JasperGold Design Tunneling flow.

Use `get_dst_mode` to display the Design Tunneling mode for the specified task.

Syntax

<code>set_dst_mode [(-task <task_name>) -env]</code>
(off on)
<code>-task</code> Enable or disable Design Tunneling for the specified task.
<code>task_name</code> Replace <code>task_name</code> with a period (.) to specify the current task.
<code>-env</code> Enable or disable Design Tunneling for all tasks.
Note: This switch removes any previous task-specific settings.
<code>off</code> Turn off Design Tunneling and use the entire cone of influence as the analysis region. This <i>option is the default</i> .

JasperGold Apps Command Reference Manual

Configuration Commands

on	Use registers as grid lines so that it is easy to Visualize the tunnel. The full cones of combinational logic driving the registers are either included or not included in the analysis region, except when you specify stopats.
<hr/>	
get_dst_mode [-task <task_name>]	
-task	Display the Design Tunneling mode setting for the specified task.
task_name	Replace <i>task_name</i> with a period (.) to specify the current task.

Default

By default, Design Tunneling is turned off and applies to the current task.

Return

get_dst_mode returns the value of the variable dst_mode.

Examples

```
% set_dst_mode on -env  
% get_dst_mode
```

See Also

[justify](#)
[prove](#)
[visualize](#)

elab_time_limit

Commands

`set_elab_time_limit`
`get_elab_time_limit`

Description

Use the `set_elab_time_limit` command to specify the maximum time that the tool spends on a single elaborate command.

Use the `get_elab_time_limit` command to display the maximum time in seconds that the tool spends on a single elaborate command.

Syntax

```
set_elab_time_limit <time_limit>
```

time_limit Use the specified time limit for elaboration.

Specify the time limit as an integer followed by one of the following (no spaces):

- *s* = seconds
- *m* = minutes
- *h* = hours

Note: To reset the default (unlimited), use 0s.

```
get_elab_time_limit
```

No arguments

Default

By default, the elaboration time limit is 0s (unlimited).

Return

`get_elab_time_limit` returns the value specified with the `set_elab_time_limit` command.

Examples

```
% set_elab_time_limit 3h  
% get_elab_time_limit
```

See Also

[configure](#)
[set_cmd_time_limit](#)
[set_prove_per_property_time_limit](#)
[set_prove_time_limit](#)
[set_property_compile_time_limit](#)
[set_task_compile_time_limit](#)

elaborate_quiet_trace

Commands

`set_elaborate_quiet_trace`
`get_elaborate_quiet_trace`

Description

Use the `set_elaborate_quiet_trace` command to control whether `elaborate` automatically generates global QuietTrace™ soft constraints.

Note: You must run this command before elaborating the design.

Use `get_elaborate_quiet_trace` to report whether `elaborate` will automatically generate global QuietTrace soft constraints.

Syntax

<code>set_elaborate_quiet_trace (on off)</code>
on Generate global QuietTrace soft constraints at elaboration.
off Do not generate global QuietTrace soft constraints at elaboration. <i>This option is the default.</i>
<code>get_elaborate_quiet_trace</code>
No arguments

Default

By default, `elaborate_quiet_trace` is off.

Return

`get_elaborate_quiet_trace` returns the value of the variable `elaborate_quiet_trace`.

Examples

```
% set_elaborate_quiet_trace on
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[elaborate](#)
[assume](#)

engine_job_timers

Commands

`set_engine_job_timers`
`get_engine_job_timers`

Description

Use the `set_engine_job_timers` command to specify the type of timer used for engine jobs in running proofs. By default, all timers used to control engine jobs are wall-clock timers, and all reported times use wall-clock time.

Use `get_engine_job_timers` to get the value of the variable `engine_job_timers`, which determines timer type used for engine jobs in running proofs.

Syntax

`set_engine_job_timers (on | off)`

`wallclock` Use wall-clock time. *This option is the default.*

`mixed` Revert to the behavior of versions prior to 2014.03, which used a mix of wall-clock and CPU time.

`get_engine_job_timers`

No arguments

Default

By default, the `engine_job_timers` variable uses wall-clock time.

Return

`get_engine_job_timers` returns the value of the variable `engine_job_timers`.

Examples

```
% set_engine_job_timers wallclock
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

No related commands

engine_mode

Commands

set_engine_mode
get_engine_mode

Description

Use `set_engine_mode` to select engines for `prove` and `visualize` that are tuned for different purposes. Specify a single engine or use curly braces to specify multiple engines (refer to “[Examples](#)” on page 1018). The default engine mode is one instance each of engines Hp, Ht, N, and B { Hp Ht N B }.

If you explicitly specify multiple engines to prove or Visualize properties in parallel, each engine triggers a separate process. In general, this feature provides a more effective way to achieve your objectives because you are not limited to a single engine. In addition, it can lead to overall faster results. Also see “[set_proofgrid_per_engine_max_jobs](#)” on page 1141 for details about specifying multiple jobs to deliver quick results with ProofGrid.

Note: Many (but not all) engines target properties serially. The first engine mode or process to converge on a proof or find a counterexample ends the proof of the target property. Use the *Proof Messages* tab in the main JasperGold Apps window to find out which engine proved the property.

Using parallel engines also provides for more robust exception handling (for example, out-of-memory conditions). This feature is most effective on high-performance machines, those with multiple processors and a large amount of physical memory. Using multiple engines on machines with limited CPU power or memory resources can lead to worse results compared to using a single-engine mode.



With regard to `set_engine_mode`...

- Only one instance of each engine is allowed except in the case of the three variants of engine H (H, Hp, Ht). However, you cannot combine H with either Hp or Ht. For example, { Hp Ht } is acceptable, but the tool issues an error when you attempt to use { H Hp }.
- The `set_engine_mode` command applies globally.
- For a shortcut, use an engine's letter, for example, use H for engineH. (Engine letters alone are case-insensitive.)

JasperGold Apps Command Reference Manual

Configuration Commands

- Refer to the “Jasper Engine Selection Guide” for details (available on support.cadence.com). This guide is also available from the tool’s *Help* menu (*Help – Mini Guides – Jasper Engine Selection*).

Use the `get_engine_mode` command to display the current engine mode.

Syntax

<code>set_engine_mode</code>	
<code>-auto N</code>	
<code> default</code>	
<code> ([engineB] [b] [engineBm] [bm]</code>	
<code> [engineC] [c] [engineC2] [c2]</code>	
<code> [engineD] [d]</code>	
<code> [engineG] [g] [engineG2] [g2]</code>	
<code> [engineH] [h]</code>	
<code> [engineHp] [hp] [engineHps] [hps]</code>	
<code> [engineHt] [ht] [engineHts] [hts]</code>	
<code> [engineI] [i] [engineJ] [j]</code>	
<code> [engineK] [k] [engineL] [l]</code>	
<code> [engineM] [m] [engineN] [n]</code>	
<code> [engineAB] [ab] [engineAD] [ad]</code>	
<code> [engineAG] [ag] [engineAM] [am]</code>	
<code> [engineQ3] [q3] [engineR] [r]</code>	
<code>) +</code>	
<code>-auto N</code>	Use the specified number of automatically selected engines. Use this command in conjunction with “set_proofgrid_per_engine_max_jobs” on page 1141 or “set_proofgrid_per_engine_max_local_jobs” on page 1144 to increase proof power. Use “get_engine_mode” on page 1009 to see which engines are currently specified.
<code>default</code>	Use the default engine mode { Ht Hp N B }.
<code>engineB b</code>	Use engine B, which only searches for traces and will generally not find proofs. Note: <ul style="list-style-type: none">Engine B cannot prove that a trace does not exist.Engine B is included in the default engine mode.

JasperGold Apps Command Reference Manual

Configuration Commands

engineBm bm	<p>Use engine Bm, which finds counterexamples for multiple safety properties in the same task concurrently.</p> <p>Note: When you ask engine Bm to prove a mix of safety and liveness assertions, the tool first proves all safety properties concurrently. Then, it proves the liveness properties one-by-one. (Other engines conduct a sequential proof of each property, both safety and liveness.)</p>
engineC c	<p>Use engine C, which can work well with a combination of C, C2, I, and N when the analysis region becomes too complex for convergence with the default engine mode.</p> <p>Note: While other engines use all the logic in the target's COI, engine C incrementally includes logic and generally converges with a subset of the analysis region.</p> <p>See also: set_engineC_optimization.</p>
engineC2 c2	<p>Use engine C2, which can work well with a combination of C, C2, I, and N when the analysis region becomes too complex for convergence with the default engine mode.</p> <p>Note: While other engines use all the logic in the target's COI, this engine incrementally includes logic and generally converges with a subset of the analysis region.</p> <p>See also: set_engineC_optimization.</p>
engineD d	<p>Use engine D, which proves or finds counterexamples for one property at a time (sequential proof of all properties in each task).</p> <p>See also: set_engineD_optimization.</p>
engineG g	<p>Use engine G, which proves or finds counterexamples for one property at a time.</p> <p>Note:</p> <ul style="list-style-type: none">■ Engines G and G2 are often better suited for verifying complex sequential properties such as data path credit or token management units.■ Engines G and G2 might encounter state-space explosion when used in combination with <code>set_dst_mode off</code> because these engine modes use all the logic from the COI for the proof. <p>See also: set_engineG_optimization.</p>

JasperGold Apps Command Reference Manual

Configuration Commands

engineG2 g2	<p>Use engine G2, which proves or finds counterexamples for one property at a time.</p> <p>Note:</p> <ul style="list-style-type: none">■ Engines G and G2 are often better suited for verifying complex sequential properties such as data path credit or token management units.■ Engines G and G2 might encounter state-space explosion when used in combination with <code>set_dst_mode off</code> because these engine modes use all the logic from the COI for the proof. <p>See also: <code>set_engineG_optimization</code>.</p>
engineH h	<p>Use engine H, which proves or finds counterexamples for multiple safety properties in the same task concurrently.</p> <p>Note:</p> <ul style="list-style-type: none">■ By default, the tool uses parallel-engine processing to improve run times: <code>set_engine_mode {hp ht n b}</code>.■ When you ask engine H to prove a mix of safety and liveness assertions, the tool first proves all safety properties concurrently. Then, it proves the liveness properties one-by-one. (Other engine modes conduct a sequential proof of each property, both safety and liveness.)■ You cannot combine this mode with engine Hp or Ht.
engineHp hp	<p>Use engine Hp, which is a variant of engine H that is focused on finding proofs for multiple safety properties in the same task concurrently.</p> <p>Note:</p> <ul style="list-style-type: none">■ Although engine Hp focuses on finding proofs, it is also capable of finding counterexamples.■ In combination with engine Ht, the two engines collaborate in finding both proofs and counterexamples.■ The tool does not support engine modes that combine engine Hp with engine H.

JasperGold Apps Command Reference Manual

Configuration Commands

engineHps hps	<p>Use engine Hps, which is a variant of engine H that is focused on finding proofs.</p> <p>Note: This engine works on one property at a time (sequential proof of all properties in each task).</p>
engineHt ht	<p>Use engine Ht, which is a variant of engine H that is focused on finding counterexamples for multiple safety properties in the same task concurrently.</p> <p>Note:</p> <ul style="list-style-type: none">■ In combination with Hp, the two engines collaborate in finding both proofs and counterexamples.■ The tool does not support engine modes that combine engine Ht with engine H.
engineHts hts	<p>Use engine Hts, which is a variant of engine H that is focused on finding counterexamples.</p> <p>Note: This engine works on one property at a time (sequential proof of all properties in each task).</p>
engineI i	<p>Use engine I, which works well with a combination of C, C2, I, and N when the analysis region becomes too complex for convergence with the default engines.</p> <p>Note: While other engines use all the logic in the target's COI, this engine incrementally includes logic and generally converges with a subset of the analysis region.</p> <p>See also: <code>set_engineD_optimization</code>.</p>

JasperGold Apps Command Reference Manual

Configuration Commands

`engineJ | j` Use engine J, which only searches for trace and will generally not find proofs.

Note:

- Engine J cannot prove that a trace does not exist.
- Engine J searches for traces for multiple safety properties in the same task concurrently, just like engine H.
- Use engine J in combination with `set_proofgrid_per_engine_max_jobs` or `set_proofgrid_per_engine_max_local_jobs` to spawn multiple instances of engine J for potentially better performance.
- Unlike other engines, when engine J's search for traces is successful, it will not necessarily generate the shortest possible traces.

See also: `set_engineJ_attempts`, `set_engineJ_restarts`, `set_engineJ_max_trace_length`, `set_engineJ_seed`

`engineK | k` Use engine K to find a bounded proof.

Note:

- Engine K only searches for traces.
- It will generally not find proofs.
- It can only prove that no trace shorter than or equal to the bound exists.
- It proceeds one property at a time.
- It exchanges information with engines C, I, and N during proofs of the same property to speed up the process.
- Engine K iteratively includes logic from the COI, minimizing the amount of logic used while searching.

JasperGold Apps Command Reference Manual

Configuration Commands

engineL l	<p>Use engine L to leverage reached cover properties to try to reach otherwise hard-to-reach expressions.</p> <p>Note:</p> <ul style="list-style-type: none">■ Engine L ignores liveness properties.■ As with engines J, M, and N, if engine L finds a trace, it will not necessarily be the shortest possible trace.■ An example application of engine L is to find a deep error state, such as those that occur in post-silicon debugging (PSD) exercises.■ You can also use engine L to increase the coverage percentage of a task with multiple targets. In this case, the tool can use multiple engine L instances (and licenses) to increase coverage.■ Use engine L in combination with <code>set_proofgrid_per_engine_max_jobs</code> or <code>set_proofgrid_per_engine_max_local_jobs</code> to spawn multiple instances of engine L.
engineM m	<p>Use engine M to find a full proof.</p> <p>Note:</p> <ul style="list-style-type: none">■ This engine works best on properties with a small COI and without many complex (non-pin) constraints.■ With this engine, the emphasis is on proving properties that are valid (assertions) or unreachable (covers). It can find traces; however, it is not well suited for that purpose.■ Engine M proves properties sequentially.■ It works well on liveness properties.■ As with engines J, L, and N, if engine M finds a trace, it will not necessarily be the shortest possible trace.

JasperGold Apps Command Reference Manual

Configuration Commands

engineN n	<p>Use engine N to find a full proof.</p> <p>Note:</p> <ul style="list-style-type: none">■ As with engine M, this engine works best on properties with a small COI and without many complex (non-pin) constraints; however, engine N is not as limited as engine M in this respect.■ As with engine M, the emphasis is on proving properties that are valid (assertions) or unreachable (covers). It can find traces; however, it is not well suited for that purpose.■ Engine N proves properties sequentially.■ It works well on liveness properties.■ As with engines J, L, and M, if engine N finds a trace, it will not necessarily be the shortest possible trace.■ Engine N is part of the CIK group of engines and will take part in the CIK cooperation.■ Engine N is included in the default engine mode.
engineAB ab engineAD ad engineAG ag engineAM am	<p>Use the specified abstraction engine to realize potentially better results than an engine it is based on.</p> <ul style="list-style-type: none">■ AB is a bounded-proof abstraction engine. Use this engine mode for cases where you might normally use engine B.■ AD is a proof-search abstraction engine. Use this engine mode for cases where you might normally use engine D or I.■ AG is a proof-search, data path abstraction engine. Use this engine mode for cases where you might normally use engine G or C.■ AM is a proof-search abstraction engine. Use this engine mode for cases where you might normally use engine M or N.

JasperGold Apps Command Reference Manual

Configuration Commands

engineQ3 | q3 Use engine Q3, which only searches for traces, and will generally not find proofs.

Note:

- Engine Q3 cannot prove that a trace does not exist.
- Engine Q3 searches for traces for multiple safety properties in the same task concurrently, just like engine J.
- Use engine Q3 with `random_diversification` on in combination with `set_proofgrid_per_engine_max_jobs` or `set_proofgrid_per_engine_max_local_jobs` to spawn multiple instances of engine Q3 for potentially better performance.
- Unlike some other engines, when engine Q3's search for traces is successful, it will not necessarily generate the shortest possible traces.
- See also "["set_engineQ3_max_trace_length"](#) on page 1076, "["set_engineQ3_random_diversification"](#) on page 1078, "["set_engineQ3_restarts"](#) on page 1082, and "["set_engineQ3_seed"](#) on page 1084.

engineR | r Use engine R, which is focused on finding proofs for all safety properties concurrently.

Note:

- Engine R is better at finding proofs (like Hp) than traces.
- It provides trace attempts, proof attempts, and `min_length` updates.
- Engine R uses a proof strategy similar to engine M. It limits resources with a timeout and automatically restarts with longer timeouts.
- This engine can find non-minimal traces.

JasperGold Apps Command Reference Manual

Configuration Commands

engineTri tri	Use engine Tri, which uses a proof strategy similar to the IFV Trident engine.
-----------------	--

Note:

- Engine Tri uses several processes at once, default 8, to process one property. (If you use ProofGrid mode shell, you will likely need to update your script to deal with this behavior.)
 - It provides only trace attempts and min_length updates.
 - It can find non-minimal traces.
-

get_engine_mode	
-----------------	--

No arguments	
--------------	--

Default

By default, the engine mode is parallel-engine processing with one instance each of engines Hp, Ht, N, and B.

Return

get_engine_mode returns the value of the variable engine_mode.

Examples

```
% set_engine_mode b
% prove expr1

% set_engine_mode {d g h}
% prove -task my_task

// In the following scenario, the tool launches three engine L instances, and
// in addition to the session license, the tool checks out two additional
// licenses. First, engine L picks start=0. Second, engine L picks start=1.
// And third, engine L picks start=3. (The default state start is 0, 1, 3,
// 5, 7, ....)
% set_engine_mode {L}
% set_proofgrid_per_engine_max_jobs 3
% prove -task ...
% get_engine_mode
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[check_assumptions](#)

[prove](#)

[visualize](#)

engine_solver

Commands

`set_engine_solver`
`get_engine_solver`

Description

Use `set_engine_solver` to specify the solver you want the engines to use in the decision procedure for the Boolean satisfiability problem.

Note: This command affects engines B, C, C2, D, H, Hp, Ht, I, K, L, M, N, AB, AD, and AM.

Use `get_engine_solver` to report the currently specified solver.

Syntax

<code>set_engine_solver (solverE solverF)</code>
<code>solverE</code> Sets the standard solver. <i>This is the default.</i>
<code>solverF</code> Sets <code>solverF</code> , which can be helpful in some cases, particularly when using the SEC Verification App.
<code>get_engine_solver</code>
No arguments

Default

The default solver is `solverE`.

Return

`get_engine_solver` returns the value of the variable `engine_solver`.

Examples

```
% set_engine_solver solverF  
% get_engine_solver
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[set_engine_mode](#)

engine_threads

Commands

set_engine_threads
get_engine_threads

Description

Use `set_engine_threads` to place an upper limit on the number of threads used by each engine. Using more than one engine thread may be especially beneficial when working with engines B, Ht, or L on properties that otherwise do not converge; that is, multiple threads can be especially useful with these engines when you are in a bug-hunting mode.

Use `get_engine_threads` to reports the current upper limit on the number of threads used by each engine.

Syntax

```
set_engine_threads <N>
```

`N` Use at most `N` threads per engine, where `N` is an integer number between 1 and 8. *The default is 1.*

Note: Setting the `engine_threads` limit to an integer larger than 1 may improve proof convergence, but it may also cause some engines to consume more memory than when the limit is set to 1.

```
get_engine_threads
```

No arguments

Default

The default threads per engine is 1.

Return

`get_engine_threads` returns the value of the variable `engine_threads`.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_engine_threads 5  
% get_engine_threads
```

See Also

[engine_mode](#)

engineB_before_first_mode

Commands

set_engineB_before_first_mode
get_engineB_before_first_mode

Description

Use the `set_engineB_before_first_mode` command to specify how engine B will treat any cycles before its first trace attempt.

Use the `get_engineB_before_first_mode` command to report how engine B will treat any cycles before its first trace attempt.

Syntax

`set_engineB_before_first_mode (skip | check | assume)`

<code>skip</code>	Do not make any assumptions or checks about the existence of traces shorter than <code>engineB_first_trace_attempt</code> in any way. <i>This option is the default.</i>
<code>check</code>	<p>Embed the check for traces shorter than <code>engineB_first_trace_attempt</code> into the first trace attempt.</p> <p>This means that if there exist traces shorter than <code>engineB_first_trace_attempt</code>, the first trace attempt will find a trace, but it is not necessarily the shortest one possible.</p>
<code>assume</code>	<p>Allow engine B to assume that there is no trace to be found before <code>engineB_first_trace_attempt</code>.</p> <p>If this assumption holds, this mode is to be preferred over <code>skip</code> or <code>check</code> because it gives the engine more information. However, if the assumption does not hold, the trace attempts may become overconstrained and may fail to find longer traces even though they exist.</p>

`get_engineB_before_first_mode`

No arguments

Default

By default, engine B does not make any assumptions or checks about the existence of traces shorter than `engineB_first_trace_attempt` (`skip`).

Return

`get_engineB_before_first_mode` returns the value of the variable `before_first_mode`: `skip`, `check`, or `assume`.

Examples

```
% set_engineB_before_first_mode assume  
% get_engineB_before_first_mode
```

See Also

[set_engineB_before_increment_mode](#)
[set_engineB_first_trace_attempt](#)
[set_engineB_trace_attempt_increment](#)
[set_first_trace_attempt](#)

engineB_before_increment_mode

Commands

`set_engineB_before_increment_mode`
`get_engineB_before_increment_mode`

Description

Use the `set_engineB_before_increment_mode` command to specify how engine B will treat any cycles between consecutive trace attempts.

Use the `get_engineB_before_increment_mode` command to report how engine B will treat any cycles between consecutive trace attempts.

Syntax

`set_engineB_before_increment_mode (skip | check | assume)`

<code>skip</code>	Do not make any assumptions or checks about the existence of traces with lengths in between the performed trace attempts. <i>This option is the default.</i>
<code>check</code>	Embed the check for traces shorter than a particular trace attempt “K” but longer than the previous trace attempt into the trace attempt “K.” This means that if such a trace exists, trace attempt “K” will find a trace, but it is not necessarily the shortest one possible.
<code>assume</code>	Allow engine B to assume that there is no trace to be found between the performed trace attempts. If this assumption holds, this mode is to be preferred over <code>skip</code> or <code>check</code> because it gives the engine more information. However, if the assumption does not hold, the trace attempts may become overconstrained and may fail to find longer traces even though they exist.

`get_engineB_before_increment_mode`

No arguments

Default

By default, engine B does not make any assumptions or checks about the existence of traces with lengths in between the performed trace attempts.

Return

`get_engineB_before_increment_mode` returns the value of the variable `before_increment_mode`: `skip`, `check`, or `assume`.

Examples

```
% set_engineB_before_increment_mode assume  
% get_engineB_before_increment_mode
```

See Also

[set_engineB_before_first_mode](#)
[set_engineB_first_trace_attempt](#)
[set_engineB_trace_attempt_increment](#)
[set_first_trace_attempt](#)

engineB_first_trace_attempt

Commands

`set_engineB_first_trace_attempt`
`get_engineB_first_trace_attempt`

Description

By default, engine B (like all engines) searches for traces starting with traces of length 1 (“trace attempt 1”). Sometimes, you can achieve better results by skipping a few early attempts, and instead, start at a higher trace attempt. Use the command `set_engineB_first_trace_attempt` to directly specify the first trace attempt engine B will make.

The following scenarios are examples of times when a higher trace attempt might be warranted:

- When slightly longer traces can be found much faster. (Sometimes trace attempt 1 may even be intractable, while at the same time, trace attempt 2 finds a trace immediately.)
- When you want to see a trace that is not too short.
- When prior proofs have established that there are no traces to be found shorter than a certain bound—the bounded proof radius—then, it is safe to start at a higher attempt in the sense that a minimal trace will still be found.

Note:

- Increasing the first trace attempt *can* shave off some proof time, but it can also have the opposite effect. Perhaps counterintuitively, it may be faster to make all trace attempts from 1 to 50 than it is to make all trace attempts from 20 to 50.
- When skipping the first few trace attempts, it is no longer guaranteed that the shortest possible trace is found.

Use the `get_engineB_first_trace_attempt` command to report the cycle at which engine B will make its first trace attempt.

Syntax

```
set_engineB_first_trace_attempt <N>
```

N Skip the first trace attempts and start with the specified trace attempt.

```
get_engineB_first_trace_attempt
```

No arguments

Default

By default, engine B searches for traces starting with traces of length 1 (“trace attempt 1”).

Return

`get_engineB_first_trace_attempt` returns the value of the variable `first_trace_attempt`, which is a natural number.

Examples

```
% set_engineB_first_trace_attempt 9  
% get_engineB_first_trace_attempt
```

See Also

[set_engineB_before_first_mode](#)
[set_engineB_before_increment_mode](#)
[set_engineB_trace_attempt_increment](#)
[set_first_trace_attempt](#)

engineB_single_property

Commands

set_engineB_single_property
get_engineB_single_property

Description

Use `set_engineB_single_property` to control whether engine B should prove all properties within a task in parallel (false) or a single property at a time (true).

Use the `get_engineB_single_property` command to determine whether engine B is set to prove all properties or a single property.

Syntax

set_engineB_single_property (true false)
true Prove a single property at a time. <i>This option is the default.</i>
false Prove all properties in parallel.
get_engineB_single_property
No arguments

Default

By default, engine B proves a single property at a time.

Return

`get_engineB_single_property` returns the value of the variable `engineB_single_property`.

Examples

```
% set_engineB_single_property false
% get_engineB_single_property
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[set_engine_mode](#)

engineB_trace_attempt_increment

Commands

`set_engineB_trace_attempt_increment`
`get_engineB_trace_attempt_increment`

Description

By default, engine B (like all engines) searches for traces at every consecutive length (trace attempt). Sometimes, you can achieve better results by incrementing the trace length more than 1 between every trace attempt. Use the `set_engineB_trace_attempt_increment` command to set a trace increment greater than 1.

The following scenarios are examples of times when a higher trace increment might be warranted:

- When the design is partly clocked by a regular slow clock and can only trigger the property every k'th cycle.
- When, for hard to understand reasons, the overall progress will be faster using a higher increment.

Note:

- As with `engineB_first_trace_attempt`, there is an element of speculation here with respect to which setting will be fastest. The best way to find out is through observation.
- When skipping the first few trace attempts, it is no longer guaranteed that engine B will find the shortest possible trace.

Use the `get_engineB_trace_attempt_increment` command to report the increment engine B uses between one trace attempt and the next.

Syntax

`set_engineB_trace_attempt_increment <N>`

<code>N</code>	Increment the trace length by the specified length between every trace attempt.
----------------	---

JasperGold Apps Command Reference Manual

Configuration Commands

get_engineB_trace_attempt_increment

No arguments

Default

By default, engine B searches for traces at every consecutive length (1).

Return

get_engineB_trace_attempt_increment returns the value of the variable trace_attempt_increment, which is a natural number.

Examples

```
% set_engineB_trace_attempt_increment 9  
% get_engineB_trace_attempt_increment
```

See Also

[set_engineB_before_first_mode](#)
[set_engineB_before_increment_mode](#)
[set_engineB_first_trace_attempt](#)
[set_first_trace_attempt](#)

engineB_trace_attempt_time_limit

Commands

set_engineB_trace_attempt_time_limit
get_engineB_trace_attempt_time_limit

Description

Use `set_engineB_trace_attempt_time_limit` to specify a trace attempt time limit for engine B.

Use the `get_engineB_trace_attempt_time_limit` command to get the limit on the amount of time engine B spends on each trace attempt.

Syntax

```
set_engineB_trace_attempt_time_limit <time_limit>
```

time_limit Use the specified time limit for each engine B trace attempt.

The tool supports three formats for the time limit:

- ISO8601: hh' : ' [mm[':' [ss['.' [ffff]]]]]
Examples: 2:30, 2:, 0:0:23, 2:30:05.235
- JasperGold: [dd'd'][hh'h'][mm'm'][ss[.ffff]'s']
Examples: 2h30m, 2h, 23s, 2h30m05.235s
- Seconds: sssss[.ffff]['s']
Examples: 9000s, 7200, 23s, 9005.235s

Note:

- If you do not specify a time unit, the default unit is seconds.
- To remove the limit, use 0s.

```
get_engineB_trace_attempt_time_limit
```

No arguments

Default

By default, engine B does not limit the amount of time spent on each trace attempt (0).

Return

`get_engineB_trace_attempt_time_limit` returns the value of the variable `engineB_trace_attempt_time_limit`.

Examples

```
% set_engineB_trace_attempt_time_limit 20s  
% get_engineB_trace_attempt_time_limit
```

See Also

[get_engineB_trace_attempt_time_limit_incr](#)
[set_engineB_trace_attempt_time_limit_incr](#)

engineB_trace_attempt_time_limit_incr

Commands

set_engineB_trace_attempt_time_limit_incr
get_engineB_trace_attempt_time_limit_incr

Description

Use `set_engineB_trace_attempt_time_limit_incr` to specify the time limit increment for engine B when the specified limit for trace attempts expires.

Use the `get_engineB_trace_attempt_time_limit_incr` command to get the trace attempt time limit increment for engine B.

Syntax

```
set_engineB_trace_attempt_time_limit_incr <N>
```

N Use the specified time limit increment when the time limit expires.

The tool supports three formats for the time limit:

- ISO8601: hh' : ' [mm[':' [ss['.' [ffff]]]]]
Examples: 2:30, 2:, 0:0:23, 2:30:05.235
- JasperGold: [dd'd'] [hh'h'] [mm'm'] [ss[.ffff]'s']
Examples: 2h30m, 2h, 23s, 2h30m05.235s
- Seconds: sssss[.ffff] ['s']
Examples: 9000s, 7200, 23s, 9005.235s

Note: If you do not specify a time unit, the default unit is seconds.

```
get_engineB_trace_attempt_time_limit_incr
```

No arguments

Default

By default, engine B does not limit the amount of time spent on each trace attempt (0).

Return

`get_engineB_trace_attempt_time_limit_incr` returns the value of the variable `engineB_trace_attempt_time_limit_incr`.

Examples

```
% set_engineB_trace_attempt_time_limit_incr 20s  
% get_engineB_trace_attempt_time_limit_incr
```

See Also

[get_engineB_trace_attempt_time_limit](#)
[set_engineB_trace_attempt_time_limit](#)

engineC_optimization

Commands

set_engineC_optimization
get_engineC_optimization

Description

Use the `set_engineC_optimization` command to specify the optimization mode for engine C.

Use the `get_engineC_optimization` command to display the optimization mode for engine C.

Syntax

set_engineC_optimization (static dynamic adaptive)
static Use the standard engine C ordering. <i>This option is the default.</i>
dynamic Use engine C2, which has a different engine ordering scheme.
adaptive Predict the best engine (engine C or C2) for the current design and use it.
get_engineC_optimization
No arguments

Default

By default, engine C optimization is static.

Return

`get_engineC_optimization` returns the current mode, static, dynamic, or adaptive.

Examples

```
% set_engineC_optimization dynamic
% get_engineC_optimization
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[set_engine_mode](#)

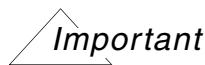
engineCG_max_mem

Commands

set_engineCG_max_mem
get_engineCG_max_mem

Description

Use the `set_engineCG_max_mem` command to set a memory limit (in megabytes) for engine C and G data structures.



With regard to `set_engineCG_max_mem`...

- In most cases, it is best to set the memory limit to a value greater than 100.
- Keep in mind that total memory consumption for a job will be greater than `engineCG_max_mem`. Consider setting a limit lower than the total amount of memory you want to use for a job, for example, 20-30 less.

Use the `get_engineCG_max_mem` command to report the current memory limit for engine C and G data structures.

Syntax

<code>set_engineCG_max_mem <memory_limit></code>	
<code>memory_limit</code>	Set the specified memory limit for engine C and G data structures.
<code>get_engineCG_max_mem</code>	
No arguments	

Default

The default `engineCG_max_mem` limit is 4096MB.

Return

`get_engineCG_max_mem` returns the current limit in megabytes.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_engineCG_max_mem 500  
% get_engineCG_max_mem
```

engineD_optimization

Commands

`set_engineD_optimization`
`get_engineD_optimization`

Description

Use the `set_engineD_optimization` command to specify the optimization mode for engines D and I.

Use the `get_engineD_optimization` command to display the optimization mode for engines D and I.

Syntax

`set_engineD_optimization (standard | high)`

`standard` Do not optimize engines D and I. *This option is the default.*

`high` For every proof or Visualize attempt made by engines D and I, simplify the task of verification to benefit successive attempts.

`get_engineD_optimization`

No arguments

Default

By default, the tool does not optimize engines D and I.

Return

`get_engineD_optimization` returns the current mode, `standard` or `high`.

Examples

```
% set_engineD_optimization high  
% get_engineD_optimization
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[set_engine_mode](#)

engineG_optimization

Commands

set_engineG_optimization
get_engineG_optimization

Description

Use the `set_engineG_optimization` command to specify the optimization mode for engine G.

Use the `get_engineG_optimization` command to display the optimization mode for engine G.

Syntax

set_engineG_optimization (static dynamic adaptive)
static Use the standard engine G ordering. <i>This option is the default.</i>
dynamic Use engine G2, which has a different engine ordering scheme.
adaptive Predict the best engine (engine G or G2) for the current design and use it.
get_engineG_optimization
No arguments

Default

By default, engine G optimization is static.

Return

`get_engineG_optimization` returns the current mode, static, dynamic, or adaptive.

Examples

```
% set_engineG_optimization dynamic
% get_engineG_optimization
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[set_engine_mode](#)

engineH_single_property

Commands

`set_engineH_single_property`
`get_engineH_single_property`

Description

Use the `set_engineH_single_property` command to control whether engine H should prove all properties within a task in parallel (`false`) or a single property at a time (`true`). Use this command if engine H does not show good performance proving tasks with multiple properties. It might improve performance.

Use the `get_engineH_single_property` command to display the `single_property` setting (`true` or `false`) for engine H.

Syntax

<code>set_engineH_single_property (true false)</code>
<code>true</code> Prove a single property at a time.
<code>false</code> Prove all properties within a task in parallel. <i>This option is the default.</i>
<code>get_engineH_single_property</code>
No arguments

Default

By default, engine H runs all properties in parallel (`false`).

Return

`get_engineH_single_property` returns `true` or `false`.

Examples

```
% set_engineH_single_property true  
% get_engineH_single_property
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[set_engine_mode](#)

engineJ_attempts

Commands

`set_engineJ_attempts`
`get_engineJ_attempts`

Description

Use the `set_engineJ_attempts` command to define the maximum number of attempts engine J will make at every cycle to satisfy all assumptions, Visualize configurations, and latch equations. If engine J fails to satisfy all assumptions within this bound, it restarts.

Use the `get_engineJ_attempts` command to return an integer representing the maximum number of attempts engine J will make at every cycle to solve all assumptions, Visualize configurations, and latch equations before restarting.

Syntax

```
set_engineJ_attempts <N>
```

N

Make the specified number of attempts at every cycle to satisfy all constraints.
Use this command with `set_engineJ_restarts` to progress farther when you have nontrivial assumptions, configurations, or latches and you get the following message:

WARNING (WAS019) :
Assumptions too hard to satisfy, gave up.

However, this command will not help if the assumptions, Visualize configurations, or latches are too complex. The only choice then is to switch to another engine mode.

```
get_engineJ_attempts
```

No arguments

Default

By default, engine J makes 1000 attempts at every cycle.

Return

get_engineJ_attempts returns the maximum number of engine J attempts to solve all constraints at every clock cycle before restarting.

Examples

```
% prove -engine J -all
WARNING (WAS019): Assumptions too hard to satisfy, gave up.
% set_engineJ_attempts 200
% set_engineJ_restarts 200
% prove -engine J -all
% get_engineJ_attempts
```

See Also

[set_engine_mode](#)
[set_engineJ_restarts](#)

engineJ_max_trace_length

Commands

`set_engineJ_max_trace_length`
`get_engineJ_max_trace_length`

Description

By default, the maximum trace length for engine J proofs and visualizations is 200 cycles. Use the `set_engineJ_max_trace_length` command to change the default.



- Engine J also honors the `max_trace_length` as an upper limit on the length of the traces it searches for. The effective `max_length` for engine J is thus the minimum (`max_trace_length` or `engineJ_max_trace_length`).
- Engine J does not make an exhaustive search for traces at every cycle and may generate very long traces. Or it might continue to search deeper and deeper even though there are shorter traces to be found. Use this command with `set_engineJ_restarts` to ensure that engine J spends the time searching for traces with lengths you want to deal with. Keep in mind that shorter traces are often easier to interpret.
- Use 0 to remove the limit.

Use the `get_engineJ_max_trace_length` command to report the maximum length setting for engine J traces.

Syntax

```
set_engineJ_max_trace_length <N>
```

`N` Limit engine J traces to the specified number of cycles.

To remove the limit, use 0. *The default value is 200.*

```
get_engineJ_max_trace_length
```

No arguments

Default

By default, engine J traces are limited to 200 cycles.

Return

`get_engineJ_max_trace_length` returns the upper limit on the length of traces searched for by engine J.

Examples

```
% prove -engine engineJ -all  
Max trace length reached.  
  
% get_engineJ_max_trace_length  
% set_engineJ_max_trace_length 5000  
  
% get_engineJ_seed  
% set_engineJ_seed 52243c71906b2166  
  
% prove -engine J -all
```

See Also

[set_engine_mode](#)
[get_engineJ_restarts](#)
[set_max_trace_length](#)

engineJ_migrate

Commands

set_engineJ_migrate
get_engineJ_migrate

Description

Use the `set_engineJ_migrate` command to instruct the tool to change engine jobs from engine J to Q3 upon reaching the `engineJ_attempts` limit instead of exiting. Engine J tends to fail due to its weak handling of constraints, but when it works, it tends to be much faster than Q3. Q3, on the other hand, has the ability to reliably solve constraints. The migration (that is, handover) provides a convenient way to get the best of both.

Note:

- If you run engine J with `migrate=on`, do not run it in parallel with Q3. It would be wasteful.
- The reported identity of the engine job does not change when it migrates to Q3. That is, the engine job keeps its `x.J` label, and the report shows that the results are from J.

Use the `get_engineJ_migrate` command to determine whether the engine job will be migrated to Q3 once the `engineJ_attempts` limit is exceeded.

Syntax

set_engineJ_migrate (on off)
on Enable engine job handover to Q3 if the <code>engineJ_attempts</code> limit is exceeded.
off Exit when the <code>engineJ_attempts</code> limit is exceeded. <i>This is the default.</i>
get_engineJ_migrate
No arguments

Default

By default, the tool exits when the `engineJ_attempts` limit is reached (`off`).

Return

`get_engineJ_migrate` returns the setting for the `engineJ_migrate` variable.

Examples

```
% prove -engine J -all  
engineJ_attempts  
% set_engineJ_migrate on  
% prove -engine J -all  
cex
```

See Also

[set_engine_mode](#)
[set_engineJ_attempts](#)
[get_engineJ_restarts](#)

engineJ_restarts

Commands

set_engineJ_restarts
get_engineJ_restarts

Description

Use the `set_engineJ_restarts` command to define the number of restarts engine J makes before abandoning the prove or Visualize job. A restart means starting again with trace attempt 1 and upwards re-searching for short traces. Two events trigger restart:

- Failing to satisfy all constraints at an cycle within the specified number of attempts
- Reaching `max_trace_length`

At every restart, the tool issues a message in the *Proof messages* tab that reports the current seed, which is a hexadecimal setting.

Use the `get_engineJ_restarts` command to report the number of times engine J will restart from trace attempt (cycle) 1 before abandoning the prove or Visualize job. You can reuse the reported seed to start from that state in a later trace search; however, reusing the seed only works as long as the prove or Visualize setup is identical, including the number of undetermined properties.

Syntax

```
set_engineJ_restarts <N>
```

N Limit engine J restarts to the specified number of attempts.
The default value is 1000.

```
get_engineJ_restarts
```

No arguments

Default

By default, engine J will restart 1000 times.

Return

`get_engineJ_restarts` returns the number of times engine J will restart from attempt (cycle) 1.

Examples

```
% prove -engine engineJ -all  
  
J: Restart 1000, Seed b45785b0fdf4a5ce  
// No trace found. Retry with changed settings:  
  
% get_engineJ_restarts  
% set_engineJ_restarts 2000  
  
% get_engineJ_seed  
% set_engineJ_seed b45785b0fdf4a5ce  
  
% prove -engine J -all
```

See Also

[set_engine_mode](#)
[set_engineJ_attempts](#)
[set_engineJ_max_trace_length](#)
[set_engineJ_seed](#)
[set_max_trace_length](#)

engineJ_seed

Commands

set_engineJ_seed
get_engineJ_seed

Description

Use the `set_engineJ_seed` command to supply a hexadecimal setting for the seed that determines how engine J searches for traces. 0 is a special value that allows engine J to pick an indeterministic seed. Other values give deterministic behavior. If you want the tool to issue a message in the *Proof Messages* tab that reports the current seed at every restart, you must specify a prove verbosity >=7. (See “[set_prove_verbosity](#)” on page 1192 or `prove...-verbosity`.) The example below shows how you can use seed information.

Use the `get_engineJ_seed` command to report the hexadecimal setting for the seed for engine J.

Syntax

`set_engineJ_seed <hex_string>`

`hex_string` Use `hex_string` as the seed for engine J.

Note: Use 0 to request an indeterministic seed.

`get_engineJ_seed`

No arguments

Default

By default, the engine J seed is 0000000000000000e.

Return

`get_engineJ_seed` returns the current seed.

Examples

The following example, Engine J finds a hard trace after a long time and after 950 restarts. Now you want to replay the proof or visualization. By picking the seed that was used at the last restart as the initial seed, you will find the trace again without a single restart.

