Setting up TCG Ruby with Sedutil

Abstract

This document describes setting up an Ultrastar® DC SN640 or DC SN840 NVMe™ TCG Ruby compliant SSD by means of a tool called sedutil. This will introduce the basics of the TCG Ruby specification, Western Digital's implementation, and some key operations on ranges. This document will also introduce the tool 'sedutil' and use sedutil to identify TCG Ruby drives, set passwords, set bands, and lock and unlock drives.

This document is not a discussion of TCG Ruby encryption details. More information can be downloaded from the Trusted Computing Group website. See notes at the end of the document.

TCG Ruby

The purpose for TCG Ruby is to provide an up-to-date enterprise Security Subsystem Class (SSC) to support NVMe datacenter drives with simplified implementation and integration.

It is part of the broader Opal SSC (such as Pyrite and Opalite) and has protocol compatibility with the Opal family. However, TCG Ruby is slightly more limited in what it supports as some features are not considered relevant to an Enterprise drive. For example, preboot authentication support is optional.

Western Digital's specific implementation of the TCG Ruby specification (see below a comparison about Opal v2.01 and TCG Ruby) makes it very similar to Opal, but preboot authentication is not supported. [1, 2]

Feature	Opal V2.01	TCG Ruby
Activation and Life Cycle	Yes	Yes
Number of Admin, Users	4 Admin, 8 Users	1 Admin, 2 Users
Min Number of Required Logical Block Addressing (LBA) Ranges	Global Range +8	Global Range (+8 on Western Digital drives)
Min Datastore Size	10MB	128 KB
Min MBR Table Size	128MB	Optional: 128MB if supported
Configurable Access Control	Yes	Yes
PSID	Yes	Yes
Media Encryption	Required	Required
Crypto Erase	Revert, Revert SP, GenKey methods for device and locking range level erase granularity	Same as Opal

The Logical Block Addressing (LBA) Ranges, are spans of LBAs into which the drive can be partitioned and that share the same encryption details. Some rules apply:

1. Ranges cannot overlap, except for Range0 which, by definition is the global range that encompass the whole capacity of the drive.
2. Once a range is set up by defining the start LBA and the LBA count, it can be locked to writes or reads/writes.
3. Any lock shall be enabled before the lock settings are effective: disabling the 'lock enable', disables locking, though the range stays in place.
4. A range can be crypto erased by re-keying which will change the encryption key for that range.

SID is the entity that can be used to authenticate to the drive and freeze or remove other entities initially.

Admin1 is the administrator of all the ranges: it can operate on any range and perform any change. User1 and User2 are entities whose access capability is configurable. For instance they might be enabled to operate only on some specific ranges, or not be enabled to erase the data but only lock reads or writes on those ranges.

Downloading and Compiling the Sedutil Tool

The first step is to download the tools from the GitHub and compile them. The test in this document was performed on Linux® Ubuntu 18.04.03 (it needs the kernel switch *libata_tpm* set at boot time), but the specific tool versions should also work on Windows and FreeBSD. Windows compilation would require VSC 2015 or later.

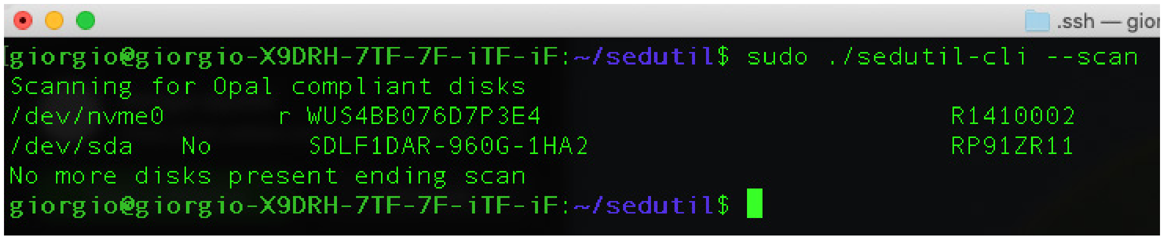
The commands to download and compile are below: please note git will create a directory called sedutil in any directory it is run from. If different is needed, please check the git manual for help.

```
git clone https://github.com/amotin/sedutil.git
autoreconf -i
./configure
make
(make install) // not required if you do not want to have the tool installed system wide
```

At the end of the process the tool will be compiled, and there will be an executable called *sedutil-cli* on the source directory. The executable *sedutil-cli* may be installed system wide the last command was issued.

