TclOK

SAVE **HOURS** ON TCL/SDC/UPF SCRIPT FAILURE IN JUST **1 SECOND**

WHY TCLOK?

TclOK is a TCL/SDC/UPF script checking tool, featuring an innovative "static + dynamic" checking mechanism. It helps users quickly scan for basic syntax errors and other logical errors that may cause script execution to terminate before running the script.

In a large project, TCL/SDC/UPF scripts can contain thousands of lines of code and may be nested in multiple script files, making it difficult to detect hidden syntax errors. Therefore, debugging TCL/SDC/UPF scripts is challenging for design engineers, who often need to repeatedly run the scripts in the EDA tool for hours before erroring out.

TclOK can detect most errors within 1 second and provide clear descriptions and locations of the issues, allowing users to quickly resolve the script problems.

KEY FEATURES

Static Check

Performs structural validation of Tcl commands, including basic syntax, argument types, and formal checks.

Dynamic Check

Uses virtual execution mechanism to handle native Tcl/SDC/UPF commands as well as EDA-tool specific commands.

Checking Mode

Offers both default and verbose checking modes to facilitate different usage preferences.

No Dependency

TclOK is not dependent on any other EDA tools. It uses initialization script to customize checking on EDA-tool specific commands.

Auto Run

Supports automatic execution of third-party programs, such as EDA tools, after successful script validation.

Script Debug

Provides a variety of debugging methods, such as variable value printing and conditional/loop statement behavior control.

DashThru

NATIVE TCL CHECKING
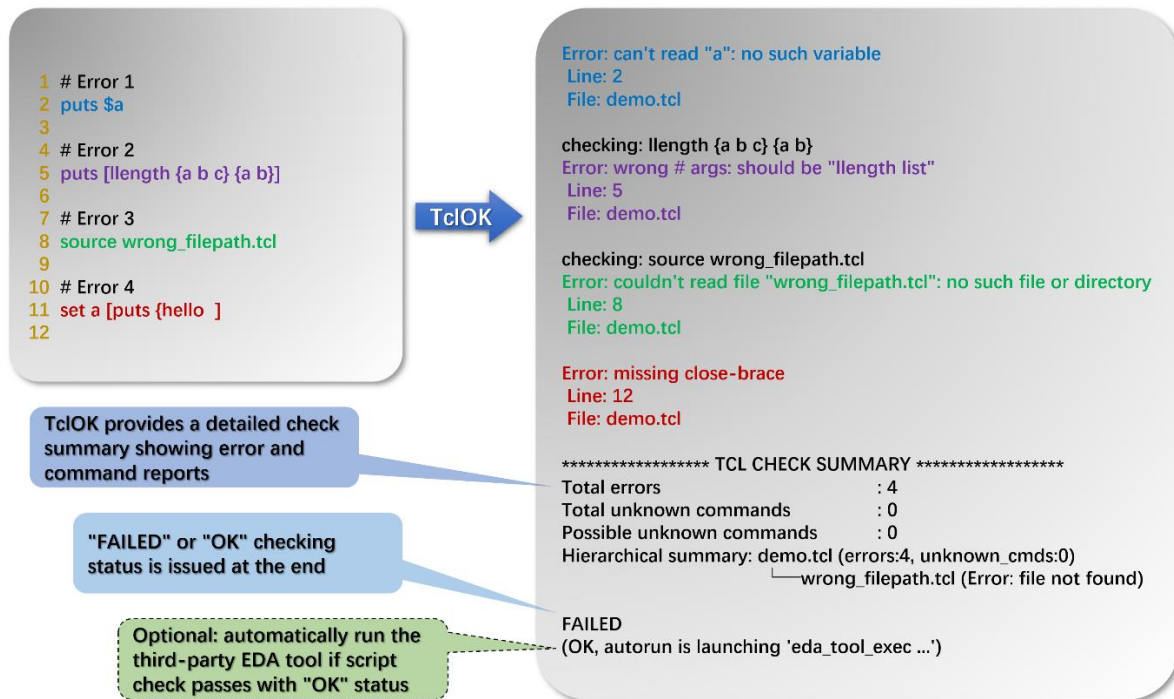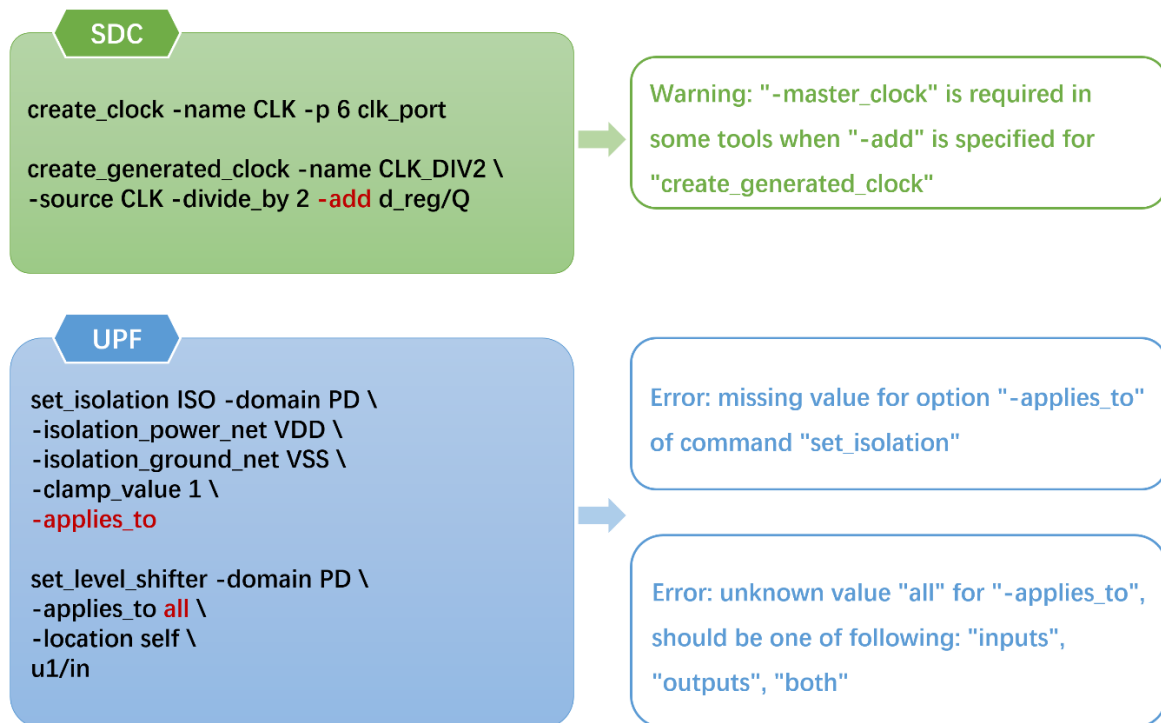
Native Tcl Checking quickly identifies syntax errors, command structure issues, and logical flaws in Tcl scripts. It provides detailed error summaries with precise line-level diagnostics, ensuring scripts are execution-ready.

```
 1  # Error 1
 2  puts $a
 3
 4  # Error 2
 5  puts [llength {a b c} {a b}]
 6
 7  # Error 3
 8  source wrong_filepath.tcl
 9
10  # Error 4
11  set a [puts {hello  ]
12
```