```
% prove -engine engineJ -all  
J: Restart 950, Seed be9f1153b0d0fa36  
  
% set_engineJ_seed be9f1153b0d0fa36  
% prove -engine engineJ -all  
  
% get_engineJ_seed
```

See Also

[set_engine_mode](#)
[set_engineJ_restarts](#)

engineJ_single_property

Commands

`set_engineJ_single_property`
`get_engineJ_single_property`

Description

Use the `set_engineJ_single_property` command to control whether engine J should prove all properties within a task in parallel (`false`) or a single property at a time (`true`). Use this command if engine H does not show good performance proving tasks with multiple properties. It might improve performance.

Use the `get_engineJ_single_property` command to display the `single_property` setting (`true` or `false`).

Syntax

<code>set_engineJ_single_property (true false)</code>	
<code>true</code>	Prove a single property at a time.
<code>false</code>	Prove all properties within a task in parallel. <i>This option is the default.</i>
<code>get_engineJ_single_property</code>	
No arguments	

Default

By default, engine J runs all properties in parallel.

Return

`get_engineJ_single_property` returns `true` or `false`.

Examples

```
% set_engineJ_single_property true  
% get_engineJ_single_property
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[set_engine_mode](#)

engineL_cache

Commands

`set_engineL_cache`
`get_engineL_cache`

Description

Use the `set_engineL_cache` command to specify whether engine L will use a cover point cache when you run prove or visualize.

When `engineL_cache` is on, the tool retrieves cached cover points from earlier engine L runs that had the same set of properties and stores all intermediate cover points found by engine L in the current run. The cache is located in the project directory.

Note: If the set of properties in the current run differs from earlier runs, the tool does not retrieve any cached information.

When the tool retrieves cached information, it prints a log message as follows:

0.L: Loaded 42 cached start states

When the tool stores information in the cache, it prints a log message as follows:

0.L: Cached 17 start states



- Use proof simplification caching (`set_cache_proof_simplification on`) to increase the impact of engine L caching.
- To clear the cache, use the command `clear -engineL_cache`.

Use the `get_engineL_cache` command to display the current `engineL_cache` setting (on or off).

Syntax

```
set_engineL_cache (on | off)
```

on Use a cover point cache when running `prove` or `visualize`.

This option is the default.

off Do not cache cover points.

```
get_engineL_cache
```

No arguments

Default

By default, the tool does not cache cover points.

Return

`get_engineL_cache` returns the current setting for `engineL_cache` (on or off).

Examples

```
% set_cache_proof_simplification on  
% set_engineL_cache on  
% get_engineL_cache  
% clear -engineL_cache
```

See Also

[clear](#)

[prove](#)

[set_engine_mode](#)

[set_cache_proof_simplification](#)

[visualize](#)

engineL_generate_trails

Commands

`set_engineL_generate_trails`
`get_engineL_generate_trails`

Description

Use the `set_engineL_generate_trails` command to enable generation of trails by engine L. A trail is the sequence of intermediate properties used to find a trace for a particular property. You can later retrieve the trail for a property by running the command `get_property_info -list engineL_trail`.

Use the `get_engineL_generate_trails` command to determine whether trails will be generated.

Syntax

<code>set_engineL_generate_trails (on off)</code>
<code>on</code> Generate trails.
<code>off</code> Do not generate trails. <i>This option is the default.</i>
<code>get_engineL_generate_trails</code>
No arguments

Default

By default, the tool does not generate trails.

Return

`get_engineL_generate_trails` returns the current setting for `engineL_generate_trails (on or off)`.

Examples

```
% set_engineL_generate_trails on
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_engineL_generate_trails
```

See Also

[get_property_info](#)

engineL_ignore_trace

Commands

`set_engineL_ignore_trace`
`get_engineL_ignore_trace`

Description

Use the `set_engineL_ignore_trace` command to specify whether engine L will keep properties that other engines (or other engine L instances) have found traces for as intermediate search targets. This command is helpful when the states in these traces are closer to the remaining targets than the current start states.

Use the `get_engineL_ignore_trace` command to display the current `engineL_ignore_trace` setting (on or off).

Syntax

`set_engineL_ignore_trace (on | off)`

`on` Keep determined targets as sources of intermediate states. *This option is the default.*

`off` Remove determined targets as sources of intermediate states.

`get_engineL_ignore_trace`

No arguments

Default

By default, the tool keeps determined targets as sources of intermediate states (on).

Return

`get_engineL_ignore_trace` returns the current setting for `engineL_ignore_trace` (on or off).

Examples

```
% set_engineL_ignore_trace on
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_engineL_ignore_trace
```

See Also

[set_engine_mode](#)

engineL_max_segment_length

Commands

`set_engineL_max_segment_length`
`get_engineL_max_segment_length`

Description

By default, engine L searches from a selection of the determined start states for new traces. Use the `set_engineL_max_segment_length` variable to control how many trace attempts the tool makes from these start states before replacing the start states. If the engine cycles through all start states without finding new traces within the limit, the tool automatically increases the limit.

Use the `get_engineL_max_segment_length` command to determine how many trace attempts the tool makes before replacing the start states.

Syntax

`set_engineL_max_segment_length <N>`

`N` Specify how many trace attempts the tool makes from current start states before replacing the start states.

`get_engineL_max_segment_length`

No arguments

Default

No limit (set to 0)

Return

`get_engineL_max_segment_length` returns the current setting for `set_engineL_max_segment_length`.

Examples

```
% set_engineL_max_segment_length 5
```

JasperGold Apps Command Reference Manual

Configuration Commands

% get_engineL_max_segment_length

See Also

No related commands

engineL_random_diversification

Commands

`set_engineL_random_diversification`
`get_engineL_random_diversification`

Description

When using several engine L instances (multiple licenses), the command `set_engineL_random_diversification on` can increase the chance of reaching deep covers by introducing more diversification in the search between the different engine L instances.

Use the `get_engineL_random_diversification` command to determine the current setting for random diversification between the searches of engine L instances.

Syntax

```
set_engineL_random_diversification (off | on)
```

on	Increase the chance of reaching deep covers by introducing more diversification in the search between the different engine L instances. <i>This option is the default.</i>
----	--

off	Do not diversify searches between multiple instances of engine L.
-----	---

	<code>get_engineL_random_diversification</code>
--	---

	No arguments
--	--------------

Default

By default, when using several engine L instances, the tool does not introduce more diversification in the search between the different engine L instances (`on`).

Return

`get_engineL_random_diversification` returns the current setting for `set_engineL_random_diversification`, `on` or `off`.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_engineL_random_diversification off  
% get_engineL_random_diversification
```

See Also

No related commands

engineL_search_mode

Commands

`set_engineL_search_mode`
`get_engineL_search_mode`

Description

Use the `set_engineL_search_mode` command to specify how engine L will target properties.

Use the `get_engineL_search_mode` command to report whether properties are targeted in order or in parallel.

Syntax

<code>set_engineL_search_mode (ordered parallel <N>)</code>	
<code>ordered</code>	Target the specified undetermined properties left to right, according to their order in the <code>prove</code> command.
<code>parallel</code>	Target the specified undetermined properties in parallel. <i>This option is the default.</i>
<code>N</code>	Target the specified number of undetermined properties in parallel at each attempt. <code>N</code> must be a positive number.
<code>get_engineL_search_mode</code>	
No arguments	

Default

By default, engine L targets specified undetermined properties in parallel.

Return

Returns `ordered`, `parallel`, or a number.

Examples

```
% get_engineL_search_mode

// With the following commands, the tool tries to find a trace to p1,
// then to p5, and finally, to p3.

% set_engine_mode L
% set_engineL_search_mode ordered
% prove -property p1 p5 p3

// With the following commands, the tool tries to find a trace to p1 or
// c1, and depending on which one of them is found first, it continues
// with p1 and c2 or c1 and c2.

% set_engine_mode L
% set_engineL_search_mode 2
% prove -property p1 c1 c2
```

See Also

[set_engine_mode](#)

engineL_state_removal

Commands

`set_engineL_state_removal`
`get_engineL_state_removal`

Description

Use `set_engineL_state_removal` to change the heuristics for start state selection in engine L. From every trace, the engine picks a start state, and this command controls the heuristics used to decide which subset of the discovered start states to use, the last found start states or start states that have been the most fruitful in the past. This setting has no effect unless you are using more than one license.

Use `get_engineL_state_removal` to report the start states engine L will use.

Syntax

`set_engineL_state_removal (fifo | lru)`

`fifo` Use the last found start states.

`lru` Use the start states that have been the most fruitful in the past. *This option is the default.*

`get_engineL_state_removal`

No arguments

Default

By default, engine L uses the last found start states (`lru`).

Return

`get_engineL_state_removal` returns the value of the variable `engineL_state_removal`, `fifo` or `lru`.

Examples

```
% set_engineL_state_removal fifo
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_engineL_state_removal
```

See Also

[set_engine_mode](#)

[set_engineL_tail_length](#)

engineL_tail_length

Commands

set_engineL_tail_length
get_engineL_tail_length

Description

By default, engine L uses one cycle before the last cycle of a trace as the start state for a new search. Use the `set_engineL_tail_length` command to specify a start state that is N number of cycles before the end of an existing trace. A higher value can help with dead-end states, where it is not possible to respect all the assumptions moving forward. This command helps make the search “wider” as the tool can use more than the concrete state from the trace. However, higher values cost more in terms of the time they take as the engine will potentially have to do N extra trace attempts.

Use `get_engineL_tail_length` to report the starting point for successive engine L trace searches.

Syntax

```
set_engineL_tail_length <N>
```

N Begin the search for a new trace at a cycle that is N cycles before the last cycle of the existing trace.

```
get_engineL_tail_length
```

No arguments

Default

By default, engine L begins the search for new traces at the end of an existing trace, that is, `tail_length = 1`.

Return

Returns a natural number.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_engineL_tail_length 5  
% get_engineL_tail_length
```

See Also

[set_engine_mode](#)
[set_engineL_state_removal](#)

engineQ3_max_trace_length

Commands

`set_engineQ3_max_trace_length`
`get_engineQ3_max_trace_length`

Description

By default, the maximum trace length (when diversification is on) for engine Q3 is 200 cycles. Use `set_engineQ3_max_trace_length` to change the default. Use 0 to impose no limit.

Note:

- When `engineQ3_random_diversification` is off, this setting has no effect.
- Engine Q3 also honors the `max_trace_length` as an upper limit on the length of traces searched for. The effective `max_length` for engine Q3 is thus the minimum (`max_trace_length` or `engineQ3_max_trace_length`).
- Engine Q3 does not make an exhaustive search for traces at every cycle and may generate very long traces. Or it might continue to search deeper and deeper even though there are shorter traces to be found. Use this command with `set_engineQ3_restarts` to ensure that engine Q3 spends the time searching for traces with lengths you want to deal with. Keep in mind that shorter traces are often easier to interpret.

Use `get_engineQ3_max_trace_length` to determine the maximum trace length for engine Q3 when diversification is on.

Syntax

`set_engineQ3_max_trace_length <N>`

`N` Limit engine Q3 traces to the specified non-negative integer cycle. To remove the limit, use 0.

`get_engineQ3_max_trace_length`

No arguments

Default

By default, the maximum trace length (when diversification is on) for engine Q3 is 200 cycles

Return

No return values

Examples

```
% set_engineQ3_random_diversification on  
% prove -engine Q3 -all  
Max trace length reached.  
  
% get_engineQ3_max_trace_length  
  
% set_engineQ3_max_trace_length 5000  
% get_engineQ3_seed  
  
% set_engineQ3_seed 52243c71  
% prove -engine Q3 -all
```

See Also

[get_engineQ3_max_trace_length](#)
[set_engine_mode](#)
[set_engineQ3_random_diversification](#)
[set_engineQ3_restarts](#)
[set_max_trace_length](#)

engineQ3_random_diversification

Commands

`set_engineQ3_random_diversification`
`get_engineQ3_random_diversification`

Description

Use `set_engineQ3_random_diversification` to enable or disable random diversification for engine Q3. When random diversification is on, engine Q3 can be further controlled using the following settings:

- `engineQ3_random_diversification_factor`
- `engineQ3_seed`
- `engineQ3_max_trace_length`
- `engineQ3_restarts`

These settings have no effect when diversification is off. When diversification is on, multiple engine Q3 instances can run in parallel.

Use `get_engineQ3_random_diversification` to report whether random diversification for engine Q3 is on or off.

Syntax

`set_engineQ3_random_diversification (on | off)`

`on` Enables random diversification for engine Q3.

`off` Disables random diversification for engine Q3. *This option is the default.*

`get_engineQ3_random_diversification`

No arguments

Default

By default, random diversification for engine Q3 is off.

Return

`get_engineQ3_random_diversification` returns the setting for `engineQ3_random_diversification`.

Examples

```
% set_engineQ3_random_diversification on
% prove -engine Q3 -per_engine_max_jobs 10 -all
2.Q3: Restart 950, Seed be9f1153

% set_engineQ3_random_diversification be9f1153
% prove -engine Q3 -all
```

See Also

[get_engineQ3_random_diversification](#)
[set_engine_mode](#)
[set_engineQ3_restarts](#)

engineQ3_random_diversification_factor

Commands

`set_engineQ3_random_diversification_factor`
`get_engineQ3_random_diversification_factor`

Description

Use `set_engineQ3_random_diversification_factor` to set the random diversification factor used by engine Q3 when random diversification is on. A high value gives more diversification, but potentially much slower progress. A very low value is probably faster, but will perhaps give too little diversification for restarts and multiple jobs to make sense. Compare the State Signatures between different jobs, and/or different restarts to see how much divergence the diversification accomplishes.

Use `get_engineQ3_random_diversification_factor` to determine the random diversification factor for engine Q3.

Syntax

`set_engineQ3_random_diversification_factor <0.0 - 1.0>`

`0.0 - 1.0` Specify the random diversification factor used by engine Q3.

`get_engineQ3_random_diversification_factor`

No arguments

Default

By default, the random diversification factor for engine Q is .5.

Return

`get_engineQ3_random_diversification_factor` returns the value for `engineQ3_random_diversification_factor`.

Examples

% `set_engineQ3_random_diversification on`

JasperGold Apps Command Reference Manual

Configuration Commands

```
% set_engineQ3_random_diversification_factor 0.001
% prove -engine Q3 -all
0.Q3: Restart 12, Seed 7, 0.1%
0.Q3: Cycle 106, State Sig: 17a49d99
1.Q3: Cycle 106, State Sig: 17a49d99
2.Q3: Cycle 106, State Sig: 17a49d99

% set_engineQ3_random_diversification_factor 0.4
% prove -engine Q3 -all
0.Q3: Restart 12, Seed 7, 40.0%
0.Q3: Cycle 106, State Sig: 17a49d99
1.Q3: Cycle 106, State Sig: 17ba23fe
2.Q3: Cycle 106, State Sig: 13235696
```

See Also

[get_engineQ3_random_diversification_factor](#)
[set_engine_mode](#)
[set_engineQ3_random_diversification](#)
[set_engineQ3_restarts](#)

engineQ3_restarts

Commands

set_engineQ3_restarts
get_engineQ3_restarts

Description

Use `set_engineQ3_restarts` to define the number of restarts engine Q3 makes when random diversification is on before abandoning the proof. A restart means starting again with trace attempt 1 and upwards re-searching for short traces. Reaching maximum trace length triggers restart.

At every restart, the tool issues a message in the Proof Messages tab that reports the current seed, which is 8 hexadecimal characters.

Use `get_engineQ3_restarts` to determine the number of restarts engine Q3 will make before abandoning the proof.

Syntax

```
set_engineQ3_restarts <N>
```

`N` Limit engine Q3 restarts to the specified non-negative integer attempts.

Note: You can reuse the reported seed to start from that state in a later trace search; however, using the seed only works as long as the proof setup is identical, including the number of undetermined properties.

```
get_engineQ3_restarts
```

No arguments

Default

By default, the limit for engine Q3 restarts is 1000.

Return

`get_engineQ3_restarts` returns the value for `engineQ3_restarts`.

Examples

```
% set_engineQ3_random_diversification on
% prove -engine Q3 -all
Q3: Restart 1000, Seed b45785b0
// No trace found. Retry with changed settings:

% get_engineQ3_restarts
% set_engineQ3_restarts 2000

% get_engineQ3_seed
% set_engineQ3_seed b45785b0

% prove -engine Q3 -all
```

See Also

[get_engineQ3_restarts](#)
[set_engine_mode](#)
[set_engineQ3_max_trace_length](#)
[set_engineQ3_random_diversification](#)
[set_engineQ3_seed](#)
[set_max_trace_length](#)

engineQ3_seed

Commands

set_engineQ3_seed
get_engineQ3_seed

Description

Use `set_engineQ3_seed` to supply a hexadecimal setting for the seed that determines how engine Q3 searches for traces when random diversification is on. 0 is a special value that allows engine Q3 to pick an indeterministic seed. Other values give deterministic behavior. If you want the tool to issue a message in the *Proof Messages* tab that reports the current seed at every restart, you must specify a `prove_verbosity >= 7`. (See “[set_prove_verbosity](#)” on page 1192 or `prove... -verbosity`.) The example shows how you can use seed information.

Use `get_engineQ3_seed` to determine the hexadecimal setting for the seed used by engine Q3.

Syntax

<code>set_engineQ3_seed <hex_string></code>
<code>hex_string</code> Specify a hexadecimal setting as the seed for engine Q3.
<code>get_engineQ3_seed</code>
No arguments

Default

By default, the hexadecimal setting for the seed used by engine Q3 is 00000007.

Return

`get_engineQ3_seed` the value for `engineQ3_seed`.

Examples

In the example below, engine Q3 finds a hard trace after a long time and after 950 restarts. Now you want to replay the proof. By picking the seed that was used at the last restart as the initial seed, you will find the trace again without a single restart.

```
% set_engineQ3_random_diversification on  
% prove -engine Q3 -all  
Q3: Restart 950, Seed be9f1153  
  
% set_engineQ3_seed be9f1153  
% prove -engine Q3 -all
```

See Also

[get_engineQ3_seed](#)
[set_engine_mode](#)
[set_engineQ3_random_diversification](#)
[set_engineQ3_restarts](#)

export_sva_bind_enable_directive

Commands

`set_export_sva_bind_enable_directive`
`get_export_sva_bind_enable_directive`

Description

Exported SVA files contain a bind statement, which binds the SVA module to the RTL design. When exporting SVA files from the property synthesis apps, you can use the `set_export_sva_bind_enable_directive` command to add a conditional compilation guard, for example, `ifdef XXX or `ifndef XXX, around the bind statement.

Use `get_export_sva_bind_enable_directive` to get the conditional compilation directive for the bind statement.

Syntax

<code>set_export_sva_bind_enable_directive directive</code>	
<code>directive</code>	Specify a conditional compilation directive for the SVA bind statement.

<code>get_export_sva_bind_enable_directive</code>	
---	--

No arguments	
--------------	--

Default

None

Return

`get_export_sva_bind_enable_directive` returns the value of the variable `export_sva_bind_enable_directive`.

Examples

```
% set_export_sva_bind_enable_directive ``ifdef SVA_ON''
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[check_bps](#)

[check_sps](#)

[export](#)

export_sva_embed_lib_modules

Commands

`set_export_sva_embed_lib_modules`
`get_export_sva_embed_lib_modules`

Description

Exported SVA files may contain JasperGold Apps library module instantiations. When exporting to SVA files from the property synthesis apps, use the command `set_export_sva_embed_lib_modules` to turn library module embedding on or off.

Use `get_export_sva_embed_lib_modules` to determine whether SVA export should embed SVA library modules.

Syntax

`set_export_sva_embed_lib_modules (true | false)`

`true` Embed library modules in exported SVA output. *This option is the default.*

`false` Do not embed library modules in exported SVA output.

`get_export_sva_embed_lib_modules`

No arguments

Default

By default, library modules are embedded in exported SVA.

Return

`get_export_sva_embed_lib_modules` returns the value of the variable `export_sva_embed_lib_modules`.

Examples

```
% set_export_sva_embed_lib_modules false
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_export_sva_embed_lib_modules
```

See Also

[check_bps](#)
[export](#)

export_sva_handshake_delay_threshold

Commands

`set_export_sva_handshake_delay_threshold`
`get_export_sva_handshake_delay_threshold`

Description

When exporting SVA from the property synthesis apps, use the `set_export_sva_handshake_delay_threshold` command to set a threshold for converting the maximum delay in the handshake property to \$.

For example, if the `export_sva_handshake_delay_threshold` variable is set to 100, then the tool converts the max delay to \$ when exporting to SVA if the difference between the min delay and max delay is more than 100. Thus, `$handshake(s1, s2, 1, 1800, 0, 0)` would create an SVA property that matches to `$handshake(s1, s2, 1, $, 0, 0)`.

Use `get_export_sva_handshake_delay_threshold` to get the threshold for converting the maximum delay to \$.

Syntax

<code>set_export_sva_handshake_delay_threshold <i>threshold</i></code>
<code><i>threshold</i></code> Specify the threshold for the delay in the handshake properties.
<code>get_export_sva_handshake_delay_threshold</code>
No arguments

Default

None

Return

`get_export_sva_handshake_delay_threshold` returns the value of the variable `export_sva_handshake_delay_threshold`.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_export_sva_handshake_delay_threshold 200
```

See Also

[scan](#)

export_sva_hierarchy_separator

Commands

`set_export_sva_hierarchy_separator`
`get_export_sva_hierarchy_separator`

Description

The Behavioral Property Synthesis App uses a hierarchy separator character when exporting SVA to create appropriate binding for exported properties. Use the `set_export_sva_hierarchy_separator` command to modify the RTL hierarchy separator character.

Use the command `get_export_sva_hierarchy_separator` to get the value of the variable `export_sva_hierarchy_separator`.

Syntax

<code>set_export_sva_hierarchy_separator <i>hierarchy_separator</i></code>	
<code><i>hierarchy_separator</i></code>	Use the specified hierarchy separator when exporting SVA.
<code>get_export_sva_hierarchy_separator</code>	
No arguments	

Default

By default, the Behavioral Property Synthesis App uses a dot (.) as a hierarchy separator when exporting SVA.

Return

`get_export_sva_hierarchy_separator` returns the value of the variable `export_sva_hierarchy_separator`.

Examples

```
% set_export_sva_hierarchy_separator /
% get_export_sva_hierarchy_separator
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

No related commands

export_sva_hierarchy_separator_escape_string

Commands

`set_export_sva_hierarchy_separator_escape_string`
`get_export_sva_hierarchy_separator_escape_string`

Description

During an SVA export, the Behavioral Property Synthesis App and the Structural Property Synthesis App replace the hierarchy separator with an alternate string (known as an “escape string”), so you can later bind exported SVA as if all signals reside in one hierarchy. Use the `set_export_sva_hierarchy_separator_escape_string` command to modify the default escape string used for the RTL hierarchy separator.

Use the command `get_export_sva_hierarchy_separator_escape_string` to get the value of the `export_sva_hierarchy_separator_escape_string` variable.

Syntax

`set_export_sva_hierarchy_separator_escape_string escape_string`

`escape_string` Use the specified escape string when exporting SVA.

`get_export_sva_hierarchy_separator_escape_string`

No arguments

Default

By default, the property synthesis apps use a double underscore (`__`) as a hierarchy separator escape string when exporting SVA.

Return

`get_export_sva_hierarchy_separator_escape_string` returns the value of the variable `export_sva_hierarchy_separator_escape_string`.

Examples

```
% set_export_sva_hierarchy_separator_escape_string __
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_export_sva_hierarchy_separator_escape_string
```

See Also

No related commands

export_sva_reset_condition

Commands

`set_export_sva_reset_condition`
`get_export_sva_reset_condition`

Description

When exporting SVA from the BPS App, use `set_export_sva_reset_condition` to disable exported SVA properties during reset. This command adds a `disable iff (sva_reset_condition)` clause to all exported SVA properties.

Use `get_export_sva_reset_condition` to return the reset condition for disabled SVA properties.

Syntax

<code>set_export_sva_reset_condition <sva_reset_condition></code>	
<code>sva_reset_condition</code>	Specify a reset condition on which to disable exported SVA properties.
<code>get_export_sva_reset_condition</code>	
No arguments	

Default

None

Return

`get_export_sva_reset_condition` returns the value of the variable `export_sva_reset_condition`

Examples

```
% set_export_sva_reset_condition ~rst_
% set_export_sva_reset_condition [waveform -reset]
% get_export_sva_reset_condition
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[export](#)

export_sva_target_environment

Commands

set_export_sva_target_environment
get_export_sva_target_environment

Description

Use `set_export_sva_target_environment` to specify the target validation environment for SVA export. With this command, the tool applies optimizations specific to one or more validation environments during SVA export.

Use `get_export_sva_target_environment` to return the target validation environment for SVA export.

Syntax

set_export_sva_target_environment	(formal simulation emulation)
formal	Optimize SVA export for formal environments. <i>This option is the default.</i>
simulation	Optimize SVA export for simulation environments.
emulation	Optimize SVA export for emulation environments.
get_export_sva_target_environment	
No arguments	

Default

By default, the tool optimizes SVA export for formal environments.

Return

`get_export_sva_target_environment` returns the value of the variable `export_sva_target_environment`

Examples

```
% set_export_sva_target_environment simulation
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

No related commands

export_sva_trace_suffix_cycles

Commands

`set_export_sva_trace_suffix_cycles`
`get_export_sva_trace_suffix_cycles`

Description

When exporting to SVA from the property synthesis apps, use `set_export_sva_trace_suffix_cycles` to debug N cycles after the event, specified as a cover property, occurred.

For example, if the `export_sva_trace_suffix_cycles` variable is set to 5, the tool generates the cover property with a suffix of the SVA delay operator as 5 additional cycles:

```
cover property (fsm == IDLE_STATE ##5 1'b1);
```

Use `get_export_sva_trace_suffix_cycles` to return the number of cycles added to the cover property.

Syntax

```
set_export_sva_trace_suffix_cycles N
```

N Specify the number of cycles that will be added to the cover property.

```
get_export_sva_trace_suffix_cycles
```

No arguments

Default

There is no default.

Return

`get_export_sva_trace_suffix_cycles` returns the value of the variable `export_sva_trace_suffix_cycles`.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_export_sva_trace_suffix_cycles 5  
% get_export_sva_trace_suffix_cycles
```

See Also

[cover](#)

export_sva_use_escaped_identifier_with_extended_names

Commands

set_export_sva_use_escaped_identifier_with_extended_names
get_export_sva_use_escaped_identifier_with_extended_names

Description

When exporting SVA from the property synthesis apps, the tool can extend property names to include additional information, such as module names. If the extended names contain hierarchy separators or other SVA tokens, such as index tokens, the tool prefixes the names with a backslash. Use the command

set_export_sva_use_escaped_identifier_with_extended_names to enable or disable the use of escaped identifiers when exporting SVA.

Use get_export_sva_use_escaped_identifier_with_extended_names to report whether the tool will use escaped identifiers when exporting SVA.

Syntax

set_export_sva_use_escaped_identifier_with_extended_names (true false)	
true	Use escaped identifiers when extending property names. <i>This option is the default.</i>
false	Do not use escaped identifiers when extending property names.
get_export_sva_use_escaped_identifier_with_extended_names	
No arguments	

Default

None

Return

get_export_sva_use_escaped_identifier_with_extended_names returns the value of export_sva_use_escaped_identifier_with_extended_names.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_export_sva_use_escaped_identifier_with_extended_names false  
% get_export_sva_use_escaped_identifier_with_extended_names
```

See Also

[export](#)

extract_high_level_fsm

Commands

set_extract_high_level_fsm
get_extract_high_level_fsm

Description

FSM detection in the BPS App and SPS App uses a higher level of heuristics than the FSM detection used in `get_design_info`. This might cause some FSM POIs to be filtered from your results on extraction, which triggers a warning message. Use the `set_extract_high_level_fsm` command to enable or disable high-level heuristics for FSM extraction.

Use `get_extract_high_level_fsm` to determine whether high-level heuristics are used for FSM extraction.

Syntax

set_extract_high_level_fsm (true false)	
true	Enable high-level heuristics for FSM extraction. <i>This option is the default.</i>
false	Disable high-level heuristics for FSM extraction.
get_extract_high_level_fsm	
No arguments	

Default

By default, the SPS App and BPS App use high-level heuristics for FSM extraction (true).

Return

`get_extract_high_level_fsm` returns the value of the variable `extract_high_level_fsm`.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_extract_high_level_fsm false  
% get_extract_high_level_fsm
```

See Also

[check_sps](#)
[check_sps_configure](#)
[scan](#)
[scope](#)

first_trace_attempt

Commands

`set_first_trace_attempt`
`get_first_trace_attempt`

Description

By default, engines search for traces starting with traces of length 1 (“trace attempt 1”). Sometimes, you can achieve better results by skipping a few early attempts, and instead, starting at a higher trace attempt. Use the `set_first_trace_attempt` command to directly specify the first trace attempt engines will make.

The following scenarios are examples of times when a higher trace attempt might be warranted:

- When slightly longer traces can be found much faster. (Sometimes trace attempt 1 may even be intractable, while at the same time, trace attempt 2 finds a trace immediately.)
- When you want to see a trace that is not too short.
- When prior proofs have established that there are no traces to be found shorter than a certain bound—the bounded proof radius—then, it is safe to start at a higher attempt in the sense that a minimal trace will still be found.

Note: Increasing the first trace attempt *can* shave off some proof time, but it can also have the opposite effect. Perhaps counterintuitively, it may be faster to make all trace attempts from 1 to 50 than it is to make all trace attempts from 20 to 50.

- When engines focused on concluding that no trace is to be found might be led away from unfruitful analysis using this command.

Note:

- When skipping the first few trace attempts, it is no longer guaranteed that the engine will find the shortest possible trace.
- Some engines do not have the capability to start at something other than trace attempt 1. Such engines will ignore this setting.
- This command is effective with engines L, B, D, K, I, N, C and with the abstraction engines AB, AD, AM, and AG. However, for engines B and L, it is possible to receive a trace shorter than the specified length.

JasperGold Apps Command Reference Manual

Configuration Commands

Use the `get_first_trace_attempt` command to report the cycle at which engines will make their first trace attempt.

Syntax

<code>set_first_trace_attempt N</code>	
<code>N</code>	Skip the first trace attempts and start with the specified trace attempt.
<code>get_first_trace_attempt</code>	
No arguments	

Default

By default, engines search for traces starting with traces of length 1 (“trace attempt 1”).

Return

`get_first_trace_attempt` returns the value of the variable `first_trace_attempt`, which is a natural number.

Examples

```
% set_first_trace_attempt 9  
% get_first_trace_attempt
```

See Also

[set_max_trace_length](#)

function_debug_visibility

Commands

set_function_debug_visibility
get_function_debug_visibility

Description

Use the `set_function_debug_visibility` command to enable debugging for functions. With this command, you have access to any signal inside a function. That is, the signals are visible and you can operate on them with debugging features. For example, if you perform a *Why*, the source pane displays the code related to the function with the function signals highlighted. When this feature is off, the tool highlights entire functions instead of signals within functions.

Use `get_function_debug_visibility` to determine whether debugging tools such as *Why* will have visibility into functions.

Syntax

set_function_debug_visibility (on | off)

on Enable debugging for functions. *This option is the default.*

off Disable debugging for functions.

get_function_debug_visibility

No arguments

Default

By default, debugging for functions is on.

Return

`get_function_debug_visibility` returns the value of the variable `function_debug_visibility`.

Examples

```
% set_function_debug_visibility off  
% get_function_debug_visibility
```

See Also

[set_cell_define_debug_visibility](#)

log_timestamp_format

Commands

set_log_timestamp_format
get_log_timestamp_format

Description

Use `set_log_timestamp_format` to specify the format for the timestamps displayed in logging messages. The format accepts a string containing a combination of regular characters and format specifiers. The format specifiers include those accepted by the `strftime` C function, in addition to the specifiers `%3` and `%6`, for milliseconds and microseconds, respectively.

Note:

- The milliseconds and microseconds format specifiers are mutually exclusive.
- The tool treats all characters after the milliseconds and microseconds format specifiers as plain text.

Use `get_log_timestamp_format` to get the current format string used for timestamps in logging messages.

Syntax

```
set_log_timestamp_format <format>
```

<code>format</code>	Set the format for the timestamp of logging messages.
---------------------	---

<code>get_log_timestamp_format</code>

No arguments

Default

The default format follows: "%Y-%m-%d %H:%M:%S: "

JasperGold Apps Command Reference Manual

Configuration Commands

Return

`get_log_timestamp_format` returns the value of the variable `log_timestamp_format`.

Examples

```
% set_log_timestamp_format "%Y-%m-%d %H:%M:%S.%6: "
% get_log_timestamp_format
```

See Also

[get_log_timestamp_mode](#)
[set_log_timestamp_mode](#)

log_timestamp_mode

Commands

set_log_timestamp_mode
get_log_timestamp_mode

Description

Use set_log_timestamp_mode to specify whether logging messages are prepended by a timestamp.

Note: This command does not apply to non-JasperGold frontend messages that begin with VERI-, VHDL-, or VDB-.

Use get_log_timestamp_mode to determine whether logging messages will be prepended by a timestamp.

Syntax

```
set_log_timestamp_mode (on | off)
```

on	Prepend logging messages by a timestamp.
----	--

off	Do not prepend logging messages by a timestamp. <i>This option is the default.</i>
-----	--

```
get_log_timestamp_mode
```

No arguments

Default

By default, the tool does not prepend logging messages by a timestamp (off).

Return

get_log_timestamp_mode returns the value of the variable log_timestamp_mode.

Examples

```
% set_log_timestamp_mode on
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_log_timestamp_mode
```

See Also

[get_log_timestamp_format](#)
[set_log_timestamp_format](#)

max_trace_length

Commands

set_max_trace_length
get_max_trace_length

Description

Use the `set_max_trace_length` command to specify the maximum length for traces in the JasperGold Apps Visualize™ window and the limit for the proof or visualization depth. When `max_trace_length` is set, if one engine reaches the length limit for some property, all other engines will stop working on that property.

Note: You can also use this command to specify a bounded proof, that is, a proof that terminates after reaching the specified maximum length.

Use the `get_max_trace_length` command to display the value specified with the companion set command.

Syntax

```
set_max_trace_length [-env | -task <task_name>] <N>
```

-env	Limit traces for all tasks. <i>This option is the default.</i>
------	--

-task <i>task_name</i>	Limit traces for the specified task. (By default, this command applies globally.)
---------------------------	---

Replace *task_name* with a period (.) to specify the current task.

<i>N</i>	Use the specified limit for the commands that search for traces.
----------	--

Note:

- To remove the limit, use 0.
- This option is useful for detecting hidden counters.
- Depending on engine mode, a possible side effect is that this will also put a lower limit on how hard proofs (`proven` and `unreachable` results) are searched for.

JasperGold Apps Command Reference Manual

Configuration Commands

```
get_max_trace_length [-task <task_name>]
```

-task
task_name Display the maximum trace length for the specified task.

Replace *task_name* with a period (.) to specify the current task.

Default

By default, the trace length is unlimited (set to 0).

Return

`get_max_trace_length` returns the value of the variable `max_trace_length` for the specified task.

Examples

```
% set_max_trace_length 25
% get_max_trace_length
% get_max_trace_length -task .
```

See Also

[prove](#)
[set_engineJ_max_trace_length](#)
[visualize](#)

message_history_limit

Commands

`set_message_history_limit`
`get_message_history_limit`

Description

Use the `set_message_history_limit` command to change the limit of user messages that can be kept in storage for `get_message -list`. When the limit is surpassed, the oldest issued messages are erased from memory and, therefore, will not be listed by the aforementioned command's switch.

Use the `get_message_history_limit` to return the message history limit.

Syntax

<code>set_message_history_limit <limit></code>
<code>limit</code> Set the limit of stored user messages issued in the current session.
<code>get_message_history_limit</code>
No arguments

Default

By default, the tool limits stored messages to 100.

Return

`get_message_history_limit` returns the current limit on user message storage.

Examples

```
% set_message_history_limit 1000  
% get_message_history_limit
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[get_message](#)

parallel_proof_mode

Commands

`set_parallel_proof_mode`
`get_parallel_proof_mode`

Description

Use the `set_parallel_proof_mode` command to enable parallel proof mode. With the parallel proof mode, you can run the `prove` and `visualize` commands while there is a background proof in progress.

Note: `set_parallel_proof_mode` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments.

Use the `get_parallel_proof_mode` command to determine whether `parallel_proof_mode` is enabled.

Syntax

```
set_parallel_proof_mode (on | off)
```

on Enable parallel proof mode.

off Disable parallel proof mode. *This option is the default.*

```
get_parallel_proof_mode
```

No arguments

Default

By default, `parallel_proof_mode` is off.

Return

`get_parallel_proof_mode` returns the value of the `parallel_proof_mode` variable.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_parallel_proof_mode on  
% get_parallel_proof_mode
```

See Also

[prove](#)

[visualize](#)

prompt1

Commands

`set_prompt1`
`get_prompt1`

Description

Use `set_prompt1` to define the normal Tcl prompt. Every time the tool is about to show the prompt, it evaluates the specified Tcl script and uses its result as the prompt.

Use `get_prompt1` to return the Tcl script used to define the Tcl prompt.

Syntax

<code>set_prompt1 <tcl_script></code>
<code>tcl_script</code> Set the Tcl script used to define the Tcl prompt.
<code>get_prompt1</code>
No arguments

Default

`If {[catch {task}]} { return "% " } else { return "\[[task]\] % " }.`

Return

`get_prompt1` returns the value of the variable `prompt1`.

Examples

`% get_prompt1`

See Also

[set_prompt2](#)

prompt2

Commands

`set_prompt2`
`get_prompt2`

Description

Use `set_prompt2` to define the Tcl prompt for commands that are not yet complete. Every time the tool is about to show this prompt, it evaluates the specified Tcl script and uses its result as the prompt.

Use `get_prompt2` to return the Tcl script used to define the Tcl prompt for incomplete commands.

Syntax

<code>set_prompt2 <tcl_script></code>
<code>tcl_script</code> Set the Tcl script used to define the Tcl prompt for incomplete commands.
<code>get_prompt2</code>
No arguments

Default

Return “> ”.

Return

`get_prompt2` returns the value of the variable `prompt2`.

Examples

% `get_prompt2`

See Also

[set_prompt1](#)

proof_simplification

Commands

set_proof_simplification
get_proof_simplification

Description

During proofs and visualization, the tool simplifies the task of verification before the actual verification starts. Simplification reduces the size of the netlist processed by the engines to achieve better overall performance. Use the `set_proof_simplification` command to disable (or re-enable) simplification. View the size of the proof netlist by clicking on the *Proof messages* tab of the main JasperGold Apps GUI.

Note: In some cases, disabling simplifications leads to faster operation, particularly with engines I, C, and C2. Simplification increases the likelihood that properties are solved during preprocessing (PRE). See “[get_property_info](#)” on page 651 for additional information.

Use the `get_proof_simplification` command to determine whether simplification is on or off.

Syntax

`set_proof_simplification (on | off)`

`on` Turn on proof simplification. *This option is the default.*

`off` Turn off proof simplification.

`get_proof_simplification`

No arguments

Default

By default, proof simplification is on.

