Hedvig Troubleshooting Guide

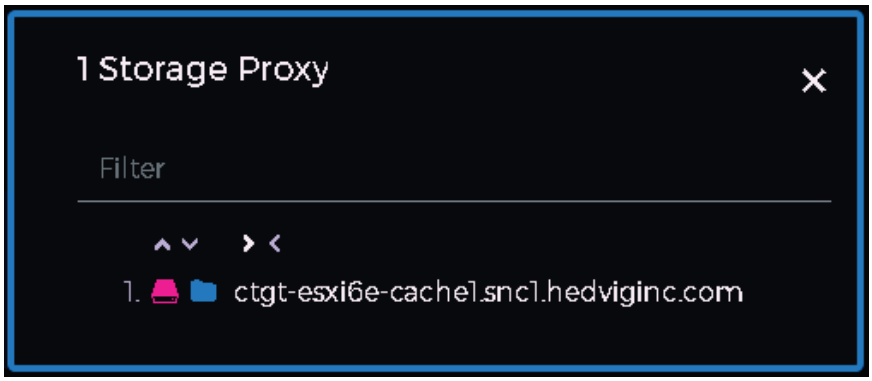
Table of Contents

Hedvig Storage Proxy service status (Hedvig WebUI)

You can determine the status of Hedvig Storage Proxy services by simply locating the storage proxy on the **Cluster Watch** page and clicking on the storage proxy.

Depending on your Hedvig theme (color scheme), the failed status will show up as either red or pink.

In this example, the tgt (block) process is in a down state.



Hedvig Storage Cluster Node service status (Hedvig WebUI)

If a Hedvig Storage Cluster Node is down due to a network connectivity outage or power outage — or if one of the main data moving or metadata services is down — you will see that depicted in the Hedvig WebUI.

In this example, a node with a service is in a down state.

You can see the down state by viewing the **Cluster Watch** page or by clicking the Alerts icon in the upper right corner.

Hedvig Storage Proxy health status (Hedvig CLI)

You can use the Hedvig CLI and log files to determine Hedvig Storage Proxy health status.

Using the crm_mon command for an HA configuration

To determine the health status of the storage proxies in a High Availably (HA) configuration, log into one of the storage proxy's Hedvig CLI, and run the following command:

```
crm_mon
```

The following example output shows two active storage proxies, where the active storage proxy is `cvm1.hedviginc.com` within the pair:

```
Online: [ cvm1.hedviginc.com cvm2.hedviginc.com ]

Active resources:

ClusterIP2     (ocf::heartbeat:IPaddr2):     Started cvm1.hedviginc.com
ClusterIP3     (ocf::heartbeat:IPaddr2):     Started cvm1.hedviginc.com
hedvigfsc      (systemd:hedvigfsc):          Started cvm1.hedviginc.com
tgt            (systemd:tgt):                Started cvm1.hedviginc.com
chkhealth      (lsb:chkhealth):              Started cvm1.hedviginc.com
```

This output shows that the storage proxy HA pair is comprised of two storage proxies with hostnames of `cvm1.hedviginc.com` and `cvm2.hedviginc.com`. It also indicates that both storage proxies are in `online` status, and that both the `hedvigfsc` (that is, NFS) and `tgt` (that is, block) services are running.

If one of the storage proxies was down, or if one of the services was down, then the output would reflect the outage(s).

Using the Linux service command

You can use the Linus `service` command to determine the health status of the two main storage proxy data moving services: `hedvigfsc` (NFS) and `tgt` (block).

