LVR Command Reference

Ramtera Inc

December 21, 2025


LVR
Layout Verification
Reporter

Contents

Copyright **1**

1 Command Reference 1

1.1 Initialization Database Management 3

1.1.1 help 3

1.1.2 init_tool_db 5

1.1.3 set_tool_db 7

1.1.4 set_debug_mode_3049 9

1.1.5 set_top 11

1.1.6 set_technode 13

1.1.7 set_testmode 15

1.1.8 lvs_enable_ports 17

1.1.9 load_db 19

1.1.10 set_log_flush_time 21

1.2 Input File Handling 23

1.2.1 set_sch_file 23

1.2.2 set_xor1_gds 25

1.2.3 set_xor2_gds 27

1.2.4 set_svs1_spice 29

1.2.5 set_svs2_spice 31

1.2.6 set_svs1_verilog 33

1.2.7 set_svs2_verilog 35

1.2.8 read_gds 37

1.2.9 read_gds2 39
1.2.10 read_spice 41
1.2.11 write_gds 43
1.2.12 read_lrf 45
1.2.13 read_verilog 47
1.2.14 expected_file 49
1.2.15 verilog2sch 51
1.2.16 test_lvs 53
1.2.17 sky130_to_spice 55
1.2.18 split_cdl_asap7 57
1.2.19 split_cdl_gpdk 59
1.2.20 split_cdl_ng15 61
1.2.21 split_cdl_ng45 63
1.2.22 split_cdl_saed32 65
1.2.23 split_cdl_sky130 67
1.2.24 split_cdl_xh018 69
1.3 Flow Resource Settings 71
1.3.1 set_gds_flow 71
1.3.2 set_lefdef_flow 73
1.3.3 set_num_cpu 75
1.3.4 set_num_cores 77
1.3.5 set_num_layers 79
1.3.6 set_num_regions 81
1.3.7 set_num_bins 83
1.3.8 cell_orient_scheme 85
1.3.9 use_max 87
1.3.10 no_compare 89
1.3.11 no_cells 91
1.3.12 no_obs 93
1.3.13 no_pwr 95

1.3.14 no_vias 97
1.3.15 no_rect 99
1.4 Run Commands 101
1.4.1 run_drc 101
1.4.2 run_qdrc 103
1.4.3 run_lvs 105
1.4.4 run_mlvs 107
1.4.5 run_svs 109
1.4.6 run_dfm 111
1.4.7 run_erc 113
1.4.8 run_xor 115
1.4.9 run_fastxor 117
1.4.10 run_antenna 119
1.4.11 run_density 121
1.4.12 run_lefdef_drc 123
1.5 Reporting 125
1.5.1 report_drc 125
1.5.2 report_lvs 127
1.5.3 report_mlvs 129
1.5.4 report_dfm 131
1.5.5 report_erc 133
1.5.6 report_xor 135
1.5.7 report_fastxor 137
1.5.8 report_svs 139
1.5.9 report_antenna 141
1.5.10 rep_antenna 143
1.5.11 report_density 145
1.5.12 rep_density 147
1.5.13 report_lefdef_drc 149
1.5.14 set_report_file_drc 151

1.5.15 set_report_file_lvs 153
1.5.16 set_report_file_mlvs 155
1.5.17 set_report_file_dfm 157
1.5.18 set_report_file_erc 159
1.5.19 set_report_file_xor 161
1.5.20 set_report_file_fastxor 163
1.5.21 set_report_file_svs 165
1.5.22 set_report_file_antenna 167
1.5.23 set_report_file_density 169
1.5.24 set_report_file_lefdef_drc 171
1.6 LVS Options Controls 173
1.6.1 lvs 173
1.6.2 lvs_write_layout_from_db 175
1.6.3 lvs_write_schematic_from_spice 177
1.6.4 lvs_set_schematic_file_name 179
1.6.5 lvs_set_spice_file_name 181
1.6.6 lvs_set_macros_cdl_file_name 183
1.6.7 lvs_set_verilog_netlist_file_name 185
1.6.8 lvs_set_datc_flow 187
1.6.9 lvs_use_datc_flow 189
1.6.10 lvs_consider_macro_pins 191
1.6.11 lvs_check_shorts 193
1.6.12 lvs_check_open 195
1.6.13 lvs_check_macro 197
1.6.14 lvs_check_matching 199
1.6.15 lvs_check_unconnected_pins 201
1.6.16 lvs_detect_short 203
1.6.17 lvs_detect_open 205
1.6.18 lvs_power_pin 207
1.6.19 lvs_ground_pin 209

