2cli User Guide

Version 2026.03

DashThru

Copyright Notice

Trademarks

Disclaimer

Contents

1

1 Introduction

2cli is a CLI interface utility developed by DashThru Technology for managing interactive EDA sessions. It supports a wide range of commercial EDA tools and can be integrated with many open-source AI agents.

Compared with MCP servers, CLI remains the more flexible and token-saving interface for AI agents. When an AI agent directly controls an EDA tool through a regular CLI workflow, the typical process involves generating a script file and launching the EDA tool for one-time execution. For EDA tools that already provide interactive command-line environments, this iterative workflow is sometimes inefficient.

2cli is specifically optimized for EDA interactive workflows. In addition to standard command transmission and response handling, it also automates tab-completion retrieval and man-page control. This makes 2cli particularly well suited for integrating AI agents with interactive EDA tool shells.

1-1 Overview

Key capabilities of 2cli include:

- Launching and managing one or more EDA tool sessions
- Sending commands interactively to EDA tools
- Retrieving command hints and completions through Tab interaction
- Automatically handling page-mode control for the man command
- Integrating with AI agents through a customized Skill

For detailed information about 2cli functionality, refer to Chapter 2, Features and Capabilities.

1-2 Installation and Deployment

- **Recommended Operating Systems**

RHEL 6.x / 7.x / 8.x or their compatible CentOS and Rocky Linux distributions are recommended. On RHEL and CentOS systems, the OS version can be checked using:

```
% cat /etc/system-release
CentOS Linux release 7.9.2009 (Core)
```

- **Installing 2cli**

2cli is included in the DashRTL distribution package. Extract the package directly to the target installation directory:

```
% tar xJvf DashRTL_v2026.03.tar.xz
```

Alternatively, 2cli can be downloaded through PyPI:

```
% pip install 2cli
```

- **Starting 2cli**

```
% which 2cli
/edatool/DashRTL_v2026.03/bin/2cli
% 2cli -start tclsh
2cli launched pid 12345
% 2cli puts hello
hello
%
% 2cli exit
2cli daemon for pid 12345 exit due to tool terminated.
```

2

2 Features and Capabilities

This chapter describes the core features of 2cli, including:

- Launching EDA Tools
- Sending Commands
- Managing Active EDA Sessions

2-1 Launching EDA Tools

To start a new EDA tool session, use the `-start` option. The following example launches the DashThru DashRTL tool:

```
% 2cli -start dashrtl -no_gui
2cli launched pid 12345
```

The EDA tool command and its startup options may also be passed as a single quoted argument:

```
% 2cli -start 'dashrtl -no_gui'
2cli launched pid 12345
```

By default, the inactivity timeout is set to 86400 seconds, and no log file is generated. The following example disables the timeout mechanism and enables logging:

```
% 2cli -timeout 0 -log 2cli.log -start dashrtl -no_gui
2cli launched pid 12345
```

2-2 Sending Commands

After launching an EDA tool session, commands can be sent directly through 2cli. Unless the command ends with a special character, 2cli automatically appends a newline (`\n`) to the command:

```
% 2cli puts hello
hello
dashrtl-tcl>
```

Tab completion is also supported for retrieving command suggestions:

```
% 2cli 'report\t'
report_black_box        report_sv_compatibility
dashrtl-tcl>
```

2cli also supports accessing built-in man pages for command documentation:

```
% 2cli man set
PROCEDURE:    set
SYNTAX:       set varName ?newValue?
DESCRIPTION:  Optionally create or modify the variable or array to
              'newValue', and return current value.

              Examples:

              set a 1  ;# -> 1
              set a    ;# -> 1
RELATED:      set_tool_var
              unset
              variable
```

When multiple EDA sessions launched by 2cli are active simultaneously, commands are sent to the most recently launched process by default. The `-pid` option can be used to target a specific session:

```
% 2cli -pid 12345 set hdl verilog
verilog
dashrtl-tcl>
```

2-3 Managing Active EDA Sessions

The `-query` option displays all currently active EDA sessions:

```
% 2cli -query
2cli Interactive Shell Manager for Agents
(v2026.03-Alpha2 on Python 3.6.8)
Copyright(c) 2026, DashThru Technology, Ltd. All rights reserved.

Active connections for user 'designer':
----------------------+------------+--------------+---------------+------------
| Laucnch Time        | Tool PID   | Daemon PID   | Tool Prompt   | Log File   |
----------------------+------------+--------------+---------------+------------
| 2026-04-29 15:50:58 | 458        | 457          | dashrtl-tcl>  | N/A        |
----------------------+------------+--------------+---------------+------------
| 2026-04-29 16:14:34 | 610        | 609          | dashrtl-tcl>  | N/A        |
----------------------+------------+--------------+---------------+------------
```

The `-cleanup` option removes records and temporary files associated with closed sessions:

```
% 2cli -cleanup
2cli cleanup removed 1 socket files and 1 log files, deleted 1 traces.
```

The `-force_cleanup` option forcibly terminates all active sessions and removes all associated records and temporary files:

```
% 2cli -force_cleanup
2cli force-cleanup killed 1 tools, removed 1 socket files and 1 log
files, and deleted trace file.
```

3

3 AI Agent Integration

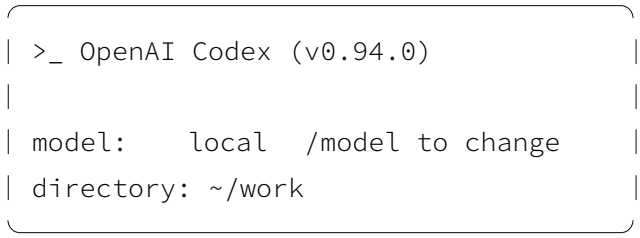
2cli is designed specifically for AI agent integration, addressing the challenges of seamlessly connecting AI agents with general-purpose interactive EDA tool environments.