From the storage proxy's Hedvig CLI, run:

```
service <service name> status
```

For example:

```
[root@CVM1]# service tgt status

Redirecting to /bin/systemctl status tgt.service

● tgt.service - Hedvig ISCSI

Loaded: loaded (/etc/systemd/system/tgt.service; enabled; vendor
preset: disabled)

Active: failed (Result: signal) since Tue 2019-04-09 16:10:58 PDT; 4min
21s ago

Process: 5555 ExecStopPost=/usr/local/hedvig/scripts/hedviglogarchive
htgt restart 2>&1 (code=exited, status=0/SUCCESS)

Main PID: 5138 (code=killed, signal=KILL)
```

Checking for Hedvig service crashes (/var/crash)

To check for Hedvig service crashes, from the storage proxy's Hedvig CLI, run:

```
ls -altr /var/crash
```

Example output:

```
[root@cvm1.hedviginc.com~]# ls -altr /var/crash

drwxr-xr-x. 22 root root      4096 Apr 12  2018 ..

-rw-------   1 root root 196276224 Apr  4 06:44

core.HedvigFSServerC.18702.cvm1.hedviginc.com.1554385465
```

Using /var/log files

You can use the `/var/log/` messages file to find various storage proxy — or underlying environment-related — problems:

From the storage proxy, run:

```
more /var/log/messages | grep -i failed
```

This example output indicates a service outage:

```
Apr 7 18:55:46 cvm1 systemd: hedvigfsc.service failed.
Apr 7 18:55:52 cvm1  systemd: Unit tgt.service entered failed state.
```

From the storage proxy, run:

```
more /var/log/messages | grep -i error
```

This example output indicates a heartbeat failure on the HA pair partner storage proxy:

```
Apr 8 15:52:38 cvm1 corosync[1333]: [TOTEM ] A processor failed,
forming new configuration.
Apr 8 22:46:30 cvm1 crmd[1653]: warning: Action 11
(chkhealth_monitor_0) on cvm2.hedviginc.com failed (target: 7 vs. rc:
0): Error
```

Parsing storage proxy service logs

You can parse the storage proxy service (`hnfs, htgt`) logs to find heath status and problems:

Here are the relevant storage proxy logs:

```
/var/log/hedvig/logs/hnfs/hnfs.ERROR
/var/log/hedvig/logs/hnfs/hnfs.WARNING for NFS vdisks
```

```
/var/log/hedvig/logs/htgt/htgt.ERROR
/var/log/hedvig/logs/hnfs/htgt.WARNING for Block vdisks
```

Here are some common parsing strings:

`more htgt.WARNING | grep -i error`

example output:

```
W0409 10:21:14.233551 25674 TgtdHandler.cpp:1487] WriteError
(retry[0]):offset:100139008length:262144:vdiskInfo: VDiskInfo vdisk1,
size: 21474836480, rf: 3, gen: 14, ver: 1, blocksize: 4096, mnt:
cvm1.hedviginc.com:50000
```

`more htgt.WARNING | grep -i exception`

example output:

```
W0409 10:21:46.591941 25700 HBlockProxy.cpp:149] HBlockRead Exception
for vdisk1: 305600 readTs: 60137391030 host: node2.hedviginc.com -
EAGAIN (timed out) timeout set (us): 5000000 elapsed time (us): 9999606
```

Hedvig Storage Proxy metrics

Relevant metrics for Hedvig Storage Proxies are recorded for each Hedvig Virtual Disk, in the following file:

NFS virtual disks: `/var/log/hedvig/logs/hnfs/hnfs.metrics`

Block virtual disks: `/var/log/hedvig/logs/htgt/htgt.metrics`

All of the entries in these files are described in the *Table 1: Hedvig Storage Proxy metrics.*

Note: There are separate entries for the NFS master disk and for each of the child virtual disks. Metrics for the NFS master disk and metrics for child virtual disks are completely independent and have no correlation with each another.

To find the proper NFS child virtual disk, log into the Hedvig Storage Cluster Node's Hedvig CLI, and run the following commands:

`lsvdisks` [lists the NFS master virtual disks]

`lschildvdisks -n <name of master vdisk>` [lists the child virtual disks associated with the master virtual disk]

For example (this has been parsed for ease of viewing):

```
cluster> lsvdisks

NFSvdisk1

cluster> lschildvdisks -n NFSvdisk1

Inode            Filename
-----------------------------------
535              client-2-flat.vmdk
845              win3-flat.vmdk
851              win3_1-flat.vmdk
860              win3_2-flat.vmdk
92               win1-flat.vmdk
```