1.6.20 lvs_timeout 211
1.6.21 lvs_max_error 213
1.6.22 lvs_option_max_error 215
1.6.23 lvs_mapping_pin_to_terminal 217
1.6.24 lvs_waiver_macro_net 219
1.6.25 lvs_waiver_netlist_net 221
1.7 DRC/ERC/SVS/MLVS/Antenna Limits 223
1.7.1 drc 223
1.7.2 erc 225
1.7.3 svs 227
1.7.4 mlvs 229
1.7.5 antenna 231
1.7.6 dfm 233
1.7.7 perc 235
1.7.8 density 237
1.7.9 xor_check 239
1.7.10 drc_max_error 241
1.7.11 erc_max_error 243
1.7.12 svs_max_error 245
1.7.13 mlvs_max_error 247
1.7.14 xor_max_error 249
1.7.15 density_max_error 251
1.7.16 antenna_max_error 253
1.7.17 dfm_max_error 255
1.8 Density DFM Settings 257
1.8.1 density_layer 257
1.8.2 density_pixel 259
1.8.3 density_window 261
1.9 GUI Display 263
1.9.1 gui_display_single_layer 263

1.9.2 gui_display_all_layers 265
1.9.3 load_gui 267
1.9.4 cmd_line_only 269
1.9.5 set_gui_xor_result 271
1.10 Logger 273
1.10.1 disable_logger_timestamp 273
1.10.2 enable_logger_timestamp 275
1.11 Geometry Queries 277
1.11.1 query_rect 277
1.11.2 set_die_area 279
1.12 Appendix 281
1.12.1 Glossary 281
1.12.2 Example Complete Flow 281

Copyright

Ramtera Design Automation, Inc. 16192 Coastal Highway, Lewes, Delaware 19958, County of Sussex, USA.

1 Command Reference

Purpose. This manual documents the LVR command-line interface for physical verification flows. It organizes commands into categories and provides a consistent two-page reference for each: a narrative description followed by syntax/arguments and examples.

Conventions. Placeholders are used in examples:

- `<string>` denotes a file path, identifier, or free-form string.
- `<int>` denotes an integer value (counts, flags, DBU coordinates).
- `<float>` denotes a real-valued parameter (distances, thresholds).

Paths are shown relative to a project root where practical. Commands that take no arguments list *(none)*.

Page Layout. Each command spans two pages: Page 1 is a tailored description with best practices; Page 2 contains a syntax table and example usage lines. Figures may be inserted later at the indicated placeholders.

Categories. Commands are grouped into: Initialization & Database Management; Input & File Handling; Flow & Resource Settings; Run Commands;

Reporting; LVS Options & Controls; DRC/ERC/SVS/MLVS/Antenna & Limits; Density & DFM Settings; GUI & Display; Logger; Geometry & Queries.

1.1 Initialization Database Management

1.1.1 `help`

Description.

Display built-in help. Without args lists all commands; some builds support ‘help ¡command¿‘ for detailed usage. Output is human-readable.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	help	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
help
```

1.1.2 `init_tool_db`

Description.

Initialize a new tool database/workspace where configuration, parsed inputs, and results are persisted. Overwrites an existing DB of the same name.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	init_tool_db	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
init\_tool\_db <string1>
```

1.1.3 `set_tool_db`

Description.

Switch active tool database context. Subsequent reads/writes/runs operate on this DB.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_tool_db	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_tool\_db <string1>
```

1.1.4 `set_debug_mode_3049`

Description.

Set debug verbosity level. Higher integers produce more verbose logs; may slow runs.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_debug_mode_3049	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
set\_debug\_mode\_3049 <int1>
set\_debug\_mode\_3049 1
```

1.1.5 `set_top`

Description.

Define the top-level design entity (root cell / top module) for hierarchy traversal.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_top	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_top <string1>
set\_top aes\_cipher\_top
```

1.1.6 `set_technode`

Description.

Set technology node/PDK id to select rule defaults, units, and device parameters.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_technode	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_technode <string1>
set\_technode sky130hd
```

1.1.7 `set_testmode`

Description.

Toggle test/regression mode. Non-zero enables reduced checks or mock IO; not for sign-off.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_testmode	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
set\_testmode <int1>
set\_testmode 1
```

1.1.8 `lvs_enable_ports`

Description.

Include top-level ports in LVS extraction; 1 enables, 0 disables.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_enable_ports	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
lvs\_enable\_ports <int1>
lvs\_enable\_ports 1
```

1.1.9 `load_db`

Description.

Load a previously saved tool database/session for reporting or incremental runs.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	load_db	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
load\_db <string1>
```

1.1.10 `set_log_flush_time`

Description.

Set periodic flush interval (seconds) for log output during long runs.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_log_flush_time	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
set\_log\_flush\_time <int1>
set\_log\_flush\_time 5
```

1.2 Input File Handling

1.2.1 `set_sch_file`

Description.

Set the main schematic/netlist file path used by schematic-consuming flows.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_sch_file	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_sch\_file <string1>
```

1.2.2 `set_xor1_gds`

Description.

Specify XOR inputs (reference vs. test) for geometric XOR comparison of two GDS databases.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_xor1_gds	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_xor1\_gds <string1>
```

1.2.3 `set_xor2_gds`

Description.

Specify XOR inputs (reference vs. test) for geometric XOR comparison of two GDS databases.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_xor2_gds	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_xor2\_gds <string1>
```

1.2.4 `set_svs1_spice`

Description.

Specify inputs for schematic-vs-schematic (SVS) comparison; accepts SPICE or Verilog sources.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_svs1_spice	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_svs1\_spice <string1>
```

1.2.5 `set_svs2_spice`

Description.