Return

`get_proof_simplification` returns the value of the variable `proof_simplification`, `on` or `off`.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_proof_simplification off  
% get_proof_simplification
```

See Also

[prove](#)
[set_cache_proof_simplification](#)
[visualize](#)

proofgrid_cluster

Commands

set_proofgrid_cluster
get_proofgrid_cluster

Description

Use the `set_proofgrid_cluster` command to set the sequence of hosts on which to spawn proof engine jobs. During `prove` (when `proofgrid=cluster`) ProofGrid spawns the first job on the first host, the second job on the second host, and so on. When the list of hosts ends, ProofGrid picks hosts from the beginning of the list again. You can specify hosts with IP numbers or host names (full or local). The hostname list accepts the following as delimiters:

- (space)
- , (comma)
- : (colon)

As a simple balancing facility, for example, when some hosts have higher capacity than others, consider the following guidelines:

- You may repeat hosts in the list.
- You may precede a host with a multiplier: `M*hostname`

Use the `get_proofgrid_cluster` command to return the sequence of hosts on which proof engine jobs will be spawned.

Syntax

<code>set_proofgrid_cluster <hostname_list></code>	
<code>hostname_list</code>	Set the sequence of hosts on which to spawn proof engine jobs.
<code>get_proofgrid_cluster</code>	
No arguments	

Default

No default values

Return

`get_proofgrid_cluster` returns the current value of the `proofgrid_cluster` variable.

Examples

```
% set_proofgrid_mode cluster  
% set_proofgrid_cluster {2*a b}  
% prove -all  
# Every third job is created on machine b, the rest on machine a.
```

See Also

[set_proofgrid_mode](#)

proofgrid_engineJ_max_jobs

Commands

`set_proofgrid_engineJ_max_jobs`
`get_proofgrid_engineJ_max_jobs`

Description

Use `set_proofgrid_engineJ_max_jobs` together with `proofgrid_per_engine_max_jobs` or `proofgrid_per_engine_max_local_jobs` to control the number of engine J jobs. This command supports the three modes described under “Syntax” below.

Use the `get_proofgrid_engineJ_max_jobs` command to display the current setting for `proofgrid_engineJ_max_jobs`.

Syntax

```
set_proofgrid_engineJ_max_jobs (N | follow | start_cap)
```

N	Use a positive integer N to limit the number of engine J jobs below the number of <code>max_jobs</code> . The number of started engine J jobs will be the minima of N and <code>max_jobs</code> .
follow	Make engine J reduce the number of engine jobs in the same way as other engines (apart from engines L and J) do when the number of remaining undetermined properties drops below <code>max_jobs</code> . The ProofGrid will not run more engine J jobs than there are undetermined properties being worked on in the current proof.
start_cap	Let the number of engine jobs be determined by the global <code>max_jobs</code> setting. The difference between this mode and the “follow” mode is that the number of jobs does not decrease as properties become determined. <i>This option is the default.</i>

```
get_proofgrid_engineJ_max_jobs
```

No arguments

Default

By default, the number of engine jobs is determined by the global `max_jobs` setting (`start_cap`).

Return

`get_proofgrid_engineJ_max_jobs` returns the current value of the `proofgrid_engineJ_max_jobs` variable.

Examples

```
# Start at most 7 engine J jobs  
% set_proofgrid_engineJ_max_jobs 7  
  
% get_proofgrid_engineJ_max_jobs
```

See Also

[set_proofgrid_engineL_max_jobs](#)
[set_proofgrid_per_engine_max_jobs](#)
[set_proofgrid_per_engine_max_local_jobs](#)

proofgrid_engineL_max_jobs

Commands

`set_proofgrid_engineL_max_jobs`
`get_proofgrid_engineL_max_jobs`

Description

Use `set_proofgrid_engineL_max_jobs` together with `proofgrid_per_engine_max_jobs` or `proofgrid_per_engine_max_local_jobs` to control the number of engine L jobs. This command supports the three modes described under “Syntax” below.

Use the `get_proofgrid_engineL_max_jobs` command to display the current setting for `proofgrid_engineL_max_jobs`.

Syntax

`set_proofgrid_engineL_max_jobs (N | follow | start_cap)`

<code>N</code>	Use a positive integer <code>N</code> to limit the number of engine L jobs below the number of <code>max_jobs</code> . The number of started engine L jobs will be the minima of <code>N</code> and <code>max_jobs</code> .
<code>follow</code>	Make engine reduce the number of engine jobs in the same way as other engines (apart from engines L and J) do when the number of remaining undetermined properties drops below <code>max_jobs</code> . The ProofGrid will not run more engine L jobs than there are undetermined properties being worked on in the current proof.
<code>start_cap</code>	Let the number of engine jobs be determined by the global <code>max_jobs</code> setting. The difference between this mode and the “follow” mode is that the number of jobs does not decrease as properties become determined. <i>This option is the default.</i>

`get_proofgrid_engineL_max_jobs`

No arguments

Default

By default, the number of engine jobs is determined by the global `max_jobs` setting (`start_cap`).

Return

`get_proofgrid_engineL_max_jobs` returns the current value of the `proofgrid_engineL_max_jobs` variable.

Examples

```
# Start at most 7 engine L jobs  
% set_proofgrid_engineL_max_jobs 7  
  
% get_proofgrid_engineL_max_jobs
```

See Also

[set_proofgrid_engineJ_max_jobs](#)
[set_proofgrid_per_engine_max_jobs](#)
[set_proofgrid_per_engine_max_local_jobs](#)

proofgrid_extra_args

Commands

`set_proofgrid_extra_args`
`get_proofgrid_extra_args`

Description

Use `set_proofgrid_extra_args` to supply extra arguments to a specified `proofgrid_mode` and thereby have additional control over how the jobs are spawned. This feature allows you to use the built-in modes for LSF, OGE, or NetworkComputer, even if you need to adhere to certain resource attributes, constraints, or policies when using the grid.

The ProofGrid mode `shell` continues to be the mode that gives you maximum flexibility, but it also demands the most of you in terms of ensuring that the jobs are spawned in a correct way. `proofgrid_extra_args` adds some flexibility to the built-in ProofGrid modes in that they can be used more often, but it also demands some additional caution to ensure that you use extra arguments that make sense and work.

Note: For proper parsing, you must prefix the first argument with a space.

Implementation notes per `proofgrid_mode`:

- `lsf` – The tool passes args to `bsub`.

Example:

```
{ -P proj -R "OSNAME==Linux && OSREL==EE50 rusage[mem=4000] " }
```

Consult your LSF documentation for more information.

- `oge` – The tool passes args to `qsub` (or `qrsh` if `proofgrid_socket_communication=off`). Consult your Grid Engine documentation for more information.
- `nc` – The tool passes args to `nc run`. Consult your NetworkComputer documentation for more information.
- `local | cluster` – The tool ignores the extra arguments you specify.
- `shell` – The specified `extra_args` will be available in the environment variable `JASPER_EXTRA_ARGS` for the shell to use (see “[set_proofgrid_shell](#)” on page 1152).

Use the `get_proofgrid_extra_args` command to display the current setting for `proofgrid_extra_args`.

Syntax

```
set_proofgrid_extra_args <args>
args      Set extra arguments that will be supplied to the active proofgrid_mode.
get_proofgrid_extra_args
No arguments
```

Default

By default, the tool passes no extra arguments to the specified proofgrid_mode.

Return

get_proofgrid_extra_args returns the current value of the proofgrid_extra_args variable.

Examples

```
% set_proofgrid_extra_args { -R "select[type==LINUX64 && mem>8000]
rusage[mem=8000]" }
```

See Also

[set_proofgrid_shell](#)

proofgrid_manager

Commands

set_proofgrid_manager
get_proofgrid_manager

Description

Use the `set_proofgrid_manager` command to turn on or off the ProofGrid™ Manager feature to orchestrate parallel proofs. By default, the feature is off.

Note:

- You can enable ProofGrid Manager at any time during a proof or Visualize, and it will show results from that point on. To capture complete results, enable ProofGrid Manager before starting the proof or Visualize.
- For additional information, refer to the “Jasper ProofGrid, Per-Engine Job Management, and ProofGrid Manager” tutorial available from Cadence Online Support (support.cadence.com).

Use the `get_proofgrid_manager` command to display the current ProofGrid Manager setting.

Syntax

```
set_proofgrid_manager (on | off)
```

on Turn on the ProofGrid Manager feature.

off Turn off the ProofGrid Manager feature. *This option is the default.*

```
get_proofgrid_manager
```

No arguments

Default

By default, ProofGrid Manager computations are off.

JasperGold Apps Command Reference Manual

Configuration Commands

Return

`get_proofgrid_manager` returns the current value of the `proofgrid_manager` variable, on or off.

Examples

```
% get_proofgrid_manager  
% set_proofgrid_manager on
```

See Also

No related commands

proofgrid_max_jobs

Commands

`set_proofgrid_max_jobs`
`get_proofgrid_max_jobs`

Description

Use the `set_proofgrid_max_jobs` command to set the maximum number of jobs ProofGrid can use to deliver results. This command, the `proofgrid_per_engine_max_jobs` variable, and the number of available licenses limit the number of jobs ProofGrid can start.

Note: For additional information, refer to the “Jasper ProofGrid, Per-Engine Job Management, and ProofGrid Manager” tutorial available from Cadence Online Support (support.cadence.com).

Use the `get_proofgrid_max_jobs` command to report the number of jobs available to ProofGrid.

Syntax

<code>set_proofgrid_max_jobs <N></code>
<code>N</code> Set a limit for the number of jobs available to ProofGrid.
<code>get_proofgrid_max_jobs</code>
No arguments

Default

By default, the number of jobs available to ProofGrid is 0 (unlimited).

Return

`get_proofgrid_max_jobs` returns the value of the variable `proofgrid_max_jobs`.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_proofgrid_max_jobs 10  
% get_proofgrid_max_jobs
```

See Also

[set_proofgrid_max_local_jobs](#)
[set_proofgrid_per_engine_max_jobs](#)

proofgrid_max_local_jobs

Commands

`set_proofgrid_max_local_jobs`
`get_proofgrid_max_local_jobs`

Description

Use the `set_proofgrid_max_local_jobs` command to set the number of jobs ProofGrid can use when `proofgrid_mode` is local. This command, the `proofgrid_per_engine_max_jobs` variable, and the number of available licenses limit the number of jobs ProofGrid can start.

Note: For additional information, refer to the “Jasper ProofGrid, Per-Engine Job Management, and ProofGrid Manager” tutorial available from Cadence Online Support (support.cadence.com).

Use the `get_proofgrid_max_local_jobs` command to report the number of jobs available to ProofGrid.

Syntax

`set_proofgrid_max_local_jobs <N>`

`N` Set a limit for the number of jobs available to ProofGrid when `proofgrid_mode` is local.

`get_proofgrid_max_local_jobs`

No arguments

Default

By default, the number of jobs available to ProofGrid when in `local` mode is unlimited (0).

Return

`get_proofgrid_max_local_jobs` returns the value of the variable `proofgrid_max_local_jobs`.

Examples

```
% set_proofgrid_max_local_jobs 8  
% get_proofgrid_max_local_jobs
```

See Also

[set_proofgrid_max_jobs](#)
[set_proofgrid_per_engine_max_local_jobs](#)

proofgrid_mode

Commands

`set_proofgrid_mode`
`get_proofgrid_mode`

Description

Use the `set_proofgrid_mode` command to specify how multiple proof engine jobs are spawned for prove or Visualize.

Note: For additional information, refer to the “Jasper ProofGrid, Per-Engine Job Management, and ProofGrid Manager” tutorial available from Cadence Online Support (support.cadence.com).

Helpful information about using Platform LSF[®], Oracle[®] Grid Engine (OGE), and Runtime NetworkComputerTM (NC) with JasperGold Apps:

- Refer to “[LSF_BINDIR](#)” on page 76, “[SGE_ROOT](#)” on page 78, or “[VOVDIR](#)” on page 78 for additional information on variables you might need to set. For more detailed instruction, consult your LSF, OGE, or NetworkComputer documentation for installation, configuration, and limitations.
- The project directory must be accessible using the same path from all machines in the server farm.
- You might need to set up authentication such that you can log in to all machines in the server farm without a password. See for example `ssh-keygen` and consult your LSF, OGE, or NetworkComputer documentation for details.
- You might get X11 forwarding warnings. You can ignore them.
- See “[set_proofgrid_socket_communication](#)” on page 1161 for more information.

Note: ProofGrid proves multiple properties in parallel using multiple licenses.

Use the `get_proofgrid_mode` command to return the value of the variable `set_proofgrid_mode`.

JasperGold Apps Command Reference Manual

Configuration Commands

Syntax

set_proofgrid_mode (local shell lsf oge nc cluster)	
local	Run the processes on the local machine (the machine running the JasperGold Apps session). <i>This option is the default.</i>
shell	Use a custom command or script to spawn the jobs. Use this command in combination with set_proofgrid_shell (see “ set_proofgrid_shell ” on page 1152).
lsf	Run the processes on the LSF server farm that is expected to be available. The LSF implementation of ProofGrid relies on the lsrun command to launch jobs over the network. The path to access this command is necessary as part of the environment setup.
oge	Run the processes on the OGE that is expected to be available.
nc	Run the processes on the NetworkComputer that is expected to be available. Note: Refer to the Runtime Design Automation website for additional information about NetworkComputer: www.rtda.com .
cluster	Spread the processes over the machines defined by set_proofgrid_cluster.
get_proofgrid_mode	
No arguments	

Default

By default, ProofGrid runs on your local machine.

Return

get_proofgrid mode returns the value of the variable proofgrid_mode (local, shell, lsf, oge, or nc).

Examples

```
% set_engine_mode d  
% set_proofgrid_mode lsf
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% set_proofgrid_per_engine_max_jobs 5  
% get_proofgrid_mode
```

See Also

[prove](#)
[set_proofgrid_cluster](#)
[set_proofgrid_per_engine_max_jobs](#)
[set_proofgrid_per_engine_max_local_jobs](#)
[set_proofgrid_per_engine_privileged_jobs](#)
[set_proofgrid_shell](#)
[set_proofgrid_socket_communication](#)
[visualize](#)

proofgrid_per_engine_max_jobs

Commands

`set_proofgrid_per_engine_max_jobs`
`get_proofgrid_per_engine_max_jobs`

Description

Use the `set_proofgrid_per_engine_max_jobs` command to specify the number of engine jobs ProofGrid can start per engine to deliver quick results. When ProofGrid runs, it attempts to start the specified number of jobs per engine and use them for parallel jobs to deliver results faster. The number cannot be lower than 1.

When setting the limit for maximum jobs per engine, consider your operating system's default limit for file descriptors and the number of file descriptors required. Each per engine job requires 1 file descriptor, each proof job requires 1 additional file descriptor (2 if socket communication is off), and additional file descriptors are required for other purposes. For example, `{Hp Ht B K I D N}` with 100 jobs per engine and `socket_communication=off` requires greater than 1104 file descriptors; however, the default limit for Linux is 1024.

The actual number of jobs per engine managed by ProofGrid might be different from what you have specified and can change over time for a number of reasons, including the following:

- ProofGrid can determine an upper limit on the level of parallelism it can put to good use. This number can vary over time while the prove or Visualize is running. In particular, it is common for this number to drop when there are only a few properties remaining. The number of jobs actually running dynamically drops accordingly.
- Fewer licenses might be available from the license server.
- The number of jobs per engine can increase, up to the limit you set, as licenses become available. The tool attempts license checkout once every 60 seconds. It will try to check out as many licenses as possible until it reaches the specified limit.

Important information about the `set_proofgrid_per_engine_max_jobs` command:

- Increasing the number of jobs per engine can speed up the prove or Visualize, but it can also increase the load on ProofGrid. For best results, you might need to monitor the memory and CPU load of the grid to ensure that it does not get out of bounds.
- This command has no effect on proofs or Visualize already in progress; you must run it before running prove or Visualize.

JasperGold Apps Command Reference Manual

Configuration Commands

- For additional information, refer to the “Jasper ProofGrid, Per-Engine Job Management, and ProofGrid Manager” tutorial available from Cadence Online Support (support.cadence.com).

Note: ProofGrid proves multiple properties in parallel using multiple licenses.

Use the `get_proofgrid_per_engine_max_jobs` command to display the maximum jobs per engine ProofGrid will check out depending on availability.

Syntax

```
set_proofgrid_per_engine_max_jobs <N>
```

`N` Specify the number of engine jobs ProofGrid can start per engine. If the specified number of licenses is not available, continue to attempt to check out this number every 60 seconds.

`N` is a positive integer. *The default is 2.*

```
get_proofgrid_per_engine_max_jobs
```

No arguments

Default

By default, ProofGrid checks out 2 licenses.

Return

`get_proofgrid_per_engine_max_jobs` returns the value of the variable `proofgrid_per_engine_max_jobs`, which will be a positive integer.

Examples

```
% set_proofgrid_per_engine_max_jobs 10  
% get_proofgrid_per_engine_max_jobs
```

See Also

[set_engine_mode](#)
[set_proofgrid_max_jobs](#)
[set_proofgrid_mode](#)
[set_proofgrid_per_engine_max_local_jobs](#)

JasperGold Apps Command Reference Manual

Configuration Commands

[set_proofgrid_shell](#)

[prove](#)

[visualize](#)

proofgrid_per_engine_max_local_jobs

Commands

`set_proofgrid_per_engine_max_local_jobs`
`get_proofgrid_per_engine_max_local_jobs`

Description

Use the `set_proofgrid_per_engine_max_local_jobs` command to specify the number of engine jobs ProofGrid can start per engine when `proofgrid_mode` is local.

This command sets a lower limit for the maximum number of jobs per engine in effect when `proofgrid_mode` is local. This command can be useful when switching back and forth between local and distributed ProofGrid modes when it is not suitable to start as many jobs on your local machine as can be started in the distributed setting.

The actual number of jobs per engine held by ProofGrid might be different from what you have specified for a number of reasons, including the following:

- ProofGrid can determine an upper limit on the level of parallelism it can put to good use. This number can vary over time while the proof or Visualize is running. In particular, it is common for this number to drop when there are only a few properties remaining. The number of jobs actually running dynamically drops accordingly.
- Fewer licenses might be available from the license server.
- The number of jobs per engine can increase, up to the limit you set, as licenses become available. The tool attempts license checkout once every 60 seconds. It will try to check out as many licenses as possible until it reaches the specified limit.

Note:

- This command has no effect on proofs or Visualize already in progress; you must run it before running prove or Visualize.
- ProofGrid proves multiple properties in parallel using multiple licenses.
- For additional information, refer to the “Jasper ProofGrid, Per-Engine Job Management, and ProofGrid Manager” tutorial available from Cadence Online Support (support.cadence.com).

Use `get_proofgrid_per_engine_max_local_jobs` command to display the limit for jobs per engine on the local machine.

Syntax

```
set_proofgrid_per_engine_max_local_jobs <N>
```

N Specify the number of engine jobs ProofGrid can start per engine when in local mode. If the specified number of licenses is not available, continue to attempt to check out this number every 60 seconds.

N is a positive integer. *The default is 1.*

```
get_proofgrid_per_engine_max_local_jobs
```

No arguments

Default

By default, ProofGrid uses a maximum of 1 local license.

Return

`get_proofgrid_per_engine_max_local_jobs` returns the value of the variable `proofgrid_per_engine_max_local_jobs`, which will be a positive integer.

Examples

```
% set_proofgrid_per_engine_max_local_jobs 2  
% get_proofgrid_per_engine_max_local_jobs
```

See Also

[set_proofgrid_max_local_jobs](#)
[set_proofgrid_per_engine_max_jobs](#)

proofgrid_per_engine_privileged_jobs

Commands

`set_proofgrid_per_engine_privileged_jobs`
`get_proofgrid_per_engine_privileged_jobs`

Description

Use the `set_proofgrid_per_engine_privileged_jobs` command to specify the number of licenses ProofGrid uses that will not be subject to surrender, that is, the number that will not be “nice” licenses. By default, all licenses, except the main license that is running the session, are nice. The value `$` makes all of the licenses privileged.

Note:

- In a “nice” session, all licenses checked out are nice, and this command has no effect (except value returned by `get_proofgrid_per_engine_privileged_jobs`).
- ProofGrid proves multiple properties in parallel using multiple licenses.
- For additional information, refer to the “Jasper ProofGrid, Per-Engine Job Management, and ProofGrid Manager” tutorial available from Cadence Online Support (support.cadence.com).

Use the `get_proofgrid_per_engine_privileged_jobs` command to report the number of licenses that are safe from surrender.

Syntax

<code>set_proofgrid_per_engine_privileged_jobs (<N> \$)</code>	
<code>N</code>	Make the specified number of licenses privileged, that is, not subject to surrender.
<code>\$</code>	Make all licenses used by ProofGrid privileged, that is, not subject to surrender.
<code>get_proofgrid_per_engine_privileged_jobs</code>	
No arguments	

Default

By default, the license running the session is not subject to surrender (1).

Return

`get_proofgrid_per_engine_privileged_jobs` returns the value of the variable `proofgrid_per_engine_privileged_jobs`.

Examples

```
% get_proofgrid_per_engine_privileged_jobs  
% set_proofgrid_per_engine_max_jobs 5  
% set_proofgrid_per_engine_privileged_jobs 3
```

See Also

[prove](#)
[set_proofgrid_mode](#)
[set_proofgrid_per_engine_max_jobs](#)
[set_proofgrid_per_engine_max_local_jobs](#)
[set_proofgrid_shell](#)

proofgrid_queue

Commands

`set_proofgrid_queue`
`get_proofgrid_queue`

Description

Use the `set_proofgrid_queue` command to send a queue name, when not "", to the selected `proofgrid_mode` and thereby control where the jobs will be spawned.

Notes per `proofgrid_mode`:

- local | cluster – The queue name will be ignored.
- shell – The queue name will be available in the environment variable JASPER_QUEUE for the shell to use. See “[set_proofgrid_shell](#)” on page 1152.
- lsf|oge|nc – The queue name will be specified using -q queue_name. You can find available queues with exec bqueues or exec qstat respectively. Consult your LSF, OGE, or NetworkComputer documentation for more information.

Use the `get_proofgrid_queue` command to report the queue name for parallel proofs.

Syntax

```
set_proofgrid_queue <queue_name>
```

```
queue_name      Set the queue name on which to run parallel proofs.
```

```
get_proofgrid_queue
```

```
No arguments
```

Default

By default, the queue name is "".

Return

`get_proofgrid_queue` returns the value of the variable `proofgrid_queue`.

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[set_proofgrid_shell](#)

proofgrid_restarts

Commands

`set_proofgrid_restarts`
`get_proofgrid_restarts`

Description

By default, ProofGrid allows 10 restarts per engine job before abandoning a job. Use the command `set_proofgrid_restarts` to control the number of restarts.

Use the `get_proofgrid_restarts` command to report how many times ProofGrid will restart each engine job before abandoning it.

Syntax

`set_proofgrid_restarts <N>`

`N` Allow ProofGrid to restart the specified number of times before abandoning a job.

`get_proofgrid_restarts`

No arguments

Default

By default, ProofGrid allows 10 restarts per engine job before abandoning a job.

Return

`get_proofgrid_restarts` returns the number of restarts ProofGrid allows per engine job.

Examples

```
% get_proofgrid_restarts  
% set_proofgrid_restarts 20
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[set_proofgrid_mode](#)

proofgrid_shell

Commands

`set_proofgrid_shell`
`get_proofgrid_shell`

Description

During prove and Visualize, the tool often spawns multiple proof engine jobs (`jg_proof` processes). The `set_proofgrid_shell` command specifies how the jobs are spawned when the `proofgrid_mode` setting is shell.

Use the `get_proofgrid_shell` command to get the value of the variable `proofgrid_shell`.



- The `proofgrid_shell` must be a string beginning with the full path to an executable. As an option, that string can continue with a space-separated set of extra arguments that will be given to the executable. You can use double and single quotes to form an argument that contains spaces. The unquoted token `%jg_session_id` will be replaced with a unique JasperGold Apps session ID (see `JASPER_SESSION_ID` below). The unquoted token `%jg_engine_job_threads` will be replaced with the effective `engine_threads` setting (see “[get_engine_threads](#)” on page 1022 and `JASPER_ENGINE_JOB_THREADS` below).
- The executable must take the job to start and its arguments as further arguments.
- The executable must propagate (that is, tunnel) `stdin/stdout` reliably from and to the job unless using socket communication. See “[set_proofgrid_socket_communication](#)” on page 1161. This arrangement is sometimes called interactive mode.
- The `proofgrid_shell` executable must have a lifetime that is as long as the `jg_proof` process it started. For example, when using Platform LSF(R) mode, use non-interactive queues, and use `-K` to cause `bsub` to wait for the ProofGrid job to complete before exiting. If you do not use `-K`, you might receive numerous LSF messages stating that the proof jobs failed and `Exited with exit code 1` due to (for example) the following:

```
jg_proof: connect: Connection refused  
jg_proof: failed to connect
```

JasperGold Apps Command Reference Manual

Configuration Commands

- If proofgrid_mode is `shell`, you must update the value of proofgrid_shell because the default value is merely a placeholder that indicates you need to modify it. The default value will cause the engine jobs to issue a message to the console and then fail; that is, it is not useful for running proofs.
- ProofGrid does not robustly support automatic re-queueing of killed jobs, but you can disable the feature with `bsub -rn` and rely on the built-in ProofGrid restart mechanism.

The following environment variables are available to the shell:

- `JASPER_INSTALL_DIR`: The location of the installation.
- `JASPER_SESSION_ID`: A string unique to this session and prove invocation.
- `JASPER_ENGINE_MODE`: The engine mode this job will run, for example, `engineHt`. See ["Examples"](#) on page 1155.
- `JASPER_ENGINE_JOB_ID`: The ID of this job, for example, `0.Ht`.
- `JASPER_ENGINE_JOB_RESTART`: The number of times this job ID has been restarted during this prove invocation. `0` means this is the first time this job ID started during this prove invocation.
- `JASPER_ENGINE_JOB_MAX_RESTART`: The number of restarts ProofGrid allows per engine job.
- `JASPER_ENGINE_JOB_THREADS`: The number of threads this job is expected to run. This corresponds to numbers of processor cores the shell must allow the engine job to consume (see, for example, `bsub -N`, `qsub -l nodes=1:ppn=`, or `nc run -dp`).
- `JASPER_PROJECT_DIR`: The project directory specific to the current session, see [session -get proj dir](#).
- `JASPER_PROVE_TIME_LIMIT`: If the current proof has a time limit, this variable contains the number of seconds; otherwise, the tool does not set this variable. This value can safely be used as a hard wall-clock limit for the job (for example, to format the time for `bsub -W`).

Note:

- For commands that set a time limit, see ["set prove time limit"](#) on page 1189 and [prove -time_limit](#).
- By default, the prove time limit is 86400 seconds (24 hours).
- `JASPER_SOCKET_COMMUNICATION`: Whether communication happens over a TCP/IP socket or standard in/out. `1` if `proofgrid_socket_communication` is on; otherwise, `0`.

JasperGold Apps Command Reference Manual

Configuration Commands

- JASPER_QUEUE: The name of the queue engine jobs should be spawned to according to the proofgrid_queue. See "[get proofgrid queue](#)" on page 1148.
- JASPER_EXTRA_ARGS: Extra arguments according to proofgrid_extra_args. See "[set proofgrid extra args](#)" on page 1130.

Note: ProofGrid jobs do not have any wait states, which may sometimes be the case with other tools that require interactive mode. The executable can use stderr for its own output, which will be sent to the terminal that starts JasperGold Apps. The executable may also issue messages on stdout under very limited circumstances:

- The messages must come before or after all messages the job produces.
- When socket_communication=off the messages must not contain the SOH – Start of heading control character \1.
- The messages will appear as Proof Messages in the designated tab and in the log. Messages beginning with = will print to the console instead of the *Proof Messages* tab.
- The project directory must be accessible using the same path from all machines in the server farm.

If these rules are not respected, JasperGold Apps ProofGrid will not work reliably.

Troubleshooting:

To get more insight into why your ProofGrid jobs fail to launch, use the built-in debugging features of the programming language you use for your proofgrid_shell as shown in the bash examples below.

1. Create the following executable file:

```
debug.sh
#!/bin/bash
exec /bin/bash -xv "$@" 2>&1
```

2. Prefix your proofgrid_shell setting with ./debug.sh:

```
set_proofgrid_shell
{myscripttodebug.sh}
```

becomes

```
set_proofgrid_shell
{./debug.sh myscripttodebug.sh}
```

With this addition to the command, the exact UNIX command executed will be echoed to the proof messages; however, if you prefer to have the exact UNIX command executed being echoed on stderr instead, use the following command:

JasperGold Apps Command Reference Manual

Configuration Commands

```
set_proofgrid_shell  
{/bin/bash -xv myscripttodebug.sh}
```

As an alternative, you can add the following to the beginning of `myscripttodebug.sh`:

```
#!/bin/bash -xv
```

If your `proofgrid_shell` is written in another language (for example, C, Tcl, or Java) use the debugging features of that language.

Syntax

```
set_proofgrid_shell <full_path_to_shell_plus_args>
```

```
full_path_to_shell_plus_args
```

A string beginning with the full path to an executable optionally continuing with a space-separated set of extra arguments that will be given to the executable. Use double and single quotes to form an argument containing spaces.

Note: When using a wrapper to spawn ProofGrid jobs through a `bsub` command, include `exec` before `bsub` in the wrapper file.

```
get_proofgrid_shell
```

```
No arguments
```

Default

```
"/bin/sh -c 'echo =proofgrid_shell is not defined; false'"
```

Return

`get_proofgrid shell` returns the value of the variable `proofgrid_shell`.

Examples

```
% set_proofgrid_shell {/usr/bin/ssh -x myhost}  
% set_proofgrid_shell {bsub -N %jg_engine_job_threads -K -q lnx64 -R "OSNAME==Linux  
&& OSREL==EE50 rusage[mem=4000] span[hosts=1]"}  
  
# In your wrapper file, "exec" the bsub command.  
% set_proofgrid_shell {./my_bsub_wrapper.sh}
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% set_proofgrid_shell {/usr/local/lsf/bin/bsub -N %jg_engine_job_threads -Ip -q  
normal -R "select[type==X86_64 && mem>2000] rusage[mem=1000] span[hosts=1]" }  
  
# Example script  
# -----  
# This is an example of how you can access the JASPER_ENGINE_MODE variable from  
# the proofgrid_shell script. You can then use information from  
# JASPER_ENGINE_MODE to decide how to submit a grid job.  
  
echo_engine.sh:  
    #!/bin/sh  
    # output must begin with '<' or be directed to stderr to not  
    # confuse the proofgrid  
    echo \< $$: $JASPER_ENGINE_MODE  
    #echo $$: $JASPER_ENGINE_MODE 1>&2  
    exec $@  
  
echo_engine.tcl:  
    set_proofgrid_mode shell  
    set_proofgrid_shell {./echo_engine.sh}  
    set_engine_mode {D B J}  
    prove -all
```

See Also

[set_proofgrid_mode](#)
[set_proofgrid_restarts](#)
[set_proofgrid_shell_stop](#)
[set_proofgrid_socket_communication](#)

proofgrid_shell_stop

Commands

set_proofgrid_shell_stop
get_proofgrid_shell_stop

Description

During prove and Visualize, the tool often spawns multiple proof engine jobs (jg_proof processes). Use the command `set_proofgrid_shell_stop` to specify a stop hook that will be executed when the `prove` or `visualize` command is about to exit. The token `%jg_session_id` will be replaced with a unique session ID. This can be useful to ensure that jobs still in a batch queue are killed as the `prove` or `visualize` exits.

Note: The command is expected to execute to completion quickly and it will block the proof thread during its execution.

Use the `get_proofgrid_shell_stop` command to get the value of the variable `proofgrid_shell_stop`.

Syntax

```
set_proofgrid_shell_stop <full_path_to_shell_plus_args>
```

```
full_path_to_shell_plus_args
```

A string beginning with the full path to an executable optionally continuing with a space-separated set of extra arguments that will be given to the executable. Use double and single quotes to form an argument containing spaces.

```
get_proofgrid_shell_stop
```

```
No arguments
```

Default

None.

Return

`get_proofgrid_shell_stop` returns the value of the variable `proofgrid_shell_stop`.

Examples

```
% set_proofgrid_shell {/usr/local/lsf/bin/bsub -J %jg_session_id }  
% set_proofgrid_shell_stop {/usr/local/lsf/bin/bkill -J %jg_session_id }  
% get_proofgrid_shell_stop
```

See Also

[set_proofgrid_shell](#)

proofgrid_skip_abandoned

Commands

`set_proofgrid_skip_abandoned`
`get_proofgrid_skip_abandoned`

Description

Use the `set_proofgrid_skip_abandoned` command to specify whether engine jobs that terminate abruptly while proving a property with `prove_per_property_time_limit_factor=0` should return to the same property after a restart or skip the “abandoned” property.

Note:

- An engine job might terminate abruptly as a result of memout, machine failure, and so forth.
- This setting is irrelevant when `prove_per_property_time_limit_factor!=0`.
- Letting engine jobs skip abandoned properties can be useful to ensure that a high level of parallelism is sustained during the proof.

Use the `get_proofgrid_skip_abandoned` command to determine whether a restarted job will return to its abandoned target.

Syntax

<code>set_proofgrid_skip_abandoned (true false)</code>
--

<code>true</code>	Do not return to the abandoned property on restart.
-------------------	---

<code>false</code>	Return to the abandoned property on restart. <i>This option is the default.</i>
--------------------	---

<code>get_proofgrid_skip_abandoned</code>

No arguments

Default

By default, the engine job will return to the abandoned property on restart (`false`).

Return

`get_proofgrid_skip_abandoned` returns the value of the variable `proofgrid_skip_abandoned`, true or false

Examples

```
% set_proofgrid_skip_abandoned true
```

See Also

[set prove_per_property_time_limit_factor](#)

proofgrid_socket_communication

Commands

`set_proofgrid_socket_communication`
`get_proofgrid_socket_communication`

Description

When `set_proofgrid_socket_communication` is on, engines exchange information using a TCP/IP socket.



When `proofgrid_socket_communication` is on:

- For Platform LSF mode, use non-interactive queues, and use `-K` to cause `bsub` to wait for the ProofGrid job to complete before exiting.
- For Oracle Engine Grid (OGE) mode, use `qsub` instead of `qrsh`, and use `-sync y` to cause `qsub` to wait for the ProofGrid job to complete before exiting.
- For shell mode, make user-specified ProofGrid shells synchronous (wait to exit until the proof or Visualize job is done).

When `proofgrid_socket_communication` is off:

- When `proofgrid_mode=shell` the shell must propagate (that is, tunnel) `stdin/stdout` reliably from and to the job. This arrangement is sometimes called interactive mode. For other ProofGrid modes this is taken care of for you.
- Your LSF setup and policies must allow jobs started with `bsub -I`. The jobs will communicate using a standard in/out socket, but will have no wait states.

Note: For additional information, refer to the “Jasper ProofGrid, Per-Engine Job Management, and ProofGrid Manager” tutorial available from Cadence Online Support (support.cadence.com).

Use the `get_proofgrid_socket_communication` command to get the value of the variable `proofgrid_shell`.

Syntax

```
set_proofgrid_socket_communication (on | off)
```

on	Exchange information during proofs over a TCP/IP socket. <i>This option is the default.</i>
----	---

off	Exchange information during proofs over a standard in/out socket.
-----	---

```
get_proofgrid_socket_communication
```

No arguments

Default

By default, `set_proofgrid_socket_communication` is on.

Return

`get_proofgrid_socket_communication` returns on if communication with proof jobs happens over a TCP/IP socket.

Examples

```
% set_proofgrid_socket_communication on
% set_proofgrid_shell {/tools/vendors/bin/nc run -v 0 -w -C jg}
% set_proofgrid_socket_communication off
% set_proofgrid_shell {/tools/vendors/bin/nc run -v 0 -Ir -C jg}
% get_proofgrid_socket_communication
```

See Also

[set_proofgrid_mode](#)

property_auto_update_db

Commands

`set_property_auto_update_db`
`get_property_auto_update_db`

Description

By default, the tool does not automatically update the database when you create new properties with `assert`, `assume`, and `cover` commands. Use `set_property_auto_update_db` to enable database updates.

Note: Enabling automatic updates can impact run-time performance.

Use `get_property_auto_update_db` to find out if the tool is set to automatically update the database when you create properties with the commands `assert`, `assume`, and `cover`.

Syntax

`set_property_auto_update_db (on | off)`

`on` Automatically update the database with properties as they are created.

`off` Do not automatically update the database with properties as they are created. *This option is the default.*

`get_property_auto_update_db`

No arguments

Default

By default, the tool does not automatically update the database when you create new properties.

Return

`get_property_auto_update_db` returns the value of the `property_auto_update_db` variable.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_property_auto_update_db on  
% get_property_auto_update_db
```

See Also

No related commands

property_compile_time_limit

Commands

`set_property_compile_time_limit`
`get_property_compile_time_limit`

Description

Use the `set_property_compile_time_limit` command to specify the time limit for compiling each property during synthesis. Property compilation runs during elaboration.

Use the `get_property_compile_time_limit` command to display the maximum time the tool can spend compiling each property during synthesis.

Syntax

```
set_property_compile_time_limit <time_limit>
```

time_limit Use the specified limit for property compilation during elaboration.

Specify the time limit as an integer followed by s, m, or h (no spaces).

- s = seconds
- m = minutes
- h = hours

JasperGold Apps commonly recognizes 0s as “unlimited.” In the case of this command, 0s sets the highest supported limit, which is 10000 seconds.

```
get_property_compile_time_limit
```

No arguments

Default

By default, the property compilation time limit is 5s.

Return

`get_property_compile_time_limit` returns the value of the variable `property_compile_time_limit`.

Examples

```
% set_property_compile_time_limit 15s  
% get_property_compile_time_limit
```

See Also

[set_elab_time_limit](#)
[set_task_compile_time_limit](#)

property_delay_cycles

Commands

`set_property_delay_cycles`
`get_property_delay_cycles`

Description

Use the `set_property_delay_cycles` command to specify the number of cycles you would like to delay properties.

Note:

- As this command affects the creation of the compiled properties for embedded and Tcl properties, you must run it before `elaborate` (or before creating the Tcl properties).
- `set_capture_elaborated_design` is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

Use the `get_property_delay_cycles` command to return the current number of delay cycles.

Syntax

`set_property_delay_cycles <N>`

Nt Delay properties by the specified number of cycles.

`get_property_delay_cycles`

No arguments

Default

By default, the property compilation time limit is 5s.

Return

`get_property_delay_cycles` returns the value of the variable `property_delay_cycles`.

Examples

```
% set_property_delay_cycles 15  
% get_property_delay_cycles
```

See Also

[set_elab_time_limit](#)
[set_task_compile_time_limit](#)

prove_no_cover_traces

Commands

`set_prove_no_cover_traces`
`get_prove_no_cover_traces`

Description

Use the command `set_prove_no_cover_traces` to suppress (or reactivate) cover trace generation during proofs. With this command, you can avoid the overhead of generating and storing cover traces. The properties show covered statuses as usual, but there are no traces to inspect. You must re-do the proof to get traces. Use this command when the cover trace data is not important, for example, when scripting a non-interactive session.

Note: This setting does not affect the `visualize` command.

Use the command `get_prove_no_cover_traces` to determine the value of the variable `prove_no_cover_traces`.

Syntax

<code>set_prove_no_cover_traces (true false)</code>	
<code>true</code>	Suppress cover trace generation during proofs.
<code>false</code>	Do not suppress cover trace generation during proofs. <i>This option is the default.</i>
<code>get_prove_no_cover_traces</code>	
<code>No arguments</code>	

Default

By default, `set_prove_no_cover_traces` is `false` and cover traces are generated during proofs.

Return

`get_prove_no_cover_traces` returns the value of the variable `prove_no_cover_traces`.

Examples

```
% set_prove_no_cover_traces true
```

See Also

[prove](#)

[set_prove_no_traces](#)

prove_no_traces

Commands

set_prove_no_traces
get_prove_no_traces

Description

Use the command `set_prove_no_traces` to suppress (or reactivate) trace generation during proofs. With this command, you can avoid the overhead of generating and storing traces. The properties show cex/covered statuses as usual, but there are no traces to inspect. You must re-do the proof to get traces. Use this command when the trace data is not important, for example, when scripting a non-interactive session.

Note: This setting does not affect the `visualize` command.

Use the command `get_prove_no_traces` to determine the value of the variable `prove_no_traces`.

Syntax

set_prove_no_traces (true false)	
true	Suppress trace generation during proofs.
false	Do not suppress trace generation during proofs. <i>This option is the default.</i>
get_prove_no_traces	
No arguments	

Default

By default, `set_prove_no_traces` is `false` and traces are generated during proofs.

Return

`get_prove_no_traces` returns the value of the variable `prove_no_traces`.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_prove_no_traces true
```

See Also

[prove](#)

[set_prove_no_cover_traces](#)

prove_per_property_max_time_limit

Commands

`set_prove_per_property_max_time_limit`
`get_prove_per_property_max_time_limit`

Description

Use the `set_prove_per_property_max_time_limit` command to set a wall clock time limit for the proof of each individual property. After at least one engine works consecutively on a property for a duration greater than the user-specified limit, all single-property engines will stop processing that property.



- ❑ This command is useful when the value of the variable `prove_per_property_time_limit_factor` is greater than one. It allows you to limit the growth of the `per_property_time_limit` variable.
- ❑ Engine modes that operate on all properties in parallel (for example, H, J, L) are not affected by this command.

Note: To remove the limit, use 0s.

Use the `get_prove_per_property_max_time_limit` command to get the time limit for the proof of each property.

JasperGold Apps Command Reference Manual

Configuration Commands

Syntax

```
set_prove_per_property_max_time_limit <time_limit>
```

time_limit Use the specified single-property engine wall clock time limit for the proof of each individual property.

The tool supports three formats for the time limit:

- ISO8601: hh' : ' [mm[':' [ss['.' [ffff]]]]]

Examples: 2:30, 2:, 0:0:23, 2:30:05.235

- JasperGold: [dd'd'] [hh'h'] [mm'm'] [ss[.ffff]'s']

Examples: 2h30m, 2h, 23s, 2h30m05.235s

- Seconds: sssss[.ffff] ['s']

Examples: 9000s, 7200, 23s, 9005.235s

To remove the limit, use 0s.

```
get_prove_per_property_max_time_limit
```

No arguments

Default

By default, the per property proof max time limit is 0s.

Return

`get_prove_per_property_max_time_limit` returns the value of the variable `prove_per_property_max_time_limit`.

Examples

```
% set_prove_per_property_max_time_limit 15m  
% get_prove_per_property_max_time_limit
```

See Also

[get_prove_per_property_time_limit](#)
[get_prove_time_limit](#)
[set_prove_per_property_time_limit](#)

JasperGold Apps Command Reference Manual

Configuration Commands

[set_prove_per_property_time_limit_factor](#)

[set_prove_time_limit](#)

prove_per_property_time_limit

Commands

`set_prove_per_property_time_limit`
`get_prove_per_property_time_limit`

Description

Use the `set_prove_per_property_time_limit` command to specify the maximum time the tool spends proving any individual assertion or cover directive. The `prove_per_property_time_limit` applies for each property; whereas, `prove_time_limit` decides the time limit for the `prove` or `visualize` command.

This command is useful for allowing a `prove` command to spend a limited amount of time on every property. Without this command, there is a risk that all the `prove_time_limit` time is spent on the first hard property even though there might be many easy properties following. This risk does not apply to engine H, J, or L since they run all properties in parallel.

By combining this with `prove_per_property_time_limit_factor`, the `prove` command does not abort when the time limit has expired for all properties; rather, it restarts with the effective `per_property_time_limit` grown by the factor. The goal of this feature is to allow the `prove` command to quickly determine the result for simple properties while giving harder properties the time they need.

In the following example, if we have 10 properties to be proven with the `prove` command, the tool allocates no more than 10 seconds for each property and 120 wall clock seconds for the `prove` command as a whole.