Taking Control of the Drive

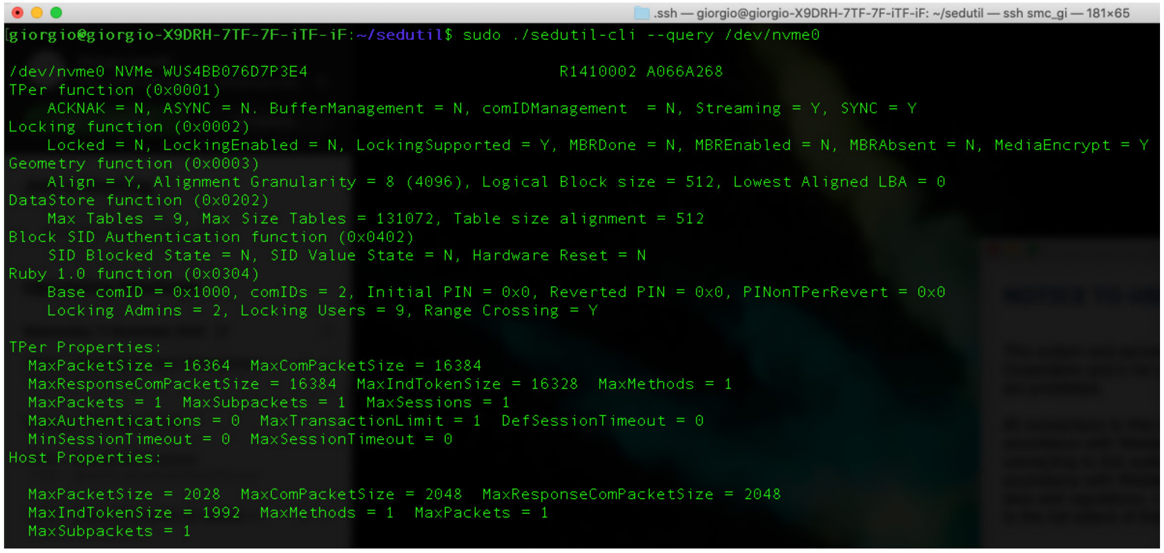
The next step is to scan the drives in the system and check for compliant drives. In this example the drive is compliant with the TCG Ruby standard, and this is identified with a small r close to the drive, as in Figure 1.



```
giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$ sudo ./sedutil-cli --scan
Scanning for Opal compliant disks
/dev/nvme0        r WUS4BB076D7P3E4                                R1410002
/dev/sda    No      SDLF1DAR-960G-1HA2                             RP91ZR11
No more disks present ending scan
giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$
```

Figure 1. Scanning drives in the system

The drive can be queried for its supported features, see Figure 2 below: note the logical block size is 512bytes for the Ultrastar DC SN640 or DC SN840 with TCG Ruby.