Specify inputs for schematic-vs-schematic (SVS) comparison; accepts SPICE or Verilog sources.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_svs2_spice	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_svs2\_spice <string1>
```

1.2.6 `set_svs1_verilog`

Description.

Specify inputs for schematic-vs-schematic (SVS) comparison; accepts SPICE or Verilog sources.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_svs1_verilog	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_svs1\_verilog <string1>
```

1.2.7 `set_svs2_verilog`

Description.

Specify inputs for schematic-vs-schematic (SVS) comparison; accepts SPICE or Verilog sources.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_svs2_verilog	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_svs2\_verilog <string1>
```

1.2.8 `read_gds`

Description.

Read a GDS stream file into the active tool DB.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	read_gds	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
read\_gds <string1>
read\_gds ./input/gds.in
```

1.2.9 `read_gds2`

Description.

Read a GDSII file (alternative reader) into the tool DB.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	read_gds2	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
read\_gds2 <string1>
read\_gds2 ./input/gds2.in
```

1.2.10 `read_spice`

Description.

Read a SPICE netlist and store for LVS/SVS flows.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	read_spice	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
read\_spice <string1>
read\_spice ./input/spice.in
```

1.2.11 `write_gds`

Description.

Export current layout database to a GDS file.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	write_gds	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
write\_gds <string1>
```

1.2.12 `read_lrf`

Description.

Load a Layout Rules File (LRF) for DRC/DFM verification.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	read_lrf	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
read\_lrf <string1>
read\_lrf ./input/lrf.in
```

1.2.13 `read_verilog`

Description.

Load a Verilog netlist for LVS/SVS or conversion flows.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	read_verilog	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
read\_verilog <string1>
read\_verilog ./input/verilog.in
```

1.2.14 `expected_file`

Description.

Set a golden results file used to compare against run outputs.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	expected_file	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
expected\_file <string1>
expected\_file ./golden/expected.json
```

1.2.15 `verilog2sch`

Description.

Convert a Verilog netlist to a schematic format compatible with the LVS engine.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	verilog2sch	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument
arg2	string (path/id)	required	—	Positional argument

Example Usage.

```
verilog2sch <string1> <string2>
verilog2sch ./design.v ./out.sch
```

1.2.16 `test_lvs`

Description.

Quick LVS between provided layout and schematic to smoke-test connectivity.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	test_lvs	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument
arg2	string (path/id)	required	—	Positional argument

Example Usage.

```
test\_lvs <string1> <string2>
test\_lvs ./layout.gds ./schematic.spice
```

1.2.17 `sky130_to_spice`

Description.

Translate Sky130 CDL/CDL-like netlist to SPICE format with tool-compatible conventions.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	sky130_to_spice	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument
arg2	string (path/id)	required	—	Positional argument

Example Usage.

```
sky130\_to\_spice <string1> <string2>
sky130\_to\_spice ./in.cdl ./out.spice
```

1.2.18 `split_cdl_asap7`

Description.

Split a monolithic CDL netlist into per-cell libraries for the named PDK/family; outputs cell files and logs.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	split_cdl_asap7	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument
arg2	string (path/id)	required	—	Positional argument
arg3	string (path/id)	required	—	Positional argument
arg4	string (path/id)	required	—	Positional argument

Example Usage.

```
split\_cdl\_asap7 <string1> <string2> <string3> <string4>
split\_cdl\_asap7 ./in.cdl ./out\_lib ./cells.map ./split.log
```

1.2.19 `split_cdl_gpdk`

Description.

Split a monolithic CDL netlist into per-cell libraries for the named PDK/family; outputs cell files and logs.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	split_cdl_gpdk	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument
arg2	string (path/id)	required	—	Positional argument
arg3	string (path/id)	required	—	Positional argument
arg4	string (path/id)	required	—	Positional argument

Example Usage.

```
split\_cdl\_gpdk <string1> <string2> <string3> <string4>
split\_cdl\_gpdk ./in.cdl ./out\_lib ./cells.map ./split.log
```

1.2.20 `split_cdl_ng15`

Description.

Split a monolithic CDL netlist into per-cell libraries for the named PDK/family; outputs cell files and logs.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	split_cdl_ng15	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument
arg2	string (path/id)	required	—	Positional argument
arg3	string (path/id)	required	—	Positional argument
arg4	string (path/id)	required	—	Positional argument

Example Usage.

```
split\_cdl\_ng15 <string1> <string2> <string3> <string4>
split\_cdl\_ng15 ./in.cdl ./out\_lib ./cells.map ./split.log
```

1.2.21 `split_cdl_ng45`

Description.

Split a monolithic CDL netlist into per-cell libraries for the named PDK/family; outputs cell files and logs.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	split_cdl_ng45	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument
arg2	string (path/id)	required	—	Positional argument
arg3	string (path/id)	required	—	Positional argument
arg4	string (path/id)	required	—	Positional argument

Example Usage.

```
split\_cdl\_ng45 <string1> <string2> <string3> <string4>
split\_cdl\_ng45 ./in.cdl ./out\_lib ./cells.map ./split.log
```

1.2.22 `split_cdl_saed32`

Description.