```
set_prove_per_property_time_limit_factor 0
set_prove_per_property_time_limit 10s
set_prove_time_limit 2m
```



- Unlike `prove_time_limit`, the time this setting limits starts when an engine job actually starts proving a particular property. Thus, it excludes time spent doing reset analysis and proof simplification, waiting in batch queues, and so forth.

JasperGold Apps Command Reference Manual

Configuration Commands

- This limit is individually respected by each concurrent proof job. Different jobs may have different startup latencies, so they are not necessarily synchronized with respect to this time limit.
- Engine modes that operate on all properties in parallel (for example, H, J, L) support this setting only in the sense that the total time is limited by this time multiplied by the number of properties.
- Using `prove_per_property_time_limit_factor`, the expiration of this limit, for all properties, triggers restart with a geometrically increased factor rather than abortion. In this case, `per_property_time_limit` will never be a reason for the `prove` command to abort. `prove_time_limit` works as usual.

Use the `get_prove_per_property_time_limit` command to display the maximum time the tool can spend proving an individual property.

Syntax

```
set_prove_per_property_time_limit <time_limit>
```

`time_limit` Use the specified time limit for each proof of an assertion or cover directive.

The tool supports three formats for the time limit:

- ISO8601: `hh' : ' [mm[' : ' [ss['.' [ffff]]]]]`
Examples: `2:30`, `2:`, `0:0:23`, `2:30:05.235`
- JasperGold: `[dd'd'] [hh'h'] [mm'm'] [ss[.ffff]'s']`
Examples: `2h30m`, `2h`, `23s`, `2h30m05.235s`
- Seconds: `ssss[.ffff] ['s']`
Examples: `9000s`, `7200`, `23s`, `9005.235s`

To remove the limit, use `0s`.

```
get_prove_per_property_time_limit
```

No arguments

Default

By default, the per property proof time limit is `1s`.

Return

`get_prove_per_property_time_limit` returns the value of the variable `prove_per_property_time_limit`.

Examples

```
% set_prove_per_property_time_limit 5m  
% get_prove_per_property_time_limit
```

See Also

[set cmd time limit](#)
[set elab time limit](#)
[set prove_per_property_time_limit_factor](#)
[set prove time limit](#)

prove_per_property_time_limit_factor

Commands

`set_prove_per_property_time_limit_factor`
`get_prove_per_property_time_limit_factor`

Description

Use the `set_prove_per_property_time_limit_factor` command to set the common ratio used for a geometric progression of `prove_per_property_time_limit` or to turn off geometric progression by setting the factor to 0. The effective `per_property_time_limit` will be the following:

`per_property_time_limit * factor ^ scan`

In this case, `scan` starts at zero and increments after each complete sweep over the properties in the current proof.

This command is useful for allowing individual engine modes within a proof to quickly prove simple properties while giving harder properties sufficient time to prove. The setting does not apply to inherently parallel modes like H, J, or L on safety properties.

At the boundary between one scan and the next, a proof message is issued indicating the effective `per_property_time_limit` for the coming scan.

Use the `get_prove_per_property_time_limit_factor` command to get the value of the variable `prove_per_property_time_limit_factor`.

JasperGold Apps Command Reference Manual

Configuration Commands

Syntax

```
set_prove_per_property_time_limit_factor <factor>
```

factor Use the specified factor to generate the geometric progression.

Note:

- To disable geometric progression, use 0.
 - The tool automatically disables geometric progression when the number of remaining properties is less than or equal to the number of held licenses.
 - When geometric progression is automatically disabled, per-property time becomes unlimited, that is, “0s”.
-

```
get_prove_per_property_time_limit_factor
```

No arguments

Default

By default, the factor is 10.

Return

`get_prove_per_property_time_limit_factor` returns the value of the variable `prove_per_property_time_limit_factor`.

Examples

```
% set_prove_per_property_time_limit 1s
% set_prove_per_property_time_limit_factor 30
# effective time limits: 1s, 30s, 15m, 7.5h

% get_prove_per_property_time_limit_factor
```

See Also

[set_prove_per_property_time_limit](#)
[set_proofgrid_skip_abandoned](#)

prove_prefer_shortest

Commands

`set_prove_prefer_shortest`
`get_prove_prefer_shortest`

Description

Certain engines and certain proof settings make the proof generate non-minimal traces without strong min_length and max_length information. See “[get_property_info](#)” on page 651. Using these engines or settings can be important to get overall good proof performance. However, in some cases, that is not enough. Sometimes you will need a shortest trace or strong bounds info. `prove_prefer_shortest on` provides a way to use engines and settings freely to get good proof performance while still getting strong bounds.

Note: Only certain engines have the ability to continue working on a property after it has received the status `cex` or `covered`. To get value out of turning this variable on, you must include at least one such engine. The supported engines are:

- Engine Ht with `engineH_single_property false` (the default)
- Engine B with `engineB_single_property false`

Use `get_prove_prefer_shortest` to determine whether `prove` will continue searching for a shorter trace after identifying a non-minimal trace.

Syntax

`set_prove_prefer_shortest (on | off)`

on	Continue searching for a shorter trace after identifying a non-minimal trace.
----	---

off	Do not continue searching for a shorter trace after identifying a non-minimal trace. <i>This option is the default.</i>
-----	---

`get_prove_prefer_shortest`

No arguments

Default

By default, `set_prove_prefer_shortest` is off.

Return

`get_prove_prefer_shortest` returns the value of the variable `prove_prefer_shortest`.

Examples

```
% set_prove_prefer_shortest on  
% get_prove_prefer_shortest
```

See Also

[prove](#)

prove_report_mem

Commands

set_prove_report_mem
get_prove_report_mem

Description

Use the `set_prove_report_mem` command to turn on or off the generation of memory consumption proof messages in the *Proof Messages* tab.

Note: The numbers given correspond to the Resident Set Size reported by the operating system and may be misleading.

Use `get_prove_report_mem` to determine whether the automatic generation of memory consumption messages is `on` or `off`.

Syntax

set_prove_report_mem (on off)
on Turn on the automatic generation of memory consumption messages.
off Turn off the automatic generation of memory consumption messages. <i>This option is the default.</i>
get_prove_report_mem
No arguments

Default

By default, the automatic generation of memory consumption messages is `off`.

Return

`get_prove_report_mem` returns the value of the variable `prove_report_mem`.

Examples

% set_prove_report_mem on

JasperGold Apps Command Reference Manual

Configuration Commands

% get_prove_report_mem

See Also

No related commands

prove_report_period

Commands

`set_prove_report_period`
`get_prove_report_period`

Description

Use the `set_prove_report_period` command to control the period with which the engine jobs receive requests for sampling their memory consumption. The argument corresponds to the number of seconds between consecutive requests, and it must be a positive integer.

Note: If you change this value while proof jobs are running in the background, the new value will only be effective after the next sample request.

Use `get_prove_report_period` to get the current value of the period used for requesting the memory reports from the proof jobs.

Syntax

```
set_prove_report_period <period>
```

`period` Set the sampling period for requesting memory reports from proof processes.

`get_prove_report_period`

No arguments

Default

By default, the period used for requesting the memory reports from the proof jobs is 10 seconds.

Return

`get_prove_report_period` returns the value of the variable `prove_report_period`.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_prove_report_period 15  
% get_prove_report_period
```

See Also

No related commands

prove_start_timer_on_engine_start

Commands

set_prove_start_timer_on_engine_start
get_prove_start_timer_on_engine_start

Description

Use the `set_prove_start_timer_on_engine_start` command to make the `prove` command timer start counting only after at least one engine has started.

Use `get_prove_start_timer_on_engine_start` to report whether the timer starts only after the first proof job starts.

Syntax

`set_prove_start_timer_on_engine_start (on | off)`

on	Start <code>prove</code> command timer only after the first proof job starts. <i>This option is the default.</i>
----	--

off	Do not wait for an engine to start.
-----	-------------------------------------

`get_prove_start_timer_on_engine_start`

No arguments

Default

By default, the `prove` command timer starts only after the first proof job starts.

Return

`get_prove_start_timer_on_engine_start` returns the value of the variable `prove_start_timer_on_engine_start`.

Examples

```
% set_prove_start_timer_on_engine_start on
% get_prove_start_timer_on_engine_start
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[set_engine_job_timers](#)

[set_prove_per_property_time_limit](#)

[set_prove_time_limit](#)

prove_time_limit

Commands

`set_prove_time_limit`
`get_prove_time_limit`

Description

Use `set_prove_time_limit` to set the time limit for the following commands:

- `check_arch -prove`
- `check_conn -prove`
- `check_csr -prove`
- `check_sec -prove`
- `check_spv -prove`
- `check_xprop -prove`
- `prove`
- `visualize`
- `check_assumptions`

The time excludes preprocessing steps such as reset analysis, which is performed during the first `prove` but not during subsequent `prove` calls. However, it includes the time spent waiting in batch queues (see “[set_proofgrid_mode](#)” on page 1138).

By default, the tool measures the time limit by wall clock time. To change the default, see “[set_engine_job_timers](#)” on page 1007.

Note: The `set_cmd_time_limit` command has no effect on the commands that honor `set_prove_time_limit`.

Use the `get_prove_time_limit` command to display the maximum time the tool can spend on commands listed above.

Syntax

```
set_prove_time_limit <time_limit>
```

time_limit Use the specified time limit for the prove and visualize commands.

The tool supports three formats for the time limit:

- ISO8601: hh' : ' [mm[' : ' [ss['.' [ffff]]]]]
Examples: 2:30, 2:, 0:0:23, 2:30:05.235
- JasperGold: [dd'd'] [hh'h'] [mm'm'] [ss[.ffff]'s']
Examples: 2h30m, 2h, 23s, 2h30m05.235s
- Seconds: sssss[.ffff] ['s']
Examples: 9000s, 7200, 23s, 9005.235s

To remove the limit, use 0s.

```
get_prove_time_limit
```

No arguments

Default

By default, the proof and Visualize time limit is 86400s (24h).

Return

get_prove_time_limit returns the value of the variable prove_time_limit.

Examples

```
% set_prove_time_limit 2h  
% get_prove_time_limit
```

See Also

[get_probe_start_timer_on_engine_start](#)
[set_cmd_time_limit](#)
[set_elab_time_limit](#)
[set_engine_job_timers](#)
[set_probe_per_property_time_limit](#)

JasperGold Apps Command Reference Manual
Configuration Commands

[set prove start timer on engine start](#)

prove_verbosity

Commands

set_probe_verbosity
get_probe_verbosity

Description

Use the `set_probe_verbosity` command to control the amount of information printed to the log file, the GUI *Console* pane, and the GUI *Proof messages* pane.

Use the `get_probe_verbosity` command to return the value of `probe_verbosity`.

Syntax

```
set_probe_verbosity <0-11>
```

<i>N</i>	Use the specified verbosity level for the <code>probe</code> command. 0 means silent. Use a higher value for more output as follows:
	<ul style="list-style-type: none">1 – Adds IPF029 assigning name to proof target2 – Adds IPF031 proof settings3 – Adds IPF035 task list message4 – Adds proof summary and generic proof error messages5 – Adds the most basic, high-level proof message tab messages6 – Adds proof simplification and brief trace attempt messages (default verbosity)7 – Adds most low-level proof message tab messages8 – Adds trace attempt time-out and brief size messages9 – Adds fine-grained design size messages and engine L- and G-specific messages11 – Adds X transform messages

JasperGold Apps Command Reference Manual

Configuration Commands

get_prove_verbosity

No arguments

Default

By default, the verbosity level for the prove command is 6.

Return

get_prove_verbosity returns the value of the variable prove_verbosity.

Examples

```
% set_prove_verbosity 0  
% get_prove_verbosity
```

See Also

[prove](#)

proven_directive

Commands

set_proven_directive
get_proven_directive

Description

By default, the tool uses proven directives as assumptions. Use the `set_proven_directive` command to turn off this feature or turn it back on.

Use the `get_proven_directive` command to display the current setting for using proven directives as assumptions. This command is context-sensitive; that is, it applies to the current task.

Syntax

set_proven_directive (true false)
true Use proven directives as assumptions. <i>This option is the default.</i>
false Do not use proven directives as assumptions.
get_proven_directive
No arguments

Default

By default, the tool uses proven directives as assumptions (`set_proven_directive true`).

Return

`get_proven_directive` returns the value of the variable `proven_directive`.

Examples

```
% set_proven_directive false
% get_proven_directive
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[prove](#)

reset_trace_inputs

Commands

`set_reset_trace_inputs`
`get_reset_trace_inputs`

Description

Use `set_reset_trace_inputs` to save input values from the saved reset trace that comes from a trace file.

By running this command, you will be able to retrieve these values with the `get_reset_info` command.



- By default, saving input values is not enabled.
- You must use this command before specifying the global reset.
- This feature applies to reset specified with `reset -vcd`, `reset -fsdb`, and `reset -shm` only.
- This command affects performance; that is, reading the trace file will be slower and will use more memory.

Use `get_reset_trace_inputs` to report whether saving input values in the saved reset trace from a VCD or FSDB file is enabled.

Syntax

```
set_reset_trace_inputs (true | false)
```

<code>true</code>	Save input values from the saved reset trace that comes from a VCD or FSDB file.
<code>false</code>	Do not save input values from the saved reset trace that comes from a VCD or FSDB file. <i>This option is the default.</i>

JasperGold Apps Command Reference Manual

Configuration Commands

get_reset_trace_inputs

No arguments

Return

Returns the value of set_reset_trace_inputs.

Examples

```
% set_reset_trace_inputs true  
% analyze -verilog my_design.v  
% elaborate -top my_design  
% clock clk  
% reset -init_state reset.vcd  
% get_reset_info -get_value input_signal_name cycle  
  
% get_reset_trace_inputs
```

See Also

[get_reset_info](#)

reset_vcd_fsdb_force_split_scope

Commands

set_reset_vcd_fsdb_force_split_scope
get_reset_vcd_fsdb_force_split_scope

Description

With `set_reset_vcd_fsdb_force_split_scope true`, JasperGold Apps recognizes the dot character (.) for splitting scopes instead of using synthesis information to determine the scopes of a signal. Use this command to fix potential differences in how JasperGold Apps splits the scope of signals while reading VCD, FSDB, or SHM files compared to other EDA tools.

Note: This command may affect the number of signals recognized in trace files and, as a consequence, may affect the reset values extracted for these signals.

Use the command `get_reset_vcd_fsdb_force_split_scope` to learn whether the tool will recognize the dot character for splitting scopes (`true`) or rely on synthesis information to determine the scopes of a signal (`false`).

Syntax

`set_reset_vcd_fsdb_force_split_scope (true | false)`

`true` Recognize the dot character (.) for splitting scopes instead of using synthesis information to determine the scopes of a signal.

`false` Use synthesis information to determine the scopes of a signal. *This option is the default.*

`get_reset_vcd_fsdb_force_split_scope`

No arguments

Return

The command `get_reset_vcd_fsdb_force_split_scope` returns `true` or `false`.

Examples

```
% set_reset_vcd_fsdb_force_split_scope true
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_reset_vcd_fsdb_force_split_scope
```

See Also

No related commands

scan_constant_relations_seeker

Commands

set_scan_constant_relations_seeker
get_scan_constant_relations_seeker

Description

Use the `set_scan_constant_relations_seeker` command to enable or disable the generation of cross-segment constant properties during property synthesis. The BPS App extracts constant relation properties over multiple segments. The ability to correlate a single signal's stable values with other stable signals in a design enables the extraction of valid register configurations from simulation traces. Cross-segment properties can be onehot, implication, equivalence, or mutex.

Note:

- Enable the constant seeker to generate constant relation properties. See [“set scan seek mode” on page 1211](#)
- You must import and scan a minimum of 5 segments for this feature to function.

Use `get_scan_constant_relations_seeker` to determine if cross segment constant properties will be generated.

Syntax

`set_scan_constant_relations_seeker (on | off)`

`on` Enable the generation of cross-segment constant properties during property synthesis.

`off` Disable the generation of cross-segment constant properties during property synthesis. *This option is the default.*

`get_scan_constant_relations_seeker`

No arguments

Return

`get_scan_constant_relations_seeker` returns the value of the variable `scan_constant_relations_seeker`.

Examples

```
% set_scan_constant_relations_seeker on  
% get_scan_constant_relations_seeker
```

See Also

[scan](#)

[set_scan_seek_mode](#)

scan_default_assume_representation_policy

Commands

```
set_scan_default_assume_representation_policy  
get_scan_default_assume_representation_policy
```

Description

Use the `set_scan_default_assume_representation_policy` command to specify the default representation for assumptions generated by the BPS App.

Use `get_scan_default_assume_representation_policy` to get the default representation for generated assumptions.

Syntax

<code>set_scan_default_assume_representation_policy (free_variables primary_inputs module_inputs)</code>	
<code>free_variables</code>	Base representation on primary inputs, outputs of black-boxed modules, and environmental stopats as free variables. <i>This option is the default.</i>
<code>primary_inputs</code>	Base representation on primary inputs only.
<code>module_inputs</code>	Base representation on all input ports in the design.
<code>get_scan_default_assume_representation_policy</code>	
No arguments	

Return

`get_scan_default_assume_representation_policy` returns the value of the variable `scan_default_assume_representation_policy`.

Examples

```
% set_scan_default_assume_representation_policy primary_inputs  
% get_scan_default_assume_representation_policy
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[check_bps](#)

[scan](#)

[set_scan_default_property_representation](#)

scan_default_property_representation

Commands

set_scan_default_property_representation
get_scan_default_property_representation

Description

Use the `set_scan_default_property_representation` command to set the default property representation for behavioral property synthesis.

When you use the BPS App `scan` command, the tool synthesizes properties with multiple representations (assertion, assumption, or cover), and then selects one of these representations as the initial representation. If design and waveform information is available, the tool uses it to determine the initial representation. However, if this information is not available, then the BPS App applies a default representation. Use the `scan_default_property_representation` variable to set the default representation for property synthesis. The default is `cover`.

Note: This setting has no effect in any of the following situations:

- The BPS App has enough information to make non-default decisions
- The chosen representation is unavailable
- The property was synthesized before using this command

Use `get_scan_default_property_representation` to get the default property representation for behavioral property synthesis.

Syntax

set_scan_default_property_representation (assert assume cover auto)
assert Set the default property representation for <code>scan</code> to assert.
assume Set the default property representation for <code>scan</code> to assume.
cover Set the default property representation for <code>scan</code> to cover. <i>This option is the default.</i>
auto Automatically detect the default property representation for behavioral property synthesis.

JasperGold Apps Command Reference Manual

Configuration Commands

get_scan_default_property_representation

No arguments

Return

get_scan_default_property_representation returns the value of the variable scan_default_property_representation.

Examples

```
% set_scan_default_property_representation auto  
% get_scan_default_property_representation
```

See Also

[check_bps](#)
[scan](#)

scan_no_sim_max_covers

Commands

`set_scan_no_sim_max_covers`
`get_scan_no_sim_max_covers`

Description

Use `set_scan_no_sim_max_covers` to define the maximum number of covers the BPS App will try to use during `scan -no_sim` or `auto_syn` flows.

Use `get_scan_no_sim_max_covers` to get the current maximum number of covers used by the BPS App with `scan -no_sim` or `auto_syn`.

Syntax

<code>set_scan_no_sim_max_covers <N></code>
<code>N</code> Set the maximum number of covers used during <code>scan -no_sim</code> or <code>auto_syn</code> flows.
<code>get_scan_no_sim_max_covers</code>
No arguments

Return

`get_scan_no_sim_max_covers` returns the value of the variable `scan_no_sim_max_covers`.

Examples

```
% set_scan_no_sim_max_covers 20  
% get_scan_no_sim_max_covers
```

See Also

[auto_syn](#)
[scan](#)

scan_per_scope_time_limit

Commands

`set_scan_per_scope_time_limit`
`get_scan_per_scope_time_limit`

Description

Use `set_scan_per_scope_time_limit` to set a time limit for scanning each scope. If the scan operation times out while scanning a waveform, the tool discards all properties extracted from that waveform and leaves all properties extracted up to that waveform untouched. The tool reports in which waveform and scope the time limit occurred, and a re-scan starts from that waveform.

For example, if you want to scan three scopes with the `scan -trace` command and you want the command to complete within 30 seconds, then set the `scan_per_scope_time_limit` to 10s. The tool then allocates no more than 10 seconds per scope and up to 30 seconds for the `scan -trace` command to complete.

Use `get_scan_per_scope_time_limit` to determine the time limit for scanning each scope.

Syntax

`set_scan_per_scope_time_limit <time_limit>`

`time_limit` Use the specified limit for the scan of each scope.
Specify the time limit as an integer followed by s, m, or h (no spaces).
■ s = seconds
■ m = minutes
■ h = hours

JasperGold Apps commonly recognizes 0s as “unlimited.” In the case of this command, 0s sets the highest supported limit, which is 10000 seconds. *The default is 0s.*

`get_scan_per_scope_time_limit`

No arguments

Return

`get_scan_per_scope_time_limit` returns the value of the variable `scan_per_scope_time_limit`.

Examples

```
% set_scan_per_scope_time_limit 2m  
% get_scan_per_scope_time_limit
```

See Also

[scan](#)
[scope](#)
[waveform](#)

scan_seek_depth

Commands

`set_scan_seek_depth`
`get_scan_seek_depth`

Description

Use `set_scan_seek_depth` to select optimizations tuned for seekers where calculating depth requires additional computational effort and may produce additional noise. Specify a depth value `n` to synthesize temporal relations up to `n` cycles deep.



- This property does not apply to all property synthesis algorithms.
- Specific algorithms, for example, handshake algorithms, may generate properties of greater temporal depth than specified by this command.
- Specify a depth value of zero to configure relevant engines to synthesize single-cycle properties.

Use `get_scan_seek_depth` to determine the maximum temporal depth for property synthesis.

Syntax

```
set_scan_seek_depth <depth>
```

`depth` Set maximum temporal depth for property synthesis. *The default depth is 3.*

```
get_scan_seek_depth
```

No arguments

Return

`get_scan_seek_depth` returns the value of the variable `scan_seek_depth`.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_scan_seek_depth 5  
% get_scan_seek_depth
```

See Also

[scan](#)

scan_seek_mode

Commands

`set_scan_seek_mode`
`get_scan_seek_mode`

Description

Use `set_scan_seek_mode` to optimize property synthesis algorithms for different types of designs and requirements. Specify a single seeker or use a comma-separated list to specify multiple seekers.



Important

- Use `all` to enable all seekers.
- Designating a specific seeker does not imply that specific types of properties will be synthesized. Results depend on the design and waveforms. However, excluding a specific seeker ensures that specific property types will *not* be synthesized.
- While the property examples below use the `assert SVA` keyword, all seekers may generate any property type (assertion, assumption, or cover).
- `set_scan_seek_mode` accepts a Tcl list.

Use `get_scan_seek_mode` to determine what seeker or seekers are enabled.

Syntax

<code>set_scan_seek_mode</code>	
	<code>([all] [constant] [equivalence] [handshake] [implication]</code> <code>[mutex] [onehot] [strobe] [template])</code>
<code>all</code>	Use all available seekers.
<code>constant</code>	Use the constant properties seeker to analyze scalars and vectors for single constant values or a range of constant values. With this argument, the tool synthesizes properties such as the following: <code>assert { stuck_at(a, 0) }</code> <code>assert { stuck_at(vec, {40'h1,40'h6-40'h8,40'ha,40'hffff}) }</code>

JasperGold Apps Command Reference Manual

Configuration Commands

equivalence	<p>Use the equivalence properties seeker to analyze equivalence relations between scalars.</p> <p>With this argument, the tool synthesizes properties such as the following:</p> <pre>assert { \$equal(a, b, ~c, d) }</pre>
handshake	<p>Use the handshake properties seeker to analyze interactions between control signals and focus on request-response protocols.</p> <p>With this argument, the tool synthesizes properties such as the following:</p> <pre>assert { \$handshake(req, ack, min_delay, max_delay, protocol_type, parameter) }</pre> <p>Note: Parameter is associated with the protocol type as follows:</p> <ul style="list-style-type: none">■ If <code>protocol_type</code> equals 2, protocol parameter is <code>deassert_count</code>, which defines the maximum number of clock cycles after response that the request remained active (<code>deassert_count >= 0</code>).■ If <code>protocol_type</code> equals 3, protocol parameter is <code>max_burst</code>, which defines the maximum number of times <code>ack</code> became active within a back-to-back request (<code>max_burst > 0</code>). <p>Note: The <code>max_burst</code> parameter is an informative parameter. No assertion is generated to check this parameter.</p> <ul style="list-style-type: none">■ If <code>protocol_type</code> equals 0, protocol parameter is zero. <p>For additional information, see the Appendix, “Handshake Protocols,” in the <i>Behavioral Property Synthesis App User’s Guide</i>. This guide is available from Cadence Online Support (support.cadence.com).</p>
implication	<p>Use the implication properties seeker to analyze interactions between scalars.</p> <p>With this argument, the tool synthesizes properties such as the following:</p> <pre>assert { expr1 -> ##n expr2[*m] }</pre>
mutex	<p>Use the mutex properties seeker to analyze interactions between control signals and focus on mutually exclusive values.</p> <p>With this argument, the tool synthesizes properties such as the following:</p> <pre>assert { \$mutex(a, b, c) }</pre>

JasperGold Apps Command Reference Manual

Configuration Commands

onehot	<p>Use the onehot properties seeker to analyze vectors and focus on various formats of onehot encoding.</p> <p>With this argument, the tool synthesizes properties such as the following:</p> <pre>assert { \$onehot0(vec1) } assert { \$onehot(~vec2) }</pre>
strobe	<p>Use the strobe properties seeker to analyze single cycle signals and find signals that have an activity duration of one cycle.</p> <p>With this argument, the tool synthesizes properties such as the following:</p> <pre>assert { a => ~a }</pre>
template	<p>Use the template properties seeker to enable extraction of user-defined properties from JasperGold Apps property template files or libraries. With templates, you can customize constraints to minimize the search and return more relevant results.</p> <p>With this argument, the tool searches for invariants specified in a given template. For example, the template specification shown below synthesizes the listed assertion.</p> <p>Template Specification:</p> <pre>property template<number N> single_cycle_grant; (gnt == N => gnt == 0); keep N.value != 0; endproperty</pre> <p>Assertion Synthesized:</p> <pre>assert (gnt == 0x1 => gnt == 0)</pre>
get_scan_seek_mode	
No arguments	

Return

`get_scan_seek_mode` returns the values of the variable `scan_seek_mode`.

Examples

```
% set_scan_seek_mode implication
% set_scan_seek_mode all
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% scan -trace  
  
% set_scan_seek_mode {mutex onehot}  
% scan -trace -fsdb my_trace.fsdb -hier_path sys.core0 -start_time 350000  
  
% get_scan_seek_mode
```

See Also

[check_bps](#)
[scan](#)
[scope](#)
[set_export_sva_handshake_delay_threshold](#)
[set_scan_trace_stuck_at_values_limit](#)
[waveform](#)

scan_trace_stuck_at_values_limit

Command

```
set_scan_trace_stuck_at_values_limit  
get_scan_trace_stuck_at_values_limit
```

Description

Use `set_scan_trace_stuck_at_values_limit` to set a limit for the number of values shown in the `stuck_at` property. If the limit is reached, the tool compacts the representation of the property, showing only the maximum constant value and the minimum constant value.



- The modified representation will take place after a new scan.
- The values limit is an integer larger than one.

In the following example, the `stuck_at` property has five values. After setting `set_scan_trace_stuck_at_values_limit` to three and using the `scan` command, the property representation will be modified.

```
stuck_at(vec, {40'h6-40'h9,40'hfffffff})
```

```
set_scan_trace_stuck_at_values_limit 3  
scan -trace
```

```
stuck_at(vec, >=40'h6, <=40'hfffffff)
```

Use `get_scan_trace_stuck_at_values_limit` to get the `stuck_at` values limit.

Syntax

```
set_scan_trace_stuck_at_values_limit <N>
```

<i>N</i>	Set the specified limit for the number of values shown in the <code>stuck_at</code> property. The default is 25.
----------	---

JasperGold Apps Command Reference Manual

Configuration Commands

get_scan_trace_stuck_at_values_limit

No arguments

Return

get_scan_trace_stuck_at_values_limit returns the value of the variable scan_trace_stuck_at_values_limit.

Examples

```
% set_scan_trace_stuck_at_values_limit 5  
% get_scan_trace_stuck_at_values_limit
```

See Also

[scan](#)

[set_scan_seek_mode](#)

scan_trace_witness_information

Command

`set_scan_trace_witness_information`
`get_scan_trace_witness_information`

Description

Use `set_scan_trace_witness_information` to enable or disable witness information collection during property synthesis.

When you use the BPS App `scan` command, the tool synthesizes properties from waveforms and stores information about each property. You may want to debug properties by looking at key design behaviors demonstrating these properties. If `set_scan_trace_witness_information` is on, which is the default, the BPS App stores key trace locations for future analysis.

Note:

- Trace locations are approximate. The tool does not point to an exact simulation time step but to a simulation start and end time pair.
- Expect minor run-time and memory cost when the `scan_trace_witness_information` variable is on.

Use `get_scan_trace_witness_information` to report whether witness information collection is enabled or disabled.

Syntax

`set_scan_trace_witness_information (on | off)`

`on` Enable witness information collection during scan. *This option is the default.*

`off` Disable witness information collection during scan.

`get_scan_trace_witness_information`

No arguments

Return

`get_scan_trace_witness_information` returns the value of the variable `scan_trace_witness_information`.

Examples

```
% set_scan_trace_witness_information off  
% get_scan_trace_witness_information
```

See Also

[check_bps](#)
[scan](#)

scan_verbosity

Command

`set_scan_verbosity`
`get_scan_verbosity`

Description

Use `set_scan_verbosity` to control the amount of information printed to the log and console when scanning multiple files. The `scan_verbosity` variable accepts four verbosity levels as described below.

Use `get_scan_verbosity` to determine how much information will be printed.

Syntax

`set_scan_verbosity (N)`

`N` Specify the verbosity level for the `scan -trace` command:

- 0 – Silent
- 1 – Include information on POIs
- 2 – Include information on POIs and waveforms (*This option is the default.*)
- 3 – Include information on POIs, waveforms, and seekers
- 4 – Include information on POIs, waveforms, seekers, and time

`get_scan_verbosity`

No arguments

Return

`get_scan_verbosity` returns the value of the variable `scan_verbosity`

Examples

```
% set_scan_verbosity 3  
% get_scan_verbosity
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[scan](#)

schematic_viewer_command

Commands

set_schematic_viewer_command
get_schematic_viewer_command

Description

Use `set_schematic_viewer_command` to change the (system) command used by JasperGold Apps to launch the schematic viewer or to override the SCHEMATICVIEWER environment variable, which governs both the schematic viewer and standalone waveform viewer. The command can be an absolute or relative path. If it is relative, the command must be in the search path for commands (that is, the PATH environment variable). The specified command must launch the Synopsys Verdi™ viewer. No other viewers are supported by this command.

Mandatory requirement: Before starting your session, add the Synopsys tool to your PATH.

Use `get_schematic_viewer_command` to display the current value of the command used to launch the schematic viewer.

Syntax

<code>set_schematic_viewer_command <command></code>
<code>command</code> Use the specified (system) command to launch the schematic viewer.
<code>get_schematic_viewer_command</code>
No arguments

Default

By default, JasperGold Apps launches Verdi (`set_schematic_viewer_command verdi`).

Return

`get_schematic_viewer_command` returns the value of the `schematic_viewer_command` variable.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% get_schematic_viewer_command
```

See Also

[show_schematic](#)
[show_waveform](#)

schematic_viewer_connection_timeout

Commands

`set_schematic_viewer_connection_timeout`
`get_schematic_viewer_connection_timeout`

Description

Use the `set_schematic_viewer_connection_timeout` command to change the length of time JasperGold Apps waits for a connection to the schematic viewer. If the tool does not establish a connection during the specified time, the command trying to establish a connection will fail.

Use the `get_schematic_viewer_connection_timeout` command to display the current setting for the schematic viewer connection timeout.

Syntax

`set_schematic_viewer_connection_timeout`

`timeout` Wait the specified amount of time for a connection to the schematic viewer.

The tool supports three formats for the time limit:

- ISO8601: `hh' : ' [mm' : ' [ss['.' [fff]]]]]`
Examples: `2:30`, `2:`, `0:0:23`, `2:30:05.235`
- JasperGold: `[dd'd'] [hh'h'] [mm'm'] [ss[.fff]'s']`
Examples: `2h30m`, `2h`, `23s`, `2h30m05.235s`
- Seconds: `ssss[.fff]['s']`
Examples: `9000s`, `7200`, `23s`, `9005.235s`

`get_schematic_viewer_connection_timeout`

No arguments

Default

By default, attempts to connect to the schematic viewer timeout after 30 seconds.

JasperGold Apps Command Reference Manual

Configuration Commands

Return

`get_schematic_viewer_connection_timeout` returns the value of the `schematic_viewer_connection_timeout` variable in seconds.

Examples

```
% set_schematic_viewer_connection_timeout 40  
% get_schematic_viewer_connection_timeout
```

See Also

[show_schematic](#)

schematic_viewer_design_script

Commands

`set_schematic_viewer_design_script`
`get_schematic_viewer_design_script`

Description

Under normal operating conditions, the schematic viewer automatically imports your design. Use the `set_schematic_viewer_design_script` command when the automatic process is not satisfactory. This command specifies a Tcl script that imports the design in Verdi. The script must return the value “1” if successful.

Note: The schematic viewer application runs the specified script; therefore, all commands in the script must be specific to the Synopsys schematic viewer you are using. (Do not include any JasperGold-specific commands.)

Use the `get_schematic_viewer_design_script` command to display the path to the script used to import the design in the schematic viewer.

Syntax

```
set_schematic_viewer_design_script <script>
```

<code>script</code>	A Tcl script that imports the design in the schematic viewer.
---------------------	---

<code>get_schematic_viewer_design_script</code>	
---	--

No arguments	
--------------	--

Default

None

Return

`get_schematic_viewer_design_script` returns the value of the `schematic_viewer_design_script` variable.

Examples

```
% set_schematic_viewer_design_script load_my_design_in_verdi.tcl  
% get_schematic_viewer_design_script
```

See Also

[show_schematic](#)

scope_clock

Commands

set_scope_clock
get_scope_clock

Description

Use the `set_scope_clock` command to change how the Behavioral Property Synthesis App sets the clock for all POIs extracted after this command is issued. For additional information on the clock detection flow, see Section 2.2.3 of the *Behavioral Property Synthesis App User's Guide*. This manual is available from Cadence Online Support (support.cadence.com).

Use `get_scope_clock` to identify the clock specified with `set_scope_clock`.

Syntax

<code>set_scope_clock <clock></code>
<code>clock</code> Sets the overwrite clock for all POIs extracted after this command is issued.
<code>get_scope_clock</code>
No arguments

Default

There is no default.

Return

`get_scope_clock` returns the overwrite clock configuration.

Examples

```
% set_scope_clock clk
% get_scope_clock
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[set_scope_clock_edge](#)
[set_scope_default_clock](#)
[set_scope_default_clock_edge](#)

scope_clock_edge

Commands

set_scope_clock_edge
get_scope_clock_edge

Description

Use the `set_scope_clock_edge` command to change how the Behavioral Property Synthesis App sets the clock edge for all POIs extracted after this command is issued. For additional information on the clock detection flow, see Section 2.2.3 of the *Behavioral Property Synthesis App User's Guide*. This manual is available from Cadence Online Support (support.cadence.com).

Use `get_scope_clock_edge` to determine whether POI sampling is at the posedge, the negedge, both edges, or auto detected.

Syntax

`set_scope_clock_edge (posedge | negedge | both_edges | auto_detect)`

`posedge` Sample POIs at the posedge.

`negedge` Sample POIs at the negedge.

`both_edges` Sample POIs at each clock change.

`auto_detect` Detect sample automatically based on waveform analysis.

`get_scope_clock_edge`

No arguments

Default

There is no default.

Return

`get_scope_clock_edge` returns the overwrite clock edge configuration.

Examples

```
% set_scope_clock_edge both_edges  
% get_scope_clock_edge
```

See Also

[set_scope_clock](#)
[set_scope_default_clock](#)
[set_scope_default_clock_edge](#)

scope_default_active_phase

Commands

`set_scope_default_active_phase`
`get_scope_default_active_phase`

Description

Use the `set_scope_default_active_phase` command to change how the BPS App sets the active phase for POI signals. Some property synthesis engines use the signal active phase to enhance property synthesis results. The signal active phase may have a value of `high`, `low`, or `auto_detect`.

The default active phase is applied to POI signals during execution of the `scope -extract` command. After `scope -extract`, the active phase may be set using the commands `scope -set_active_high` and `scope -set_active_low`. When the active phase value is `auto_detect`, the BPS App uses the signal waveform values to choose a high or low active phase.

Use `get_scope_default_active_phase` to determine the current default active phase configuration.

Syntax

<code>set_scope_default_active_phase (high low auto_detect)</code>	
<code>high</code>	Set the active phase for POI signals to high.
<code>low</code>	Set the active phase for POI signals to low.
<code>auto_detect</code>	Use the signal waveform value to choose a high or low active phase. <i>This option is the default.</i>
<code>get_scope_default_active_phase</code>	
No arguments	

Default

By default, the tool automatically detects the active phase.

JasperGold Apps Command Reference Manual

Configuration Commands

Return

`get_scope_default_active_phase` returns the default active phase configuration.

Examples

```
% set_scope_default_active_phase high  
% get_scope_default_active_phase
```

See Also

[scope](#)

scope_default_clock

Commands

set_scope_default_clock
get_scope_default_clock

Description

Use the `set_scope_default_clock` command to provide a default clock should the Behavioral Property Synthesis App be unable to determine the clock for a POI during extraction. For additional information on the clock detection flow, see Section 2.2.3 of the *Behavioral Property Synthesis App User's Guide*. This manual is available from Cadence Online Support (support.cadence.com).

Use `get_scope_default_clock` to determine the default clock configuration.

Syntax

set_scope_default_clock <clock>	
<code>clock</code>	Set the default clock.
<hr/>	
get_scope_default_clock	
<hr/>	
No arguments	

Default

There is no default.

Return

`get_scope_default_clock` returns the default clock configuration.

Examples

```
% set_scope_default_clock clk
% get_scope_default_clock
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[set_scope_clock](#)
[set_scope_clock_edge](#)
[set_scope_default_clock_edge](#)

scope_default_clock_edge

Command

set_scope_default_clock_edge
get_scope_default_clock_edge

Description

Use the `get_scope_default_clock_edge` command to provide a default clock edge should the Behavioral Property Synthesis App be unable to determine the clock edge for a POI during extraction. If the parameter is `true`, the POI will be sampled at the posedge; otherwise, it will be sampled at the negedge. For additional information on the clock detection flow, see Section 2.2.3 of the *Behavioral Property Synthesis App User's Guide*. This manual is available from Cadence Online Support (support.cadence.com).

Use `get_scope_default_clock_edge` to determine whether the POI will be sampled at the posedge or negedge.

Syntax

set_scope_default_clock_edge (true | false)

true Sample POI at the posedge.

false Sample POI at the negedge.

get_scope_default_clock_edge

No arguments

Default

There is no default.

Return

`get_scope_default_clock_edge` returns the default clock edge configuration.

Examples

```
% set_scope_default_clock_edge true
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_scope_default_clock_edge
```

See Also

[set_scope_clock](#)
[set_scope_clock_edge](#)
[set_scope_default_clock](#)

scope_extract_min_ports_threshold

Commands

`set_scope_extract_min_ports_threshold`
`get_scope_extract_min_ports_threshold`

Description

Use `set_scope_extract_min_ports_threshold` to limit the number of module interfaces extracted with `scope -extract module_interface`. When this variable is set, the tool extracts only those interfaces with port counts greater than or equal to the specified threshold (port count = number of interface signals).

Use `get_scope_extract_min_ports_threshold` to determine the minimum port count threshold for module interface extraction.

Syntax

<code>set_scope_extract_min_ports_threshold <threshold></code>
<code>threshold</code> Set the minimum port count threshold for <code>scope -extract module_interface</code> . <i>The default is 6.</i>
<code>get_scope_extract_min_ports_threshold</code>
No arguments

Default

The default value is 6.

Return

`get_scope_extract_min_ports_threshold` returns the minimum port count threshold for `scope -extract module_interface`.

Examples

```
% set_scope_extract_min_ports_threshold 10  
% get_scope_extract_min_ports_threshold
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[scope](#)

scope_signal_width_threshold

Commands

`set_scope_signal_width_threshold`
`get_scope_signal_width_threshold`

Description

Use `set_scope_signal_width_threshold` to set the maximum signal width threshold when extracting module, module interface, and user-defined POIs.

Use `get_scope_signal_width_threshold` to determine the maximum signal width threshold when extracting module, module interface, and user-defined POIs.

Syntax

<code>set_scope_signal_width_threshold <threshold></code>
<code>threshold</code> Set the maximum signal width threshold when extracting module, module interface, and user-defined POIs. <i>The default is 64.</i>
<code>get_scope_signal_width_threshold</code>
No arguments

Default

The default value is 64.

Return

`get_scope_signal_width_threshold` returns the value of the `scope_signal_width_threshold` variable.

Examples

```
% set_scope_signal_width_threshold 128  
% get_scope_signal_width_threshold
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[scope](#)

sec_auto_prove_prelqualification_efforts

Commands

`set_sec_auto_prove_prelqualification_efforts`
`get_sec_auto_prove_prelqualification_efforts`

Description

Use `set_sec_auto_prove_prelqualification_efforts` to tell the Sequential Equivalence Checking App autoprove how much effort to put into generating helper assertions that contribute to full convergence on verification targets.

