MyWebUniversity
python
Red Hat Enterprise Linux Manual Pages
FPDF Library PDF generator
FPDF
Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'nvme-id-ns.1' command

$ man nvme-id-ns.1

NVME-ID-NS(1)   NVMe Manual   NVME-ID-NS(1)

NAME

nvme-id-ns - Send NVMe Identify Namespace, return result and structure

SYNOPSIS

```
nvme id-ns <device> [-v | --vendor-specific] [-b | --raw-binary]
                [--namespace-id=<nsid> | -n <nsid>] [--force]
                [--human-readable | -H]
                [--output-format=<fmt> | -o <fmt>]
```

DESCRIPTION

For the NVMe device given, sends an identify namespace command and provides the result and returned structure.

The <device> parameter is mandatory and may be either the NVMe character device (ex: /dev/nvme0), or a namespace block device (ex: /dev/nvme0n1). If the character device is given, the '--namespace-id' option is mandatory, otherwise it will use the ns-id of the namespace for the block device you opened. For block devices, the ns-id used can be overridden with the same option.

On success, the structure may be returned in one of several ways depending on the option flags; the structure may be parsed by the program or the raw buffer may be printed to stdout.

OPTIONS

-n <nsid>, --namespace-id=<nsid>

Retrieve the identify namespace structure for the given nsid. This

is required for the character devices, or overrides the block nsid
if given. If the controller supports namespace management
capability and 0xFFFFFFFF is given, then the controller returns the
identify namespace structure that specifies common capabilities
across namespaces for the controller.

--force

Request controller return the identify namespace structure even if
the namespace is not attached to the controller. This is valid only
for controllers at or newer than revision 1.2. Controllers at
revision lower than this may interpret the command incorrectly.

-b, --raw-binary

Print the raw buffer to stdout. Structure is not parsed by program.
This overrides the vendor specific and human readable options.

-v, --vendor-specific

In addition to parsing known fields, this option will dump the
vendor specific region of the structure in hex with ascii
interpretation.

-H, --human-readable

This option will parse and format many of the bit fields into
human-readable formats.

-o <format>, --output-format=<format>

Set the reporting format to normal, json, or binary. Only one
output format can be used at a time.

EXAMPLES

? Has the program interpret the returned buffer and display the known
fields in a human readable format:

```
# nvme id-ns /dev/nvme0n1
```

? If using the character device or overriding namespace id:

```
# nvme id-ns /dev/nvme0 -n 1
# nvme id-ns /dev/nvme0n1 -n 1
# nvme id-ns /dev/nvme0 --namespace-id=1
```

? In addition to showing the known fields, have the program to
display the vendor unique field:

```
    # nvme id-ns /dev/nvme0n1 --vendor-specific

    # nvme id-ns /dev/nvme0n1 -v
```

The above will dump the 'vs' buffer in hex since it doesn?t know how to interpret it.

? Have the program return the raw structure in binary:

```
    # nvme id-ns /dev/nvme0n1 --raw-binary > id_ns.raw

    # nvme id-ns /dev/nvme0n1 -b > id_ns.raw
```

It is probably a bad idea to not redirect stdout when using this mode.

? Alternatively you may want to send the data to another program that can parse the raw buffer.

```
    # nvme id-ns /dev/nvme0n1 --raw-binary | nvme_parse_id_ns
```

The parse program in the above example can be a program that shows the structure in a way you like. The following program is such an example that will parse it and can accept the output through a pipe, '|', as shown in the above example, or you can 'cat' a saved output buffer to it.

```
/* File: nvme_parse_id_ns.c */

#include <linux/nvme.h>

#include <stdio.h>

#include <unistd.h>

int main(int argc, char **argv)

{

        unsigned char buf[sizeof(struct nvme_id_ns)];

        struct nvme_id_ns *ns = (struct nvme_id_ns *)buf;

        if (read(STDIN_FILENO, buf, sizeof(buf)))

                return 1;

        printf("nsze : %#llx\n", ns->nsze);

        printf("ncap : %#llx\n", ns->ncap);

        return 0;

}
```

NVME