Table 1: Hedvig Storage Proxy metrics

entry	units	description
writeLatency	milliseconds	Complete turnaround time for write I/O
writeOpSize	kb	Write I/O size
concWrites	simple count	Number of threads performing concurrent writes
readLatency	milliseconds	Complete turnaround time for read I/O
readOpSize	kb	I/O size for read operation
readPages	milliseconds	Time to read metadata for a given read I/O
readBlocks	milliseconds	Time to read actual data/blocks for a given read I/O
concReads	simple count	Number of threads performing concurrent reads
metaCacheLookupHit	microseconds	Time to fetch metadata from Hedvig Storage Proxy cache, when lookup was successful
metaCacheLookupMiss	microseconds	Time for an unsuccessful lookup of metadata in Hedvig Storage Proxy cache
Note: The following entries, which correspond to the Hedvig Storage Proxy cache (also known as the block cache) are computed globally, and not for each Hedvig Virtual Disk.		
/flache/lookupHit	microseconds	Time for a successful lookup of blocks in Hedvig Storage Proxy cache
/flache/lookupMiss	microseconds	Time for an unsuccessful lookup of blocks in Hedvig Storage Proxy cache

Interpreting Hedvig Storage Proxy metrics

To interpret the Hedvig Storage Proxy metrics in the `hnfs.metrics` file, use the following equations.

Read throughput

Select any Hedvig Virtual Disk, and look for the latest average (`avg`) values for all of the entries mentioned below, from the `hnfs.metrics` file, and compute read throughput as:

```
Read throughput = ( readOpSize * concReads ) / readLatency
```

Write throughput

Select any Hedvig Virtual Disk, and look for the latest average (`avg`) values for all of the entries mentioned below, from the `hnfs.metrics` file, and compute write throughput as:

```
Write throughput = ( writeOpSize * concWrites ) / writeLatency
```

Meta cache hit ratio

Look for the latest values of `nupdates` for `metaCacheLookupMiss` and `metaCacheLookupHit`.

```
Meta cache hit ratio =
  ( metaCacheLookupHit / metaCacheLookupHit + metaCacheLookupMiss ) * 100
```

Block cache hit ratio

Look for the latest values of `nupdates` for `/flache/lookupHit` and `/flache/lookupMiss`.

```
Block cache hit ratio = ( lookupHit / lookupHit + lookupMiss ) * 100
```

Example of Hedvig Storage Proxy metrics

Here are some sample metrics for a Hedvig Virtual Disk named `mydisk1` in an `hnfs.metrics` file.

```
/vdisk/mydisk1/readOpSize/avg                 4 kb
/vdisk/mydisk1/readLatency/avg                3 ms
/vdisk/mydisk1/concReads/avg                  64 num
/vdisk/mydisk1/writeLatency/avg               25 ms
/vdisk/mydisk1/writeOpSize/avg                512 kb
/vdisk/mydisk1/concWrites/avg                 8 num
/vdisk/mydisk1/metaCacheLookupHit/nupdates    5000
/vdisk/mydisk1/metaCacheLookupMiss/nupdates   100
/flache/lookupHit/nupdates                    10000
/flache/lookupMiss/nupdates                   1000
```

Read throughput

The calculation for read throughput would be:

```
( 4 * 1024 * 64 ) / ( 3/1000 ) ~ 83 MB / second
```

Write throughput

The calculation for write throughput would be:

```
( 512 * 1024 * 8 ) / ( 25/1000) ~ 160 MB / second
```

Meta cache hit ratio

The calculation for the meta cache hit ratio would be:

```
( 5000 / 5100 ) * 100 ~ 98 %
```

Block cache hit ratio

The calculation for block cache hit ratio would be:

```
( 10000 / 11000 ) * 100 ~ 90 %
```

Changing Hedvig Storage Proxy log file settings

To properly diagnose problems on a storage proxy, you may sometimes need to change the logging level.

By default, the logging verbosity level on a storage proxy is set to `-1` (as in the example).