Use `get_sec_auto_prove_prelqualification_efforts` to determine the current effort level setting for the autoprove prequalification process for the Sequential Equivalence Checking App.

Syntax

```
set_sec_auto_prove_prelqualification_efforts  
(normal | high | medium)
```

Configure the SEC autoprove prequalification efforts level.

```
get_sec_auto_prove_prelqualification_efforts
```

```
No arguments
```

Default

The default level is `normal`.

Return

`get_sec_auto_prove_prelqualification_efforts` returns the value of the `sec_auto_prove_prelqualification_efforts` variable.

Examples

```
% set_sec_auto_prove_prelqualification_efforts high  
% get_sec_auto_prove_prelqualification_efforts
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

No related commands

sec_autoprove

Commands

set_sec_autoprove
get_sec_autoprove

Description

Use this command to specify the proof flow SEC uses, autoprove or prove. With set_sec_autoprove false, check_sec -prove runs a low-level prove command rather than the default autoprove.

Use set_sec_autoprove to report which proof flow SEC will use, autoprove or prove.

Syntax

set_sec_autoprove (true | false)

true Use autoprove settings when running check_sec -prove. *This is the default.*

false Run a low-level prove command with no autoprove settings when running check_sec -prove.

get_sec_autoprove

No arguments

Default

By default, SEC uses autoprove settings when running check_sec -prove.

Return

get_sec_autoprove returns the value of the variable sec_autoprove.

Examples

```
% set_sec_autoprove false  
% get_sec_autoprove
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

No related commands

sec_autoprove_cutpoints_on_internal_signals

Commands

`set_sec_autoprove_cutpoints_on_internal_signals`
`get_sec_autoprove_cutpoints_on_internal_signals`

Description

Use `set_sec_autoprove_cutpoints_on_internal_signals` to add cutpoints on internal signals when running `check_sec -prove` with `-autoprove_mode sscp` or `-autoprove_mode dscp`.

Use `get_sec_autoprove_cutpoints_on_internal_signals` to report whether SEC will add cutpoints on internal signals when running `check_sec -prove` in autoprove mode.

Syntax

<code>set_sec_autoprove_cutpoints_on_internal_signals (true false)</code>	
<code>true</code>	Add cutpoints on internal signals when running <code>check_sec -prove</code> in autoprove mode.
<code>false</code>	Do not add cutpoints on internal signals when running <code>check_sec -prove</code> in autoprove mode. <i>This is the default.</i>
<code>get_sec_autoprove_cutpoints_on_internal_signals</code>	
No arguments	

Default

By default, SEC does not add cutpoints on internal signals when running `check_sec -prove` with `-autoprove_mode sscp` or `-autoprove_mode dscp`.

Return

`get_sec_autoprove_cutpoints_on_internal_signals` returns the value of the variable `sec_autoprove_cutpoints_on_internal_signals`.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_sec_autoprove_cutpoints_on_internal_signals true  
% get_sec_autoprove_cutpoints_on_internal_signals
```

See Also

No related commands

sec_autoprove_cutpoints_on_latches

Commands

`set_sec_autoprove_cutpoints_on_latches`
`get_sec_autoprove_cutpoints_on_latches`

Description

Use `set_sec_autoprove_cutpoints_on_latches` to add cutpoints on latches when running `check_sec -prove` with `-autoprove_mode sscp` or `-autoprove_mode dscp`.

Use `get_sec_autoprove_cutpoints_on_latches` to report whether SEC will add cutpoints on latches when running `check_sec -prove` in autoprove mode.

Syntax

<code>set_sec_autoprove_cutpoints_on_latches (true false)</code>	
<code>true</code>	Add cutpoints on latches when running <code>check_sec -prove</code> in autoprove mode.
<code>false</code>	Do not add cutpoints on latches when running <code>check_sec -prove</code> in autoprove mode. <i>This is the default.</i>
<code>get_sec_autoprove_cutpoints_on_latches</code>	
No arguments	

Default

By default, SEC does not add cutpoints on latches when running `check_sec -prove` with `-autoprove_mode sscp` or `-autoprove_mode dscp`.

Return

`get_sec_autoprove_cutpoints_on_latches` returns the value of the variable `sec_autoprove_cutpoints_on_latches`.

Examples

```
% set_sec_autoprove_cutpoints_on_latches true
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_sec_autoprove_cutpoints_on_latches
```

See Also

No related commands

sec_autoprove_max_number_of_cutpoints

Commands

`set_sec_autoprove_max_number_of_cutpoints`
`get_sec_autoprove_max_number_of_cutpoints`

Description

Use `set_sec_autoprove_max_number_of_cutpoints` to specify that you want the tool to automatically add no more than the specified number of cutpoints when running `check_sec -prove` with `-autoprove_mode sscp` or `-autoprove_mode dscp`.

Use `get_sec_autoprove_max_number_of_cutpoints` to report the maximum number of cutpoints SEC adds on signals when running `check_sec -prove` in autoprove mode.

Syntax

`set_sec_autoprove_max_number_of_cutpoints (true | false)`

`N` Add up to *N* cutpoints on signals when running `check_sec -prove` in autoprove mode. *The default is 10000.*

`get_sec_autoprove_max_number_of_cutpoints`

No arguments

Default

By default, SEC adds up to 10000 cutpoints on signals when running `check_sec -prove` in autoprove mode.

Return

`get_sec_autoprove_max_number_of_cutpoints` returns the value of the variable `sec_autoprove_max_number_of_cutpoints`.

Examples

```
% set_sec_autoprove_max_number_of_cutpoints 11000  
% get_sec_autoprove_max_number_of_cutpoints
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

No related commands

sec_autoprove_mode

Commands

`set_sec_autoprove_mode`
`get_sec_autoprove_mode`

Description

Use `set_sec_autoprove_mode` to run a `check_sec -prove` command with cutpoint refinement loop handling.

Use `get_sec_autoprove_mode` to report which mode of cutpoint refinement loop handling SEC uses with the `check_sec -prove` command.

Syntax

`set_sec_autoprove_mode (sscp | dscp | none)`

`sscp` Run the proof with single-sided cutpoints.

`dscp` Run the proof with double-sided cutpoints.

`none` Run the proof without cutpoints. *This is the default.*

`get_sec_autoprove_mode`

No arguments

Default

By default, SEC runs the proof without cutpoints.

Return

`get_sec_autoprove_mode` returns the value of the variable `sec_autoprove_mode`.

Examples

```
% set_sec_autoprove_mode sscp  
% get_sec_autoprove_mode
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

No related commands

sec_prove_assumption_lifting

Commands

set_sec_prove_assumption_lifting
get_sec_prove_assumption_lifting

Description

Use `set_sec_prove_assumption_lifting` to include only a relevant subset of assumptions that is sufficient for the proof using `check_sec -prove`.

Use `get_sec_prove_assumption_lifting` to report whether SEC will include only a relevant subset of assumptions sufficient for the proof.

Syntax

set_sec_prove_assumption_lifting (true false)
true Include only a relevant subset of assumptions sufficient for the proof.
false Include all assumptions. <i>This is the default.</i>
get_sec_prove_assumption_lifting
No arguments

Default

By default, SEC includes all assumptions
(`set_sec_prove_assumption_lifting=false`).

Return

`get_sec_prove_assumption_lifting` returns the value of the variable `sec_prove_assumption_lifting`.

Examples

```
% set_sec_prove_assumption_lifting true
% get_sec_prove_assumption_lifting
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

No related commands

sec_prove_levelize_helpers

Commands

`set_sec_prove_levelize_helpers`
`get_sec_prove_levelize_helpers`

Description

Use `set_sec_prove_levelize_helpers` to specify an iterative, level-by-level proof over the levels that were previously calculated with the `check_sec -prove` command.

Use `get_sec_prove_levelize_helpers` to report whether SEC will run a level-by-level proof.

Syntax

`set_sec_prove_levelize_helpers (true | false)`

`true` Specify a level-by-level SEC proof.

`false` Do not specify a level-by-level SEC proof. This *is the default*.

`get_sec_prove_levelize_helpers`

No arguments

Default

By default, SEC does not run a level-by-level proof over the levels that were previously calculated with the `check_sec -prove` command.

Return

`get_sec_prove_levelize_helpers` returns the value of the variable `sec_prove_levelize_helpers`.

Examples

```
% set_sec_prove_levelize_helpers true  
% get_sec_prove_levelize_helpers
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

No related commands

sec_prove_order_targets

Commands

`set_sec_prove_order_targets`
`get_sec_prove_order_targets`

Description

Use `set_sec_prove_order_targets` to specify that you want the `check_sec -prove` command to attempt to order the `prove` dispatch for the helper assertions based on internal ranking of what is best to start with rather than ordering them one by one. The latter can be a time-consuming process that results in a time-out on hard-to-prove helper assertions. With the `set_sec_prove_order_targets` mode, the easy-to-prove helper assertions will converge quickly, thus benefiting proof convergence.

Use `get_sec_prove_order_targets` to report whether SEC will attempt to order the `prove` dispatch for the helper assertions based on internal ranking.

Syntax

`set_sec_prove_order_targets (by_gates | by_flops | none)`

`by_gates` Attempt to order the `prove` dispatch for the helper assertions based on internal ranking by gates.

`by_flops` Attempt to order the `prove` dispatch for the helper assertions based on internal ranking by flops.

`none` Order the `prove` dispatch for the helper assertions one by one. *This is the default.*

`get_sec_prove_order_targets`

No arguments

Default

By default, SEC does not attempt to order the `prove` dispatch for the helper assertions.

Return

`get_sec_prove_order_targets` returns the value of the variable `sec_prove_order_targets`.

Examples

```
% set_sec_prove_order_targets by_gates  
% get_sec_prove_order_targets
```

See Also

No related commands

sec_prove_suppress_traces

Commands

set_sec_prove_suppress_traces
get_sec_prove_suppress_traces

Description

Use `set_sec_prove_suppress_traces` to suppress trace generation when using the `check_sec -prove` command.

Use `get_sec_prove_suppress_traces` to report whether SEC will suppress trace generation.

Syntax

set_sec_prove_suppress_traces (true false)
true Configure <code>check_sec -prove</code> to suppress trace generation.
false Do not configure <code>check_sec -prove</code> to suppress trace generation. This <i>is the default</i> .
get_sec_prove_suppress_traces
No arguments

Default

By default, SEC does not suppress trace generation with the `check_sec -prove` command.

Return

`get_sec_prove_suppress_traces` returns the value of the variable `sec_prove_suppress_traces`.

Examples

```
% set_sec_prove_suppress_traces true
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_sec_prove_suppress_traces
```

See Also

No related commands

sps_arithmetic_overflow_style

Commands

`set_sps_arithmetic_overflow_style`
`get_sps_arithmetic_overflow_style`

Description

Use `set_sps_arithmetic_overflow_style` to control whether SPS generates one overflow for the assignment that contains the arithmetic operation (statement style) or one for every arithmetic operation within the expression in the assignment statement (expression style).

Use `get_sps_arithmetic_overflow_style` to report whether SPS generates one overflow for the assignment that contains the arithmetic operation or one for every arithmetic operation within the expression in the assignment statement.

Syntax

`set_sps_arithmetic_overflow_style (statement | expression)`

<code>statement</code>	Generate one arithmetic overflow for the assignment that contains the arithmetic operation. <i>This is the default.</i> This checks overflow condition only on the top level of the assignment of the value expression assigned to the LHS variable.
<code>expression</code>	Generate a check for every arithmetic operation within the expression of the assignment statement.

`get_sps_arithmetic_overflow_style`

No arguments

Default

By default, the tool generates one arithmetic overflow for the assignment (`statement`).

JasperGold Apps Command Reference Manual

Configuration Commands

Return

`get_sps_arithmetic_overflow_style` returns the value of the variable `sps_arithmetic_overflow_style`.

Examples

```
% set_sps_arithmetic_overflow_style statement  
% get_sps_arithmetic_overflow_style
```

See Also

[check_sps_configure](#)

sps_deadcode_full_case_filter

Commands

`set_sps_deadcode_full_case_filter`
`get_sps_deadcode_full_case_filter`

Description

Use `set_sps_deadcode_full_case_filter` to prevent the generation of dead code checks for defaults of case if the case is already full (all cases specified).

Use `get_sps_deadcode_full_case_filter` to report whether SPS generates dead code checks for defaults of case if the case is already full.

Syntax

<code>set_sps_deadcode_full_case_filter (true false)</code>	
<code>true</code>	Prevent generation of dead code checks for defaults of case if the case is already full.
<code>false</code>	Allow generation of dead code checks for defaults of case if the case is already full. <i>This is the default.</i>
<code>get_sps_deadcode_full_case_filter</code>	
No arguments	

Default

By default, the tool generates dead code checks for defaults of case if the case is already full.

Return

`get_sps_deadcode_full_case_filter` returns the value of the variable `sps_deadcode_full_case_filter`.

Examples

```
% set_sps_deadcode_full_case_filter true
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_sps_deadcode_full_case_filter
```

See Also

[check_sps_configure](#)

sps_no_duplicates_per_check

Commands

set_sps_no_duplicates_per_check
get_sps_no_duplicates_per_check

Description

Use `set_sps_no_duplicates_per_check` to prevent duplication POIs that would generate duplicate properties within checks.

Use `get_sps_no_duplicates_per_check` to report whether duplication POIs are allowed.

Note: Removing duplication for dead code checks limits the ability to roll up coverage for all the blocks in the RTL.

Syntax

`set_sps_no_duplicates_per_check (true | false)`

`true` Do not allow duplication of POIs.

`false` Allow duplication of POIs. *This is the default.*

`get_sps_no_duplicates_per_check`

No arguments

Default

By default, duplicates are allowed (`false`).

Return

`get_sps_no_duplicates_per_check` returns the value of the variable `sps_no_duplicates_per_check`.

Examples

```
% set_sps_no_duplicates_per_check true
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_sps_no_duplicates_per_check
```

See Also

[check_sps_configure](#)

sps_signals_flops

Commands

set_sps_signals_flops
get_sps_signals_flops

Description

Use `set_sps_signals_flops` to filter out trivially true stuck-at and toggle properties by limiting them to the flops of selected signals.

Note: You can use this command in combination with `set_sps_signals_outputs`. If the setting for both `sps_signals_outputs` and `sps_signals_flops` is false, the tool considers all signals.

Use `get_sps_signals_flops` to determine whether generation of stuck-at and toggle properties is limited to the flops of selected signals.

Syntax

set_sps_signals_flops (true false)	
true	Limit generation of stuck-at and toggle properties to the flops of the selected signals.
false	Generate stuck-at and toggle properties for all signals. <i>This is the default.</i>
get_sps_signals_flops	
No arguments	

Default

By default, SPS generates stuck-at and toggle properties for all signals.

Return

`get_sps_signals_flops` returns the value of the variable `sps_signals_flops`.

Examples

```
% set_sps_signals_flops true  
% get_sps_signals_flops
```

See Also

[check_sps_configure](#)
[set_sps_signals_outputs](#)
[set_sps_stuck_at_compact](#)

sps_signals_max_bits

Commands

`set_sps_signals_max_bits`
`get_sps_signals_max_bits`

Description

Use `set_sps_signals_max_bits` to limit the number of bits on which the tool generates toggle or stuck-at properties when `set_sps_stuck_at_compact` is set to true. Use a non-positive value to unlimit max bits.

Use `get_sps_signals_max_bits` to determine the number of bits on which the tool will generate toggle or stuck-at properties.

Syntax

`set_sps_signals_max_bits <N>`

`N` Set the number of bits on which to generate toggle and stuck-at properties when `set_sps_stuck_at_compact` is set to true. Use a non-positive value to unlimit max bits.

`get_sps_signals_max_bits`

No arguments

Default

By default, SPS limits max bits to 3.

Return

`get_sps_signals_max_bits` returns the value of the variable `set_sps_signals_max_bits`.

Examples

```
% set_sps_signals_max_bits 4  
% get_sps_signals_max_bits
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[check_sps_configure](#)
[set_sps_stuck_at_compact](#)

sps_signals_outputs

Commands

set_sps_signals_outputs
get_sps_signals_outputs

Description

Use `set_sps_signals_outputs` to filter out trivially true stuck-at and toggle properties by limiting them to the outputs of selected signals.

Note: You can use this command in combination with `set_sps_signals_flops`. If the setting for both `sps_signals_outputs` and `sps_signals_flops` is `false`, the tool considers all signals.

Use `get_sps_signals_outputs` to determine whether generation of stuck-at and toggle properties is limited to the outputs of selected instances.

Syntax

set_sps_signals_outputs (true false)	
true	Limit generation of stuck-at and toggle properties to the outputs of the selected instances.
false	Generate stuck-at and toggle properties for all signals. <i>This is the default.</i>
get_sps_signals_outputs	
No arguments	

Default

By default, SPS generates stuck-at and toggle properties for all signals.

Return

`get_sps_signals_outputs` returns the value of the variable `set_sps_signals_outputs`.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_sps_signals_outputs true  
% get_sps_signals_outputs
```

See Also

[check_sps_configure](#)
[set_sps_signals_flops](#)
[set_sps_stuck_at_compact](#)

sps_stuck_at_compact

Commands

set_sps_stuck_at_compact
get_sps_stuck_at_compact

Description

Use `set_sps_stuck_at_compact` to control the bit blasting of stuck-at properties in SPS. When set to `true`, stuck-at properties are not bit blasted.

Use `get_sps_stuck_at_compact` to report whether bit blasting of stuck-at properties is allowed. When set to `true`, stuck-at properties are not bit blasted.

Syntax

set_sps_stuck_at_compact (true false)
true Prevent bit blasting of stuck-at properties. <i>This is the default.</i>
false Allow bit blasting of stuck-at properties.
get_sps_stuck_at_compact
No arguments

Default

By default, SPS prevents bit blasting of stuck-at properties.

Return

`get_sps_stuck_at_compact` returns the value of the variable `sps_stuck_at_compact`.

Examples

```
% set_sps_stuck_at_compact true
% get_sps_stuck_at_compact
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[check_sps_configure](#)
[set_sps_signals_flops](#)
[set_sps_signals_outputs](#)

sps_sva_max_depth

Commands

`set_sps_sva_max_depth`
`get_sps_sva_max_depth`

Description

Use `set_sps_sva_max_depth` to set the maximum depth (in terms of gate complexity) for SVA expressions.

Use `get_sps_sva_max_depth` to report the current maximum depth for SVA expressions..

Syntax

```
set_sps_sva_max_depth <N>
```

`N` Set the maximum depth (in terms of gate complexity) for SVA expressions.

```
get_sps_sva_max_depth
```

No arguments

Default

By default, the maximum depth for SVA expressions is 15.

Return

`get_sps_sva_max_depth` returns the value of the variable `sps_sva_max_depth`.

Examples

```
% set_sps_sva_max_depth 10  
% get_sps_sva_max_depth
```

See Also

[check_sps_configure](#)

sps_sva_max_length

Commands

set_sps_sva_max_length
get_sps_sva_max_length

Description

Use `set_sps_sva_max_length` to set the maximum length (in number of characters) for SVA expressions.

Use `get_sps_sva_max_length` to report the current maximum length for SVA expressions.

Syntax

set_sps_sva_max_length <N>	
N	Set the maximum length (in number of characters) for SVA expressions.
get_sps_sva_max_length	
No arguments	

Default

By default, the maximum length for SVA expressions is 1000.

Return

`get_sps_sva_max_length` returns the value of the variable `sps_sva_max_length`.

Examples

```
% set_sps_sva_max_length 100
% get_sps_sva_max_length
```

See Also

[check_sps_configure](#)

spv_include_control_path

Commands

`set_spv_include_control_path`
`get_spv_include_control_path`

Description

By default, SPV properties consider all paths between source and destination nodes to check for data propagation. Use `set_spv_include_control_path` to include or exclude control paths from the data propagation analysis.

Use `get_spv_include_control_path` to report whether control paths will be included in the data propagation analysis.

Syntax

<code>set_spv_include_control_path (on off_)</code>	
<code>on</code>	Include control paths in the data propagation analysis. <i>This option is the default.</i>
<code>off</code>	Exclude control paths from the data propagation analysis.
<code>get_spv_include_control_path</code>	
No arguments	

Default

By default, the tool includes control paths in the data propagation analysis (`set_spv_include_control_path=on`).

Return

`get_spv_include_control_path` returns the value of the variable `spv_include_control_path`.

Examples

```
% analyze -sv $RTL_DIR/dut.sv
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% elaborate -top dut

% clock clk
% reset rst

% set_spv_include_control_path off
% check_spv -create -from in -to out -name prop_not_through_control_paths

% set_spv_include_control_path on
% check_spv -create -from in -to out -name prop_through_all_paths

% check_spv -prove all

% get_spv_include_control_path
```

See Also

[check_spv](#)

sst_default_trace_length

Commands

`set_sst_default_trace_length`
`get_sst_default_trace_length`

Description

Use `set_sst_default_trace_length` to set the default state-space tunneling trace length. This length will be the length of the SST trace if you do not set the maximum or minimum length for Visualize.

Use `get_sst_default_trace_length` to get the current state-space tunneling default trace length.

Syntax

`set_sst_default_trace_length <N>`

`N` Set the default SST trace length. *The default is 15.*

`get_sst_default_trace_length`

No arguments

Default

The default trace length is 15.

Return

`get_sst_default_trace_length` returns the current value of the `sst_default_trace_length` variable.

Examples

```
% set_sst_default_trace_length 10
```

```
% get_sst_default_trace_length
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[set_sst_undetermined_helpers](#)

[visualize](#)

sst_undetermined_helpers

Commands

`set_sst_undetermined_helpers`
`get_sst_undetermined_helpers`

Description

Use `set_sst_undetermined_helpers` to specify how the tool should handle undetermined helper assertions when requesting a state-space tunneling trace with the command `visualize -sst -prop <sst_target>`.

Use `get_sst_undetermined_helpers` to get the state-space tunneling mode for undetermined helper assertions.

Syntax

<code>set_sst_undetermined_helpers (conjunct assume ignore conjunct_force)</code>	
<code>conjunct</code>	AND all the helper assertions together with the target property to create the <code>sst_property</code> , which will fail exactly (and only) at the last cycle of the SST trace. The failure happens if any of the properties in the conjunct fail.
<code>assume</code>	Use all undetermined helper assertions as assumptions and assume them in the generated SST trace. The generated trace will be a CEX for the target property (<code>sst_target</code>).
<code>ignore</code>	Ignore all the undetermined helper assertions while generating the SST trace. The generated trace will be a CEX for the target property.
<code>conjunct_force</code>	AND all the helper assertions together with the target property to create the <code>sst_property</code> , which will fail exactly (and only) at the last cycle of the SST trace. The failure happens only if the target property in the conjunct fails. Other helper assertions might fail too.

JasperGold Apps Command Reference Manual

Configuration Commands

get_sst_undetermined_helpers

No arguments

Default

The default value is conjunct.

Return

`get_sst_undetermined_helpers` returns the current value of the `sst_undetermined_helpers` variable.

Examples

```
% set_sst_undetermined_helpers conjunct  
% get_sst_undetermined_helpers
```

See Also

[set_sst_default_trace_length](#)
[visualize](#)

stop_on_cex_limit

Commands

`set_stop_on_cex_limit`
`get_stop_on_cex_limit`

Description

Use `set_stop_on_cex_limit` to direct the `prove` command to stop right after it reports the first `stop_on_cex_limit` counterexamples.

Use `get_stop_on_cex_limit` to get the value of the `stop_on_cex_limit` variable.

Syntax

<code>set_stop_on_cex_limit <N></code>	
<code>N</code>	Set the stop on counterexample limit.
<code>get_stop_on_cex_limit</code>	
No arguments	

Default

The default value is 0 (unlimited).

Return

`get_stop_on_cex_limit` returns the current value of the `stop_on_cex_limit` variable.

Examples

```
% set_stop_on_cex_limit 1  
% get_stop_on_cex_limit
```

See Also

[prove](#)

stop_on_unreachable_limit

Commands

`set_stop_on_unreachable_limit`
`get_stop_on_unreachable_limit`

Description

Use `set_stop_on_unreachable_limit` to direct the `prove` command to stop right after it reports the first `stop_on_unreachable_limit` unreachable covers.

Use `get_stop_on_unreachable_limit` to get the value of the `stop_on_unreachable_limit` variable.

Syntax

<code>set_stop_on_unreachable_limit <N></code>
<code>N</code> Set the stop on unreachable limit.
<code>get_stop_on_unreachable_limit</code>
No arguments

Default

The default value is 0 (unlimited).

Return

`get_stop_on_unreachable_limit` returns the current value of the `stop_on_unreachable_limit` variable.

Examples

```
% set_stop_on_unreachable_limit 1  
% get_stop_on_unreachable_limit
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

[prove](#)

tag_irrelevant_values

Commands

set_tag_irrelevant_values
get_tag_irrelevant_values

Description

Use the `set_tag_irrelevant_values` command to specify whether the tool should replace X values with concrete values or show that they are irrelevant to the proof or Visualize target. This command applies to all future traces during your session. It does not change a trace that the tool is currently displaying.

With `set_tag_irrelevant_values true`, the tool shows signal values that do not affect the analysis result as don't cares with x or X. These don't cares provide additional debugging information, allowing you to focus on the signals that actually affect the analysis result. To use this feature, you must first turn on the irrelevant values computation with `set_trace_optimization -irrelevant_value_computation true`.

Use the `get_tag_irrelevant_values` command to show the current setting for whether irrelevant values are tagged as X in the Visualize window.

Syntax

`set_tag_irrelevant_values (true | false)`

`true` The tool shows inputs irrelevant to the related assumptions and targets as X. The values of other signals are consistent with three-value simulation.

`false` The tool assigns 0 to inputs that are irrelevant to the related assumptions and targets so that there are no X values in the trace. *This option is the default.*

`get_tag_irrelevant_values`

No arguments

Default

The default value is `false`. JasperGold Apps assigns 0 to inputs that are irrelevant to the related assumptions and targets so that there are no X values in the trace.

Return

`get_tag_irrelevant_values` returns the value of the variable `tag_irrelevant_values`.

Examples

```
% set_tag_irrelevant_values false  
% get_tag_irrelevant_values
```

See Also

[set_trace_optimization](#)
[visualize](#)

task_compile_time_limit

Commands

`set_task_compile_time_limit`
`get_task_compile_time_limit`

Description

Use the `set_task_compile_time_limit` command to specify the per-task time limit for compiling properties during synthesis. This process runs during elaboration.

Use the `get_task_compile_time_limit` command to display the maximum time the tool can spend compiling a task's properties during elaboration.

Syntax

```
set_task_compile_time_limit <time_limit>
```

`time_limit` Use the specified limit for compiling each task's properties during elaboration.

Specify the time limit as an integer followed by `s`, `m`, or `h` (no spaces).

- `s` = seconds
- `m` = minutes
- `h` = hours

JasperGold Apps commonly recognizes `0s` as “unlimited.” In the case of this command, `0s` sets the highest supported limit, which is 10000 seconds.

```
get_task_compile_time_limit
```

No arguments

Default

By default, the task compilation time limit is 120s.

Return

`get_task_compile_time_limit` returns the value of the variable `task_compile_time_limit`.

Examples

```
% set_task_compile_time_limit 45s  
% get_task_compile_time_limit
```

See Also

[set_elab_time_limit](#)
[set_property_compile_time_limit](#)

tcl_no_precondition

Commands

`set_tcl_no_precondition`
`get_tcl_no_precondition`

Description

By default, the tool automatically generates precondition covers for liveness assertions you create with the `assert` command. Use `set_tcl_no_precondition on` if you do not want automatically generated covers.

Use the `get_tcl_no_precondition` command to determine whether the tool will automatically generate precondition covers for liveness assertions created with the `assert` command.

Syntax

<code>set_tcl_no_precondition (on off</code>	
<code>on</code>	Do not generate precondition covers for liveness assertions created with the <code>assert</code> command.
<code>off</code>	Generate precondition covers for liveness assertions created with the <code>assert</code> command. <i>This option is the default.</i>
<code>get_tcl_no_precondition</code>	
No arguments	

Default

By default, the tool automatically generates precondition covers for liveness assertions you create with the `assert` command (`off`).

Return

`get_tcl_no_precondition` returns the value of the variable `tcl_no_precondition`.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_tcl_no_precondition on  
% get_tcl_no_precondition
```

See Also

No related commands

time_format

Commands

set_time_format
get_time_format

Description

Use the `set_time_format` command to change the format for time values that are printed in JasperGold Apps messages. By default, time is expressed in seconds carried out to three decimal places.

Use the `get_time_format` command to display the current time format.

Syntax

```
set_time_format (second | minute | hour)
```

second	Express time values in seconds. <i>This option is the default.</i>
--------	--

minute	Express time values in minutes.
--------	---------------------------------

hour	Express time values in hours.
------	-------------------------------

```
get_time_format
```

```
No arguments
```

Default

By default, time is expressed in seconds carried out to three decimal places.

Return

`get_time_format` returns the value of the variable `time_format`.

Examples

```
% set_time_format minute
```

```
% get_time_format
```

JasperGold Apps Command Reference Manual

Configuration Commands

See Also

No related commands

trace_extension

Commands

set_trace_extension
get_trace_extension

Description

Use the `set_trace_extension` command to extend a trace by a specified number of cycles that continue to satisfy assumptions.



- Trace extensions apply only to finite (non-looping) traces; therefore, this command has no effect on traces for liveness properties.
- A trace extension is part of the global property environment (just like clock and reset) and alters the meaning of your properties. The tool clears proof results when you change the trace extension. Running a new proof after extending the trace can change the property status from cex or covered to proven or unreachable
- Trace extensions force the proof engines to look for longer traces, which is likely to slow the engines whether a trace exists or not.
- \$ is a special value meaning infinite. If you use \$, you will get looping traces for safety properties.
- The trace extension is visible in the waveforms as additional cycles in which all assumptions are respected, but the property `signal` itself may behave arbitrarily.
- When you specify a trace extension, the full trace length and the cycle in which a property is triggered may not coincide as they normally do. The following describes how this distinction manifests in the tool:
 - The property bounds – as seen in the *Bound* column in the main window property table and as accessible using `get_property_info -list {min_length max_length}` – will always refer to the cycle triggering the property, that is, the length excluding any trace extension. Thus, as long as there are no non-extendable dead-end traces, the bound numbers will be comparable regardless of extension.
 - The *Trace Attempts* and the trace length will refer to the full length including any trace extension.

Note:

- You can see *Trace Attempts* in the ProofGrid Manager window and the *Proof Messages* tab.
- You can see the trace length with `get_property_info -list trace_length` or `visualize -get_length`.
- When using engine L, consider adding the trace extension to the `engineL_tail_length`. (See “[set_engineL_tail_length](#)” on page 1074.) Doing so instructs engine L to extract new start states at (or before) the actual property violation rather than a few cycles after.

Use the `get_trace_extension` command to report the extra number of cycles specified beyond the cycle that triggered a property.

Syntax

<code>set_trace_extension <N></code>
<code>N</code> Extend a trace <i>N</i> extra cycles beyond the cycle that triggered a property.
<code>get_trace_extension</code>
No arguments

Default

By default, trace extension is 0.

Return

`get_trace_extension` returns the value of the variable `trace_extension`.

Examples

```
% set_trace_extension 10  
% get_trace_extension
```

See Also

No related commands

trace_optimization

Commands

set_trace_optimization
get_trace_optimization

Description

Use the `set_trace_optimization` command to change the optimization in trace computations.

The following list includes things you should know about usage.

- In addition to their impact on performance, trace optimization settings are reflected in saved trace files:
`visualize -save (-vcd | -fsdb).`

For example:

- If trace optimization is `ar`, then only the analysis region is saved into the trace file. (This is the default setting.)
 - If trace optimization is `standard`, then the entire design is dumped into the trace file.
- To tag irrelevant values, that is, show signal values that do not affect the analysis result as don't cares with `x` or `X`, you must first turn on the irrelevant values computation with `set_trace_optimization -irrelevant_value_computation true`. (Tagging provides additional debugging information, allowing you to focus on the signals that actually affect the analysis result.)

Use the `get_trace_optimization` command to report whether trace optimizations are enabled.

JasperGold Apps Command Reference Manual

Configuration Commands

Syntax

```
set_trace_optimization (ar | coi | maximum | standard)
```

ar	<p>Use the analysis region for optimizations. <i>This option is the default.</i></p> <p>If Design Tunneling mode is off, this setting is the same as <code>coi</code>. This setting is faster than the standard setting because, initially, the tool simulates only the analysis region. As you add logic from outside the AR, the tool simulates it on the fly.</p>
coi	<p>Use the cone of influence for optimizations.</p> <p>This setting is faster than the standard setting, but the values of the signals outside the cone of influence will be shown as Xs.</p>
maximum	<p>Use <code>maximum</code> to include the <code>ar</code> optimization and turn off global constant manipulation.</p> <p>If you use this setting when you are using Design Tunneling, some proofs might not terminate as quickly and gated clocks might not be detected fully. (You might get messages about flops with no compatible clocks.)</p>
standard	<p>Use the standard set of optimizations for the full design.</p> <p>This setting is designed for full design exploration mode with good time and memory performance in trace generation, Why computations, and so forth.</p>

```
set_trace_optimization -irrelevant_value_computation (true | false)
```

Turn on or off the trace computation to detect irrelevant values.

- `true` turn on trace computation to detect irrelevant values.
- `false` turn off trace computation to detect irrelevant values.

The `set_tag_irrelevant_values` feature uses the computation result. *The default argument is false.*

```
get_trace_optimization
```

No arguments

Default

By default, trace generation uses the analysis region for optimizations and the irrelevant value computation is off.

Return

`get_trace_optimization` returns the value of the variable `trace_optimization` (standard, maximum, and so forth).

Examples

```
% set_trace_optimization maximum  
% get_trace_optimization
```

See Also

[set tag irrelevant values](#)
[visualize](#)

trace_show_reset

Commands

set_trace_show_reset
get_trace_show_reset

Description

Use the `set_trace_show_reset` command to disable or re-enable the feature that makes it possible to display the reset sequence as a prefix in the Visualize window.



- You must use this command before reset analysis.
- The tool only displays the register values of a trace file specified with `reset -vcd`, `reset -shm`, or `reset -fsdb` and only if you do not use `reset -expression` or `reset -sequence`.
- If you use `reset -expression` or `reset -sequence` with other `reset` options (`-init_state`, `-vcd`, `-fsdb`, `-non_resettable_regs`) or abstract `-init_value`, it is possible to have inconsistencies in the trace prefix (the portion that shows the reset values). This situation occurs because the trace prefix shows the results of the reset simulation (`-expression` or `-sequence`), and the tool applies value changes the other commands impose after the reset simulation. Therefore, the Visualize window uses the final reset values imposed by other commands beginning at cycle 1.
- By default, the Visualize window does not display the reset sequence. To display the prefix in the waveforms pane with negative cycle numbers, use the menu *View – Show Reset Cycles*.

Use the `get_trace_show_reset` command to learn whether the reset prefix feature is enabled.

Syntax

```
set_trace_show_reset (true | false)
```

true	Enable the reset prefix feature. <i>This option is the default.</i>
------	---

JasperGold Apps Command Reference Manual

Configuration Commands

false	Turn off the reset prefix feature.
-------	------------------------------------

get_trace_show_reset

No arguments

Default

By default, the feature that enables you to display the reset sequence in traces is on.

Return

get_trace_show_reset returns the value of the variable trace_show_reset (true or false).

Examples

```
% analyze -verilog my_design.v
% elaborate -top my_design
% clock clk
% reset {resetN==1'b0}
% prove -property {<embedded>>::my_property}
% visualize -violation -property {<embedded>>::my_property}
// Use the View menu to display the reset sequence.

% get_trace_show_reset
```

See Also

[reset](#)

[visualize](#)

tunnel_library_cell

Commands

set_tunnel_library_cell
get_tunnel_library_cell

Description

Use the `set_tunnel_library_cell` command to enable tunneling in designs with library cells. With `set_tunnel_library_cell on`, the tool “jumps over” cell define registers during manual tunneling, which allows you to access the signals you want to justify without accessing the logic inside the library cell. You must use this command before the `prove` command to impact subsequent traces.

Note: Use this command in an interactive JasperGold Design Tunneling flow.

Use the `get_tunnel_library_cell` command to learn whether tunneling in designs with library cells is enabled.

Syntax

```
set_tunnel_library_cell (on | off)
```

on Enable tunneling for designs with library cells.

off Turn off support for manual tunneling in designs with library cells. *This option is the default.*

get_tunnel_library_cell

No arguments

Default

By default, tunneling library cells is not supported.

Return

`get_tunnel_library_cell` returns the value of the variable `tunnel_library_cell` (`on` or `off`).

Examples

```
% analyze -verilog my_design.v -v my_lib.v
% elaborate -top my_design
% clock clk
% reset {resetN==1'b0}
% set_dst_mode -env on
% set_tunnel_library_cell on
% prove -property {<embedded>::my_property}
% visualize -violation -property {<embedded>::my_property}

% get_tunnel_library_cell
```

See Also

[justify](#)
[prove](#)
[stopat](#)
[suggest](#)

visualize_auto_check_props

Commands

`set_visualize_auto_check_props`
`get_visualize_auto_check_props`

Description

By default, the tool does not automatically compute exercised covers and violated assertions. Use `set_visualize_auto_check_props` to turn on the computation. If the computation is turned on, the properties in the Indexed Behaviors (Visualize™ window) display the cycles in which covers are exercised and assertions are failing. Look for this information in the *At* column.

Note: Enabling this computation can impact run-time performance.

Use `get_visualize_auto_check_props` to find out if Visualize is set to automatically check property status for indexed behaviors when generating waveforms.

Syntax

<code>set_visualize_auto_check_props (on off)</code>
on Automatically compute exercised covers and violated assertions.
off Do not compute exercised covers and violated assertions. <i>This option is the default.</i>
<code>get_visualize_auto_check_props</code>
No arguments

Default

By default, the tool does not compute exercised covers and violated assertions automatically (off).

Return

`get_visualize_auto_check_props` returns the value of the `visualize_auto_check_props` variable.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_visualize_auto_check_props on  
% get_visualize_auto_check_props
```

See Also

No related commands

visualize_auto_load_debugging_tables

Commands

`set_visualize_auto_load_debugging_tables`
`get_visualize_auto_load_debugging_tables`

Description

By default, the tool does not automatically load debugging tables. Use the `set_visualize_auto_load_debugging_tables` command to set the tool to automatically load debugging tables for all.

Note: Enabling debugging tables can impact run-time performance.

Use `get_visualize_auto_load_debugging_tables` to find out if Visualize is set to automatically load debugging tables.

Syntax

<code>set_visualize_auto_load_debugging_tables (on off)</code>
<code>on</code> Automatically load debugging tables.
<code>off</code> Do not automatically load debugging tables. <i>This option is the default.</i>
<code>get_visualize_auto_load_debugging_tables</code>
No arguments

Default

By default, the tool does not automatically load debugging tables (`off`).

Return

`get_visualize_auto_load_debugging_tables` returns the value of the `visualize_auto_load_debugging_tables` variable.

JasperGold Apps Command Reference Manual

Configuration Commands

Examples

```
% set_visualize_auto_load_debugging_tables on  
% get_visualize_auto_load_debugging_tables
```

See Also

No related commands

waveform_import_multiple_segments

Commands

set_waveform_import_multiple_segments
get_waveform_import_multiple_segments

Description

By default, the tool extracts properties only from the last non-reset segment in a single trace. Use the `set_waveform_import_multiple_segments` command to enable the extraction of properties from multiple non-reset segments in a single trace.

Use the `get_waveform_import_multiple_segments` command to learn whether the tool extracts properties over multiple non-reset segments of a single trace.

Syntax

set_waveform_import_multiple_segments (on off)
on Extract properties over multiple non-reset segments of a single trace.
off Extract properties only from the last non-reset segment of the trace. <i>This option is the default.</i>
get_waveform_import_multiple_segments
No arguments

Default

By default, the tool does not extract properties from multiple non-reset segments in a single trace (`off`).

Return

`get_waveform_import_multiple_segments` returns the value of the variable `waveform_import_multiple_segments`, on or off.

Examples

```
% set_waveform_import_multiple_segments on
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_waveform_import_multiple_segments
```

See Also

[scan](#)
[waveform](#)

waveform_import_secondary_hier_path

Commands

`set_waveform_import_secondary_hier_path`
`get_waveform_import_secondary_hier_path`

Description

Use `set_waveform_import_secondary_hier_path` to specify a secondary hierarchical path. When you have two top-level designs and want to analyze both hierarchies to get cross-hierarchy properties, this command provides the path to the signals not specified in the `waveform` or `scan` commands.

Use the `get_waveform_import_secondary_hier_path` command to determine the path to the signals not specified in the `waveform` or `scan` commands.

Syntax

<code>set_waveform_import_secondary_hier_path <hierarchical_path></code>
<i>hierarchical_path</i>
Set a secondary hierarchical path to signals not specified in the <code>waveform</code> or <code>scan</code> commands.
<code>get_waveform_import_secondary_hier_path</code>
No arguments

Default

There is no default.