TclOK →

```
Error: can't read "a": no such variable
  Line: 2
  File: demo.tcl

checking: llength {a b c} {a b}
Error: wrong # args: should be "llength list"
  Line: 5
  File: demo.tcl

checking: source wrong_filepath.tcl
Error: couldn't read file "wrong_filepath.tcl": no such file or directory
  Line: 8
  File: demo.tcl

Error: missing close-brace
  Line: 12
  File: demo.tcl


****************** TCL CHECK SUMMARY ******************
Total errors                    : 4
Total unknown commands          : 0
Possible unknown commands       : 0
Hierarchical summary: demo.tcl (errors:4, unknown_cmds:0)
                        └──wrong_filepath.tcl (Error: file not found)

FAILED
(OK, autorun is launching 'eda_tool_exec ...')
```

TclOK provides a detailed check summary showing error and command reports

"FAILED" or "OK" checking status is issued at the end

Optional: automatically run the third-party EDA tool if script check passes with "OK" status

BUILD–IN SDC/UPF CHECKING

Built-in SDC/UPF Checking validates SDC and UPF commands. It thoroughly checks the legality of SDC/UPF command options and arguments, ensuring they meet structural and syntax requirements as well as EDA tool compatibility.

SDC

```
create_clock -name CLK -p 6 clk_port

create_generated_clock -name CLK_DIV2 \
-source CLK -divide_by 2 -add d_reg/Q
```
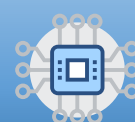
→ Warning: "-master_clock" is required in some tools when "-add" is specified for "create_generated_clock"

UPF

```
set_isolation ISO -domain PD \
-isolation_power_net VDD \
-isolation_ground_net VSS \
-clamp_value 1 \
-applies_to

set_level_shifter -domain PD \
-applies_to all \
-location self \
u1/in
```

→ Error: missing value for option "-applies_to" of command "set_isolation"

Error: unknown value "all" for "-applies_to", should be one of following: "inputs", "outputs", "both"

Contact Us

contact@dashthru.com

https://www.dashthru.com

Resources

TclOK User Guide

TclOK Video

Get Free Trial Now

DashThru