Split a monolithic CDL netlist into per-cell libraries for the named PDK/family; outputs cell files and logs.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	split_cdl_saed32	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument
arg2	string (path/id)	required	—	Positional argument
arg3	string (path/id)	required	—	Positional argument
arg4	string (path/id)	required	—	Positional argument

Example Usage.

```
split\_cdl\_saed32 <string1> <string2> <string3> <string4>
split\_cdl\_saed32 ./in.cdl ./out\_lib ./cells.map ./split.log
```

1.2.23 `split_cdl_sky130`

Description.

Split a monolithic CDL netlist into per-cell libraries for the named PDK/family; outputs cell files and logs.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	split_cdl_sky130	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument
arg2	string (path/id)	required	—	Positional argument
arg3	string (path/id)	required	—	Positional argument
arg4	string (path/id)	required	—	Positional argument

Example Usage.

```
split\_cdl\_sky130 <string1> <string2> <string3> <string4>
split\_cdl\_sky130 ./in.cdl ./out\_lib ./cells.map ./split.log
```

1.2.24 `split_cdl_xh018`

Description.

Split a monolithic CDL netlist into per-cell libraries for the named PDK/family; outputs cell files and logs.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	split_cdl_xh018	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument
arg2	string (path/id)	required	—	Positional argument
arg3	string (path/id)	required	—	Positional argument
arg4	string (path/id)	required	—	Positional argument

Example Usage.

```
split\_cdl\_xh018 <string1> <string2> <string3> <string4>
split\_cdl\_xh018 ./in.cdl ./out\_lib ./cells.map ./split.log
```

1.3 Flow Resource Settings

1.3.1 `set_gds_flow`

Description.

Select primary import flow (GDS-based or LEF/DEF-based).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_gds_flow	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
set\_gds\_flow <int1>
set\_gds\_flow 1
```

1.3.2 `set_lefdef_flow`

Description.

Select primary import flow (GDS-based or LEF/DEF-based).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_lefdef_flow	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
set\_lefdef\_flow <int1>
set\_lefdef\_flow 1
```

1.3.3 set_num_cpu

Description.

Configure resources/partitioning for performance: parallelism, layers, regions, bins.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_num_cpu	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
set\_num\_cpu <int1>
set\_num\_cpu 8
```

1.3.4 `set_num_cores`

Description.

Configure resources/partitioning for performance: parallelism, layers, regions, bins.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_num_cores	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
set\_num\_cores <int1>
set\_num\_cores 8
```

1.3.5 `set_num_layers`

Description.

Configure resources/partitioning for performance: parallelism, layers, regions, bins.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_num_layers	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
set\_num\_layers <int1>
set\_num\_layers 8
```

1.3.6 set_num_regions

Description.

Configure resources/partitioning for performance: parallelism, layers, regions, bins.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_num_regions	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
set\_num\_regions <int1>
set\_num\_regions 8
```

1.3.7 `set_num_bins`

Description.

Configure resources/partitioning for performance: parallelism, layers, regions, bins.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_num_bins	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
set\_num\_bins <int1>
set\_num\_bins 8
```

1.3.8 `cell_orient_scheme`

Description.

Choose the cell orientation scheme used during import/checks (e.g., R0/MX/MY).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	cell_orient_scheme	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
cell\_orient\_scheme <int1>
```

1.3.9 use_max

Description.

Enable/disable specific checks/filters to tailor import/verification.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	use_max	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
use\_max <int1>
use\_max 1
```

1.3.10 no_compare

Description.

Enable/disable specific checks/filters to tailor import/verification.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	no_compare	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
no\_compare <int1>
no\_compare 1
```

1.3.11 no_cells

Description.

Enable/disable specific checks/filters to tailor import/verification.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	no_cells	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
no\_cells <int1>
no\_cells 1
```

1.3.12 no_obs

Description.

Enable/disable specific checks/filters to tailor import/verification.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	no_obs	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
no\_obs <int1>
no\_obs 1
```

1.3.13 no_pwr

Description.

Enable/disable specific checks/filters to tailor import/verification.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	no_pwr	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
no\_pwr <int1>
no\_pwr 1
```

1.3.14 `no_vias`

Description.

Enable/disable specific checks/filters to tailor import/verification.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	no_vias	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
no\_vias <int1>
no\_vias 1
```

1.3.15 `no_rect`

Description.

Enable/disable specific checks/filters to tailor import/verification.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	no_rect	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
no\_rect <int1>
no\_rect 1
```

1.4 Run Commands

1.4.1 `run_drc`

Description.

Execute the DRC flow with current inputs/settings and write results.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	run_drc	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
run\_drc
```

1.4.2 `run_qdrc`

Description.

Run quick/fast DRC with an accelerated kernel or subset of rules.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	run_qdrc	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
run\_qdrc
```

1.4.3 `run_lvs`

Description.

Execute the LVS flow with current inputs/settings and write results.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	run_lvs	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
run\_lvs
```

1.4.4 `run_mlvs`

Description.