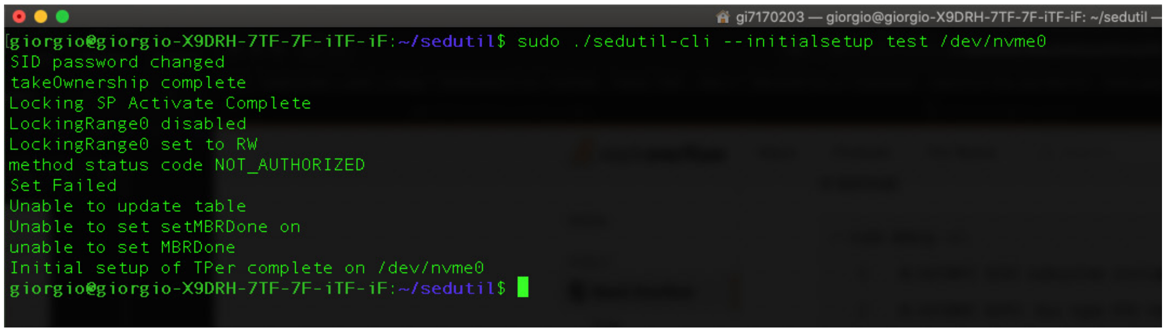

.ssh — giorgio@giorgio-X9DRH-7TF-7F-iTF-iF: ~/sedutil — ssh smc_gi — 181×65

```
giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$ sudo ./sedutil-cli --query /dev/nvme0

/dev/nvme0 NVMe WUS4BB076D7P3E4                          R1410002 A066A268
TPer function (0x0001)
    ACKNAK = N, ASYNC = N. BufferManagement = N, comIDManagement  = N, Streaming = Y, SYNC = Y
Locking function (0x0002)
    Locked = N, LockingEnabled = N, LockingSupported = Y, MBRDone = N, MBREnabled = N, MBRAbsent = N, MediaEncrypt = Y
Geometry function (0x0003)
    Align = Y, Alignment Granularity = 8 (4096), Logical Block size = 512, Lowest Aligned LBA = 0
DataStore function (0x0202)
    Max Tables = 9, Max Size Tables = 131072, Table size alignment = 512
Block SID Authentication function (0x0402)
    SID Blocked State = N, SID Value State = N, Hardware Reset = N
Ruby 1.0 function (0x0304)
    Base comID = 0x1000, comIDs = 2, Initial PIN = 0x0, Reverted PIN = 0x0, PINonTPerRevert = 0x0
    Locking Admins = 2, Locking Users = 9, Range Crossing = Y

TPer Properties:
  MaxPacketSize = 16364  MaxComPacketSize = 16384
  MaxResponseComPacketSize = 16384  MaxIndTokenSize = 16328  MaxMethods = 1
  MaxPackets = 1  MaxSubpackets = 1  MaxSessions = 1
  MaxAuthentications = 0  MaxTransactionLimit = 1  DefSessionTimeout = 0
  MinSessionTimeout = 0  MaxSessionTimeout = 0
Host Properties:

  MaxPacketSize = 2028  MaxComPacketSize = 2048  MaxResponseComPacketSize = 2048
  MaxIndTokenSize = 1992  MaxMethods = 1  MaxPackets = 1
  MaxSubpackets = 1
```

Figure 2. Query of Ultrastar DC SN640 drive model

Taking control of the drive is a process called "Take Ownership". This will change the default passwords (SID and admin1), initially set to an electronically readable value referred as MSID, to your password.

With sedutil this involves issuing the command in Figure 3, changing the password, in this example, to "test":



```
gi7170203 — giorgio@giorgio-X9DRH-7TF-7F-iTF-iF: ~/sedutil —
[giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$ sudo ./sedutil-cli --initialsetup test /dev/nvme0
SID password changed
takeOwnership complete
Locking SP Activate Complete
LockingRange0 disabled
LockingRange0 set to RW
method status code NOT_AUTHORIZED
Set Failed
Unable to update table
Unable to set setMBRDone on
unable to set MBRDone
Initial setup of TPer complete on /dev/nvme0
giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$
```

Figure 3. Taking ownership

The error message shown in Figure 3 comes from the fact the TCG Ruby implementation is not supporting MBR boot, and can safely be ignored. Note that both admin1 and SID passwords are changed. All the commands below use the admin1 password and eventually SID and admin1 can be changed one by one.

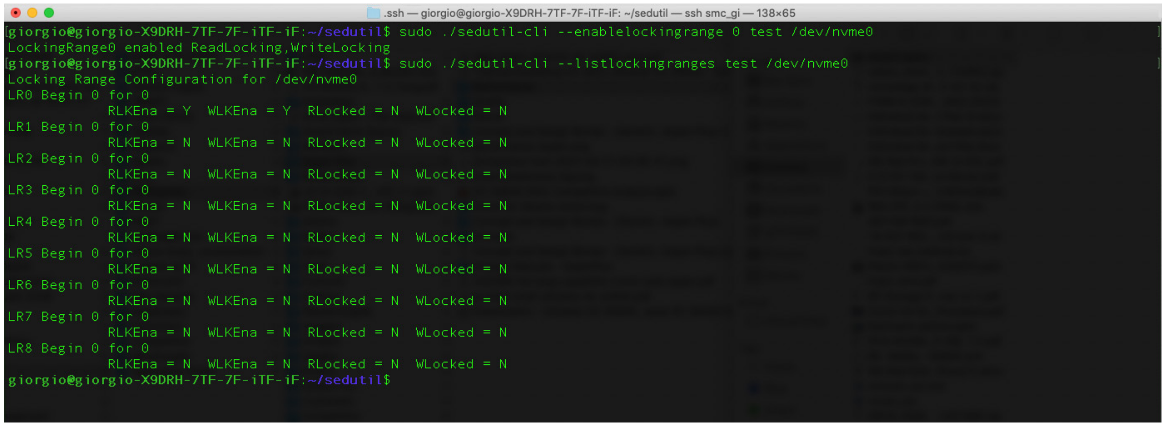
At this stage encryption has been initialized and the passwords are now different from their default values. It is good practice to change the SID and Admin1 passwords to different values after the initial setup. This can be accomplished with:

--setSIDPassword <SIDPassword> <newSIDPassword> <device>
--setPassword <oldpassword> admin1 <newPassword> <device>

Note that except for *revertTPer* command which returns the drive to default values, and the *PSIDrevert* which uses the PSID printed on drive's label, any command uses the admin1 password.

Setting up Bands and Locking

A TCG Ruby compliant device has at least one global band (called band0) and optionally more. In Western Digital's implementation up to 8 bands are supported. You can list the bands and their status with the command --*listlockingranges*, shown in Figure 4.



```
.ssh — giorgio@giorgio-X9DRH-7TF-7F-iTF-iF: ~/sedutil — ssh smc_gi — 138×65
[giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$ sudo ./sedutil-cli --enablelockingrange 0 test /dev/nvme0
LockingRange0 enabled ReadLocking,WriteLocking
[giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$ sudo ./sedutil-cli --listlockingranges test /dev/nvme0
Locking Range Configuration for /dev/nvme0
LR0 Begin 0 for 0
            RLKEna = Y  WLKEna = Y  RLocked = N  WLocked = N
LR1 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR2 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR3 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR4 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR5 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR6 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR7 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR8 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$
```

Figure 4. Listing and enabling global range

Locking a band is a 2-step process: reads or reads/writes needs to be locked (reflected in *RLocked*, *WLocked*), as well as write and read locks need to be enabled (reflected in *RLKEna* and *WLKEna*). Setting a lock but not enabling it, will result in no lock.

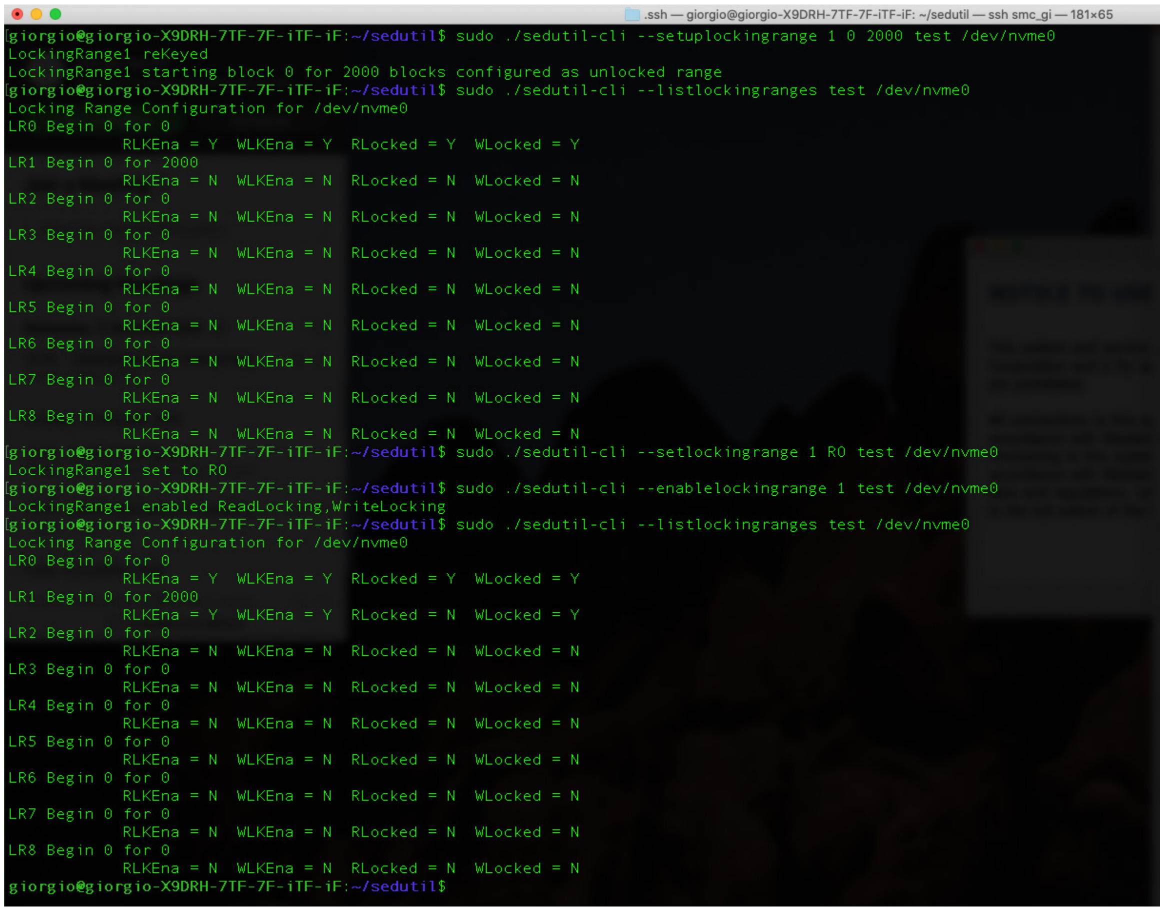
Figure 4 demonstrates enabling the global range (with the option --*enablelockingrange* and using our password "test") and then lists the status of the bands.