Return

`get_waveform_import_secondary_hier_path` returns the value of the variable `waveform_import_secondary_hier_path`.

Examples

```
% set_waveform_import_secondary_hier_path tb.t
```

JasperGold Apps Command Reference Manual

Configuration Commands

```
% get_waveform_import_secondary_hier_path
```

See Also

[scan](#)
[waveform](#)

word_level_reduction

Commands

`set_word_level_reduction`
`get_word_level_reduction`

Description

Use the `set_word_level_reduction` command to run the proof flow with word-level transformations or reductions to optimize the design before calling the proof engines.

Note:

- The word-level flow is an initial release to gather feedback from early adopters and finalize implementation for an upcoming release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.
- This feature is not supported in the following cases in the current release:
 - When using engine modes P5, Q1, or Q2
 - When using the X-Prop Verification App or SPV App
 - When using the Coverage App with proof-core calculation enabled.

Use the `get_word_level_reduction` to determine if the proof flow will be run with word-level transformations or reductions.

Syntax

```
set_word_level_reduction (on | off)
```

on	Enable word-level transformations or reductions during the proof flow.
----	--

off	Disable word-level transformations or reductions during the proof flow. <i>This is the default.</i>
-----	--

```
get_word_level_reduction
```

No arguments

Default

By default, word-level reduction is disabled (`off`).

Return

`get_word_level_reduction` returns the value of the variable `word_level_reduction`.

Examples

```
% set_word_level_reduction on  
% get_word_level_reduction
```

See Also

No related commands

xprop_compute_highlights

Commands

set_xprop_compute_highlights
get_xprop_compute_highlights

Description

Use `set_xprop_compute_highlights` to disable or re-enable X-Propagation Verification App highlights.

Use `get_xprop_compute_highlights` to determine whether X-Propagation highlights are enabled or disabled.

Syntax

<code>set_xprop_compute_highlights (on off) [-task <task_name>]</code>	
<code>on</code>	Turn on X-Prop highlights. <i>This option is the default.</i>
<code>off</code>	Turn off X-Prop highlights. This setting can improve performance in run times.
<code>-task</code> <i>task_name</i>	Change the setting for the specified task instead of all tasks.
<hr/> <code>get_xprop_compute_highlights [-task <task_name>]</code>	
<code>-task</code> <i>task_name</i>	Get the setting for the specified task instead of all tasks.
Note:	
<ul style="list-style-type: none">■ This switch is helpful because it is possible to use <code>set_xprop_compute_highlights</code> to change the configuration variable for a specific task instead of all tasks.■ A task configuration value has priority over global configuration values.	

Default

By default, X-Propagation highlight are on for all tasks.

Return

`get_xprop_compute_highlights` returns the value of the `xprop_compute_highlights` variable.

Examples

```
% set_xprop_compute_highlights off  
% get_xprop_compute_highlights
```

See Also

No related commands

xprop_use_all_undriven

Commands

set_xprop_use_all_undriven
get_xprop_use_all_undriven

Description

By default, the tool treats only black-boxed outputs as sources of X. Use set_xprop_use_all_undriven to change the default behavior and treat all undriven nets (black-boxed outputs, inputs, stopat nets, and internal undriven nets) as sources of X during X-propagation checks.



Using the command `set_xprop_use_all_undriven` on accomplishes the same result as the following commands combined:

- `set_xprop_use_bbox_outputs` on
- `set_xprop_use_inputs` on
- `set_xprop_use_stopats` on
- `set_xprop_use_internal_undriven` on

CAUTION:

- If `xprop_use_all_undriven` is on, it *supersedes* any prior and future instances of the commands listed above.
- If `xprop_use_all_undriven` is off, the tool *obeys* any prior and future instances of the commands listed above.

Use `get_xprop_use_all_undriven` to determine whether the tool treats all undriven nets as sources of X during X-propagation checks.

Syntax

```
set_xprop_use_all_undriven (on | off) [-task <task_name>]
```

on	Treat all undriven nets as sources of X.
----	--

JasperGold Apps Command Reference Manual

Configuration Commands

off	Do not treat all undriven nets as sources of X. <i>This option is the default.</i>
-task task_name	Change the sources of X for the specific task instead of for all tasks. Note: <ul style="list-style-type: none">■ If you use the -task switch, the tool clears the proof status for properties in the specified task.■ If a specific task behavior is set, it has higher priority over global behaviors. However, if you set a global behavior, the tool clears all previously set task-specific behaviors.
get_xprop_use_all_undriven [-task <task_name>]	
-task task_name	Use -task to get the value of the variable for the specified task. A task configuration value has priority over global configuration values.

Default

By default, the tool treats only black-boxed outputs as sources of X during X-propagation checks (off) and includes all tasks.

Return

get_xprop_use_all_undriven returns the value of the xprop_use_all_undriven variable.

Examples

```
% set_xprop_use_all_undriven on
% get_xprop_use_all_undriven
```

See Also

[set_xprop_use_bbox_outputs](#)
[set_xprop_use_inputs](#)
[set_xprop_use_internal_undriven](#)
[set_xprop_use_low_power](#)
[set_xprop_use_reset_state](#)
[set_xprop_use_stopats](#)
[set_xprop_use_x_assignments](#)

xprop_use_bbox_outputs

Commands

set_xprop_use_bbox_outputs
get_xprop_use_bbox_outputs

Description

By default, the tool treats black-boxed outputs as sources of X during X-propagation checks. Use the `set_xprop_use_bbox_outputs` command to change the default behavior so that the tool does not treat black-boxed outputs as sources of X during X-propagation checks.

Important

This command is intended for fine-grained control over what the tool considers as sources of X during X-propagation. If you want to treat all undriven nets (black-boxed outputs, inputs, stopat nets, and internal undriven nets) as sources of X, use the command `set_xprop_use_all_undriven on` instead of this command.

CAUTION:

- If `xprop_use_all_undriven` is `on`, it *supersedes* any prior and future instances of `set_xprop_use_bbox_outputs`.
- If `xprop_use_all_undriven` is `off`, the tool *obeys* any prior and future instances of `set_xprop_use_bbox_outputs`.

Use the `get_xprop_use_bbox_outputs` command to determine whether the tool treats black-boxed outputs as sources of X during X-propagation checks.

Syntax

```
set_xprop_use_bbox_outputs (on | off) [-task <task_name>]
```

on	Treat black-boxed outputs as sources of X. <i>This option is the default.</i>
----	---

off	Do not treat black-boxed outputs as sources of X.
-----	---

JasperGold Apps Command Reference Manual

Configuration Commands

```
-task      Change the sources of X for the specific task instead of for all tasks.  
task_name
```

Note:

- If you use the `-task` switch, the tool clears the proof status for properties in the specified task.
- If a specific task behavior is set, it has higher priority over global behaviors. However, if you set a global behavior, the tool clears all previously set task-specific behaviors.

```
get_xprop_use_bbox_outputs [-task <task_name>]
```

```
-task      Use -task to get the value of the variable for the specified task.  
task_name
```

A task configuration value has priority over global configuration values.

Default

By default, the tool treats black-boxed outputs as sources of X during X-propagation checks (on) and includes all tasks.

Return

`get_xprop_use_bbox_outputs` returns the value of the configuration variable `xprop_use_bbox_outputs`, on or off.

Examples

```
% set_xprop_use_bbox_outputs off  
% get_xprop_use_bbox_outputs
```

See Also

[set_xprop_use_all_undriven](#)
[set_xprop_use_inputs](#)
[set_xprop_use_internal_undriven](#)
[set_xprop_use_low_power](#)
[set_xprop_use_reset_state](#)
[set_xprop_use_stopats](#)
[set_xprop_use_x_assignments](#)

xprop_use_inputs

Commands

set_xprop_use_inputs
get_xprop_use_inputs

Description

By default, the tool does not treat primary inputs as sources of X during X-propagation checks. Use `set_xprop_use_inputs` to change the default behavior and treat the primary inputs as sources of X during X-propagation checks.



This command is intended for fine-grained control over what the tool considers as sources of X during X-propagation. If you want to treat all undriven nets (black-boxed outputs, inputs, stopat nets, and internal undriven nets) as sources of X, use the command `set_xprop_use_all_undriven on` instead of this command.

CAUTION:

- If `xprop_use_all_undriven` is `on`, it *supersedes* any prior and future instances of `set_xprop_use_inputs`.
- If `xprop_use_all_undriven` is `off`, the tool *obeys* any prior and future instances of `set_xprop_use_inputs`.

Use `get_xprop_use_inputs` to determine whether the tool treats primary inputs as sources of X during X-propagation checks.

Syntax

```
set_xprop_use_inputs (on | off) [-task <task_name>]
```

on	Treat primary inputs as sources of X.
----	---------------------------------------

off	Do not treat primary inputs as sources of X. <i>This option is the default.</i>
-----	---

JasperGold Apps Command Reference Manual

Configuration Commands

-task Change the sources of X for the specific task instead of for all tasks.
 task_name

Note:

- If you use the **-task** switch, the tool clears the proof status for properties in the specified task.
- If a specific task behavior is set, it has higher priority over global behaviors. However, if you set a global behavior, the tool clears all previously set task-specific behaviors.

get_xprop_use_inputs
[-task <task_name>]

-task Use **-task** to get the value of the variable for the specified task.
 task_name
A task configuration value has priority over global configuration values.

Default

By default, the tool does not treat primary inputs as sources of X during X-propagation checks (**off**) and includes all tasks.

Return

`get_xprop_use_inputs` returns the value of the `xprop_use_inputs` variable.

Examples

```
% set_xprop_use_inputs on  
% get_xprop_use_inputs
```

See Also

[set_xprop_use_all_undriven](#)
[set_xprop_use_bbox_outputs](#)
[set_xprop_use_internal_undriven](#)
[set_xprop_use_low_power](#)
[set_xprop_use_reset_state](#)
[set_xprop_use_stopats](#)
[set_xprop_use_x_assignments](#)

xprop_use_internal_undriven

Commands

set_xprop_use_internal_undriven
get_xprop_use_internal_undriven

Description

By default, the tool does not treat internal undriven nets as sources of X during X-propagation checks. Use set_xprop_use_internal_undriven to change the default behavior and treat internal undriven nets (other than black-boxed outputs, inputs and stopat nets) as sources of X during X-propagation checks.



This command is intended for fine-grained control over what the tool considers as sources of X during X-propagation. If you want to treat all undriven nets (black-boxed outputs, inputs, stopat nets, and internal undriven nets) as sources of X, use the command set_xprop_use_all_undriven on instead of this command.

CAUTION:

- If xprop_use_all_undriven is on, it *supersedes* any prior and future instances of set_xprop_use_internal_undriven.
- If xprop_use_all_undriven is off, the tool *obeys* any prior and future instances of set_xprop_use_internal_undriven.

Use get_xprop_use_internal_undriven to determine whether the tool treats undriven nets as sources of X during X-propagation checks.

Syntax

```
set_xprop_use_internal_undriven (on | off) [-task <task_name>]
```

on Treat internal undriven nets as sources of X.

off Do not treat internal undriven nets as sources of X. *This option is the default.*

JasperGold Apps Command Reference Manual

Configuration Commands

`-task
task_name` Change the sources of X for the specific task instead of for all tasks.

Note:

- If you use the `-task` switch, the tool clears the proof status for properties in the specified task.
- If a specific task behavior is set, it has higher priority over global behaviors. However, if you set a global behavior, the tool clears all previously set task-specific behaviors.

`get_xprop_use_internal_undriven [-task <task_name>]`

`-task
task_name` Use `-task` to get the value of the variable for the specified task.
A task configuration value has priority over global configuration values.

Default

By default, the tool does not treat internal undriven nets as sources of X during X-propagation checks (`off`) and includes all tasks.

Return

`get_xprop_use_internal_undriven` returns the value of the `xprop_use_internal_undriven` variable .

Examples

```
% set_xprop_use_internal_undriven on  
% get_xprop_use_internal_undriven
```

See Also

[set_xprop_use_all_undriven](#)
[set_xprop_use_bbox_outputs](#)
[set_xprop_use_inputs](#)
[set_xprop_use_low_power](#)
[set_xprop_use_reset_state](#)
[set_xprop_use_stopats](#)
[set_xprop_use_x_assignments](#)

xprop_use_low_power

Commands

set_xprop_use_low_power
get_xprop_use_low_power

Description

By default, X-Prop checks ignore Xs generated by any low-power configurations. Use `set_xprop_use_low_power` to consider Xs generated by low-power configurations or to reinstate the default.

Use `get_xprop_use_low_power` to determine whether the tool considers Xs generated by low-power configurations as sources of X.

Syntax

`set_xprop_use_low_power (on | off)`

`on` Consider Xs generated by low-power configurations as sources of X.

Note: To consider just X assignments from low-power configurations, set the `xprop_use_x_assignments` variable to off.

`off` Ignore Xs generated by low-power configurations. *This option is the default.*

`get_xprop_use_low_power`

No arguments

Default

By default, the tool does not consider Xs generated by low-power configurations as sources of X.

Return

`get_xprop_use_low_power` returns the value of the `xprop_use_low_power` variable .

Examples

```
% analyze -sv $RTL_DIR/dut.sv
% elaborate -top dut

% check_lpv -load_jpf lowpower.jpf
% check_lpv -generate_power_design

% clock clk
% reset -expression ~rst_n

% get_reset_info -x_value

% assert -xprop -to a_out
% assert -xprop -to b_out

% set_xprop_use_x_assignments off
% set_xprop_use_low_power on
% prove -task
```

See Also

[assert](#)
[elaborate](#)

xprop_use_reset_abstraction

Commands

set_xprop_use_reset_abstraction
get_xprop_use_reset_abstraction

Description

By default, X-Prop checks do not treat flop reset abstractions as sources of X. Use `set_xprop_use_reset_abstraction` to enable flop reset abstractions as sources of X or reinstate the default.

Use `get_xprop_use_reset_abstraction` to determine whether the tool will treat flop reset abstractions as source of X.

Syntax

set_xprop_use_reset_abstraction (on off) [-task task_name]
on Use flop reset abstractions as sources of X..
off Ignore flop reset abstractions as sources of X. <i>This option is the default.</i>
[-task task_name]
Apply this command to the specified task.
Note:
<ul style="list-style-type: none">■ If you specify a task, the tool only clears proof status for the specified task's properties.■ A task configuration value has priority over global configuration values, but if you subsequently apply the command to the global environment, the tool clears the previously specified task-specific configuration.

get_xprop_use_reset_abstraction

No arguments

Default

By default, the tool ignores flop reset abstractions as sources of X for all tasks.

Return

`get_xprop_use_reset_abstraction` returns the value of the `xprop_use_reset_abstraction` variable.

Examples

```
% analyze -sv $RTL_DIR/dut.sv
% elaborate -top dut

% clock clk
% reset -expression ~rst_n

% set_xprop_use_reset_abstraction on

% abstract -reset_value a_out
% assert -xprop -to a_out

% prove -task
```

See Also

[assert](#)
[reset](#)

xprop_use_reset_state

Commands

set_xprop_use_reset_state
get_xprop_use_reset_state

Description

By default, X-propagation checks treat uninitialized registers as sources of X. Use set_xprop_use_reset_state to ignore uninitialized registers or reinstate the default.

Use the get_xprop_use_reset_state command to determine whether the tool treats uninitialized registers as sources of X.

Syntax

```
set_xprop_use_reset_state (on | off) [-task <task_name>]
```

on	Use uninitialized registers as sources of X. <i>This option is the default.</i>
----	---

off	Ignore uninitialized registers.
-----	---------------------------------

-task	Change the sources of X for the specific task instead of for all tasks.
-------	---

task_name

Note:

- If you use the -task switch, the tool clears the proof status for properties in the specified task.
- If a specific task behavior is set, it has higher priority over global behaviors. However, if you set a global behavior, the tool clears all previously set task-specific behaviors.

```
get_xprop_use_reset_state [-task <task_name>]
```

-task	Use -task to get the value of the variable for the specified task.
-------	--

task_name

A task configuration value has priority over global configuration values.

Default

By default, the tool treats uninitialized flops as sources of X and includes all tasks.

Return

`get_xprop_use_reset_state` returns the value of the configuration variable `xprop_use_reset_state`, on or off.

Examples

```
% analyze -sv $RTL_DIR/dut.sv
% elaborate -top dut

% clock clk
% reset -expression ~rst_nxp

% get_reset_info -x_value

% assert -xprop -to a_out
% assert -xprop -to b_out

% set_xprop_use_reset_state on
% prove -task

% get_xprop_use_reset_state
```

See Also

[assert](#)
[elaborate](#)

xprop_use_stopats

Commands

set_xprop_use_stopats
get_xprop_use_stopats

Description

By default, the tool does not treat stopat nets as sources of X during X-propagation checks. Use `set_xprop_use_stopats` to change the default behavior and treat the stopat nets (including instance outputs after instance stopat) as sources of X during X-propagation checks.



This command is intended for fine-grained control over what the tool considers as sources of X during X-propagation. If you want to treat all undriven nets (black-boxed outputs, inputs, stopat nets, and internal undriven nets) as sources of X, use the command `set_xprop_use_all_undriven on` instead of this command.

CAUTION:

- If `xprop_use_all_undriven` is on, it *supersedes* any prior and future instances of `set_xprop_use_stopats`.
- If `xprop_use_all_undriven` is off, the tool *obeys* any prior and future instances of `set_xprop_use_stopats`.

Use `get_xprop_use_stopats` to determine whether the tool treats stopat nets as sources of X during X-propagation checks.

Syntax

```
set_xprop_use_stopats (on | off) [-task <task_name>]
```

on Treat stopat nets as sources of X.

off Do not treat stopat nets as sources of X. *This option is the default.*

JasperGold Apps Command Reference Manual

Configuration Commands

-task Change the sources of X for the specific task instead of for all tasks.
 task_name

Note:

- If you use the **-task** switch, the tool clears the proof status for properties in the specified task.
- If a specific task behavior is set, it has higher priority over global behaviors. However, if you set a global behavior, the tool clears all previously set task-specific behaviors.

get_xprop_use_stopats [-task <task_name>]

-task Use **-task** to get the value of the variable for the specified task.
 task_name
A task configuration value has priority over global configuration values.

Default

By default, the tool does not treat stopat nets as sources of X during X-propagation checks (`off`) and includes all tasks.

Return

`get_xprop_use_stopats` returns the value of the `xprop_use_stopats` variable.

Examples

```
% set_xprop_use_stopats on  
% get_xprop_use_stopats
```

See Also

[set_xprop_use_all_undriven](#)
[set_xprop_use_bbox_outputs](#)
[set_xprop_use_inputs](#)
[set_xprop_use_internal_undriven](#)
[set_xprop_use_low_power](#)
[set_xprop_use_reset_state](#)
[set_xprop_use_x_assignments](#)

xprop_use_x_assignments

Commands

set_xprop_use_x_assignments
get_xprop_use_x_assignments

Description

By default, the tool handles Xs in the design as sources of X during X-propagation checks. Use elaborate -disable_x_handling if you do not want Xs treated as sources of X during X-propagation checks. After elaborating the design, use set_xprop_use_x_assignments off to ignore Xs in the design during X-propagation checks.

Use get_xprop_use_x_assignments to determine whether the tool treats all Xs in the design as sources of X during X-propagation checks.

Syntax

set_xprop_use_x_assignments (on off) [-task <task_name>]	
on	Treat all Xs in the design as sources of X. <i>This option is the default.</i>
off	Do not treat Xs in the design as sources of X.
-task task_name	Change the sources of X for the specific task instead of for all tasks.
Note:	
<ul style="list-style-type: none">■ If you use the -task switch, the tool clears the proof status for properties in the specified task.■ If a specific task behavior is set, it has higher priority over global behaviors. However, if you set a global behavior, the tool clears all previously set task-specific behaviors.	
get_xprop_use_x_assignments [-task <task_name>]	
-task task_name	Use -task to get the value of the variable for the specified task. A task configuration value has priority over global configuration values.

Default

By default, the tool treats Xs as sources of X during X-propagation checks and includes all tasks.

Return

`get_xprop_use_x_assignments` Returns the value of the configuration variable `xprop_use_x_assignments`, on or off.

Examples

```
% analyze -v2k my_hdl_file.v
% elaborate
% get_xprop_use_x_assignments
% set_xprop_use_x_assignments off // Ignore Xs during X-Prop checks.
```

See Also

[set_xprop_use_all_undriven](#)
[set_xprop_use_bbox_outputs](#)
[set_xprop_use_inputs](#)
[set_xprop_use_internal_undriven](#)
[set_xprop_use_low_power](#)
[set_xprop_use_reset_state](#)
[set_xprop_use_stopats](#)

PSL Support

This appendix lists JasperGold Apps support for IEEE® Std 1850™ *Standard for PSL: Property Specification Language*. By default, JasperGold Apps analyzes embedded PSL. The table below lists various clarifications, exceptions, and limitations.

Note: To learn more about PSL, download the application note “Introduction to PSL” from Cadence Online Support (support.cadence.com). The goal of this application note is to provide you with sufficient knowledge of PSL so that you can write your own design-specific set of properties to be formally verified.

Table A-1 PSL Support Status

Item	Supported	Description
'event, rising_edge and falling_edge (VHDL)		
Yes*	JasperGold Apps does not support 'event in regular properties in the VHDL flavor of PSL.	
*In clock expressions only	As a workaround, use rose(a) or fell(a) instead of a 'event in the case of a Boolean signal a.	
Binding verification units of an HDL flavor different from the design	No	It is not possible to bind a verification unit written in the VHDL flavor to a Verilog or SystemVerilog design (and vice versa).
Binding verification units to design instances	No	Binding verification units to design instances is not supported.

JasperGold Apps Command Reference Manual
PSL Support

Table A-1 PSL Support Status, *continued*

Item	Supported	Description
Binding verification units to modules analyzed with the <code>analyze -req</code> command		
No		<p>Binding verification units to modules analyzed with the <code>analyze -req</code> command is not supported. However, you can use embedded PSL in these modules.</p> <p>If you want to bind a PSL verification unit to a requirements module that contains modeling code to be used in the properties, read the module as part of the design with the <code>analyze</code> command instead of <code>analyze -req</code> and use <code>connect -bind</code> to instantiate it in the design hierarchy.</p> <p>Example of standard requirements module usage:</p> <pre># Analyze the DUV files analyze... # Analyze the requirements files analyze -req... # Elaborate design hierarchy elaborate... # Instantiate requirements module connect...</pre> <p>Example of requirements module usage for PSL vunits:</p> <pre># Analyze the DUV files analyze... # Analyze the requirements files analyze... # Analyze the PSL vunit files analyze -psl... # Elaborate design hierarchy elaborate... # Instantiate requirements module connect -bind... -elaborate</pre>
Binding verification units to modules retrieved through the <code>analyze -y</code> and <code>-v</code> mechanisms		
Yes		If there are multiple modules with the name specified in the PSL binding, the tool uses the first one.

JasperGold Apps Command Reference Manual
PSL Support

Table A-1 PSL Support Status, *continued*

Item	Supported	Description
Compound_SERE operators &&, &, and	Yes*	The Compound_SERE operators &&, &, and do not parse with the correct precedences. This can lead to erroneous properties or strange parse error messages. As a workaround, use braces {} to control the operands to these operators.
Design wire overriding (modeling layer)	No	Design wire overriding (modeling layer) is not supported. This feature allows design nets to be overridden if the same signal name is specified in the modeling layer (having the same type). As a workaround, consider the following scenario. If the name of a modeling layer signal is the same as the name for a signal in the module to which the verification unit binds (but the intention was not to override it), change the name of the modeling layer signal to something that does not clash with a design signal name.
GDL	No	The GDL flavor, which is also known as EDL, is not supported.
Inheritance of symbols	No	Inheritance of symbols is not supported. For example, consider a case where we declare a property p in a vunit v that is inherited by vunit w. In this case, we cannot access p within w. The workaround in this case is to move (that is, copy) the definition of p to w.
isunknown()	Yes*	<ul style="list-style-type: none"> ■ isunknown(<constant>) reports true (1'b1) if the constant contains x or z. ■ Note: isunknown(<constant>) is supported in Verilog only. ■ isunknown(<signal>) always reports false (1'b0).
	*See description for restrictions.	

JasperGold Apps Command Reference Manual
PSL Support

Table A-1 PSL Support Status, *continued*

Item	Supported	Description
Local variables		
	No	Local variables in PSL properties are not supported.
Macro layer		
Yes*		With the following exceptions, the macro layer is supported inside vunits:
*See description for restrictions.		<ul style="list-style-type: none"> ■ String expressions involving %for variables are not supported. ■ The VHDL flavor #if and #elsif pre-processing directives are not supported.
modeling layer next() function		
Yes		<p>The modeling layer <code>next()</code> function is supported; however, it is not allowed directly in properties. It is a construct used in modelling code.</p> <p>Refer to Section 5.2.3.2 of the IEEE 1850 PSL LRM for additional limitations.</p> <p>Note: This item is not related to the <code>next</code> keyword for properties.</p>
Multiple clocks		
Yes*		The tool elaborates properties sampled by more than one clock, provided you use the <code>-multiple_clock</code> switch.
*See description for restrictions.		<p>Note: Using this switch can, in some cases, increase the complexity of a prove or Visualize job.</p>
Native PSL (VHDL 2008)		
No		The use of Native PSL in VHDL 2008 is not supported.
Non-determinism		
Yes*		The <code>union</code> keyword is supported; however, the built-in functions <code>nondet()</code> and <code>nondet_vector()</code> are not supported.
*See description for restrictions.		

JasperGold Apps Command Reference Manual
PSL Support

Table A-1 PSL Support Status, *continued*

Item	Supported	Description
Optional branching extension layer		
No		This CTL part of PSL is not supported. It deals with branching tree logic. This means there is no support for the operators AX, AG, AF, EX, EG, EF, A [U] and E [U].
Parameterized properties		
No		Parameterized properties are not supported.
Parameterized SERE		
No		Parameterized SEREs are not supported.
Replicated properties		
Yes*		With the following limitations, the specification of replicated properties using the <code>forall</code> keyword is supported within the VHDL flavor only: <ul style="list-style-type: none"> ■ The optional <code>Index_Range</code> field is not supported. ■ The use of replicated properties is not allowed in embedded PSL code. For more information, refer to Section 6.2.3 of the IEEE-1850 2005 LRM, <i>IEEE Standard for Property Specification Language (PSL)</i> .
*See description for restrictions		
SystemVerilog (modeling layer)		
Yes*		Verilog data types, for example, <code>reg</code> , <code>integer</code> , and <code>time</code> , are supported, but SystemVerilog data types, for example, <code>logic</code> and <code>enum</code> , are not supported.
*See description for restrictions		
Temporal parameters		
Yes*		Passing temporal parameters, sequences to sequences or sequences and properties to properties, is only supported in the VHDL flavor.
*See description for restrictions		

JasperGold Apps Command Reference Manual

PSL Support

Table A-1 PSL Support Status, *continued*

Item	Supported	Description
Unbound verification units		
No		<p>There is no support for the inheritance or elaboration of unbound verification units for use as independent property definitions.</p> <p>For more information, refer to Section 7.2.3 of the IEEE-1850 2010 LRM, <i>IEEE Standard for Property Specification Language (PSL)</i>.</p>
Using keywords <code>vunit</code> , <code>vmode</code> , and <code>vprop</code> in VHDL code		
No		<p>The PSL keywords <code>vunit</code>, <code>vmode</code>, and <code>vprop</code>, in “normal” VHDL code, are always treated as keywords; however, since PSL keywords are case sensitive, it is acceptable to use, for example, <code>vUnit</code> as a normal identifier within both VHDL and PSL code.</p> <p>In general, if an identifier is one of the PSL keywords listed above, then you cannot directly reference that identifier from within PSL expressions. Instead, use an extended identifier, for example, <code>\vunit\</code>.</p>
Verilog constructs specified in the IEEE-1850 2010 LRM, Section 8.1		
No		<p>The Verilog constructs <code>integer</code> ranges and structures, which are specified in Section 8.1 of the IEEE-1850 2010 LRM, <i>IEEE Standard for Property Specification Language (PSL)</i>, are not supported. The tool parses these constructs but ignores them and issues a warning.</p>
<code>vprop</code> cover directives		
Yes		<p>The tool accepts cover directives in <code>vprop</code> verification units and issues a warning.</p>

SVA Support

This appendix lists IEEE Std 1800TM-2005 and IEEE Std 1800-2009 SystemVerilog Assertions (SVA) support for JasperGold Apps. It also gives a brief introduction to the use flow. This appendix includes the following sections:

- [SVA Support Status](#) on page 1340
- [Additional Support for SystemVerilog 2009](#) on page 1349

SVA Support Status

JasperGold Apps supports IEEE 1800-2005 and IEEE Std 1800-2009 SystemVerilog Assertions. You can embed SVA in the design files or in a separate requirements file.

Note: See [Table B-2](#) on page 1349 for additional SystemVerilog 2009 items the tool supports if you use the command `analyze -sv09`.

The primary use flow for SVA is:

```
# If you are using SV 2009, replace "analyze -sv" with "analyze -sv09".  
% analyze -sv file.v  
% elaborate  
# SVA properties will now appear in the <embedded> task.  
% prove -all
```

[Table B-1](#) on page 1341 lists various clarifications, exceptions, and limitations in SVA support.

Note: To learn more about SVA, download the application note “Introduction to SVA” and the “SVA Quick Reference Guide” from Cadence Online Support (support.cadence.com). The goal of these publications is to provide you with sufficient knowledge of SVA so that you can write your own design-specific set of properties to be formally verified.

Table B-1 SVA Support Status

Item	Supported	Description/Additional Information
\$posedge expr \$negedge expr expr or expr	No*	When used as property and sequence arguments, these are not supported. *In clock expressions only (see Event expressions below).
\$isunknown()	Yes*	<ul style="list-style-type: none"> ■ \$isunknown(<constant>) reports true (1'b1) if the constant contains x or z. ■ \$isunknown(<signal>) and \$isunknown(<expression>) always report false (1'b0). ■ If you use elaborate -enable_sva_isunknown, \$isunknown(<signal>) and \$isunknown(<expression>) report true (1'b1) if X propagates through the specified signal or expression. <p>Note: Temporal expressions are not accepted.</p>
action_block	No	action_block specifies what actions are taken upon success or failure of the assertion. It is not applicable to formal verification. The tool ignores it and does not issue a warning.

JasperGold Apps Command Reference Manual
SVA Support

Table B-1 SVA Support Status, *continued*

Item	Supported	Description/Additional Information
bind statement	Yes*	<p>JasperGold Apps supports binding to modules and instances; however, the <code>bind</code> statement requires you to analyze the file with <code>-sv</code>.</p> <p>*See description for restrictions and guidelines.</p> <p>The tool adds properties to the <code><embedded></code> task.</p> <p>The <code>bind</code> statement can be located in the following places:</p> <ul style="list-style-type: none"> ■ Inside a separate module or instance ■ Outside any module or instance ■ Inside a property module or instance ■ Inside a design module or instance <p><i>Property files must be read as part of the design, not as requirements files.</i> If you read a property file with <code>analyze -req</code>, you will encounter the following:</p> <ul style="list-style-type: none"> ■ If <code>bind</code> is in the requirements file, the tool generates an error message (design module or instance of module not found). ■ If <code>bind</code> is in the design file or a separate module or instance, the elaboration does not find the property file, and it automatically black boxes the module or instance. <p>Parameters are supported in module instantiations.</p> <p>TIP: It is a good practice to write the <code>bind</code> statement outside modules.</p>
Binding to modules retrieved through the <code>analyze -y</code> and <code>-v</code> mechanisms	Yes	If there are multiple modules with the name specified in the SVA binding, the tool uses the first one.

JasperGold Apps Command Reference Manual

SVA Support

Table B-1 SVA Support Status, *continued*

Item	Supported	Description/Additional Information
	Binding SVA checker modules to VHDL entities	
Yes*	*See description for restrictions and guidelines.	<ul style="list-style-type: none">■ The tool accepts the following in the bind statement:<ul style="list-style-type: none"><input type="checkbox"/> Operators %, +, -, *, /, >, <, >=, <=, ==, !=, &&, , <<, >>, <<<, >>>, **, !<input type="checkbox"/> Concatenation<input type="checkbox"/> Type casts<input type="checkbox"/> Integer constants<input type="checkbox"/> Function calls<input type="checkbox"/> Array indexing<input type="checkbox"/> Array slicing<input type="checkbox"/> Selected names■ The tool supports binding to entities and architectures. To bind to an architecture, specify the name as \entity(arch). This format is required by Verilog lexical rules. If you bind to an entity name only, it will bind to all architectures of that entity.■ The tool supports binding by position and name. Use the .* syntax to bind to all ports of the checker module matching names in the architecture.■ Use constants and parameters.■ Bind statements are global and you cannot undo them. After the tool analyzes a bind statement, the only way to remove it is with clear -all. If elaboration fails, you must use analyze -clear to clear the parse tree; otherwise, the tool ignores the bind statement that failed and elaboration might finish without generating an error.

Continued below.

JasperGold Apps Command Reference Manual
SVA Support

Table B-1 SVA Support Status, *continued*

Item	Supported	Description/Additional Information
Binding SVA checker modules to VHDL entities (Continued)		
		<ul style="list-style-type: none"> ■ The tool supports interfaces in bind statement port declarations but only if each interface member is explicitly identified by its name. ■ If you enable the hierarchical reference (HR) handler released with JasperGold Apps version 2015.03 (<code>elaborate -enable_new_hierref_handler</code>), the tool supports VHDL record fields in bind statement port declarations. <p>Note: Refer to “-enable_new_hierref_handler” on page 520 for additional requirements for enabling this HR handler.</p>
<code>cover property</code>		
Yes		<p><code>cover property</code> is supported for sequences and properties. If possible, <code>cover property</code> will identify the shortest span of cycles where an instance of the property or sequence is attempted (exercised) and succeeds. This trace will be annotated with advanced debugging indicators.</p> <p>Example:</p> <pre>assert property (F); cover property (F);</pre> <ul style="list-style-type: none"> ■ If the assertion is proven and it is not vacuously true, then, in most cases, the cover will produce an interesting sanity trace for F. ■ If the assertion is vacuously true, the cover will not be satisfied. ■ If the assertion gives a counterexample, the cover might still produce a cover trace. Although not all instances are satisfied, this trace indicates an instance where the property is satisfied.
<code>dist</code>		
No		<p><code>dist</code> targets randomized simulation. It is not applicable to formal verification. The tool ignores it and does not issue a warning.</p>

JasperGold Apps Command Reference Manual
SVA Support

Table B-1 SVA Support Status, *continued*

Item	Supported	Description/Additional Information
Event expressions		
Yes*		Event operators are fully supported inside always blocks.
*See description for restrictions.		Otherwise, the only event expressions supported are Boolean expressions, and <code>posedge</code> and <code>negedge</code> on Boolean expressions.
		The following operators are not supported:
		<ul style="list-style-type: none"> ■ <code>or</code> ■ <code>,</code> ■ <code>iff</code> (This operator is supported if you use the command <code>analyze -sv09</code> or <code>analyze -sv</code>.)
expect		
No		<code>expect</code> is not applicable to formal verification. The tool ignores it and does not issue a warning.
<code>first_match(...)</code>		
Yes*		The following are not supported within an operand sequence of <code>first_match</code> :
*See description for restrictions.		<ul style="list-style-type: none"> ■ The operators <code>##0</code>, <code>[=n:m]</code>, <code>[=n:\$]</code>, <code>[>n:m]</code>, and <code>[>n:\$]</code> ■ Local variable assignments
Immediate assertions		
Yes		Immediate assertions are also known as in-line assertions.

Table B-1 SVA Support Status, *continued*

Item	Supported	Description/Additional Information
Liveness properties		
Yes*	*See description for restrictions.	<ul style="list-style-type: none"> ■ Visualize supports liveness properties as targets. Refer to the documentation for <code>visualize -force</code> and <code>visualize -at_least_once</code> for additional details about configuring liveness constraints. ■ Liveness properties containing local variable assignments are not supported for assumptions; however, they are supported for assertions. <p>To learn about liveness properties, see Appendix C, “Liveness vs. Safety.”</p>
Local variable assignments		
Yes*	*See stipulations in the next column	<p>The <code>sequence_match_item(operator_assignment or inc_or_dec_expression)</code> is also known as a local variable assignment. SVA local variable assignments are supported under the following conditions:</p> <ul style="list-style-type: none"> ■ Assignments are allowed in sequences that are in a negative context such as in the left-hand side of a suffix implication or in the scope of negation. ■ In other contexts, assignments are only allowed if, for each instance, the subsequence to the left of and including the assignment can only match in one way. <p>Note:</p> <ul style="list-style-type: none"> ■ The tool disables local variable assignments in multi-clocked properties. ■ Local variables can be used in assumptions only if for each match of the property along the trace the cycle distance between an assignment and a use of the variable is the same. For example, <pre>(c1, lvar=exp1) ##1 (c2, lvar++) [*5] => lvar<exp2;</pre>

Continued below.

JasperGold Apps Command Reference Manual

SVA Support

Table B-1 SVA Support Status, *continued*

Item	Supported	Description/Additional Information
Local variable assignments (Continued)		
Examples of local variables used in a sequence and properties:		
<pre>sequence data_check; int x; a ##1 (!a, x = data_in) ##1 !b[*0:\$] ##1 b && (data_out == x); endsequence</pre>		
<pre>property data_check_p; int x; a ##1 (!a, x = data_in) ##1 !b[*0:\$] ##1 b > (data_out == x); endproperty</pre>		
<pre>property fifo_overflow; int cnt; not ((empty, cnt=0) ##0 (1'b1, cnt = cnt+push-pop)[*0:\$] ##1 cnt==8) endproperty</pre>		
Note:		
<ul style="list-style-type: none">■ Do not attempt visualize -sanity on properties that contain local variables.■ Liveness properties containing local variable assignments are not supported for assumptions.		
Recursive properties		
No	The tool does not support directly or mutually recursive properties.	

JasperGold Apps Command Reference Manual

SVA Support

Table B-1 SVA Support Status, *continued*

Item	Supported	Description/Additional Information
Sequence properties		
Yes		<p>In SystemVerilog 2009, sequences can be either weakly embedded or strongly embedded. For example, in <code>r =>weak(s)</code>, <code>s</code> is weakly embedded, and in <code>r =>strong(s)</code>, <code>s</code> is strongly embedded. By default, without the operators <code>weak</code> and <code>strong</code>, a sequence is weakly embedded in an assertion or an assumption and strongly embedded in a cover.</p> <p>In SystemVerilog 2005 only strong embedding is available. If you intend to use properties written for 2005 with <code>analyze -sv09</code>, you can use <code>elaborate -sv09_strong_embedding</code> for backward compatibility between 2009 and 2005 versions.</p> <p>Note: The tool converts SystemVerilog 2005 strong sequence expressions into weak ones with <code>elaborate -weaken_embedding</code>, which is the opposite of <code>-sv09_strong_embedding</code>.</p>
Verification statements in initial blocks		
No		The tool ignores verification statements (<code>assert</code> , <code>assume</code> , or <code>cover</code>) in initial blocks along with everything else in initial blocks, and it issues a warning.

Additional Support for SystemVerilog 2009

In addition to the items in [“SVA Support Status”](#) on page 1341, JasperGold Apps offers support for the following IEEE Std 1800-2009 SystemVerilog Assertion items if you use the command `analyze -sv09`.

Table B-2 Additional Support for System Verilog 2009

Item	Description/Additional Information
case statements inside properties	For example: <pre>property P (logic [1:0] delay); @(posedge clk) case (delay) 2'd0: a -> b; 2'd1: a -> ##2 b; 2'd2: a -> ##4 b; 2'd3: a -> ##8 b; default: 0; endcase endproperty</pre>
restrict statements	
.triggered	
let	
##[*] and ##[+]	
->	
<->	
expr implies expr	
expr iff expr	

Table B-2 Additional Support for System Verilog 2009, continued

Item	Description/Additional Information
nexttime	
s_nexttime	
s_always	
until	
s_until	
until_with	
s_until_with	
s_eventually	
eventually	
reject_on	
sync_reject_on	
accept_on	
sync_accept_on	
default disable iff	
Sequence properties	<p>In SystemVerilog 2009, sequences can be either weakly embedded or strongly embedded. For example, in <code>r =>weak(s)</code>, s is weakly embedded, and in <code>r =>strong(s)</code>, s is strongly embedded. By default, without the operators <code>weak</code> and <code>strong</code>, a sequence is weakly embedded in an assertion or an assumption and strongly embedded in a cover.</p> <p>In SystemVerilog 2005 only strong embedding is available. If you intend to use properties written for 2005 with <code>analyze -sv09</code>, you can use <code>elaborate -sv09_strong_embedding</code> for backward compatibility between 2009 and 2005 versions.</p> <p>Note: The tool converts SystemVerilog 2005 strong sequence expressions into weak ones with <code>elaborate -weaken_embedding</code>, which is the opposite of <code>-sv09_strong_embedding</code>.</p>

Liveness vs. Safety

A safety property states that some condition must always be fulfilled or that some condition must never happen.

A liveness property states that some condition must occur at some unspecified time in the future.

Look at the following two properties:

- Property A: After start is active, busy must be active for three consecutive cycles.
- Property B: After start is active, end must eventually become active.

Property A is a safety property since it states a condition that must always be fulfilled. Property B, on the other hand, is a liveness property since it states that the end signal must be active at some unspecified time in the future. An important difference between liveness and safety properties is that when a safety property is false, each counterexample is finite, which is not generally the case for liveness properties. A counterexample for Property B must show that it is possible for *end* to never become active, which is not possible in a finite trace.