Execute the MLVS flow with current inputs/settings and write results.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	run_mlvs	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
run\_mlvs
```

1.4.5 `run_svs`

Description.

Execute the SVS flow with current inputs/settings and write results.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	run_svs	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
run\_svs
```

1.4.6 `run_dfm`

Description.

Execute the DFM flow with current inputs/settings and write results.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	run_dfm	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
run\_dfm
```

1.4.7 `run_erc`

Description.

Execute the ERC flow with current inputs/settings and write results.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	run_erc	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
run\_erc
```

1.4.8 `run_xor`

Description.

Execute the XOR flow with current inputs/settings and write results.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	run_xor	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
run\_xor
```

1.4.9 `run_fastxor`

Description.

Execute the FASTXOR flow with current inputs/settings and write results.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	run_fastxor	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
run\_fastxor
```

1.4.10 run_antenna

Description.

Execute the ANTENNA flow with current inputs/settings and write results.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	run_antenna	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
run\_antenna
```

1.4.11 `run_density`

Description.

Execute the DENSITY flow with current inputs/settings and write results.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	run_density	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
run\_density
```

1.4.12 `run_lefdef_drc`

Description.

Execute the LEFDEF_DRC flow with current inputs/settings and write results.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	run_lefdef_drc	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
run\_lefdef\_drc
```

1.5 Reporting

1.5.1 `report_drc`

Description.

Generate a DRC report from the most recent run, writing to the configured report file.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	report_drc	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
report\_drc
```

1.5.2 `report_lvs`

Description.

Generate a LVS report from the most recent run, writing to the configured report file.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	report_lvs	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
report\_lvs
```

1.5.3 `report_mlvs`

Description.

Generate a MLVS report from the most recent run, writing to the configured report file.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	report_mlvs	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
report\_mlvs
```

1.5.4 `report_dfm`

Description.

Generate a DFM report from the most recent run, writing to the configured report file.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	report_dfm	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
report\_dfm
```

1.5.5 `report_erc`

Description.

Generate a ERC report from the most recent run, writing to the configured report file.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	report_erc	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
report\_erc
```

1.5.6 `report_xor`

Description.

Generate a XOR report from the most recent run, writing to the configured report file.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	report_xor	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
report\_xor
```

1.5.7 `report_fastxor`

Description.

Generate a FASTXOR report from the most recent run, writing to the configured report file.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	report_fastxor	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
report\_fastxor
```

1.5.8 `report_svs`

Description.

Generate a SVS report from the most recent run, writing to the configured report file.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	report_svs	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
report\_svs
```

1.5.9 `report_antenna`

Description.

Generate a ANTENNA report from the most recent run, writing to the configured report file.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	report_antenna	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
report\_antenna
```

1.5.10 `rep_antenna`

Description.

Generate a ANTENNA report from the most recent run, writing to the configured report file.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	rep_antenna	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
rep\_antenna
```

1.5.11 `report_density`

Description.

Generate a DENSITY report from the most recent run, writing to the configured report file.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	report_density	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
report\_density
```

1.5.12 `rep_density`

Description.

Generate a DENSITY report from the most recent run, writing to the configured report file.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	rep_density	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
rep\_density
```

1.5.13 `report_lefdef_drc`

Description.

Generate a LEFDEF_DRC report from the most recent run, writing to the configured report file.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	report_lefdef_drc	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
report\_lefdef\_drc
```

1.5.14 `set_report_file_drc`

Description.

Set output path for a specific report category. Must be set before running the report.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_report_file_drc	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_report\_file\_drc <string1>
set\_report\_file\_drc ./reports/drc.rpt
```

1.5.15 `set_report_file_lvs`

Description.

Set output path for a specific report category. Must be set before running the report.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_report_file_lvs	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_report\_file\_lvs <string1>
set\_report\_file\_lvs ./reports/lvs.rpt
```

1.5.16 `set_report_file_mlvs`

Description.

Set output path for a specific report category. Must be set before running the report.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_report_file_~~mlv~~s	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_report\_file\_mlvs <string1>
set\_report\_file\_mlvs ./reports/mlvs.rpt
```

1.5.17 `set_report_file_dfm`

Description.

Set output path for a specific report category. Must be set before running the report.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_report_file_dfm	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_report\_file\_dfm <string1>
set\_report\_file\_dfm ./reports/dfm.rpt
```

1.5.18 `set_report_file_erc`

Description.

Set output path for a specific report category. Must be set before running the report.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_report_file_~~erc~~	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_report\_file\_erc <string1>
set\_report\_file\_erc ./reports/erc.rpt
```

1.5.19 `set_report_file_xor`

Description.

Set output path for a specific report category. Must be set before running the report.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_report_file_~~xor~~	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_report\_file\_xor <string1>
set\_report\_file\_xor ./reports/xor.rpt
```

1.5.20 `set_report_file_fastxor`

Description.