In this case the 8 bands are undefined, and the global band has been enabled.

Next is an example of setting up one band, specifically band 1.

Bands are defined as starting from a block number (512bytes in our case, see the query at the beginning for the specific value for each drive type) and extending from that block for a certain number of blocks. Bands cannot overlap except for the global band, which by definition extends over the entire capacity.

Figure 5 demonstrates setting up band1 from block 0 for 2000 blocks using the option --*setuplockingrange <band#> <from block #> <to block #>*. The password in Figure 5 is "test":


.ssh — giorgio@giorgio-X9DRH-7TF-7F-iTF-iF: ~/sedutil — ssh smc_gi — 181×65

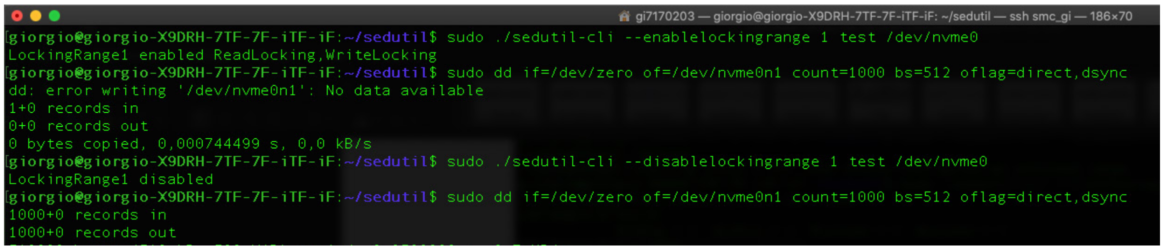
```
[giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$ sudo ./sedutil-cli --setuplockingrange 1 0 2000 test /dev/nvme0
LockingRange1 reKeyed
LockingRange1 starting block 0 for 2000 blocks configured as unlocked range
[giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$ sudo ./sedutil-cli --listlockingranges test /dev/nvme0
Locking Range Configuration for /dev/nvme0
LR0 Begin 0 for 0
            RLKEna = Y  WLKEna = Y  RLocked = Y  WLocked = Y
LR1 Begin 0 for 2000
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR2 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR3 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR4 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR5 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR6 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR7 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR8 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
[giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$ sudo ./sedutil-cli --setlockingrange 1 RO test /dev/nvme0
LockingRange1 set to RO
[giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$ sudo ./sedutil-cli --enablelockingrange 1 test /dev/nvme0
LockingRange1 enabled ReadLocking,WriteLocking
[giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$ sudo ./sedutil-cli --listlockingranges test /dev/nvme0
Locking Range Configuration for /dev/nvme0
LR0 Begin 0 for 0
            RLKEna = Y  WLKEna = Y  RLocked = Y  WLocked = Y
LR1 Begin 0 for 2000
            RLKEna = Y  WLKEna = Y  RLocked = N  WLocked = Y
LR2 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR3 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR4 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR5 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR6 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR7 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
LR8 Begin 0 for 0
            RLKEna = N  WLKEna = N  RLocked = N  WLocked = N
giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$
```

