User Manual

SSDynamics

May 8, 2025

Carter K. Chas D. Connor A. Charles D. Savannah Chappus

Chris Ortiz
Senior Technologist, SSD Validation
Western Digital Corp., Flash Business Unit

John Lee
Senior Director, SSD Validation
Western Digital Corp., Flash Business Unit

Introduction

We are pleased that you have chosen NVMe-Gentest for your business needs. NVMe-Gentest is a powerful system for validating the behavior of NVMe storage devices through formal specification-driven, real-world testing. that has been custom-designed to meet your needs. Some of the key highlights include:

- Integration with TLA+ specifications to define valid NVMe state transitions and behaviors
- Randomized test scenario generation with seed resampling for bug isolation
- Direct NVMe CLI command execution on real hardware to ensure authentic results
- A rolling log system with detailed error summaries for efficient debugging

The purpose of this user manual is to help you, the client, successfully install, operate, and maintain NVMe-Gentest in your testing environment. We aim to ensure you can depend on it as a reliable, long-term solution for NVMe device validation.

Installation

Installing our NVMe-Gentest program is as simple as cloning the GitHub repository. Since the program consists of Python scripts, with a bash script to run the Python script, it's easily portable as long as the requirements are installed. The NVMe-Gentest platform only requires the base Python 3.12 standard libraries and the nvme-cli Debian/Ubuntu library. Installing the NVMe-Gentest platform just requires these two libraries from any package manager that has these.

Step-by-Step instructions to install:

1) Run git clone https://github.com/SSDynamics-Capstone/nvme-gentest.git in a terminal window.
2) Run `sudo apt-get install nvme-cli python3-full` to install the required libraries.
3) Change directory into the cloned repository.
4) Run `sudo ./nvme-gentest <nvme-device-path> <tla-specification>` to run the NVMe-Gentest platform against an NVMe drive.

Configuration and Daily Operation

Once NVMe-Gentest is installed and the environment is set up, the following steps will help you configure the system and begin using it for your daily SSD testing workflow. This section covers everything from initial configuration to regular test execution and logging review.

1. Initial Configuration

Before running any tests, a few configuration tasks should be completed:

- **Configure Admin Access:**
  If using in a multi-user test environment, restrict usage by setting permissions on the test directory or wrapping NVMe-Gentest in a privileged shell script.

- **Verify System Requirements:**
  Ensure the following are available on your test system:
  - Linux environment with `nvme-cli` installed
  - Root or sudo privileges for issuing NVMe passthrough commands
  - Python 3.9+ environment

- **Edit Constants File (optional):**
  The file `constants.json` can be modified to set initial conditions such as available namespaces, expected states, or testing limits. This may be required based on the TLA+ specification

2. Running a Basic Test

Use the CLI interface to run a basic test:

```
sudo ./nvme_gentest -c 10 /dev/nvme0n1 tla_specs/YourSpec.tla
```

- `-c`: Number of steps or state transitions to explore
- `/dev/nvme0n1`: Your NVMe target device (not the namespace)
  - Can easily find all available devices with `sudo nvme list`
- `tla_specs/YourSpec.tla`: File path to your TLA+ test specification file

3. Advanced Test Configuration (Flags)

NVMe-Gentest supports additional CLI flags to configure test behavior:

- `-h, --help` show help message and exit
- `-c, --count:` number of states to explore
- `-C --constants:` constant file path (default constants.json)
- `-S, --seed`: Set a specific test seed (for reproducibility)
- `-r, --retry_max:` Max retries on resampled seed (default 100)
- `-b, --bug_max:` threshold on resampled seed before test termination
- `-d, --detached:` run in detached mode
- `-o, --output:` specifies output file
- `-g, --generative`: Enable ongoing seed sampling after each complete run
- `-D, --debug`: Run without submitting commands to the NVMe CLI
- `-L, --log-lines`: Set max number of log lines to retain in rolling log
- `-s, --silent:` Silent mode, avoids printing to console

Example:

```
sudo ./nvme_gentest -l 000 -c 20 -S 12345 --generative
/dev/nvme0n1 tla_specs/SSD_NAU_TEST.tla
```

4. Monitoring Output

- All output is written to `nvme.log` in the working directory.
- The log includes a summary of all errors at the end, which can be used to debug issues and compare against expected TLA+ results.

5. Daily Tasks and Best Practices

- **Track Seeds and Errors:**
  If a bug is detected, the seed is saved for reproduction. Keep a record of bug-triggering seeds and output logs for later review.

- **Validate Device State:**
  After a test, especially one that ends with an error, verify that the NVMe device is

still accessible and in the expected format.

- **Update TLA+ Specs as Needed:**
  As testing goals evolve, update or expand your TLA+ specs to reflect new state transitions or command sequences.

Maintenance

- **Update NVMe Cli:**
  Keep the most up to date version of NVMe Cli, if any changes occur to commands you may have to change some tla+ logic for a specification.

- **Update Pluspy:**
  If any updates occur to pluspy, you should update the pluspy in the codebase, there may be pluspy updates that are needed to keep the TLA+ accurate.

- **Update parsing:**
  In the case that the NVMe Cli has a large change and the output to the command line change this may break the parsing, so that would need to be updated.

Troubleshooting

Code

Problem	Cause	Fix
Nvme command fails	Nvme-cli not installed or in $PATH	Run “which nvme” or attempt to install with sudo apt install nvme-cli
Permission issues	Program needs to be run as root/sudo	Either run as sudo, su or modify permission groups
Python subprocess errors	Incorrect command syntax or missing CLI	Validate the last command in CLI manually
Result mismatch false positive	Compatibility issues	Check versioning, may need to modify the comparison code to your needs

Module import errors	Wrong working directory, should be executing from project root	If you do not wish to do it this way, you can also do absolute imports, but makes the project less portable
Script hangs	The nvme cli is likely blocking or there is some deadlock	May need to add timeouts

TLA+

Problem	Cause	Fix
Model dosen't match real-world behavior	Specification too broad or incorrect	Validate system invariants and outputs
Syntax errors in .tla	TLA+ is white space/indent sensitive and can often cause issues	Validate spacing
TLA+ unknown operator	TLA+ and PlusPy are two different interpreters acting on the same specification	May need to modify code or use more basic TLA+ code, (cannot use PlusCal)

Virtualization

Problem	Cause	Fix
NVMe devices not visible in VM	Passthrough not enabled	Check IOMMU and passthrough when using virt-manager/qemu
Script behaves differently in VM than host	Environment mismatch	Standardized test environment

Hardware

Problem	Cause	Fix
Device not listed	Kernel module not loaded or faulty device	Use lsblk or dmesg
Frequent timeouts Intermediate I/O errors	Power/Thermal issues	Check nvme smart-log
NVMe device disappears	Driver/firmware crash	Check lsblk and dmesg

Conclusion

We are proud to formally close this document with deep gratitude for the opportunity to work with you on NVMe-Gentest. It has been a rewarding experience to contribute to the development of a reliable and specification-driven SSD validation tool, and we sincerely hope it brings you many years of productive use.

Your organization’s commitment to robust NVMe device testing has inspired us to deliver a tool that is both powerful and user-friendly. It’s our belief that NVMe-Gentest will serve as a solid foundation for your validation workflows, helping uncover edge cases and streamline device qualification processes.

Thank you again for your trust and collaboration.