In most cases when the logging level needs to be increased, the new verbosity level will be set to `1`.

Important: Changing the logging verbosity level should be done only under the guidance of the Hedvig Customer Support staff.

Example

Here is an example of the `<logging>` section of the storage proxy `config.xml` file, located in `/var/log/hedvig/`:

```
<logging>
     <level>INFO</level>
     <verbosity>-1</verbosity>
     <directory>/var/log/hedvig/logs</directory>
     <prefix>0</prefix>
     <stderr>0</stderr>
     <rotation_size>1000</rotation_size>
</logging>
```

Hedvig Storage Cluster Node health status and problem diagnosis (Hedvig CLI)

You can use the Hedvig CLI and log files to determine Hedvig Storage Cluster Node health status, gather statistics, and diagnose problems.

Each storage node uses three Hedvig-specific node processes: HPod, HBlock, and Pages. Each node process has its own set of log files that can be parsed to determine health status and to diagnose problems. [For definitions of HBlock, Pages, etc., see the *Glossary*.]

Using the Linux service command

To determine the status of any node process, use the Linux `service` command, for example:

```
service hedvighpod status
service hedvigpages status
service hedvighblock status
```

In this example, the `hedvighblock` service is down:

```
[root@node1]# service hedvighblock status

Redirecting to /bin/systemctl status hedvighblock.service

● hedvighblock.service - Hedvig Hblock

Loaded: loaded (/etc/systemd/system/hedvighblock.service; enabled;
vendor preset: disabled)

Active: failed (Result: signal) since Tue 2019-04-09 17:53:42 PDT; 3s
ago

Process: 27386 ExecStopPost=/usr/local/hedvig/scripts/hedviglogarchive
hblock restart 2>&1 (code=exited, status=0/SUCCESS)

Process: 23025 ExecStart=/bin/bash -c /usr/local/hedvig/hblock/start-
server.sh > /usr/local/hedvig/hblock/nohup.out 2>&1 & (code=exited,
status=0/SUCCESS)

Main PID: 23028 (code=killed, signal=KILL)
```

If a service does not start up, or if a service does not start up properly, then refer to the startup files located here for the Hedvig HBlock and Hedvig Pages processes:

```
/tmp/pages-<hostname>.out and /tmp/hblock-<hostname>.out
```

Using the nohup.out file

An important file for the node processes is the `nohup.out` file, which is generated on service start up. These files can be found here:

```
/usr/local/hedvig/hpod/nohup-hpod.out
/usr/local/hedvig/pages/nohup.out
/usr/local/hedvig/hblock/nohup.out
```

Parsing storage node log files

Each of the node processes has log files that contain useful health status and diagnosis-related data that can be easily parsed, much like the storage proxy logs described in *Parsing storage proxy service logs.*

For the HPod process, the logs are located here:

```
/var/log/hedvig/logs/hpod
```

The main log file to parse for diagnosis information is the `system.log`. The most commonly used parsing strings are `error`, `exception`, and `null`.

Example:

```
more system.log | grep -i exception
```

For the Hedvig HBlock and Pages processes, the logs are located here:

```
/var/log/hedvig/logs/pages
/var/log/hedvig/logs/hblock
```

The most commonly used parsing strings are `error`, `exception`, and `null`.

Example:

```
[root@node1]# more system.log | grep -i exception
```

Process log files are rotated every hour and then archived once every 24-hour period. To parse the logs for the previous few hours, you can add `*` to the parsing string.

Example:

```
[root@node1]# more system.log* | grep -i exception
```

Viewing commit logs

To diagnose the health status and stability of storage nodes, it is important to view the number of commit logs for the Hedvig Pages and HBlock processes.