Figure 5. Setting up band1

At this point, band 1 (or locking range) is set and write locked (*WLocked=Y*). –This example uses the --*setlockingrange* option with the RO (read only) switch. It is also possible to enable the full lock if needed, using the LK switch, and to remove it, using RW.

Testing the Locked Drive

The locked drive can be tested, and only when the lock is disabled can it be written. This is demonstrated in Figure 6.



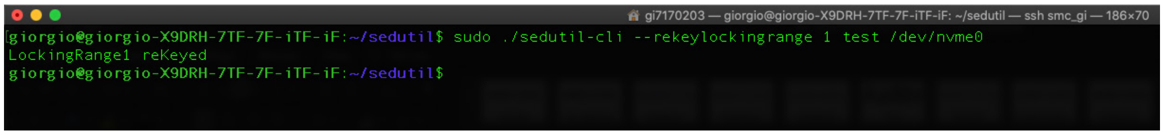
```
gi7170203 — giorgio@giorgio-X9DRH-7TF-7F-iTF-iF: ~/sedutil — ssh smc_gi — 186×70
[giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$ sudo ./sedutil-cli --enablelockingrange 1 test /dev/nvme0
LockingRange1 enabled ReadLocking,WriteLocking
[giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$ sudo dd if=/dev/zero of=/dev/nvme0n1 count=1000 bs=512 oflag=direct,dsync
dd: error writing '/dev/nvme0n1': No data available
1+0 records in
0+0 records out
0 bytes copied, 0,000744499 s, 0,0 kB/s
[giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$ sudo ./sedutil-cli --disablelockingrange 1 test /dev/nvme0
LockingRange1 disabled
[giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$ sudo dd if=/dev/zero of=/dev/nvme0n1 count=1000 bs=512 oflag=direct,dsync
1000+0 records in
1000+0 records out
```

Figure 6. Enabling and disabling locking

The list of operations performed are:

1. Enable locking (although only the write lock was selected with this command we enable both the lock and setting up the lock).
2. Try to write to the drive with dd. The command fails.
3. Disable locking.
4. Write to the device and succeed.

Finally the locking range can be rekeyed which will change the encryption key for that specific band. This will properly erase the band since the original key is changed and cannot be recovered.

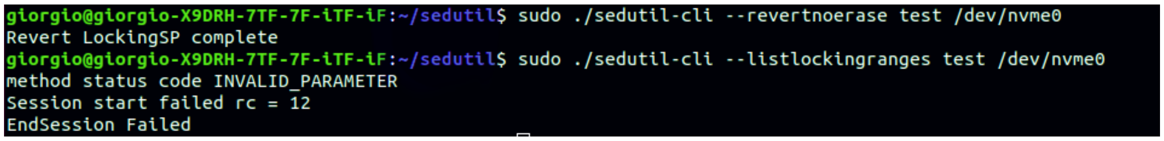


```
gi7170203 — giorgio@giorgio-X9DRH-7TF-7F-iTF-iF: ~/sedutil — ssh smc_gi — 186×70
[giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$ sudo ./sedutil-cli --rekeylockingrange 1 test /dev/nvme0
LockingRange1 reKeyed
giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$
```

Figure 7. Rekey locking range 1

Reverting to Defaults

You can revert to defaults without erasing the drive with --*revertNoErase*, as shown in Figure 8.



```
giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$ sudo ./sedutil-cli --revertnoerase test /dev/nvme0
Revert LockingSP complete
giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$ sudo ./sedutil-cli --listlockingranges test /dev/nvme0
method status code INVALID_PARAMETER
Session start failed rc = 12
EndSession Failed
```

Figure 8. Revertnoerase command

Reverting to Defaults in Case the SID Password is Lost

It is always possible to reset the drive to defaults if you forget your password, albeit losing all data by means of the PSID password, that is written on the disk's label (Figure 8), and that cannot be read electronically from the drive.

