NVMECMD USER GUIDE

www.epicutils.com

OVERVIEW

nvmecmd is a command line utility to run NVMe Admin commands on Windows or Linux computers. The utility was written in C++ and uses the OS drivers to run the commands. Commands are defined in a json file called a command file which is entered as the first command line parameter. Below is an example of how to read NVMe information using the read.cmd.json file. Run this command from a terminal or console window:

nvmecmd read.cmd.json

This command produces the sample output here:

By default, the above command creates three files in the local directory. Nvmecmd.trace.log is a copy of the display output (same as above), nvme.info.json is a json file with the NVMe information, and read.summary.json is a json file that summarizes the command. These files are described in more detail below.

READ COMMAND TYPE

There are currently two command types: Read and Self-Test. The plan is to add more types in the future. This section describes the read command type. The read command type specifies to read the NVMe information. It has the following parameters defined in the command file.

Parameter	Comment
command type	The command type. This must be read for this type
short description	A short description of the command, in this example Read Info
samples [1]	Number of times to read the drive information
interval in ms [1]	Time between samples in milliseconds

compare type	Possible values are Default, Firmware, and ALL. See Compare section below for details
compare file [1]	File to compare. If not used then specify empty string “”
display sample rate	Updates display at this rate
log sample rate	Logs information to file at this rate
rules file [1]	Rules file to verify. If not used then specify empty string “”
Fail limit	Abort after this number of failures
high priority	On Windows OS, run the application at higher priority. On Linux has no effect.
high resolution timer	Request Windows OS to run with a higher resolution system timer. This is typically 4mS instead of 16mS. On Linux has no effect.
log hex data	Log the raw hex data in the nvme.info.json file
read system data	Read the system data from the OS
read extended system data	Read additional system data from the OS
read identify controller	Read the controller information. Must be true
read identify namespace	Read the namespace information
read feature <nn>	Read feature <nn> where <nn> is the feature number (e.g. 01h)
read log page >nn>	Read log page <nn> where <nn> is the log page number (e.g. 01h)

[1] These parameters can also be specified on the command line

For example, a command file can be created to only read log page 2 by specifying read log page 02h as true and all other read parameters as false. The release package contains several example command files such as read.cmd.json, logpage02.cmd.json, and logpage03.cmd.json.

LOG FILES

The command above creates the three log files below.

- read.summary.json Summary of the command run in json format
- nvmecmd.trace.log Capture of information displayed when command runs
- nvme.info.json NVMe drive information in json format

The read.summary.json file contains the results of the command, time begun and end, settings used, admin command completion times, and some other metadata.

The nvmecmd.trace.log contains the display output. In some rare cases, specifically when errors occur, this file may contain additional data that was not displayed.

The nvme.info.json file contains the NVMe and system information for one sample. In the case where multiple samples were run there can be multiple files created. If a compare or verify is done then this file contains the number of mismatches from the compare and the rule violations from the verify.

COMPARE INFORMATION

There are three methods used for the comparison: DEFAULT, firmware and ALL. The ALL comparison compares the values of all parameters. If any parameter has a different value from the value in the compare file a mismatch is displayed. The DEFAULT comparison compares static parameters and, if the same drive, Log Page 2 counters. Specifically, the DEFAULT comparison will:

- Compare all parameters from the Admin Identify Controller and Identify Namespace commands
- Compare all PCI and driver parameters from the OS
- NOT compare any log pages except for:
    - Log Page 3
    - Log Page 2 except for:
        - composite temperature
        - temperature sensors
        - If the same drive remaining counter parameters must be greater or equal to the previous values in the compare file
- Compare features except for:
    - Feature 2 (Power Management)
    - Feature 0xE (Timestamp)
    - Feature 0xD (Host Memory Buffer)
        - Host Memory Descriptor List Address (HMDLAL)
        - Host Memory Descriptor List Address (HMDLAU)

The firmware compare is the same as above except ignores log page 3.

VERIFY INFORMATION AND RULES

Rule files are json files that contain an array of strings where each string defines a rule. The files have a .rules.json extension. Below is an example file:

```
{
  "rules": [
    "'Critical Warnings' match No",
    "'Composite Temperature' > 20 C",
    "'Composite Temperature' < 82 C",
    "'Media and Data Integrity Errors' = 0",
    "'Warning Composite Temperature Time' = 0 Min",
    "'Critical Composite Temperature Time' = 0 Min",
    "'Device Self-test Command' match Supported",
    "'Number Of Failed Self-Tests' = 0",
    "'Extended Device Self-test Time (EDSTT)' > 0 Min",
    "'PCI Width' match x2",
    "'PCI Speed' match Gen3 8.0GT/s",
    "'RTD3 Entry Latency (RTD3E)'  < 500000 uS",
    "'RTD3 Resume Latency (RTD3R)' < 1000000 uS",
    "'Firmware Activation Without Reset' match Supported",
    "'Power State 0 Maximum Power (MP)' < 5 Watts",
    "'Power State 1 Maximum Power (MP)' < 4 Watts",
    "'Power State 2 Maximum Power (MP)' < 3 Watts",
    "'Power State 3 Maximum Power (MP)' < 0.1 Watts",
    "'Power State 4 Maximum Power (MP)' < 0.01 Watts"
  ]
}
```

Each string has the format: <parameter> <operator> <value> where…

<parameter> is the parameter name in single quotes
<operator> can be a numeric operator (<, <=, =, !=, >=, or >) or a string operator (match, not-match)
<value> is the value of the parameter to verify using the operator

Numeric operators take into account units so the value must include the proper unit. For example, a power measurement value can be 5 Watts but not 5.

FINDING PARAMETER NAMES

Creating new rules requires knowing the parameter name. The parameter name can be found in the parameter_list.json file which can be created with the –export command line option.

nvmecmd –export

Wherever possible parameter names match the NVMe specification.

SELF-TEST COMMAND TYPE

The self-test command type has the following parameters in the command file:

Parameter	Comment
command type	The command type. This must be self-test for this type
short description	A short description of the command, in this example Self-Test
display sample interval in sec	Number of seconds between updates to the display
extended self-test	Run the extended self-test instead of the short self-test
read command times	Logs the admin command times in self-test.summary.json. This allows the user to measure the amount of time each command took and the overall distribution of times
read smart data	Reads log page 2 SMART data on each update. This allows the displaying of temperature on each update
abort time	Number of seconds to wait before aborting self-test. A value of 0 indicates not to abort and complete the test. This parameter allows testing of the abort function.
high priority	On Windows OS, run the application at higher priority. On Linux has no effect.
high resolution timer	Request Windows OS to run with a higher resolution system timer. This is typically 4mS instead of 16mS. On Linux has no effect.

Here is the self-test.cmd.json file included with the release package.

```
{
  "cmd type": "self-test",
  "short description": "Self-Test",
  "display sample interval in sec": 10,
  "extended self-test": false,
  "read command times": true,
  "read smart data": true,
  "abort time": 0,
  "high priority": false,
  "high resolution timer": false
}
```

Run this command from a terminal or console window to complete a short self-test.

nvmecmd self-test.cmd.json

Run this command to complete an extended self-test.

nvmecmd self-test.cmd.json --extended

COMMAND LINE OPTIONS

To view the available command line options run this command:

nvmecmd -h

Command line options take precedence over the command file.

ACKNOWLEDGEMENTS

The following very cool add-ins are used in the project:

- jsoncpp https://github.com/open-source-parsers/jsoncpp
- plog https://github.com/SergiusTheBest/plog
- boost https://www.boost.org/