- To view the number of commit logs for the Hedvig Pages process, follow these steps:

  a. Log into a storage node.

  b. Change to the `commitlog` directory:

  ```
  cd /mnt/d2/commitlog
  ```

  c. Run:

  ```
  ls -altr | grep -i commit | wc
  ```

  d. Example:

  ```
  [root@node1]# ls -altr | grep -i commit | wc
        2      18     148
  ```

- To view the number of commit logs for the Hedvig HBlock process, follow these steps:

  a. Log into a storage node.

  b. Change to the `commitlog` directory:

  ```
  cd /hedvig/d2/commitlog
  ```

  c. Run:

  ```
  ls -altr | grep -i commit | wc
  ```

  d. Example:

  ```
  [root@node1]# ls -altr | grep -i commit | wc
        2      18     148
  ```

Monitoring rereplication

As part of normal storage node operation, a background process known as *rereplication* will run peridocially.

However, under some conditions this process can become impacted or stalled, and you must monitor rereplication progress and status.

To monitor a rereplication process, follow these steps:

1. Find the rereplication processes in progress, by running the following Hedvig CLI command using the `-d (date)` parameter for the date of the rereplication process(es) that you wish to review.

   ```
   showallrereplicationids -d <date>
   ```

   Example:

   ```
   cluster> showallrereplicationids -d 20190410
   ```

   Example output:

   ```
   | 10-4-2019 08:48:59:483| 905b1c131981ac05e1cfe9a709200c62|
   node1.hedviginc.com:7000| 8D860250-2491-58BA-66BA-3DA0F629A7F0|
   ```

2. To follow progress, use the `rereplicationstats` command with the `-d (date)`, `-s (storage ID)`, and `-r (rereplication ID)` options.

   You can derive these entries from the `showallrereplicationids` command output.

   Example:

   ```
   cluster> rereplicationstats -d 20190410 -s 905b1c131981ac05e1cfe9a709200c62
               -r 8D860250-2491-58BA-66BA-3DA0F629A7F0
   ```

   Example output:

```
Total # of vDisks that participated as part of 8D860250-2491-58BA-66BA-3DA0F629A7F0: is 3

Name:3118
Total # of Containers for Vdisk:1
Elapsed Time:00:00:00
    _____________________________________________________________________________________________
   |Container Idx|# of Chunks|# of Processed Chunks|# of failed Blks|# of Repaired failed Blks |
   |=============================================================================================|
1. | 1           | 1         | 0                   | 0              | 0                        |

Total Number of Chunks:1
Total Number of Processed Chunks:0
Total Number of Failed Blocks:0
Total Number of Repaired Blocks:0


Name:1016
Total # of Containers for Vdisk:2
Elapsed Time:00:00:00

    _____________________________________________________________________________________________
   |Container Idx|# of Chunks|# of Processed Chunks|# of failed Blks|# of Repaired failed Blks |
   |=============================================================================================|
1. | 1           | 4         | 0                   | 0              | 0                        |
2. | 2           | 23        | 0                   | 0              | 0                        |

Total Number of Chunks:27
Total Number of Processed Chunks:0
Total Number of Failed Blocks:0
Total Number of Repaired Blocks:0
```

Hedvig Storage Cluster Node metrics

Relevant metrics for Hedvig Storage Cluster Nodes are recorded in the HBlock and Pages metrics files.

Note: For definitions of HBlock, Pages, etc., see the *Glossary.*

HBlock metrics file

```
/var/log/hedvig/logs/hblock.metrics
```

Pages metrics file

```
/var/log/hedvig/logs/pages.metrics
```

HBlock and Pages metrics file explanation

All of the entries in the HBlock and Pages metrics files are described in the following table.

Table 2: Hedvig Storage Cluster Node metrics

entry	units	description
HBlock metrics, which are computed for each Hedvig Virtual Disk:		
WRITE-LATENCY	milliseconds	Time to write blocks in hblock
READ-LATENCY	milliseconds	Time to read blocks in hblock
Pages metrics, which are computed globally:		
ROW-WRITE-LATENCY	milliseconds	Time to write metadata
ROW-READ-LATENCY	milliseconds	Time to fetch metadata

Data collection from your production environment

If you have any problems, back up the following files, and send them to Hedvig.