This chapter provides common integration examples, including:

3-1 Installing the Skill

Most AI agents support Skill installation by copying the Skill directory into the agent's designated skills path.

For OpenAI Codex, copy the `examples/agent_skills/eda-2cli` directory from the 2cli installation path into `~/.codex/skills/`, then restart Codex to load the new Skill:



3-2 Codex Workflow Example

Once the Skill is loaded, the AI agent can autonomously explore interactive EDA environments through 2cli and gradually become proficient with the tool. The following prompt demonstrates how Codex can explore the DashThru DashRTL:

Use 2cli to launch dashrtl -no_gui, explore the tool environment, and summarize typical dashrtl workflows.

Codex successfully loads the Skill during startup:

- Explored
  └ Read SKILL.md

- Now I understand the skill. Let me start by launching dashrtl and then exploring its environment.

- Ran 2cli -start dashrtl -no_gui
  └ (no output)
  └ 2cli Interactive Shell Manager for Agents
    (v2026.03-Alpha2 on Python 3.6.8)
    … +8 lines

On a local setup equipped with an RTX 5090 GPU and a Qwen 3.6-27B-UD_Q4 model, Codex can complete the task within minutes:

- I've completed the exploration. Here's a summary:

  ———

  DashRTL is a digital EDA tool with a Tcl-based interactive shell (also supports Python mode). It's designed for RTL analysis, elaboration, and physical design preparation. Here are the typical workflows:

  ### Typical DashRTL Workflows

  1. RTL Analysis and Elaboration Flow

  - analyze — Read in HDL files (Verilog/SystemVerilog) with options

for language standard, defines, and file type. Supports auto-detection by file extension.
- read_libs — Load timing libraries before elaboration to prevent macros from becoming black boxes.
- read_flist — Process file lists with compile options.
- elaborate — Build a generic netlist for a specified top module (top-down elaboration with parameter support).
- current_design / current_instance — Set/query the active design and hierarchy context.

2. Physical Design Preparation

- read_lefs — Import technology LEF and cell LEF files for layout data.
- read_def — Read floorplan and scanchain DEF files, with optional site mapping.
- get_floorplan — Query DEF floorplan objects.

3. Design Object Querying (Consistent Pattern)
All get_* commands share a unified interface with flags: -hier, -of, -filter, -exact, -regexp, -nocase, -quiet. Key object types:

- get_cells, get_modules, get_nets, get_pins, get_ports — RTL/Netlist objects
- get_lib_cells, get_lib_pins, get_libs — Library objects
- get_lef_cells, get_lef_pins, get_lef_sites, get_lef_tech — LEF technology objects
- get_functions, get_interfaces — SystemVerilog constructs
- Iteration via foreach_in_objects, filtering with filter_objects, and property access via get_property.

4. Design Management

- reset_design — Clear all design data.
- reload_design — Reset and redo the full analyze → elaborate → read_def flow.
- report_black_box — Report unresolved black boxes, optionally writing dummy .lib, .lef, and Verilog files.
- set_sv_compatibility / report_sv_compatibility — Manage SystemVerilog compatibility settings.

5. Shell and Scripting

- Tcl mode (default) — Full Tcl support with source, proc, foreach, etc.
- Python mode — Switch with pymode(), return with tclmode().
- Plugin framework — plugin load/call/exists for extensibility.
- Cross-probing — crossprobe opens an external editor to navigate to source locations of objects or errors.

6. Tool Configuration

- 15+ tool variables control behavior (e.g., search_path, hdl_default_std, host_max_threads). Managed via list_tool_var, get_tool_var, set_tool_var, reset_tool_var.

7. Help System

- help <topic> — Quick tips for rtl, physical, shell, tcl, objects, tool_var, gui.
- man <command> — Detailed documentation with syntax and related commands.
- <command> -help — Usage summary (not available for all commands).
- Tab completion available throughout.

A

Appendix A: 2cli Options List

`%` **2cli** [-start <eda_tool_exec> [<eda_option>...]]

Launches a new EDA tool process and connects it to the 2cli daemon.

[-log [<log_file_name>]]

Specifies the log file generated by 2cli. A custom log file name may be provided; otherwise, the default log file name is used. This option must appear before the `-start` option.

[-timeout <seconds>]

Specifies the inactivity timeout for 2cli. If the session remains inactive longer than the specified value, the process is automatically terminated.
Default value is `86400` seconds. Set to `0` to disable the timeout mechanism. This option must appear before the `-start` option.

[-query]

Lists all active processes launched by the current user.

[-cleanup]

Removes records and files associated with daemon processes that have already exited.

[-force_cleanup]

Removes all daemon records and files, and forcibly terminates all active processes started by 2cli.

[-pid <pid>]

Specifies the target process PID for sending EDA commands. If not specified, commands are sent to the most recently launched process by default. This option must appear before `<tool_cmd_or_stdin>`.

[<tool_cmd_or_stdin>]
: Sends EDA commands to the 2cli daemon, such as interactive EDA tool commands.
If the command does not end with a special character (for example `\t`), 2cli automatically appends a newline character (`\n`).

[-version | --version]
: Displays the current tool version.

[-help | -h | --help]
: Displays help information.