DATA INTEGRITY ON VNX

Abstract
This white paper explains the data integrity features on EMC® VNX™.

March 2016
# Table of Contents

**Executive summary** ................................................................. 4
  Audience ..................................................................................... 4

**Introduction** ................................................................................. 5
  Protocols ...................................................................................... 5
  Ethernet Frame Check Sequence .................................................. 5
  Internet Protocol .......................................................................... 6
  Transmission Control and User Datagram Protocols ...................... 7
  Fibre Channel Protocol ................................................................ 7

**Services** ..................................................................................... 8
  Common Anti-Virus Agent ........................................................... 8
  File-Level Retention .................................................................... 8
  MirrorView .................................................................................. 9
  VNX Replicator ........................................................................... 10

**File System Check** .................................................................... 11
  Mirrored SP Write Cache ............................................................ 11
  The VAULT .................................................................................. 13
  RAID Protection .......................................................................... 13
  Parity Shedding .......................................................................... 14
  Write Journaling .......................................................................... 16
  Non-Volatile Random-Access Memory ......................................... 17

**520-Byte Sectoring** .................................................................... 17
  Shed Stamp .................................................................................. 18
  Checksum .................................................................................... 19
  Write Stamp ................................................................................ 20
  Time Stamp ................................................................................ 21
  SNiiFFER ..................................................................................... 23
  Background Verify ....................................................................... 24
  Uncorrectable Errors ................................................................... 24

**Conclusion** ................................................................................ 25

**References** ................................................................................ 26
  Websites ....................................................................................... 26
  White Papers ............................................................................... 26
Executive summary
This white paper explains the methods used by EMC® VNX™ to ensure the end-to-end integrity of data.

Audience
This white paper is intended for users interested in the technology used to ensure the safety and validity of their data.
**Introduction**

Data integrity refers to maintaining accuracy, completeness, and consistency of information throughout its entire life cycle. If the data is not valid, any analysis of that data will also be invalid. Data is at risk for corruption at any time, while in transit or at rest. EMC VNX strives to provide end-to-end data integrity on a hardware and software level, which is crucial to prevent catastrophic events that can lead to extended periods of data unavailability or even data loss.

This paper provides information on how to maintain the integrity of data as it travels from the source through VNX. This paper also provides information on how to verify data integrity on previously stored data to ensure it is still valid before it is read.

**Protocols**

Many standard protocols that are used in a data center already have built-in data integrity functionality. This most commonly comes in the form of a checksum, which is a fixed-size hash that is calculated against a dataset. Any changes in the dataset result in a different checksum. A Cyclic Redundancy Check (CRC) is a form of a checksum designed for error detection in data storage or transmission. The following 32-bit polynomial is used for the CRC calculation:

\[ X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1 \]

**Ethernet Frame Check Sequence**

Ethernet is a data link layer (layer 2) protocol commonly used for Local Area Networks.

Figure 1 shows the contents of an Ethernet frame, which consist of:

- 8-byte preamble
- 6-byte destination MAC address
- 6-byte source MAC address
- 2-byte frame type or length
- 46-1500 byte payload
- 4-byte Frame Check Sequence (FCS)

The FCS is a 4-byte CRC that is used to detect corrupted data within the entire frame. When the source system assembles the frame, it calculates the CRC using the destination MAC address, source MAC address, frame type or length, and the entire payload. The calculated CRC is stored at the end of the frame and is transmitted along with the frame.
When the frame arrives at the destination, the target system calculates the CRC and compares it with the FCS. If the calculated CRC does not match the FCS, this indicates that the data was either altered or corrupted during transit, and the entire frame can be dropped. It is the responsibility of the upper layer protocols to handle dropped frames.

Internet Protocol

The Internet Protocol (IP) is a network layer (layer 3) protocol used for routing packets across networks. IPv4 packets consist of a header and the payload.

Figure 2 shows the contents of an IP packet header, which consist of:

- 4-bit IP version
- 4-bit header length
- 8-bit service type
- 16-bit total length
- 16-bit identification tag
- 3-bit fragment flags
- 13-bit fragment offset
- 8-bit time to live (TTL)
- 8-bit protocol
- 16-bit header checksum
- 32-bit source IP address
- 32-bit destination address
- Variable options and padding

The header checksum is calculated against the entire header. The payload is not considered in the calculation, because the data that follows generally has its own checksum. Another reason for this calculation is that some of the fields in the header change as they pass through each
router whereas the data never changes. This decreases the processing time, because only the parts that may change are recalculated instead of the entire packet.