Set output path for a specific report category. Must be set before running the report.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_report_file_fastxor	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_report\_file\_fastxor <string1>
set\_report\_file\_fastxor ./reports/fastxor.rpt
```

1.5.21 set_report_file_svs

Description.

Set output path for a specific report category. Must be set before running the report.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_report_file_svs	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_report\_file\_svs <string1>
set\_report\_file\_svs ./reports/svs.rpt
```

1.5.22 `set_report_file_antenna`

Description.

Set output path for a specific report category. Must be set before running the report.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_report_file_antenna	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_report\_file\_antenna <string1>
set\_report\_file\_antenna ./reports/antenna.rpt
```

1.5.23 `set_report_file_density`

Description.

Set output path for a specific report category. Must be set before running the report.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_report_file_density	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_report\_file\_density <string1>
set\_report\_file\_density ./reports/density.rpt
```

1.5.24 `set_report_file_lefdef_drc`

Description.

Set output path for a specific report category. Must be set before running the report.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_report_file_lefdef_drc	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
set\_report\_file\_lefdef\_drc <string1>
set\_report\_file\_lefdef\_drc ./reports/lefdef\_drc.rpt
```

1.6 LVS Options Controls

1.6.1 `lvs`

Description.

Invoke the LVS engine with the provided argument (file or rule id).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
lvs <string1>
lvs ./rule\_or\_input\_file
```

1.6.2 `lvs_write_layout_from_db`

Description.

Export current layout view from DB to a file for debug or downstream use.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_write_layout_from_db	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
lvs\_write\_layout\_from\_db <string1>
```

1.6.3 `lvs_write_schematic_from_spice`

Description.

Emit a schematic representation derived from SPICE for LVS inspection or GUI view.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_write_schematic_from_spice			Command keyword
arg1	string (path/id)	required	—	Positional argument
arg2	string (path/id)	required	—	Positional argument

Example Usage.

```
lvs\_write\_schematic\_from\_spice <string1> <string2>
```

1.6.4 `lvs_set_schematic_file_name`

Description.

Set an LVS-related input path or mode (schematic, SPICE, macros, Verilog, DATC flow).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_set_schematic_file_name	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
lvs\_set\_schematic\_file\_name <string1>
lvs\_set\_schematic\_file\_name ./netlist.spice
```

1.6.5 `lvs_set_spice_file_name`

Description.

Set an LVS-related input path or mode (schematic, SPICE, macros, Verilog, DATC flow).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_set_spice_file_name	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
lvs\_set\_spice\_file\_name <string1>
lvs\_set\_spice\_file\_name ./netlist.spice
```

1.6.6 `lvs_set_macros_cdl_file_name`

Description.

Set an LVS-related input path or mode (schematic, SPICE, macros, Verilog, DATC flow).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_set_macros_cdl_file_name	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
lvs\_set\_macros\_cdl\_file\_name <string1>
lvs\_set\_macros\_cdl\_file\_name ./netlist.spice
```

1.6.7 `lvs_set_verilog_netlist_file_name`

Description.

Set an LVS-related input path or mode (schematic, SPICE, macros, Verilog, DATC flow).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_set_verilog_netlist_file_name			Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
lvs\_set\_verilog\_netlist\_file\_name <string1>
lvs\_set\_verilog\_netlist\_file\_name ./netlist.spice
```

1.6.8 `lvs_set_datc_flow`

Description.

Set an LVS-related input path or mode (schematic, SPICE, macros, Verilog, DATC flow).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_set_datc_flow	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
lvs\_set\_datc\_flow <int1>
lvs\_set\_datc\_flow ./netlist.spice
```

1.6.9 `lvs_use_datc_flow`

Description.

Toggle the DATC reference flow for LVS preprocessing/import (0/1).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_use_datc_flow	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
lvs\_use\_datc\_flow <int1>
lvs\_use\_datc\_flow 1
```

1.6.10 `lvs_consider_macro_pins`

Description.

(Description TBD) Sets configuration or triggers a run in the LVR flow.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_consider_macro_pins	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
lvs\_consider\_macro\_pins <int1>
lvs\_consider\_macro\_pins 1
```

1.6.11 `lvs_check_shorts`

Description.

Enable/disable LVS check for shorts (0/1).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_check_shorts	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
lvs\_check\_shorts <int1>
lvs\_check\_shorts 1
```

1.6.12 `lvs_check_open`

Description.

Enable/disable LVS check for open (0/1).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_check_open	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
lvs\_check\_open <int1>
lvs\_check\_open 1
```

1.6.13 `lvs_check_macro`

Description.

Enable/disable LVS check for macro (0/1).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_check_macro	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
lvs\_check\_macro <int1>
lvs\_check\_macro 1
```

1.6.14 `lvs_check_matching`

Description.

Enable/disable LVS check for matching (0/1).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_check_matching	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
lvs\_check\_matching <int1>
lvs\_check\_matching 1
```

1.6.15 `lvs_check_unconnected_pins`

Description.