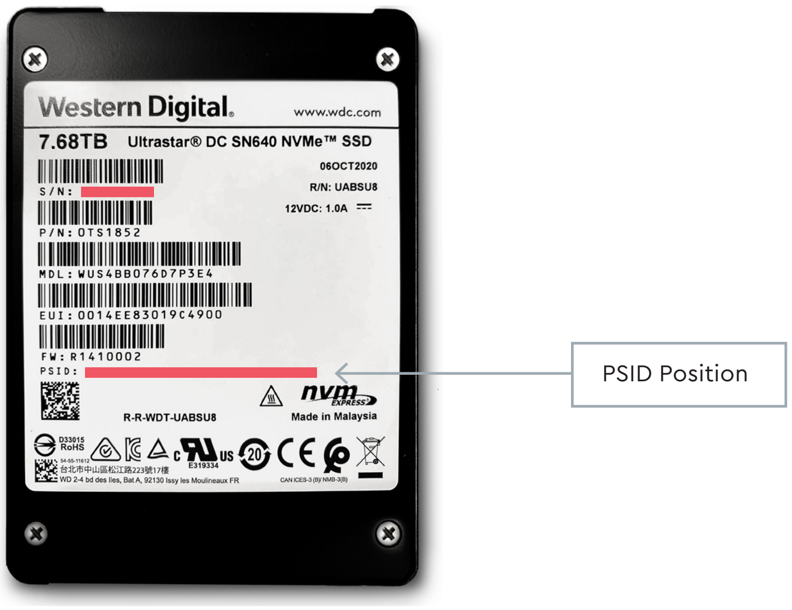
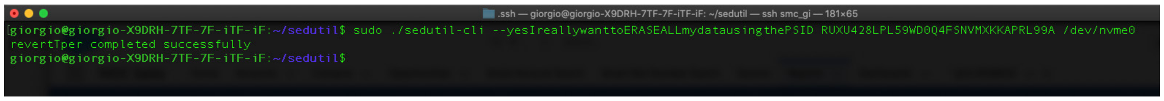


Figure 9. PSID position on the label

The command clearly warns you about erasing your data (Figure 9).


.ssh — giorgio@giorgio-X9DRH-7TF-7F-iTF-iF: ~/sedutil — ssh smc_gi — 181×65

```
giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$ sudo ./sedutil-cli --yesIreallywanttoERASEALLmydatausingthePSID RUXU428LPL59WD0Q4FSNVMXKKAPRL99A /dev/nvme0
revertTper completed successfully
giorgio@giorgio-X9DRH-7TF-7F-iTF-iF:~/sedutil$
```

Figure 10. Restoring defaults with PSID

Conclusions

This paper presents an open source sedutil tool that allows a user to configure encryption on an Ultrastar DC SN640 or DC SN840 NVMe SSD TCG Ruby compliant SSD. Basic concepts and commands have been discussed, showing how to take control of the drive, encrypt the data, erase and restore the drive to defaults.

References
[1] https://trustedcomputinggroup.org/wp-content/uploads/TCG_Storage_SSC_Ruby_v1_r1_pub.pdf
[2] https://trustedcomputinggroup.org/wp-content/uploads/TCG_Storage-Opal_SSC_v2.01_rev1.00.pdf
[3] https://www.flashmemorysummit.com/English/Collaterals/Proceedings/2018/20180807_SECU-102B-1_Tipton.pdf
[4] https://trustedcomputinggroup.org/wp-content/uploads/Storage-Ruby-SSC-v1.0-Specification-FAQ_20182811_Final.pdf
[5] https://nvmexpress.org/wp-content/uploads/TCGandNVMe_Joint_White_Paper-TCG_Storage_Opal_and_NVMe_FINAL.pdf

Western Digital.

5601 Great Oaks Parkway
San Jose, CA 95119, USA
www.westerndigital.com