Note: When you have a mix of liveness assumptions and safety assertions, it is important to understand that counterexamples to safety properties are always finite. Thus, if the tool is able to find a CEX for a safety assertion, it does not consider the trace beyond the cycle where the assertion fails for the first time. And typically, the liveness assumption cannot constrain this type of case.

The following example is one case that illustrates the issue. This case results in a CEX of length 1 for the safety assertion. And in this case, the tool does not look beyond cycle 1 to see if the assumption eventually holds.

```
assume property (!exp1);
assume property (c1 |=> ##[0:$] exp1);
assert property (!c1);
```

JasperGold Apps Command Reference Manual
Liveness vs. Safety

D

Tcl Support for SVA Expressions

JasperGold Apps supports SVA operators and functions in Tcl commands. The tables in this appendix list various clarifications and restrictions in temporal expression support.

- [“SVA Operators Support” on page 1354](#)
- [“SVA Functions Support” on page 1357](#)
- [“Additional Support with SystemVerilog 2009 Expression Mode” on page 1357](#)

Table D-1 SVA Operators Support

Item	Supported	Description/Additional Information
Temporal Operators		
->	overlapping implication	
Yes		Note: JasperGold Apps supports the <code>s #-# p</code> operator, which is an alias for <code>not (s -> not p)</code> .
=>	nonoverlapping implication	
Yes		Note: JasperGold Apps supports the <code>s #=# p</code> operator, which is an alias for <code>not (s => not p)</code> .
##n cycle delay		
Yes*		<code>n</code> is a non-negative integer.
*See description for clarification.		JasperGold Apps supports the following delay ranges: <ul style="list-style-type: none"> ■ <code>#<num></code> ■ <code>#<constant id></code> ■ <code>#(<expr>)</code> ■ <code># [<left> : <right>]</code> ■ <code># [<leftLim> : \$]</code> Note: <code><expr></code> , <code><left></code> and <code><right></code> must compute to constants.

Table D-1 SVA Operators Support, continued

Item	Supported	Description/Additional Information
Repetition Operators		
<p>Note: When defining an interval using parameters or constant identifiers, the expression parser sometimes does not recognize the interval unless you add white space before and after the colon as shown below and in the examples in this appendix:</p>		
	<value1> : <value2> <value1> : \$	
e[*n]	consecutive repeat	<p>Yes*</p> <p>*See description for clarification.</p> <p>This operator supports non-negative integers and the following interval specification:</p> $<leftLim> : <rightLim>$ $<leftLim> : \$$ <p>Note: $<leftLim>$ and $<rightLim>$ are expressions that must compute to constants.</p>
e[=n]	nonconsecutive repeat	<p>Yes*</p> <p>*See description for clarification.</p> <p>This operator supports non-negative integers and the following interval specification:</p> $<leftLim> : <rightLim>$ $<leftLim> : \$$ <p>Note: $<leftLim>$ and $<rightLim>$ are expressions that must compute to constants.</p>
e[>-n]	goto repeat	<p>Yes*</p> <p>*See description for clarification.</p> <p>This operator supports non-negative integers and the following interval specification:</p> $<leftLim> : <rightLim>$ $<leftLim> : \$$ <p>Note: $<leftLim>$ and $<rightLim>$ are expressions that must compute to constants.</p>

Table D-1 SVA Operators Support, continued

Item	Supported	Description/Additional Information
Sequence Operators		
disable iff	Yes*	Using the syntax <code>disable_ifff</code> is not supported.
	*See description for clarification.	
Array Operators		
\$left \$right \$low \$high \$increment \$size \$dimensions \$unpacked_dimensions	No	
Set Membership Operator		
inside	No	
Clocks		
Yes*	Only one clock per temporal expression is supported. By default, temporal expressions use the primary clock. Use the following operators to change the expression's clock:	
*See description for restrictions.	<pre>@(posedge signal) texpr @(negedge signal) texpr</pre>	
	<p>Use <code>clock -both_edges</code> if you are targeting phases of the clock different than the one declared by the <code>clock</code> command. (See “clock” on page 455.)</p>	

Table D-2 SVA Functions Support

Item	Supported	Description/Additional Information
Temporal Functions		
\$past	Yes*	In the current implementation of \$past, \$past flops are left undriven in the reset cycle and their reset values are defined by reset analysis. *See description for clarification.
Math Functions		
\$clog2	No	
Data Query Functions		
\$isunbounded	No	
Hierarchical Functions		
\$root	No	
User-Defined Functions		
	No	

Note: [Table D-3](#) on page 1357 lists additional support provided with the elaborate -sv09_expression_mode command.

Table D-3 Additional Support with SystemVerilog 2009 Expression Mode

Item	Supported	Description/Additional Information
expr implies expr		
expr iff expr		
Yes		

Table D-3 Additional Support with SystemVerilog 2009 Expression Mode, *continued*

Item	Supported	Description/Additional Information
nexttime s_nexttime s_always expr until expr expr s_until expr expr until_with expr expr s_until_with expr s_eventually eventually reject_on sync_reject_on accept_on sync_accept_on	Yes *See description for clarification.	You must use the following syntax with operators <code>reject_on</code> , <code>sync_reject_on</code> , <code>accept_on</code> , and <code>sync_accept_on</code> : <code><operator>(expr) expr</code>
Sequence Operators		
[+] and [*] .triggered	Yes	
weak strong		
Yes*		To have a <code>sequence_expr</code> at the right-hand side of <code> =></code> , it needs to be “lifted” to the level of a <code>property_expr</code> . This is called “lifting embedding.” See the following SVA 2009 examples where <code>r</code> and <code>s</code> are both <code>sequence_expr</code> :
*See description for clarification.		<ul style="list-style-type: none"> ■ <code>r => weak(s)</code> ■ <code>r => strong(s)</code>
Sample Functions		
\$changed \$sampled	Yes	



OVL Support

OVL is the Open Verification Library that defines assertion checkers implemented in Verilog[®], SVA, PSL, and VHDL. JasperGold Apps supports OVL versions 1.6 through 2.8.1. For OVL-specific technical information and the latest OVL version, refer to the Accellera committee home page <http://www.accellera.org/activities/working-groups/ovl>. This appendix includes the following sections:

- [OVL v2.8.1 Support](#) on page 1360
- [Special Handling for OVL Assertion Checkers](#) on page 1360
- [Basic Flow for OVL Use](#) on page 1361

OVL v2.8.1 Support

JasperGold Apps supports all OVL v2.8.1 SVA assertion checkers except `ovl_next_state`. The following table provides additional information.

Checker	Implementation	Support	Issue
<code>ovl_next_state</code>	SVA	None	This checker contains mixed blocking and non-blocking assignments, which JasperGold Apps does not support.

JasperGold Apps also supports the VHDL implementation of OVL. To use the VHDL implementation, you must use the synthesizable source, which does not contain the `path_name` attribute. The code can be found inside the directory `std_ovl/vhdl193/syn_src`. Refer to the example in [“VHDL Implementation”](#) on page 1363.

Special Handling for OVL Assertion Checkers

The following OVL assertion checkers require special handling when used with JasperGold Apps.

Assertion Checker	<code>assert_frame</code> <code>assert_handshake</code> <code>assert_quiescent_state</code> <code>assert_width</code>	<code>ovl_frame</code> <code>ovl_handshake</code> <code>ovl_quiescent_state</code> <code>ovl_reg_loaded</code> <code>ovl_width</code>
Issue	The SVA versions of these OVL assertion checkers use the <code>\$rose()</code> function in the antecedent expression of their properties. If the assertion checkers are implemented as assumptions, they might not behave as expected. For example, consider a case where <code>\$rose(test_expression)</code> is in the antecedent of the property. If <code>test_expression</code> is asserted in the first cycle during the proof, its rising edge might not be detected as expected with the <code>\$rose()</code> function.	

Workaround	Prevent the <code>test_expression</code> from being asserted in the first cycle of the proof by adding the following in your Tcl script: <code>assume -bound 1 {<test_expression>} == 1'b0}</code>
-------------------	---

Assertion Checker	<code>assert_handshake</code>	<code>ovl_handshake</code>
Issue	If you elaborate with <code>-sv</code> or <code>-sv05</code> , the SVA implementation of this assertion checker uses a strong embedding approach.	
	To allow the tool to process this assertion checker, add the <code>-weaken_embedding</code> option to the <code>elaborate</code> command in your Tcl scripts.	
Workaround	Note: This workaround applies to designs elaborated with <code>-sv</code> and <code>-sv05</code> . If you elaborate with <code>-sv09</code> , ignore the workaround. Refer to “Sequence properties” on page 1348 for additional information about elaboration differences for <code>-sv09</code> .	

Basic Flow for OVL Use

If the design contains embedded OVL assertion checkers, analyze the standard OVL assertion checker files as design modules not as library modules. Refer to the example Tcl scripts below.

Note:

- Analyzing the OVL assertion checkers with the `OVL_SVA` or `OVL_PSL` define tells the tool to extract properties.
- JasperGold Apps is able to handle X-value checking; however, you must elaborate with `-enable_sva_isunknown` in these cases.

SVA Implementation

```
# -----
# Sample Tcl Script for SVA Implementation of OVL v2.8.1
# -----
analyze \
+define+OVL_SVA \
+define+OVL_ASSERT_ON \
+define+OVL_COVER_ON \
+define+OVL_GATING_OFF \
+define+OVL_COVERGROUP_OFF \
+define+OVL_SYNTHESIS \
+incdir+./std_ovl \
-sv [glob ./std_ovl/ovl_*.v]
analyze <design_files>
elaborate -top <top module>
```

PSL Implementation

```
# -----
# Sample Tcl Script for PSL Implementation of OVL v2.8.1
# -----
analyze \
+define+OVL_PSL \
+define+OVL_ASSERT_ON \
+define+OVL_COVER_ON \
+define+OVL_GATING_OFF \
+define+OVL_COVERGROUP_OFF \
+define+OVL_SYNTHESIS \
+incdir+./std_ovl \
-sv [glob ./std_ovl/ovl_*.v]
analyze \
+define+OVL_PSL \
+define+OVL_ASSERT_ON \
+define+OVL_COVER_ON \
+define+OVL_GATING_OFF \
+define+OVL_COVERGROUP_OFF \
+define+OVL_SYNTHESIS \
+incdir+./std_ovl \
-sv -psl [glob ./std_ovl/psl11/vunits/*.psl]
```

```
analyze <design_files>
elaborate -top <top module>
```

Note: For the PSL version of OVL assertion checkers, JasperGold Apps creates separate tasks for each verification unit by default. However, you can elaborate with the switch `-no_psl_tasks` if you prefer to create properties in the `<embedded>` task. Or you can link separate PSL tasks to the `<embedded>` task with the following command:

```
task -link <PSL task name> -to <embedded>
# Use "task -list" to identify extracted tasks
...
```

VHDL Implementation

```
# -----
# Sample Tcl Script for VHDL Implementation of OVL v2.8.1
# -----
analyze -vhdl std_ovl/std_ovl.vhd
analyze -vhdl std_ovl/std_ovl_components.vhd
analyze -vhdl std_ovl/vhdl93/syn_src/std_ovl_procs_syn.vhd
analyze -vhdl std_ovl/std_ovl_clock_gating.vhd
analyze -vhdl std_ovl/std_ovl_reset_gating.vhd
analyze -vhdl [glob std_ovl/ovl_*.vhd]
analyze -vhdl [glob std_ovl/vhdl93/syn_src/ovl_*.vhd]
analyze <design_files>
elaborate -top <top module>
```

Note: The VHDL implementation does not generate any property that is extracted by JasperGold Apps. Create properties manually using the “fire” port of the checker instance.

Example:

```
assert {my_ovl_checker.fire(0) = '0'}
assume {my_ovl_assumption.fire(0) = '0'}
```

JasperGold Apps Command Reference Manual
OVL Support

IP-XACT Support

You can load the IP-XACT information with the `ipxact` Tcl command. The tool processes the IP-XACT data in three different ways:

- Compares RTL-related content to the RTL
- Uses connectivity-related content with the Connectivity Verification App
- Uses register-related content with the CSR Verification App

For meta-data that can help JasperGold Apps with the verification effort but cannot be expressed with the IP-XACT standard, use the available JasperGold Apps-specific vendor extensions. See the examples in [Table F-1](#) on page 1366.

Table F-1 Vendor Extension Description

Vendor Extension	Description
<cadence:busLatencies> <jasper:busLatencies>	Specifies the latency of read and write operations in a register, field, or instance.
<cadence:accessPriority> <jasper:accessPriority>	Specifies which operation takes precedence in the case of a simultaneous read/write.
<cadence:hwAccessWriteEnable>, <cadence:hwAccessWriteLatency>, and <cadence:hwAccessWriteData> or <jasper:hwAccessWriteEnable>, <jasper:hwAccessWriteLatency>, and <jasper:hwAccessWriteData>	Defines direct write access to a register or field, while <cadence:hwAccessModifiedWriteValue> and <jasper:hwAccessModifiedWriteValue> describe how the data is manipulated before being written.
<cadence:busReadCheck> and <cadence:busWriteCheck> or <jasper:busReadCheck> and <jasper:busWriteCheck>	Further defines access, for example, read_undefined.

Summary of IP-XACT Standard

Following is a list of the most important IP-XACT elements that JasperGold flows use:

- <spirit:component>
- <spirit:design>

■ `<spirit:abstraction>` and `<spirit:busDefinition>`

The `<spirit:component>` and `<spirit:design>` elements may reference each other to describe the design. `<spirit:component>` contains most of the meta-data describing design features, while `<spirit:design>` elements define how the components are assembled. Likewise, pairs of `<spirit:abstraction>` and `<spirit:busDefinition>` describe bus interfaces. The former does so with port information and the latter with instantiation rules.

`<spirit:component>` describes the design implementation ([Table F-2](#) on page 1367). Port configuration, bus interfaces, and memory mapping comprise the most useful information for JasperGold Apps. The Connectivity Verification App extracts the most important connections from the port configuration and bus interfaces, while the CSR Verification App uses memory mapping information to extract register definitions and their access rules.

Table F-2 Highlights of IP-XACT Component Tags

IP-XACT Tag	Tag Description
<code><spirit:component></code>	Design implementation. Defines ports, memory, and interface implementations.
<code><spirit:memoryRemap></code>	Remapping rules for memory (<code>remapPort</code> and <code>remapState</code>).
<code><spirit:memoryMap></code>	Mapping rules for memory.
<code><spirit:addressBlock></code>	A block of addresses inside the memory.
<code><spirit:register></code>	Defines the register's access type, reset value, and behavior.
<code><spirit:field></code>	One slice of the register. May define specific access type and behavior.
<code><spirit:busInterface></code>	Implementation of an interface with mapping between interface and design ports.

`<spirit:design>` defines the design hierarchy and interface and port connections ([Table F-3](#) on page 1368). This information is important for finding where each signal is

located in the design and for pairing connected interfaces and exploring their connection implementation, thus generating the relevant checks.

Table F-3 Highlights of IP-XACT Design Tags

IP-XACT Tag	Tag Description
<spirit:design>	Design hierarchy and connections.
<spirit:componentInstance>	<spirit:design> references its child instances defined in <spirit:component> elements.
<spirit:hierConnection>	A connection between interfaces in a module and one of its descendants.
<spirit:interconnection>	A connection between interfaces in different branches of the hierarchy.
<spirit:adHocConnection>	A connection between two ports.

<spirit:abstraction> and <spirit:busDefinition> describe the interfaces implemented by the design, including port details and interface instantiation rules ([Table F-4](#) on page 1368). With this information, JasperGold Apps is able to verify the interface connections throughout the design.

Table F-4 Highlights of IP-XACT Interface Tags

IP-XACT Tag	Tag Description
<spirit:abstractionDefinition>	Defines the port configuration of a bus interface, with port type, direction, and other details.
<spirit:busDefinition>	Defines instantiation rules of a bus interface.

Using IP-XACT for Connectivity Verification

Use the `ipxact` and `check_conn -load_ipxact` commands to load IP-XACT and generate connectivity maps. For additional details see “[ipxact](#)” on page 696 and “[check_conn](#)” on page 241.

Using IP-XACT for CSR Verification

Use the `ipxact` and `check_csr -load_ipxact` commands to load IP-XACT and generate CSV files for the CSR Verification App. For additional details, see “[ipxact](#)” on page 696 and “[check_csr](#)” on page 291.

Vendor Extensions

Use the JasperGold Apps-specific vendor extensions in [Table F-5](#) on page 1369 to express meta-data that can help the verification effort.

Note: JasperGold Apps supports both the `jasper:` and `cadence:` prefixes in vendor extensions.

Table F-5 Vendor Extensions

CSR Attribute	Definition Level I = Instance C = CSR F = FIELD	VendorExtension: Tag Followed by Type Description
ACCESS_PRIORITY	I, C, F	<cadence:accessPriority> <jasper:accessPriority>
		Comma-separated list of up to five values from bus_read, bus_read_action, direct_read, bus_write, and direct_write
BUS_LATENCIES	I, C, F	<cadence:busLatencies> <jasper:busLatencies>
		Two natural numbers separated by a comma
BUS_READ_LATENCY	I, C, F	<cadence:busReadLatency> <jasper:busReadLatency>
		Natural number
BUS_WRITE_LATENCY	I, C, F	<cadence:busWriteLatency> <jasper:busWriteLatency>
		Natural number

JasperGold Apps Command Reference Manual
IP-XACT Support

Table F-5 Vendor Extensions

CSR Attribute	Definition Level I = Instance C = CSR F = FIELD	VendorExtension: Tag Followed by Type Description
DIRECT_READ	C, F	<pre><cadence:hwAccessReadEnable> <jasper:hwAccessReadEnable></pre> <p>Boolean expression</p> <pre><cadence:hwAccessReadData> <jasper:hwAccessReadData></pre> <p>Expression</p> <pre><cadence:hwAccessReadLatency> <jasper:hwAccessReadLatency></pre> <p>Natural number</p>
DIRECT_WRITE	C, F	<pre><cadence:hwAccessModifiedWriteValue> <jasper:hwAccessModifiedWriteValue></pre> <p>oneToClear, oneToSet, oneToToggle, zeroToClear, zeroToSet, zeroToToggle, clear or set</p> <pre><cadence:hwAccessWriteEnable> <jasper:hwAccessWriteEnable></pre> <p>Boolean expression</p> <pre><cadence:hwAccessWriteLatency> <jasper:hwAccessWriteLatency></pre> <p>Natural number</p> <pre><cadence:hwAccessWriteLatency> <jasper:hwAccessWriteLatency></pre> <p>Natural number</p> <pre><cadence:hwAccessWriteData> <jasper:hwAccessWriteData></pre> <p>Expression</p> <pre><cadence:fieldAlias> <jasper:fieldAlias></pre> <p><i><register name>.<field_name></i></p>

JasperGold Apps Command Reference Manual
IP-XACT Support

Table F-5 Vendor Extensions

CSR Attribute	Definition Level I = Instance C = CSR F = FIELD	VendorExtension: Tag Followed by Type Description
LOCKABLE_BY	C	<cadence:lockableBy> <jasper:lockableBy>
		Register name
REGISTER_ALIAS_FOR	C	<cadence:registerAlias> <jasper:registerAlias>
		Register name
READ_MASKED_BY	C	<cadence:readMaskedBy> <jasper:readMaskedBy>
		Register name
WRITE_MASKED_BY	C	<cadence:writeMaskedBy> <jasper:writeMaskedBy>
		Register name
BUS_READ_CHECK	I, C, F	<cadence:busReadCheck> <jasper:busReadCheck>
		read_forbidden, read_one, read_zero, or read_undefined
BUS_WRITE_CHECK	I, C, F	<cadence:busWriteCheck> <jasper:busWriteCheck>
		write_no_effect, write_forbidden, write_reset, write_once, or write_undefined
TEST_OPTION	I, C, F	<cadence:testOption> <jasper:testOption>
		ast or asm
LOCK_SEMANTIC	C	<cadence:lockSemantic> <jasper:lockSemantic>
		SEMANTIC1 or SEMANTIC2
LOCK_VAL	C	<cadence:lockVal> <jasper:lockVal>
		Expression

JasperGold Apps Command Reference Manual
IP-XACT Support

Table F-5 Vendor Extensions

CSR Attribute	Definition Level I = Instance C = CSR F = FIELD	VendorExtension: Tag Followed by Type Description
UNLOCK_VAL	C	<cadence:unlockVal> <jasper:unlockVal>
		Expression
UNLOCK_PIN	C	<cadence:unlockPin> <jasper:unlockPin>
		Signal name
DEFAULT_RESET_SIGNAL	I	<cadence:resetSignal> <jasper:resetSignal>
		Boolean expression
RESET_SIGNAL	C, F	<cadence:resetSignal> <jasper:resetSignal>
		Boolean expression
ACCESSED_BY		<cadence:indexValue> <jasper:indexValue>
		Register name
BUS_WRITE_ENABLE	C, F	<cadence:busWriteEnable> <jasper:busWriteEnable>
		Expression
	C, F	<cadence:resetValueExpression> <jasper:resetValueExpression>
		Expression

Note: When defining <cadence:resetSignal> or <jasper:resetSignal> at the instance level, you must define it in <spirit:memoryMap>, not <spirit:component>. The reset signal is used only for the registers contained by the memory map. For all other instance extensions, define it in <spirit:component>.

Examples

The examples in this section use the `jasper:` prefix; however, JasperGold Apps supports both the `cadence:` and `jasper:` prefixes.

Latencies

For CSR verification, you can specify latency configuration for bus interface read, bus interface write, direct read, and direct write. You can specify the first two together with `<cadence:busLatencies>` as shown below:

```
<spirit:field>
  <spirit:name>Low</spirit:name>
  ...
  <spirit:vendorExtensions>
    <cadence:busLatencies>2,3</cadence:busLatencies>
  </spirit:vendorExtensions>
</spirit:field>
```

Or you can specify it separately with `<cadence:busReadLatency>` and `<cadence:busWriteLatency>`:

```
<spirit:field>
  <spirit:name>Low</spirit:name>
  ...
  <spirit:vendorExtensions>
    <cadence:busWriteLatency>2</cadence:busWriteLatency>
    <cadence:busReadLatency>3</cadence:busReadLatency>
  </spirit:vendorExtensions>
</spirit:field>
```

In both examples, the field `Low` has bus write latency of 2 and bus interface read latency of 3. In the case of direct read and write, the latency is specified together with the direct read or direct write semantics.

Direct Read and Write

Sometimes, the design may access a register or field directly. For these cases, you can use vendor extensions to describe the direct access semantics. In the example below, the signal `internal_controller` writes directly to register control. The signal `internal_controller_event` indicates when such an event happens. The latency in this

JasperGold Apps Command Reference Manual

IP-XACT Support

example is 2 and each bit with value one in the signal `internal_controller` clears the corresponding bit in the register.

```
<spirit:register>
  <spirit:name>control</spirit:name>
  ...
  <spirit:vendorExtensions>
    <cadence:hwAccessWriteEnable>
      top.internal_controller_event</cadence:hwAccessWriteEnable>
    <cadence:hwAccessWriteData>
      top.internal_controller</cadence:hwAccessWriteData>
    <cadence:hwAccessWriteLatency>
      2</cadence:hwAccessWriteLatency>
    <cadence:hwAccessModifiedWriteValue>
      oneToClear</cadence:hwAccessModifiedWriteValue>
  </spirit:vendorExtensions>
</spirit:register>
```

Similar extensions exist for direct reads as shown below:

```
<spirit:register>
  <spirit:name>data</spirit:name>
  ...
  <spirit:vendorExtensions>
    <cadence:hwAccessReadEnable>
      top.monitor_enable</cadence:hwAccessReadEnable>
    <cadence:hwAccessReadData>
      top.monitor_din</cadence:hwAccessReadData>
    <cadence:hwAccessReadLatency>
      1</cadence:hwAccessReadLatency>
  </spirit:vendorExtensions>
</spirit:register>
```

Simultaneous Read/Write

If multiple operations can happen at the same time, it is important to define which values get written to a register or read from them. Specify these semantics with `<cadence:accessPriority>`, which takes a list of at most five operations.

In the example below, direct writes happen before bus interface reads.

```
<spirit:register>
    <spirit:name>control</spirit:name>
    ...
    <spirit:vendorExtensions>
        ...
        <cadence:accessPriority>direct_write,bus_read</cadence:accessPriority>
    </spirit:vendorExtensions>
</spirit:register>
```

If you do not specify all five values, the tool generates the missing ones with the default order: bus_read, bus_read_action, direct_read, bus_write, direct_write.

Note: direct_read and direct_write are valid when there are direct read and direct write configurations.

Different Access Type Semantics with BUS WRITE CHECK

You can specify specific access types with the extensions <cadence:busReadCheck> and <cadence:busWriteCheck>.

Note: In the current release, <cadence:busReadCheck> applies to write-only, read-only, and read-write access types, while <cadence:busWriteCheck> applies to all access types.

In the example below, only the first bus write operation updates the value of field High, although bus writes are allowed at any time. Furthermore, the read operation is forbidden.

```
<spirit:field>
    <spirit:name>High</spirit:name>
    ...
    <spirit:access>write-only<spirit:access>
    <spirit:vendorExtensions>
        <cadence:busWriteCheck>write_undefined</cadence:busWriteCheck>
        <cadence:busReadCheck>read_forbidden</cadence:busReadCheck>
        <cadence:testOption>assert_bus_constraint</cadence:testOption>
    </spirit:vendorExtensions>
</spirit:field>
```

When the read operation is forbidden, the tool generates the assertion AST_FORBID_READ or the assumption ASM_FORBID_READ. The latter is the default, but the former is generated instead in this example since <cadence:testOption> has value assert_bus_constraint.

The other values for <cadence:busWriteCheck> are write_no_effect, write_forbidden, write_reset, write_undefined, and write_once. For <cadence:busReadCheck>, the other options are read_one, read_zero, and read_undefined.

Indexed or Window Registers

You can define an extension to the register bus using alternate registers and the extensions <cadence:indexValue> and <cadence:indexRegister>. A common use case is to write the index of the register to be accessed on a specific register, called index. The value can then be read from or written to another register, called data.

The example below shows the definition of a set of indexed registers and the two helper registers.

```
<spirit:register>
  <spirit:name>index</spirit:name>
  <spirit:description>Index Register</spirit:description>
  <spirit:addressOffset>0x0</spirit:addressOffset>
  <spirit:size>8</spirit:size>
  <spirit:access>write-only</spirit:access>
  <spirit:reset>
    <spirit:value>0x00</spirit:value>
    <spirit:mask>0xFF</spirit:mask>
  </spirit:reset>
</spirit:register>
<spirit:register>
  <spirit:name>data</spirit:name>
  <spirit:description>Data Register</spirit:description>
  <spirit:addressOffset>0x1</spirit:addressOffset>
  <spirit:size>8</spirit:size>
  <spirit:access>read-write</spirit:access>
  <spirit:reset>
    <spirit:value>0x00</spirit:value>
    <spirit:mask>0xFF</spirit:mask>
  </spirit:reset>
  <spirit:alternateRegisters>
    <spirit:alternateRegister>
      <spirit:name>IndexedRegister1</spirit:name>
      <spirit:access>read-write</spirit:access>
      <spirit:reset>
```

JasperGold Apps Command Reference Manual

IP-XACT Support

```
<spirit:value>0x00</spirit:value>
<spirit:mask>0xFF</spirit:mask>
</spirit:reset>
<spirit:alternateGroups>
    <spirit:alternateGroup>IndexedRegisters</spirit:alternateGroup>
</spirit:alternateGroups>
<spirit:vendorExtensions>
    <cadence:indexRegister>index</cadence:indexRegister>
    <cadence:indexValue>1</cadence:indexValue>
</spirit:vendorExtensions>
</spirit:alternateRegister>
<spirit:alternateRegister>
<spirit:name>IndexedRegister2</spirit:name>
<spirit:access>read-only</spirit:access>
<spirit:reset>
    <spirit:value>0x00</spirit:value>
    <spirit:mask>0xFF</spirit:mask>
</spirit:reset>
<spirit:alternateGroups>
    <spirit:alternateGroup>IndexedRegisters</spirit:alternateGroup>
</spirit:alternateGroups>
<spirit:vendorExtensions>
    <cadence:indexRegister>index</cadence:indexRegister>
    <cadence:indexValue>2</cadence:indexValue>
</spirit:vendorExtensions>
</spirit:alternateRegister>
</spirit:alternateRegisters>
</spirit:register>
```

The indexed registers are defined as `<spirit:alternateRegister>` elements of the data register. Their indexes are defined by the vendor extension `<cadence:indexValue>` and the index register is specified by `<cadence:indexRegister>`.

Note: Only one index register can be used for each set of alternate registers.

Multiple Security Levels

You can define different configurations for the same registers, like security levels. To do so, use `<spirit:memoryRemap>` elements. Each memory remap element has a corresponding remap state, which describes the condition necessary to activate the mapping.

JasperGold Apps Command Reference Manual

IP-XACT Support

JasperGold Apps defines two special remap states: JDA:SECURE and JDA:NON_SECURE. These depend on the values of the wr_secure and rd_secure bits connected to the CSR Checker Instance. You can find additional information on these remap states in the CSR application guide (*Help – Application Guides – CSR Verification*).

IP-XACT defines the `<spirit:remapState>` element to describe mapping conditions. The condition is met when all ports have the corresponding value described in the `<spirit:remapPort>` elements. See an example below with the remap state LEVEL_0.

```
<spirit:remapStates>
  <spirit:remapState>
    <spirit:name>LEVEL_0</spirit:name>
    <spirit:remapPorts>
      <spirit:remapPort spirit:portNameRef="security_level">0x0
      </spirit:remapPort>
    </spirit:remapPorts>
  </spirit:remapState>
</spirit:remapStates>
...
<spirit:memoryMap>
  ...
  <spirit:memoryRemap spirit:state="LEVEL_0">
    <spirit:addressBlock>
      ...
      <spirit:register>
        <spirit:name>REG1</spirit:name>
        <spirit:access>read-only</spirit:access>
        ...
      </spirit:register>
      ...
    </spirit:addressBlock>
  </spirit:memoryRemap>
  <spirit:memoryRemap spirit:state="JDA:SECURE">
    <spirit:addressBlock>
      ...
      <spirit:register>
        <spirit:name>REG2</spirit:name>
        <spirit:access>read-write</spirit:access>
        ...
      </spirit:register>
      ...
    </spirit:addressBlock>
  </spirit:memoryRemap>
```

```
</spirit:addressBlock>
</spirit:memoryRemap>
</spirit:memoryMap>
```

In the example above, REG1 is a read-only register when the port security_level has value 0x0. Similarly, REG2 is a read-write register when the wr_secure and rd_secure bits are high.

Reset Expression

The IP-XACT definition format requires you to use a constant as the reset value of registers. If you want to use other expression types, use the vendor extension <cadence:resetValueExpression> as shown below:

```
<spirit:field>
  <spirit:name>Low</spirit:name>
  ...
  <spirit:vendorExtensions>
    <cadence:resetValueExpression>
      {top.reset_high,top.reset_low}
    </cadence:resetValueExpression>
  </spirit:vendorExtensions>
</spirit:field>
```

In the example, the reset value for the field Low is defined by an expression. In this way, it is possible to use a reset value that changes over time or depends on dynamic logic.

Note:

- If you use <cadence:resetValueExpression>, the tool ignores <spirit:mask>.
- For the CSR Verification App, reset is active low.

JasperGold Apps Command Reference Manual
IP-XACT Support

Proof Setup

This appendix includes a summary of proof setup commands to guide you as you initiate your verification session. For each command listed, you will find a short description and the following information:

- **Task column** – Indicates context sensitivity; for example, only applies to the current task. In some cases, this column provides a way to make the command context sensitive.
- **Global column** – Indicates whether the command applies to the global environment. In some cases, this column provides a way to make the command global.

To set the task using the command line, refer to “[task](#)” on page 847.

Note:

- For more help with a command, click on the hyperlinked cross-reference, which takes you to the complete command reference.
- Also see the “Jasper Engine Selection Guide” manual available on the Cadence Online Support website (support.cadence.com) and from the tool (*Help – Mini Guides – Jasper Engine Selection*).

Table G-1 Proof Setup Commands

Command	Task	Global
abstract		
Create a counter, initial-value, register-value, reset-value, or CDC abstraction.	Default	Use the <code>-env</code> switch.

See “[abstract](#)” on page 146.

assert		
Add an assertion about the signals in the design to the current task.	Only	N/A

See “[assert](#)” on page 174.

JasperGold Apps Command Reference Manual

Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
assume		
Add an assumption about the signals in the design to the current task or all tasks.	Default	Use the <code>-env</code> switch.
Note: The <code>clock</code> and <code>reset</code> commands only consider global assumptions.		
See “ assume ” on page 187.		
autoprove		
Orchestrates a dynamic approach to verification based on the <code>prove</code> command.	Default	Use the <code>-all</code> switch to prove all assertions and covers in all tasks.
See “ autoprove ” on page 207.		
clock		
Specify a clock-related configuration.	N/A	Only
Note: The <code>clock</code> command only considers global assumptions and stopats.		
See “ clock ” on page 455.		
configure		
List or unlimit the current configuration or change it to default values.	N/A	Only
See “ configure ” on page 464.		
cover		
Add a cover directive for signals in the design to the current task.	Only	N/A
See “ cover ” on page 471.		
get_needed_assumptions		
Find the minimal set of assumptions to achieve a target status.	Only	N/A
See “ get_needed_assumptions ” on page 645.		

JasperGold Apps Command Reference Manual

Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>justify</code>		
Set up the proof to justify the values in the specified signals.	Only	N/A
See “ justify ” on page 729.		
<code>prove</code>		
Prove a single property or multiple properties with respect to a task.	Default when you use <code>prove <expr></code>	Use the <code>-all</code> switch to prove all assertions and covers in all tasks.
See “ prove ” on page 741.		
<code>reset</code>		
Specify the design reset condition.	N/A	Only
Note: The <code>reset</code> command only considers global assumptions and stopats.		
See “ reset ” on page 761.		
<code>sanity_check</code>		
Facilitate the successful setup of the global environment.	Use the <code>-task <task_name></code> switch.	Default
See “ sanity_check ” on page 782.		
<code>set_auto_disable_related_covers</code>		
Configure the tool to automatically disable related covers.	N/A	Only
Default: <code>true</code>		
See “ set_auto_disable_related_covers ” on page 952.		
<code>set_cache_proof_simplification</code>		
Enable or disable proof simplification caching. Use the <code>clear -cache_proof_simplification</code> command to clear the cache.	N/A	Only
Default: <code>off</code>		
See “ set_cache_proof_simplification ” on page 960.		
See “ clear ” on page 453.		

JasperGold Apps Command Reference Manual
Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_clock_auto_stabilize</code>		
Enable support for RTL with potential race hazards due to looping clocking signals.	N/A	Only
Default: off		
See “ set_clock_auto_stabilize ” on page 968.		
<code>set_clock_handle_self_disabling_registers</code>		
Enable support for flops that gate their own clock.	N/A	Only
Default: on		
See “ set_clock_handle_self_disabling_registers ” on page 970.		
<code>set_clock_process_reconvergent_logic</code>		
Enable the processing of reconvergent logic in clock pins to remove glitches.	N/A	Only
Default: off		
See “ set_clock_process_reconvergent_logic ” on page 972.		
<code>set_clock_structural_glitch_analysis</code>		
Enable or disable structural glitch analysis.	N/A	Only
Default: off		
See “ set_clock_structural_glitch_analysis ” on page 974.		
<code>set_dst_mode</code>		
Enable or disable Design Tunneling.	Default	Use the -env switch.
Default: off		
See “ set_dst_mode ” on page 1001.		

JasperGold Apps Command Reference Manual
Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_elaborate_quiet_trace</code>		
Specify whether elaborate automatically generates global QuietTrace soft constraints.	N/A	Only
Default: off		
See “ set_elaborate_quiet_trace ” on page 1005.		
Also see “ assume ” on page 187 (-soft -quiet, -soft -noisy, -soft -quiet_from, and -soft -max_from).		
<code>set_engine_job_timers</code>		
Specify the timers used for engine jobs.	N/A	Only
<ul style="list-style-type: none"> ■ wallclock – Use wall-clock time. ■ mixed – Revert to the behavior of versions prior to 2014.03, which used a mix of wall-clock and CPU time. 		
Default: wall-clock		
See “ set_engine_job_timers ” on page 1007.		
<code>set_engine_mode</code>		
Optimize the proof process for different types of designs and requirements. Specify a single engine or use curly braces to specify parallel-engine processing.	N/A	Only
Default: {Hp Ht N B}		
See “ set_engine_mode ” on page 1009.		
<code>set_engine_solver</code>		
Specify the solver you want the engines to use in the decision procedure for the Boolean satisfiability problem.	N/A	Only
Default: solverE		
See “ set_engine_solver ” on page 1020.		

JasperGold Apps Command Reference Manual
Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_engine_threads</code>		
Specify the upper bound on the number of threads used by each engine. Using more than one engine thread may be especially beneficial when working with engines B, Ht, or L on properties that otherwise do not converge; that is, multiple threads can be especially useful with these engines when you are in a bug-hunting mode.	N/A	Only
Default: 1		
See “ set_engine_threads ” on page 1022.		
<code>set_engineB_before_first_mode</code>		
Specify how engine B will treat any cycles before its first trace attempt.	N/A	Only
Default: skip		
See “ set_engineB_before_first_mode ” on page 1024.		
<code>set_engineB_before_increment_mode</code>		
Specify how engine B will treat any cycles between consecutive trace attempts.	N/A	Only
Default: skip		
See “ set_engineB_before_increment_mode ” on page 1026.		
<code>set_engineB_first_trace_attempt</code>		
Specify the cycle at which engine B will make its first trace attempt.	N/A	Only
Default: trace attempt 1		
See “ set_engineB_first_trace_attempt ” on page 1028.		
<code>set_engineB_single_property</code>		
Control whether engine B should prove all properties within a task in parallel (<code>false</code>) or one property at a time (<code>true</code>).	N/A	Only
Default: <code>true</code>		
See “ set_engineB_single_property ” on page 1030.		

JasperGold Apps Command Reference Manual
Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_engineB_trace_attempt_increment</code>		
Specify the increment engine B uses between one trace attempt and the next.	N/A	Only
Default: at every consecutive length (1)		
See “ set_engineB_trace_attempt_increment ” on page 1032.		
<code>set_engineB_trace_attempt_time_limit</code>		
Specify a trace attempt time limit for engine B.	N/A	Only
Default: unlimited (0)		
See “ set_engineB_trace_attempt_time_limit ” on page 1034.		
<code>set_engineB_trace_attempt_time_limit_incr</code>		
Specify the time limit increment for engine B when the specified limit for trace attempts expires.	N/A	Only
Default: unlimited (0)		
See “ set_engineB_trace_attempt_time_limit_incr ” on page 1036.		
<code>set_engineC_optimization</code>		
Specify the optimization mode for engine C:	N/A	Only
<ul style="list-style-type: none"> ■ <code>static</code> – Use the standard engine C ordering. ■ <code>dynamic</code> – Use engine C2, which has a different engine ordering scheme. ■ <code>adaptive</code> – Predict the best engine (engine C or C2) for the current design and use it. 		
Default: <code>static</code>		
See “ set_engineC_optimization ” on page 1038.		

JasperGold Apps Command Reference Manual

Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_engineCG_max_mem</code>		
Specify the memory limit for engines C and G data structures.	N/A	Only

■ It is best to set the memory limit to a value greater than 100.

■ Total memory consumption for a job will be greater than `engineCG_max_mem`.

Default: 4096MB.

See “[set_engineCG_max_mem](#)” on page 1040.

<code>set_engineD_optimization</code>		
Specify the optimization mode for engines D and I.	N/A	Only

■ standard – Do not optimize engines D and I.

■ high – For every proof attempt made by engines D and I, simplify the task of verification to benefit successive attempts.

Default: standard

See “[set_engineD_optimization](#)” on page 1042.

JasperGold Apps Command Reference Manual
Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_engineG_optimization</code>		
Specify the optimization mode for engine G:	N/A	Only
<ul style="list-style-type: none"> ■ <code>static</code> – Use the standard engine G ordering. ■ <code>dynamic</code> – Use engine G2, which has a different engine ordering scheme. ■ <code>adaptive</code> – Predict the best engine (engine G or G2) for the current design and use it. 		
Default: <code>static</code>		
See “ set_engineG_optimization ” on page 1044.		
<code>set_engineH_single_property</code>		
Control whether engine H should prove all properties within a task in parallel (<code>false</code>) or one property at a time (<code>true</code>).	N/A	Only
Default: <code>false</code>		
See “ set_engineH_single_property ” on page 1046.		
<code>set_engineJ_attempts</code>		
Define the maximum number of attempts engine J will make at every cycle to satisfy all assumptions, Visualize configurations, and latch equations. If engine J fails to satisfy all assumptions within this bound, it restarts.	N/A	Only
Default: 1000 attempts per cycle		
See “ set_engineJ_attempts ” on page 1048.		
<code>set_engineJ_max_trace_length</code>		
Change the default maximum trace length for engine J proofs.	N/A	Only
Default: 200 cycles		
See “ set_engineJ_max_trace_length ” on page 1050.		