Enable/disable LVS check for unconnected pins (0/1).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_check_unconnected_pins	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
lvs\_check\_unconnected\_pins <int1>
lvs\_check\_unconnected\_pins 1
```

1.6.16 `lvs_detect_short`

Description.

Enable detection of shorts/opens during LVS (0=off, 1=on).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_detect_short	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
lvs\_detect\_short <int1>
lvs\_detect\_short 1
```

1.6.17 `lvs_detect_open`

Description.

Enable detection of shorts/opens during LVS (0=off, 1=on).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_detect_open	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
lvs\_detect\_open <int1>
lvs\_detect\_open 1
```

1.6.18 `lvs_power_pin`

Description.

Declare canonical power/ground pin names for special handling in LVS equivalence.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_power_pin	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
lvs\_power\_pin <string1>
lvs\_power\_pin VDD
```

1.6.19 `lvs_ground_pin`

Description.

Declare canonical power/ground pin names for special handling in LVS equivalence.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_ground_pin	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
lvs\_ground\_pin <string1>
lvs\_ground\_pin VDD
```

1.6.20 `lvs_timeout`

Description.

Set timeout (seconds) for LVS phase or run to prevent pathological runtimes.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_timeout	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
lvs\_timeout <int1>
```

1.6.21 lvs_max_error

Description.

Limit the number of errors recorded for the category; extra errors suppressed after the cap.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_max_error	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
lvs\_max\_error <int1>
```

1.6.22 lvs_option_max_error

Description.

Limit the number of errors recorded for the category; extra errors suppressed after the cap.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_option_max_error	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
lvs\_option\_max\_error <int1>
```

1.6.23 `lvs_mapping_pin_to_terminal`

Description.

Map a schematic pin to a layout terminal name for name-equivalence resolution.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_mapping_pin_to_terminal	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument
arg2	string (path/id)	required	—	Positional argument

Example Usage.

```
lvs\_mapping\_pin\_to\_terminal <string1> <string2>
```

1.6.24 lvs_waiver_macro_net

Description.

Declare an LVS waiver for known differences (macro net or netlist net), suppressing selected errors.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_waiver_macro_net	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument
arg2	string (path/id)	required	—	Positional argument

Example Usage.

```
lvs\_waiver\_macro\_net <string1> <string2>
lvs\_waiver\_macro\_net U\_MACRO\_1 NET\_A
```

1.6.25 `lvs_waiver_netlist_net`

Description.

Declare an LVS waiver for known differences (macro net or netlist net), suppressing selected errors.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	lvs_waiver_netlist_net	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
lvs\_waiver\_netlist\_net <string1>
lvs\_waiver\_netlist\_net U\_MACRO\_1 NET\_A
```

1.7 DRC/ERC/SVS/MLVS/Antenna Limits

1.7.1 `drc`

Description.

Invoke the DRC engine with the provided argument (file or rule id).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	drc	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
drc <string1>
drc ./rule\_or\_input\_file
```

1.7.2 `erc`

Description.

Invoke the ERC engine with the provided argument (file or rule id).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	erc	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
erc <string1>
erc ./rule\_or\_input\_file
```

1.7.3 svs

Description.

Invoke the SVS engine with the provided argument (file or rule id).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	svs	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
svs <string1>
svs ./rule\_or\_input\_file
```

1.7.4 `mlvs`

Description.

Invoke the MLVS engine with the provided argument (file or rule id).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	mlvs	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
mlvs <string1>
mlvs ./rule\_or\_input\_file
```

1.7.5 `antenna`

Description.

Invoke the ANTENNA engine with the provided argument (file or rule id).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	antenna	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
antenna <string1>
antenna ./rule\_or\_input\_file
```

1.7.6 `dfm`

Description.

Invoke the DFM engine with the provided argument (file or rule id).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	dfm	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
dfm <string1>
dfm ./rule\_or\_input\_file
```

1.7.7 `perc`

Description.

Invoke the PERC engine with the provided argument (file or rule id).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	perc	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
perc <string1>
perc ./rule\_or\_input\_file
```

1.7.8 `density`

Description.

Invoke the DENSITY engine with the provided argument (file or rule id).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	density	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
density <string1>
density ./rule\_or\_input\_file
```

1.7.9 xor_check

Description.

Invoke the XOR_CHECK engine with the provided argument (file or rule id).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	xor_check	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
xor\_check <string1>
xor\_check ./rule\_or\_input\_file
```

1.7.10 drc_max_error

Description.

Limit the number of errors recorded for the category; extra errors suppressed after the cap.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	drc_max_error	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
drc\_max\_error <int1>
```

1.7.11 erc_max_error

Description.

Limit the number of errors recorded for the category; extra errors suppressed after the cap.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	erc_max_error	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
erc\_max\_error <int1>
```

1.7.12 svs_max_error

Description.

Limit the number of errors recorded for the category; extra errors suppressed after the cap.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	svs_max_error	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
svs\_max\_error <int1>
```

1.7.13 `mlvs_max_error`

Description.

Limit the number of errors recorded for the category; extra errors suppressed after the cap.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	mlvs_max_error	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
mlvs\_max\_error <int1>
```

1.7.14 xor_max_error

Description.