Hedvig Storage Proxy logs

1. /var/log/hedvig/logs/hnfs/*
2. /var/log/hedvig/logs/htgt/*
3. /var/log/messages

Hedvig Storage Cluster Node logs

1. /var/log/hedvig/logs/pages/*.log
2. /var/log/hedvig/logs/hblock/*.log
3. /var/log/hedvig/logs/hpod/*.log
4. /var/log/hedvig/logs/pages.metrics*
5. /var/log/hedvig/logs/hblock.metrics*
6. /usr/local/hedvig/pages/nohup.out
7. /usr/local/hedvig/hblock/nohup.out
8. /usr/local/hedvig/pages/gc.log
9. /usr/local/hedvig/hblock/gc.log

Data collection using the vm-support command

If you have any problems, send in the `vm-support` log bundle with all support cases.

Simply follow the instructions in this Knowledge Base article from VMware:

Collecting diagnostic information using the vm-support command in VMware ESX/ESXi

SSD specifications and recommendations

When you are troubleshooting performance in general, it is recommended that you perform a sanity check around SSDs/Flash.

For example, here are some recommendations and specifications for two SSDs:

- This SSD is recommended (write latency 0.55ms):

  http://www.samsung.com/global/business/semiconductor/minisite/SSD/downloads/document/Samsung_SSD_845DC_PRO_Brochure.pdf

- This SSD is not recommended (write latency 3.0ms):

  http://www.samsung.com/global/business/semiconductor/file/media/SM843T_Product_Overview-0.pdf

Glossary

This glossary contains definitions of terms used in this document. See also the Hedvig Storage Glossary.

Table 3: Glossary of terms

term	definition
block cache	See *Hedvig Storage Proxy cache.*
child virtual disk	A *child virtual disk*, also known as a *redo log* or a *delta link*, provides an ongoing way to save changes to a specific virtual disk, without actually altering this virtual disk, which is then known as the *parent virtual disk.* See also *parent virtual disk.*
controller, CVM	See *Hedvig Storage Proxy.*
HBlock	See *Hedvig Block Process (HBlock).*
HDD	A *hard disk drive* is the traditional spinning hard drive, which provides basic nonvolatile storage on a computer.
Hedvig Block Process (HBlock)	The *Hedvig Block Process* is responsible for storing the physical blocks.
Hedvig Metadata Process (Pages)	The *Hedvig Metadata Process* is responsible for tracking all system *metadata*.
Hedvig Storage Cluster	A *Hedvig Storage Cluster* is an elastic cluster, formed by using any type of commodity server(s).
Hedvig Storage Cluster Node	A *Hedvig Storage Cluster Node* is an individual commodity server running *Hedvig Storage Service* software.
Hedvig Storage Proxy	A *Hedvig Storage Proxy* is a lightweight software component that deploys at the application tier as a virtual machine or Docker container, or on bare metal, to provide storage access to any physical host or virtual machine in the application tier. A storage proxy provides intelligent access to the hyperscale storage nodes, directing I/O requests to the relevant backend storage nodes based on latency response times.

term	definition
Hedvig Storage Proxy cache	The *Hedvig Storage Proxy cache* is also known as the *block cache*.
Hedvig Storage Service	The *Hedvig Storage Service* is Hedvig's patented distributed systems engine. It installs on commodity x86 servers, or cloud instances, and transforms existing server and storage assets — including SSD/flash media and hard disk — into fully featured elastic storage clusters. The software deploys to an on-premise infrastructure, or to hosted or public clouds, to create a single storage cluster that is implicitly hybrid.
Pages	See *Hedvig Metadata Process (Pages).*
parent virtual disk	A *parent virtual disk* is a virtual disk for which at least one *child virtual disk* has been made. See also *child virtual disk.*
SSD	A *solid-state drive* (or *flash* or *pin-to-flash*) is a solid-state storage device that uses integrated circuit assemblies as memory to store data persistently.
virtual disk	A *virtual disk* is an abstracted logical disk volume presented to a computer or application for read/write use.
VMDK	*Virtual machine disk* is a file format that describes containers for virtual HDDs to be used in VMs.