JasperGold Apps Command Reference Manual
Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_engineJ_migrate</code>		
Enable (or disable) engine job handover to engine Q3 if the engineJ_attempts limit is exceeded.	N/A	Only
Default: <code>off</code>		
See “ set_engineJ_migrate ” on page 1052.		
<code>set_engineJ_restarts</code>		
Define the number of restarts engine J makes before abandoning the proof. A restart means starting again with trace attempt 1 and upwards re-searching for short traces. Two events trigger restart:	N/A	Only
<ul style="list-style-type: none"> ■ Failing to satisfy all constraints at a cycle within the specified number of attempts ■ Reaching <code>max_trace_length</code> 		
Default: 1000 times		
See “ set_engineJ_restarts ” on page 1054.		
<code>set_engineJ_seed</code>		
Supply a hexadecimal setting for the seed that determines how engine J searches for traces. 0 allows engine J to pick an indeterministic seed.	N/A	Only
Default: 00000000000000e		
See “ set_engineJ_seed ” on page 1056.		
<code>set_engineJ_single_property</code>		
Control whether engine J should prove all properties within a task in parallel (<code>false</code>) or one property at a time (<code>true</code>).	N/A	Only
Default: <code>false</code>		
See “ set_engineJ_single_property ” on page 1058.		

JasperGold Apps Command Reference Manual
Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_engineL_cache</code>		
Specify whether to cache cover points for engine L. Use the <code>clear -engineL_cache</code> command to clear the cache.	N/A	Only
Default: <code>off</code>		
See “ set_engineL_cache ” on page 1060.		
See “ clear ” on page 453.		
<code>set_engineL_generate_trails</code>		
Specify whether engine L will generate trails, that is, the sequence of intermediate properties used to find a trace for a particular property.	N/A	Only
Default: <code>off</code>		
See “ set_engineL_generate_trails ” on page 1062.		
<code>set_engineL_ignore_trace</code>		
Specify whether to keep determined targets as sources of intermediate states.	N/A	Only
Default: <code>on</code>		
See “ set_engineL_ignore_trace ” on page 1064.		
<code>set_engineL_max_segment_length</code>		
Specify the segment length limit.	N/A	Only
Default: unlimited (set to 0)		
See “ set_engineL_max_segment_length ” on page 1066.		
<code>set_engineL_random_diversification</code>		
Specify whether engine L will find different traces for the same property to diversify start states for other properties	N/A	Only
Default: <code>on</code>		
See “ set_engineL_random_diversification ” on page 1068.		

JasperGold Apps Command Reference Manual
Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_engineL_search_mode</code>		
Specify whether engine L will target properties in order or in parallel. Default: parallel	N/A	Only
See “ set_engineL_search_mode ” on page 1070.		
<code>set_engineL_state_removal</code>		
Specify whether engine L should pick start states based on the last-found start states (fifo) or the start states that have been most fruitful (lru).	N/A	Only
Default: lru		
See “ set_engineL_state_removal ” on page 1072.		
<code>set_engineL_tail_length</code>		
Specify how many cycles before the end of the found traces the new search will start.	N/A	Only
Default: 1 cycle		
See “ set_engineL_tail_length ” on page 1074.		
<code>set_engineQ3_max_trace_length</code>		
Specify the trace length upper limit for searches by engine Q3 when diversification is on.	N/A	Only
Default: 200		
See “ set_engineQ3_max_trace_length ” on page 1076.		
<code>set_engineQ3_random_diversification</code>		
Enable random diversification for engine Q3.	N/A	Only
Default: off		
See “ set_engineQ3_random_diversification ” on page 1078.		

JasperGold Apps Command Reference Manual
Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_engineQ3_random_diversification_factor</code>		
Specify the random diversification factor used by engine Q3. Default: .5	N/A	Only
See “ set_engineQ3_random_diversification_factor ” on page 1080.		
<code>set_engineQ3_restarts</code>		
Limit engine Q3 restarts to the specified non-negative integer attempts. Default: 1000	N/A	Only
See “ set_engineQ3_restarts ” on page 1082.		
<code>set_engineQ3_seed</code>		
Specify a hexadecimal setting as the seed for engine Q3. Default: 00000007	N/A	Only
See “ set_engineQ3_seed ” on page 1084.		
<code>set_first_trace_attempt</code>		
Specify the cycle at which engines will make their first trace attempt. Default: trace attempt 1	N/A	Only
See “ set_first_trace_attempt ” on page 1106.		
<code>set_max_trace_length</code>		
Specify the maximum length for traces in the JasperGold Visualize window. Default: unlimited (set to 0)	Use the -task <task_name> switch.	Default
See “ set_max_trace_length ” on page 1114.		

JasperGold Apps Command Reference Manual

Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_parallel_proof_mode</code>		
Enable parallel proof mode. With this variable enabled, you can run the <code>prove</code> and <code>visualize</code> commands while there is a background proof in progress.	N/A	Only
Note: This command is included with this release as a beta feature to gather feedback from early adopters and finalize implementation for a formal release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.		
Default: <code>off</code>		
See “ set_parallel_proof_mode ” on page 1118.		
<code>set_proof_simplification</code>		
Disable (or re-enable) simplification. Simplification reduces the size of the netlist processed by the engines to achieve better overall performance, particularly with engines I, C, and C2.	N/A	Only
Default: <code>on</code>		
See “ set_proof_simplification ” on page 1122.		
<code>set_proofgrid_cluster</code>		
Specify the sequence of hosts on which to spawn proof engine jobs.	N/A	Only
Default: <code>none</code>		
See “ set_proofgrid_cluster ” on page 1124.		

JasperGold Apps Command Reference Manual
Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_proofgrid_engineJ_max_jobs</code>		
Control the number of engine J jobs.	N/A	Only
<ul style="list-style-type: none"> ■ <code>N</code> – Use a positive integer <code>N</code> to limit the number of engine J jobs below the number of <code>max_jobs</code>. ■ <code>follow</code> – Make engine J reduce the number of engine jobs in the same way as other engines do when the number of remaining undetermined properties drops below <code>max_jobs</code>. ■ <code>start_cap</code> – Let the number of engine jobs be determined by the global <code>max_jobs</code> setting. 		
Default: <code>start_cap</code>		
See “ set_proofgrid_engineJ_max_jobs ” on page 1126.		
<code>set_proofgrid_engineL_max_jobs</code>		
Control the number of engine L jobs.	N/A	Only
<ul style="list-style-type: none"> ■ <code>N</code> – Use a positive integer <code>N</code> to limit the number of engine L jobs below the number of <code>max_jobs</code>. ■ <code>follow</code> – Make engine L reduce the number of engine jobs in the same way as other engines do when the number of remaining undetermined properties drops below <code>max_jobs</code>. ■ <code>start_cap</code> – Let the number of engine jobs be determined by the global <code>max_jobs</code> setting. 		
Default: <code>start_cap</code>		
See “ set_proofgrid_engineL_max_jobs ” on page 1128.		

JasperGold Apps Command Reference Manual

Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_proofgrid_cluster</code>		
Specify the sequence of hosts on which to spawn proof engine jobs.	N/A	Only
Default: none		
See “ set_proofgrid_cluster ” on page 1124.		
<code>set_proofgrid_extra_args</code>		
Set extra arguments that will be supplied to the active <code>proofgrid_mode</code> .	N/A	Only
Default: “ ”		
See “ set_proofgrid_extra_args ” on page 1130.		
<code>set_proofgrid_manager</code>		
Turn on (or off) the ProofGrid Manager feature.	N/A	Only
Note: ProofGrid Manager begins collecting data when you enable it. Activity prior to that point is not captured. For complete data collection, enable it prior to running a prove or Visualize.		
Default: <code>off</code>		
See “ set_proofgrid_manager ” on page 1132.		
<code>set_proofgrid_max_jobs</code>		
Set the limit for jobs used by ProofGrid (when not in local mode).	N/A	Only
Default: unlimited (set to 0)		
See “ set_proofgrid_max_jobs ” on page 1134.		
<code>set_proofgrid_max_local_jobs</code>		
Set the limit for jobs used by ProofGrid when in <code>local</code> mode.	N/A	Only
Default: unlimited (set to 0)		
See “ set_proofgrid_max_local_jobs ” on page 1136.		

JasperGold Apps Command Reference Manual

Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_proofgrid_mode</code>		
Specify how multiple proof engine jobs are spawned for <code>prove</code> and <code>visualize</code> .	N/A	Only
Default: <code>local</code>		
See “ set_proofgrid_mode ” on page 1138.		
<code>set_proofgrid_per_engine_max_jobs</code>		
Specify the limit for engine jobs per engine started by ProofGrid when not in <code>local</code> mode.	N/A	Only
Default: 2		
See “ set_proofgrid_per_engine_max_jobs ” on page 1141.		
<code>set_proofgrid_per_engine_max_local_jobs</code>		
Set the limit for engine jobs per engine started by ProofGrid when in <code>local</code> mode.	N/A	Only
Default: 1		
See “ set_proofgrid_per_engine_max_local_jobs ” on page 1144.		
<code>set_proofgrid_per_engine_privileged_jobs</code>		
Specify the number of licenses ProofGrid uses that will not be subject to surrender.	N/A	Only
Default: 1		
See “ set_proofgrid_per_engine_privileged_jobs ” on page 1146.		
<code>set_proofgrid_queue</code>		
Set the queue name on which to run parallel proofs.	N/A	Only
Default: “ ”		
See “ set_proofgrid_queue ” on page 1148.		

JasperGold Apps Command Reference Manual
Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
set_proofgrid_restarts		
Specify the number of restarts ProofGrid allows per engine job. Default: 10 See “ set_proofgrid_restarts ” on page 1150.	N/A	Only
set_proofgrid_shell		
Set the command used to launch ProofGrid™ jobs. Default: "/bin/sh -c 'echo =proofgrid_shell is not defined; false'" Note: If proofgrid_mode is shell, you must update the value of proofgrid_shell because the default value is merely a placeholder that indicates you need to modify it. The default value will cause the engine jobs to issue a message to the console and then fail; that is, it is not useful for running proofs.	N/A	Only
See “ set_proofgrid_shell ” on page 1152.		
set_proofgrid_shell_stop		
Set the command used to clean up ProofGrid jobs when proofgrid_mode is shell. Default: none See “ proofgrid_shell_stop ” on page 1157.	N/A	Only
set_proofgrid_skip_abandoned		
Specify whether a restarted job will return to its abandoned target. Default: off See “ set_proofgrid_skip_abandoned ” on page 1159.	N/A	Only

JasperGold Apps Command Reference Manual
Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
set_proofgrid_socket_communication		
Specify whether communication happens over a TCP/IP socket or standard in/out.	N/A	Only
Default: on		
See “ set_proofgrid_socket_communication ” on page 1161.		
set_property_delay_cycles		
Specify the number of cycles you would like to delay properties.	N/A	Only
Default: 0 cycles		
See “ set_property_delay_cycles ” on page 1167.		
set_prove_no_cover_traces		
Specify whether to generate cover traces during proofs.	N/A	Only
Default: false		
See “ set_prove_no_cover_traces ” on page 1169.		
set_prove_no_traces		
Specify whether to generate traces during proofs.	N/A	Only
Default: false		
See “ set_prove_no_traces ” on page 1171.		
set_prove_per_property_max_time_limit		
Specify the single-property engine time limit for the proof of each property.	N/A	Only
Default: unlimited (0s)		
See “ set_prove_per_property_max_time_limit ” on page 1173.		
set_prove_per_property_time_limit		
Specify the maximum time allowed for proving each property.	N/A	Only
Default: 1s		
See “ set_prove_per_property_time_limit ” on page 1176.		

JasperGold Apps Command Reference Manual
Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_prove_per_property_time_limit_factor</code>		
Set the common ratio used for a geometric progression of <code>prove_per_property_time_limit</code> or turn off geometric progression by setting the factor to 0.	N/A	Only
Default: 10		
When used with <code>set_per_property_time_limit</code> , the effective limit is the following: <code>per_property_time_limit * factor ^ iter</code>		
See “ set_prove_per_property_time_limit_factor ” on page 1179 and “ set_prove_per_property_time_limit ” on page 1176.		
<code>set_prove_prefer_shortest</code>		
Control whether <code>prove</code> will continue searching for a shorter trace after identifying a non-minimal trace.	N/A	Only
Default: <code>off</code>		
See “ set_prove_prefer_shortest ” on page 1181.		
<code>set_prove_report_mem</code>		
Specify whether to generate frequent engine job memory consumption messages.	N/A	Only
Default: <code>off</code>		
See “ set_prove_report_mem ” on page 1183.		
<code>set_prove_report_period</code>		
Set the sampling period for requesting memory reports from proof processes.	N/A	Only
Default: 10		
See “ set_prove_report_period ” on page 1185.		

JasperGold Apps Command Reference Manual

Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_prove_start_timer_on_engine_start</code>		
Start the <code>prove</code> command timer only after the first proof job starts.	N/A	Only
Default: <code>on</code>		
See “ set prove start timer on engine start ” on page 1187.		
<code>set_prove_time_limit</code>		
Specify the maximum time allowed for a <code>prove</code> command.	N/A	Only
Default: 86400 seconds (24h)		
See “ set prove time limit ” on page 1189.		
<code>set_prove_verbosity</code>		
Control the amount of information printed to the log file, the GUI <i>Console</i> pane, and the GUI <i>Proof messages</i> pane.	N/A	Only
Default: 6		
See “ set prove verbosity ” on page 1192.		
<code>set_proven_directive</code>		
Specify whether the tool uses proven directives as assumptions.	N/A	Only
Default: <code>true</code>		
See “ set proven directive ” on page 1194.		
<code>set_stop_on_cex_limit</code>		
Set the value of the <code>stop_on_cex_limit</code> variable.	N/A	Only
Default: 0 (unlimited)		
See “ set stop on cex limit ” on page 1283.		

JasperGold Apps Command Reference Manual

Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_stop_on_unreachable_limit</code>		
Set the value of the <code>stop_on_unreachable_limit</code> variable. Default: 0 (unlimited)	N/A	Only
See “ set_stop_on_unreachable_limit ” on page 1284.		
<code>set_tcl_no_precondition</code>		
Specify whether the tool generates precondition covers for liveness assertions. Default: off	N/A	Only
See “ set_tcl_no_precondition ” on page 1290.		
<code>set_trace_extension</code>		
Extend a trace <i>N</i> extra cycles beyond the cycle that triggered a property. Default: 0	N/A	Only
See “ set_trace_extension ” on page 1294.		
<code>set_word_level_reduction</code>		
Enable word-level transformations or reductions during the proof flow. Default: off	N/A	Only
See “ set_word_level_reduction ” on page 1311.		
<code>set_xprop_compute_highlights</code>		
Enable or disable X-propagation highlights. Default: on	N/A	Only
See “ set_xprop_compute_highlights ” on page 1313.		

JasperGold Apps Command Reference Manual

Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_xprop_use_all_undriven</code>		
Enable or disable treatment of all undriven nets as sources of X during X-Prop checks.	Use the <code>-task <task_name></code> switch.	Default
IMPORTANT: Using the command <code>set_xprop_use_all_undriven</code> on accomplishes the same result as the following commands combined:		
<code>set_xprop_use_bbox_outputs on</code> <code>set_xprop_use_inputs on</code> <code>set_xprop_use_stopats on</code> <code>set_xprop_use_internal_undriven on</code>		
CAUTION:		
<ul style="list-style-type: none">■ If <code>xprop_use_all_undriven</code> is on, it <i>supersedes</i> any prior and future instances of the commands listed above.■ If <code>xprop_use_all_undriven</code> is off, the tool <i>obeys</i> any prior and future instances of the commands listed above.		
Default: off		
See “ set_xprop_use_all_undriven ” on page 1315.		
<code>set_xprop_use_bbox_outputs</code>		
Disable or re-enable treatment of black-box outputs as sources of X during X-Prop checks.	Use the <code>-task <task_name></code> switch.	Default
Default: on		
See “ set_xprop_use_bbox_outputs ” on page 1317.		

JasperGold Apps Command Reference Manual
Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_xprop_use_inputs</code>		
Enable or disable treatment of primary inputs as sources of X during X-Prop checks.	Use the -task <task_name> switch.	Default
Default: off		
See “ set_xprop_use_inputs ” on page 1319.		
<code>set_xprop_use_internal_undriven</code>		
Enable or disable treatment of internal undriven nets as sources of X during X-Prop checks.	Use the -task <task_name> switch.	Default
Default: off		
See “ set_xprop_use_internal_undriven ” on page 1321.		
<code>set_xprop_use_low_power</code>		
Enable or disable the consideration of Xs generated by low-power configurations as sources of X.	N/A	Default
Default: off		
“ set_xprop_use_low_power ” on page 1323		
<code>set_xprop_use_reset_abstraction</code>		
Enable flop reset abstractions as sources of X or reinstate the default.	Use the -task <task_name> switch.	Default
Default: off		
See “ set_xprop_use_reset_abstraction ” on page 1325.		
<code>set_xprop_use_reset_state</code>		
Ignore uninitialized registers or reinstate the default register values for prove and Visualize jobs.	Use the -task <task_name> switch.	Default
Default: on		
See “ set_xprop_use_reset_state ” on page 1327.		

JasperGold Apps Command Reference Manual

Proof Setup

Table G-1 Proof Setup Commands, *continued*

Command	Task	Global
<code>set_xprop_use_stopats</code>		
Enable or disable treatment of stopat nets as sources of X during X-Prop checks. Default: off	Use the <code>-task <task_name></code> switch.	Default
See “ set_xprop_use_stopats ” on page 1329.		
<code>set_xprop_use_x_assignments</code>		
Disable or re-enable treatment of all Xs in the design as sources of X during X-Prop checks. Default: on	Use the <code>-task <task_name></code> switch.	Default
See “ set_xprop_use_x_assignments ” on page 1331.		
<code>stopat</code>		
Stop traversing a netlist at the specified signals/instances.	Default	Use the <code>-env</code> switch.
Note: You cannot use <code>-env</code> and <code>-soft</code> in the same command.		
The <code>clock</code> and <code>reset</code> commands only consider global stopats.		
See “ stopat ” on page 840.		

JasperGold Apps Command Reference Manual
Proof Setup

Custom Radix Files

JasperGold Apps supports user-created radix files to define opcodes. Use this feature to display predefined radix values (instead of the original values) on the waveform. This feature is useful for debugging in a processor setting where you want to have the instruction type displayed instead of the bits that form the opcode. This appendix includes the following sections:

- [Radix File Format](#) on page 1408
- [Example Radix File](#) on page 1410
- [Related Documentation](#) on page 1411

Note: If you load a radix file and then manually add signals to the trace, newly added signals are unaffected by the radix file. As a workaround, you can re-load the radix file so the use clause will set the radix of the newly added signals.

Radix File Format

A radix file consists of definition clauses and use clauses. Definition clauses map values to tags and use clauses map waveform signal names to a defined radix. Use clauses are optional.

Definition Clauses

Important information about your definition clauses follows:

- Values are zero extended if the size is not completely specified. For example, 10 is represented by 00000010 if the size is 8 and the base is binary.
- If you have defined the size and base, they specify the size and base for values used in the TAG <value> pairs within the begin and end block only.
 - If you do not define the size, the tool uses 8 as the default size.
 - If you do not define the base, the tool uses binary as the default base.
- If you do not specify an opcode for some values, the Visualize window displays the default radix:
 - For a signal vector that is greater than 4-bit, hexadecimal is the default base.
 - For a signal vector that is 4-bit or less, binary is the default base.
- A value must be matched by only one tag.

```
radix <radix_name> // Radix name to be displayed in GUI.
size 8              // Number of bits. Optional. Default size is 8.
base b|o|d|h       // Optional. Default base is binary.
begin
    TAG1 <value>      // Value specified in defined base.
    TAG2 <value>      // "x" is allowed as a wildcard. It is the only
                      // supported wildcard. It matches 0 or 1 characters like
                      // the regular expression wildcard "?".
                      // No wildcards are supported in decimal base.

    .
    .

end
```

Use Clauses

Important information about your use clauses follows:

- They are optional.
- You can specify the radix you want to apply to listed signals with a radix name or signal name:

```
use radix <radix_name> clause
use radix_signal <signal_name> clause
```
- When using a radix name (use radix <radix_name>), the radix you want to apply must be defined previously in the same file.
- Do not use wildcards in signal names.

```
use [radix <radix_name> | radix_signal <signal_name>]
begin
    signal signal_1, signal_2
    signal signal_3
end
```

Predefined Radix Names

To use the predefined radix names (hex, bin, dec, signed_dec¹, oct), add the following to your radix file before the use clause.

```
radix bin
begin
end
```

```
radix hex
begin
end
```

```
radix dec
begin
end
```

1. The Visualize window uses the two's complement method to represent the signed decimal notation.

```
radix signed_dec  
begin  
end
```

```
radix oct  
begin  
end
```

You can set the radix for signals to one of these five predefined radix names in use clauses, for example:

```
use radix dec  
begin  
    signal <signal_1>, <signal_2>  
    signal <signal_3>  
end  
  
use radix hex  
begin  
    signal <signal_h1>, <signal_h2>  
    signal <signal_h3>  
end
```

Example Radix File

An example of a radix file follows:

Note: The following example defines the TAG for values 0, 1, 2, and 3. Any value that falls outside of that range (0-3) will be displayed in the default radix in the Visualize window.

```
radix radix_2b  
size 3  
base d  
begin  
SIZE_0 0  
SIZE_1 1  
SIZE_2 2  
SIZE_3 3  
end
```

Related Documentation

Refer to the following sections for information about related commands and GUI options.

- “[load radix file](#)” on page 736
- “[visualize](#)” on page 864
- “Menu bar” in Chapter 4, “Visualize” of the *JasperGold Apps User’s Guide*

JasperGold Apps Command Reference Manual
Custom Radix Files

Automatically Extracted Type Properties

JasperGold Apps extracts and uses type information in prove and Visualize runs. That is, during elaboration, the tool uses the VHDL type information and SystemVerilog enum variables to create type properties. This appendix includes the following sections:

- [Introduction to VHDL Type Properties](#) on page 1414
- [Introduction to SystemVerilog Type Properties](#) on page 1415
- [Procedural Notes](#) on page 1415
- [Type Property Limitations](#) on page 1415

Introduction to VHDL Type Properties

JasperGold Apps synthesizes system VHDL signals with integer types into bit vectors during the elaboration phase. The exact number of bits used for the synthesis is the smallest number needed to encode all the values defined by the type declaration. For instance, an integer signal with the range zero to seven will be synthesized into three bits to be able to represent all values. In some cases though, an integer signal will have a range that does not contain an exact power of two elements. A signal with the type zero to six will still require three bits, but there will be one value (namely seven) that can be encoded in the bit vector that the signal should not be able to get according to its type definition. JasperGold Apps automatically extracts verification directives that check that such type properties are not violated. This feature is useful for the following main purposes:

- Environment properties – When a type property directive is generated for an input port at the top elaboration level, the tool generates an assumption for that signal that ensures that the port is never driven with a value outside of the type range. This is the only type of integer signal for which the tool generates assumptions. For all other signals, the tool generates assertions instead.
- Bug detection – If a violation of the type property can actually occur, this is a design bug that should be fixed. Type overflows can occur in synthesized versions of the integer signal, and verifying such directives provides a quick sanity check of the design under verification.
- State Space Tunneling (SST) – Values outside the type definition should not be reachable from any legal state of a design. Type properties are therefore a natural starting point for an SST attempt, especially as type violations would likely render a design inoperable.

Just like integer signals, signals of enumeration types will be synthesized as bit vectors. Type properties will therefore be generated for such signals if the number of enumeration literals in the type declaration is not an exact power of two. For instance, directives will be generated for signals of the type “Color” below since it contains three literals, but not for the type “Direction,” which contains four.

- Type Color is (Red, Blue, Green)
- Type Direction is (North, East, South, West)

Note: JasperGold Apps does not generate type properties for variables.

Introduction to SystemVerilog Type Properties

SystemVerilog enumerated data types provide the capability to abstractly declare strongly typed variables without either a data type or data value and later adds the required data type and value for designs that require more definition. Enumerated data types also can be easily referenced or displayed using the enumerated names as opposed to the enumerated values.

To avoid the assignment of non-valid values for the enumeration type inputs, the tool creates assumptions that guarantee that only the listed values will be considered. On the other hand, for enumeration type outputs, the generated properties are assertions that check whether, in any situation, an invalid value for the signal can be reached.

The tool automatically extracts type properties for enumerations in which the number of enumeration literals is not an exact power of two. An enumerated type declares a set of integral named constants.

Procedural Notes

By default, the tool extracts VHDL type properties. However, for SystemVerilog, you must explicitly specify extraction with the `elaborate` switch `-enable_sv_type_properties`. With automatic extraction enabled, the tool creates the `<constraints>` task for type properties. By default, all the other tasks inherit the properties in the `<constraints>` task.

Note:

- See “[elaborate](#)” on page 501 for command details.
- `elaborate -enable_sv_type_properties` is included with this release as a beta feature to gather feedback from early adopters and finalize implementation for a formal release. Contact support@cadence.com to provide feedback or comments targeting the upcoming production version.

Type Property Limitations

This section includes a list of known limitations for extracting type properties.

- If you use the VHDL attribute `enum_encoding`, extracted type properties for enums may have the wrong expression.

Example:

```
attribute enum_encoding : string ;
type my_state is (start, stop, ready, off, warmup);
```

JasperGold Apps Command Reference Manual

Automatically Extracted Type Properties

```
attribute enum_encoding of my_state : type is "001 010 011 100 111";
```

- The tool may not extract type properties when integer ranges depend on VHDL generics specified with `elaborate -parameter`.

Example:

```
entity fifo is
  generic (fifo_length : fifo_length := 8);
  port (level      : out integer range 0 to fifo_length);
end fifo;

analyze -vhdl test.vhd
elaborate -vhdl -top fifo -parameter fifo_length 512
```

- JasperGold Apps does not extract type properties from Proof Accelerator modules.
- Because undriven SystemVerilog enums are possible sources of spurious counterexamples in other assertions, the tool creates type assertions for them instead of assumptions.
- Type properties in black-boxed modules or instances may not work as expected.
- Type properties in signals with extended names or escaped identifiers may not work as expected.

Vacuity Checking with Automatically Extracted Covers

By default, JasperGold Apps extracts cover properties for PSL and SVA assertions and assumptions that include triggering expressions. These cover properties are called preconditions. If the tool proves that an extracted cover property is unreachable, the related assertion is vacuous and the property table displays a red exclamation point in the same cell with the property status indicator. In addition to the automatically extracted covers, you can specify that you want the tool to extract late precondition and witness covers.

This appendix includes the following:

- [Precondition Reachability Checking](#) on page 1418
- [Refining Vacuity Checking with Additional Covers](#) on page 1420
- [Managing Vacuity Checking](#) on page 1421
- [Enabling and Disabling Related Covers](#) on page 1421
- [Limitations](#) on page 1422

Precondition Reachability Checking

JasperGold Apps elaborates PSL and SVA properties embedded in design files or in separate verification modules. Assertions and assumptions often use a precondition (triggering expression) to specify when the test expression should be checked. For assertions, if the precondition is unreachable, that is, it never triggers (always false: `1'b0`), the assertion becomes vacuous. In other words, the property does not check what was intended, and the tool will never find a violation trace regardless of the logic driving the test expression.

In the default operating mode, JasperGold Apps automatically extracts precondition reachability checks for liveness assertions added on the fly with the `assert` command and for embedded PSL and SVA assertions and assumptions. The tool creates additional cover properties for each property (assertion or assumption) where it detects preconditions and adds them to the task in which the original property resides. These cover properties check whether preconditions are indeed reachable based on the related logic and environment constraints. To specify that you do not want the tool to extract precondition reachability checks for embedded properties, use the command `elaborate -no_preconditions`.

Note:

- Automatically extracted precondition reachability checks also apply to liveness assertions you create on the fly with the `assert` command.
- By default, the tool names an automatically extracted cover property after the original property with a `:preconditionN` suffix.

Property Examples

For the following PSL and SVA examples, the tool extracts a cover property equivalent to `req` evaluates to true:

- PSL property (Verilog)
`assert (always ({req} |=> {ack})) @ (posedge clk);`
- PSL property (VHDL)
`assert (always ({req = '1'}) |=> {ack = '1'})) @ (rose (clk));`
- SVA property
`assert property (@(posedge clk) req |=> ack);`

SVA Immediate Properties

The following example shows SVA immediate assertions embedded in `always` blocks. The tool extracts cover properties for the assertions `assert1`, `assert2`, and `assert3`.

```
always @(posedge clk) begin
    if (cond1) begin
        ...
        assert1: assert (test_expr1);
        // The tool extracts a cover property such as "cond1 is asserted".
    end

    else if (cond2)
        ...
        assert2: assert (test_expr2);
        // The tool extracts a cover property such as
        // "cond1 is deasserted and cond2 is asserted".
    end

    else begin
        ...
        assert3: assert (test_expr3);
        // The tool extracts a cover property such as
        // "cond1 and cond2 are deasserted".
    end

end
```

Nested Implications

As shown in the following SVA examples, the tool generates a separate precondition for every implication in the property:

- For properties of the form `s1 | -> (s2 | -> P)`, the tool produces two precondition covers, `s1` and `s1##1s2`.
- For sub-properties consisting of nested instances of `if a ... else ...`, the tool takes this context into account. For example, given the following property,

```
a | ->
    if (b)
        if_b
    else if (c)
        if_c
    else if (d)
        if_d
    else if (e)
```

```
if_e  
$display ("something");
```

the tool generates the following set of preconditions:

- a
- a&&b
- a&&!b&&c
- a&&!b&&!c&&d
- a&&!b&&!c&&!d&&e

Refining Vacuity Checking with Additional Covers

To refine vacuity checking for embedded properties, use the command `elaborate -create_related_covers ...`. This switch accepts Tcl lists that specify one or more types of cover statements (`precondition`, `late_precondition`, and `witness`).

Note: Using `-create_related_covers` in `-f` files is prohibited. It triggers an error.

The tool creates related covers as follows. In these examples, `s1` and `s2` are sequences.

- The late precondition cover of an SVA implication `s1 | => s2` will be a cover of the sequence `(s1 ##1 1'b1)`.
- The witness cover of an SVA implication `s1 | => s2` will be the cover of the sequence `(s1 ##1 s2)`.
- The witness cover of an SVA implication `s1 | -> s2` will be the cover of the sequence `(s1 ##0 s2)`.

Note:

- In many cases, when a property contains a negated sequence, for example, `B | -> not (A[*4:4])`, the tool does not produce a witness cover.
- The tool's default behavior is equivalent to `elaborate -create_related_covers precondition`.
- It is an error to use `elaborate -no_preconditions` in combination with the switch `-create_related_covers precondition`.
- The elaborate switches `-no_preconditions` and `-create_related_covers` do not apply to liveness assertions created on the fly with the `assert` command. Instead,

use the switch `-no_precondition` with the `assert` command or the command `set_tcl_no_precondition` on.

- The tool does not extract witness precondition covers or late precondition covers for liveness assertions created with the `assert` command.
- The tool does not support witness precondition covers for properties with local variable assignments.

The recommended engines for optimal precondition reachability checks are the following:

- **Multiple properties** – engine Ht
- **Single property** – engine B

Managing Vacuity Checking

Default preconditions cover the majority of vacuous proof situations; however, in some cases, the basic precondition is reachable but the witness precondition of the same property is not. This kind of vacuous proof can go unnoticed if you rely only on the default preconditions. Furthermore, the tool is able to generate basic precondition covers only for implication properties. Non-implication properties do not have default preconditions, but the tool can generate witness preconditions for them and thereby provide broader vacuity detection.

Witness preconditions have the disadvantage of increasing computational overhead, so we do not recommend using them in all cases. Carefully consider your design and proof environment before taking this step. If the majority of your assertions have implications and your environment has few assumptions, witnesses will probably add little value. However, if your proof environment has many related assumptions or few implication assertions, witness preconditions may greatly improve vacuity detection.

In general, we recommend a mixed-use approach. For daily work and nightly regressions, generate only the default preconditions to get reasonable confidence with fast performance. Then, periodically generate the related covers for a deep scan of the proof environment.

Enabling and Disabling Related Covers

By default, if you disable a property, its enabled related covers are also disabled. Hence, related covers can be disabled for two reasons:

- You have explicitly disabled them.
- You have explicitly disabled all their main properties; that is, you have disabled the properties that own the related covers.

This is the handling you can expect:

- If you disable the main property, the related covers are also disabled unless you run the command `set_auto_disable_related_covers off`.
- If you subsequently re-enable the main property, its related covers are also re-enabled unless you directly disabled the related covers. In that case, the covers remain disabled regardless of the state of their main properties.
- If you explicitly enable a disabled related cover, it is re-enabled even if the main property is disabled.
- Explicitly enabling a related cover does not automatically enable the main property.

Limitations

JasperGold Apps does not automatically generate cover properties for the following cases:

- Properties written in Verilog or VHDL. These properties do not have explicit preconditions.
- Non-liveness assertions you create with the `assert` command.
- Assumptions you create with the `assume` command.

Understanding Visualize Semantics with SVA Equivalents

JasperGold Apps adds the keywords `BT` (beyond trace) and `RC` (reset cycle) and the Boolean `NOCEX(p)` to the SVA language for the purpose of explaining the meaning of the `visualize` command. This appendix begins with an explanation of those keywords and the Boolean followed by SVA equivalents of `visualize` commands. The topics in this appendix are listed below:

- [SystemVerilog BT/RC Keywords and NOCEX Boolean](#) on page 1424
- [SystemVerilog Equivalent of Visualize At Least Once Options](#) on page 1424
- [SystemVerilog Equivalent of Visualize Force Options](#) on page 1426
- [SystemVerilog Equivalent of Visualize Not Options](#) on page 1427
- [SystemVerilog Equivalent of Visualize for a Signal Sequence](#) on page 1427

SystemVerilog BT/RC Keywords and NOCEX Boolean

This section describes the added SystemVerilog BT and RC keywords and NOCEX (p) Boolean.

SystemVerilog BT Keyword

The SystemVerilog BT keyword added by JasperGold Apps ensures that the cover covers all of the finite trace produced. This is only needed for the [n:\$] case of -force. Refer to ["SystemVerilog Equivalent of Visualize Force Options"](#) on page 1426.

The following BT expression is something that is only true beyond the finite trace considered here. You can think of it as having the meaning `cycle>trace_length` where `cycle` is the number of the current cycle and `trace_length` is the length of the current trace. This cover will match on an extension (by one cycle) of the finite trace that is produced.

```
cover property ##(n-1) exprA[*0:$] ##1 BT;
```

SystemVerilog RC Keyword

The SystemVerilog RC keyword added by JasperGold Apps matches the reset cycle, which is the cycle immediately before the trace. In the examples found in this appendix, the RC signal (reset cycle signal) is thought of as true at and only at the cycle before the first cycle.

SystemVerilog NOCEX(p) Boolean

The SystemVerilog NOCEX (p) Boolean added by JasperGold Apps is true at a cycle if and only if there is no counterexample to the property ending in that cycle. For example, where `p = 'A |=>B'`, `NOCEX(p) = !((A ##1 !B).triggered)`.

SystemVerilog Equivalent of Visualize At Least Once Options

The following examples show `visualize -at_least_once`, `visualize -at_least_once -after`, `visualize -at_least_once -before`, and `visualize`

JasperGold Apps Command Reference Manual

Understanding Visualize Semantics with SVA Equivalents

-at_least_once -after -before commands with their SVA equivalents when taking expressions as arguments.

Visualize Construct	Equivalent SVA Cover Property
visualize -at_least_once exprA n:m	RC ##[n:m] exprA
visualize -at_least_once exprA n:\$	RC ##[n:\$] exprA
visualize -at_least_once exprA \$:\$	exprA ##1 BT
visualize -at_least_once exprA n:m -after exprB	exprB ##[n:m] exprA
visualize -at_least_once exprA n:\$ \ -after exprB	exprB ##[n:\$] exprA
visualize -at_least_once exprA n:m -before exprB	exprA ##[n:m] exprB
visualize -at_least_once exprA n:\$ \ -before exprB	exprA ##[n:\$] exprB
visualize -at_least_once exprA n:m -after exprB -before exprC	exprB ##1 exprA[=n:m] ##1 exprC
visualize -at_least_once exprA n:\$ \ -after exprB -before exprC	exprB ##1 exprA[=n:\$] ##1 exprC

The following examples show visualize -at_least_once, visualize -at_least_once -after, and visualize -at_least_once -before commands with their SVA equivalents when taking sequences as arguments.

Visualize Construct	Equivalent SVA Cover Property
visualize -at_least_once seqA n:m	RC ##[n:m] seqA
visualize -at_least_once seqA n:\$	RC ##[n:\$] seqA
visualize -at_least_once seqA \$:\$	RC ##[\$:\$] seqA
visualize -at_least_once seqA n:m \ -after exprB	exprB ##[n:m] seqA
visualize -at_least_once seqA n:\$ \ -after exprB	exprB ##[n:\$] seqA
visualize -at_least_once seqA n:m \ -before exprB	seqA ##[n:m] exprB
visualize -at_least_once seqA n:\$ \ -before exprB	seqA ##[n:\$] exprB

JasperGold Apps Command Reference Manual

Understanding Visualize Semantics with SVA Equivalents

Visualize Construct	Equivalent SVA Cover Property
visualize -at_least_once exprA n:m -after exprB -before exprC	exprB ##1 exprA[=n:m] ##1 exprC
visualize -at_least_once exprA n:\$ \ -after exprB -before exprC	exprB ##1 exprA[=n:\$] ##1 exprC

SystemVerilog Equivalent of Visualize Force Options

To help explain `visualize -force`, the following examples show `visualize` commands with their SVA equivalents when taking expressions as arguments.

Visualize Construct	Equivalent SVA Cover Property
visualize -force exprA n:m	RC ##n exprA[*m-(n-1)]
visualize -force exprA n:\$	RC ##n exprA[*1:\$] ##1 BT
visualize -force exprA \$:\$	exprA ##1 BT
visualize -force exprA n:m \ -after exprB	exprB ##n exprA[*m-(n-1)]
visualize -force exprA n:\$ \ -after exprB	exprB ##n exprA[*1:\$] ##1 BT
visualize -force exprA n:m \ -before exprB	exprA[*m-(n-1)] ##n exprB
visualize -force exprA n:\$ \ -before exprB	RC ##1 exprA[*1:\$] ##n exprB

To help explain `visualize -force`, the following examples show `visualize` commands with their SVA equivalents when taking properties as arguments.

Visualize Construct	Equivalent SVA Cover Property
visualize -force propertyA n:m	RC ##n nocex(propertyA) [*m-(n-1)]
visualize -force propertyA n:\$	RC ##n nocex(propertyA) [*1:\$] ##1 BT
visualize -force propertyA \$:\$	nocex(propertyA) ##1 BT
visualize -force propertyA n:m \ -after exprB	exprB ##n nocex(propertyA) [*m-(n-1)]
visualize -force propertyA n:\$ \ -after exprB	exprB ##n nocex(propertyA) [*1:\$] ##1 BT

Visualize Construct	Equivalent SVA Cover Property
visualize -force propertyA n:m \ -before exprB	nocex(propertyA) [*m-(n-1)] ##n exprB
visualize -force propertyA n:\$ \ -before exprB	RC ##1 nocex(propertyA) [*1:\$] ##n exprB

SystemVerilog Equivalent of Visualize Not Options

To help explain `visualize -not`, the following examples show `visualize` commands with their SVA equivalents.

Visualize Construct	Equivalent SVA Cover Property
visualize -not exprA n:m	RC ##n ~exprA[*m-(n-1)]
visualize -not exprA n:\$	RC ##n ~exprA[*1:\$] ##1 BT
visualize -not exprA \$:\$	~exprA ##1 BT
visualize -not exprA n:m \ -after exprB	exprB ##n ~exprA[*m-(n-1)]
visualize -not exprA n:\$ \ -after exprB	exprB ##n ~exprA[*1:\$] ##1 BT
visualize -not exprA n:m \ -before exprB	~exprA[*m-(n-1)] ##n exprB
visualize -not exprA n:\$ \ -before exprB	RC ##1 ~exprA[*1:\$] ##n exprB

SystemVerilog Equivalent of Visualize for a Signal Sequence

To help explain `visualize -signal -sequence`, the following examples show `visualize` commands with their SVA equivalents.

Visualize Construct	Equivalent SVA Cover Property
visualize -signal A -sequence \ {0 1 1 0 0 1 1 1 0}	cover property (RC ##1 A==0 ##1 A==1 ##1 A==1 ##1 A==0 ##1 A==0 ##1 A==1 ##1 A==1 ##1 A==1 ##1 A==0);

JasperGold Apps Command Reference Manual

Understanding Visualize Semantics with SVA Equivalents

Visualize Construct	Equivalent SVA Cover Property
visualize -signal B -sequence \ {0 1{3} 0 1 0}	cover property (RC ##1 B==0 ##1 B==1[*3] ##1 B==0 ##1 B==1 ##1 B==0);
visualize -signal C -sequence \ {0 1 1 1 0 0 x x 0}	cover property (RC ##1 C==0 ##1 C==1 ##1 C==1 ##1 C==1 ##1 C==0 ##1 C==0 ##1 C==x ##1 C==x ##1 C==0);
visualize -signal D -sequence \ {0 1{3} 0{2} x{3} 0}	cover property (RC ##1 D==0 ##1 D==1[*3] ##1 D==0[*2] ##1 D==x[*3] ##1 D==0);
visualize -signal busA -sequence \ {8'b0001 0000 8'hffff \ 8'h0 8'd0}	cover property (RC ##1 busA==8'b0001 ##1 busA==0000 ##1 busA==8'hffff ##1 busA==8'h0 ##1 busA==8'd0);
visualize -signal busB -sequence \ {8'b0001 0000 8'hffff \ 8'h0{2}}	cover property (RC ##1 busB==8'b0001 ##1 busB==0000 ##1 busB==8'hffff ##1 busB==8'h0[*2]);