Limit the number of errors recorded for the category; extra errors suppressed after the cap.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	xor_max_error	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
xor\_max\_error <int1>
```

1.7.15 density_max_error

Description.

Limit the number of errors recorded for the category; extra errors suppressed after the cap.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	density_max_error	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
density\_max\_error <int1>
```

1.7.16 antenna_max_error

Description.

Limit the number of errors recorded for the category; extra errors suppressed after the cap.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	antenna_max_error	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
antenna\_max\_error <int1>
```

1.7.17 dfm_max_error

Description.

Limit the number of errors recorded for the category; extra errors suppressed after the cap.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	dfm_max_error	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
dfm\_max\_error <int1>
```

1.8 Density DFM Settings

1.8.1 density_layer

Description.

Configure density analysis: target layer, pixel size, and window/stride parameters.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	density_layer	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
density\_layer <string1>
density\_layer VDD
```

1.8.2 `density_pixel`

Description.

Configure density analysis: target layer, pixel size, and window/stride parameters.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	density_pixel	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
density\_pixel <int1>
density\_pixel 2
```

1.8.3 density_window

Description.

Configure density analysis: target layer, pixel size, and window/stride parameters.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	density_window	—	—	Command keyword
arg1	integer	required	—	Positional argument
arg2	integer	required	—	Positional argument
arg3	integer	required	—	Positional argument
arg4	integer	required	—	Positional argument
arg5	integer	required	—	Positional argument

Example Usage.

```
density\_window <int1> <int2> <int3> <int4> <int5>
density\_window 1000 1000 100 100 0
```

1.9 GUI Display

1.9.1 gui_display_single_layer

Description.

GUI: display only the specified layer (hides all others).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	gui_display_single_layer	—	—	Command keyword
arg1	string (path/id)	required	—	Positional argument

Example Usage.

```
gui\_display\_single\_layer <string1>
gui\_display\_single\_layer VDD
```

1.9.2 `gui_display_all_layers`

Description.

GUI: display all layers.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	gui_display_all_layers	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
gui\_display\_all\_layers
```

1.9.3 `load_gui`

Description.

Launch or attach GUI for interactive viewing of layout/rules/markers.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	load_gui	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
load\_gui
```

1.9.4 `cmd_line_only`

Description.

Force command-line mode (no GUI). Suited for servers and CI.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	cmd_line_only	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
cmd\_line\_only
```

1.9.5 `set_gui_xor_result`

Description.

Toggle rendering of XOR results in GUI after a run (0/1).

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_gui_xor_result	—	—	Command keyword
arg1	integer	required	—	Positional argument

Example Usage.

```
set\_gui\_xor\_result <int1>
set\_gui\_xor\_result 1
```

1.10 Logger

1.10.1 `disable_logger_timestamp`

Description.

Disable/enable timestamps in logger output.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	disable_logger_timestamp	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
disable\_logger\_timestamp
```

1.10.2 `enable_logger_timestamp`

Description.

Disable/enable timestamps in logger output.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	enable_logger_timestamp	—	—	Command keyword
—	(none)	—	—	No arguments

Example Usage.

```
enable\_logger\_timestamp
```

1.11 Geometry Queries

1.11.1 `query_rect`

Description.

Query rectangles within DBU coordinate window (lx ly ux uy) for CLI debugging/inspection.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	query_rect	—	—	Command keyword
arg1	integer	required	—	Positional argument
arg2	integer	required	—	Positional argument
arg3	integer	required	—	Positional argument
arg4	integer	required	—	Positional argument

Example Usage.

```
query\_rect <int1> <int2> <int3> <int4>
query\_rect 0 0 1000 1000
```

1.11.2 set_die_area

Description.

Define die bounding box in DBU coordinates (lx ly ux uy). Affects reporting/visualization.

Best practices: • Validate paths before invocation; prefer relative project paths. • Use consistent naming for DBs and reports to simplify CI. • When scripting, check exit codes and report presence.

Syntax and Arguments.

Parameter	Type	Required	Default	Notes
Command	set_die_area	—	—	Command keyword
arg1	integer	required	—	Positional argument
arg2	integer	required	—	Positional argument
arg3	integer	required	—	Positional argument
arg4	integer	required	—	Positional argument

Example Usage.

```
set\_die\_area <int1> <int2> <int3> <int4>
set\_die\_area 0 0 1000 1000
```

1.12 Appendix

1.12.1 Glossary

DRC Design Rule Checking.

LVS Layout vs. Schematic comparison.

SVS Schematic vs. Schematic comparison.

GDS Graphic Data System layout format.

CDL Circuit Description Language netlist.

PDK Process Design Kit.

DBU Database Unit, the internal coordinate unit.

1.12.2 Example Complete Flow

```
init\_tool\_db project\_db
set\_tool\_db project\_db
set\_technode sky130hd
set\_top aes\_cipher\_top
read\_gds ./input/chip.gds
read\_lrf ./rules/sky130\_drc.lrf
run\_drc
report\_drc
read\_spice ./netlist/chip.spice
lvs\_enable\_ports 1
run\_lvs
report\_lvs
```