The checksum is calculated by dividing the header into 16-bit words and summing them with a one's complement calculation. A one's complement calculation is obtained by inverting the bits of a binary number (changing 1s to 0s and vice-versa). The calculated sum is recalculated with one's complement and inserted into the checksum field. The checksum is verified at each router, and the entire packet is discarded silently if the checksum is invalid. The router can also modify fields in the header, such as the TTL, recalculate the checksum, and then save the new checksum.

IPv6 has a completely revamped packet structure, and the checksum has been eliminated altogether. This was because the checksum is computed on the Ethernet layer, and that was deemed to be sufficient. This improves performance, because routers no longer need to compute and validate the checksum for each packet.

**Transmission Control and User Datagram Protocols**

The Transmission Control Protocol (TCP) is a transport layer (layer 4) protocol designed to provide reliable and in-order data transfer between computers. The User Datagram Protocol (UDP) is designed for time-sensitive applications where error correction is not required. TCP emphasizes accuracy over timely delivery while UDP does the opposite. CIFS runs on TCP port 445, NFS runs on TCP/UDP port 2049, and iSCSI runs on TCP ports 860/3260.

TCP and UDP include a 6-bit checksum calculated against the pseudo IP header, TCP/UDP header, and data. This is required for TCP but optional for UDP. The checksum is calculated in the same way as the IP checksum, but the 16-bit words are first padded with zeros, if necessary. If the calculated checksum is zero, it will be sent as the one's complement of zero, which is binary 1111111111111111 (65535), because zero is reserved for when there is no checksum. In IPv6, the checksum uses the 128-bit IPv6 addresses instead of the 32-bit addresses used in IPv4.

TCP also features guaranteed delivery, ordering, and duplicate detection. TCP assigns an incrementing sequence number to each byte of sent data, resembling the identification tag in IP. The receiver sends an acknowledgement to the sender with the last sequence number it receives plus one. This indicates the sequence number that is expected on the next transfer. If any other sequence number is sent, the receiver requests a retransmission. This also allows the receiver to reassemble data that is received out of order or detect when duplicate data is received.

**Fibre Channel Protocol**

The Fibre Channel protocol transmits data in 2148-byte frames. Of those 2148 bytes, 2112 are reserved for the data and 36 are overhead.
Figure 3 shows the contents of a Fibre Channel frame, which consist of:

- 4-byte start of frame
- 24-byte frame header
- 2112-byte payload
- 4-byte CRC
- 4-byte end of frame

The CRC will always be placed between the data and the end of the frame marker. This value is calculated using the entire header and payload but does not include the start and end of the frames.

The recipient reads the frame, bit by bit, and calculates the CRC. Once the recipient reaches the end of the frame, it confirms that the calculated CRC matches the one found in the frame. If the CRC does not match, the frame is assumed to be corrupt and can be dropped.

**Services**

**Common Anti-Virus Agent**

Data integrity can also be compromised by malware, such as viruses, worms, or other malicious binaries. The Common Anti-Virus Agent (CAVA) automatically scans files to identify and eliminate these threats. CAVA runs in conjunction with a Windows server that hosts a third-party antivirus (AV) engine, such as Computer Associates, Kaspersky, McAfee, Sophos, Symantec, or Trend Micro, along with the CAVA agent. The agent communicates with the CIFS server on VNX to scan and clean, quarantine, or delete infected files. CAVA offers several features, such as load balancing and fault tolerance across multiple CAVA servers, scan on first read, scan after virus definition update, and scan on write.

For more information on CAVA, refer to the *Using VNX Event Enabler* document available on [EMC Online Support](https://www.emc.com).

**File-Level Retention**

File-Level Retention (FLR) can be used to protect files from modification or deletion until a specified retention date. Standard disks on the VNX can be used to create a permanent, unalterable set of files and directories. File-Level Retention–Compliance (FLR-C) protects data from changes made by users through NAS protocols, such as CIFS, NFS, and FTP, and backup
restores. FLR-C also protects against administrator actions and meets the requirements of the
SEC Rule 17a-4(f). You cannot delete an FLR-C file system that has files in the locked state.

One requirement of the SEC rule 17a-4(f) found in SEC Ruling and Requirements is to
automatically verify the quality and accuracy of the storage media recording process. To
address this, an FLR-C file system provides block-level checksums and bit-level verification,
also known as disk scrubbing.

- Block-level checksums are calculated at the time the data is recorded and maintained
  by the disk storage system. These checksums are verified by the disk storage system
  when the data is read back from the disk to ensure that it has not been altered since it
  was written.

- Periodically, a bit-level verification of the physical storage media and block-level
  checksums is performed to ensure that there are no hard failures at the media level.

Although this feature prevents destructive actions against protected content at the file system
level for FLR-C file systems, it does not protect against committed content at the disk level. An
authorized administrator can destroy the LUNs on which the file system resides by using
management tools, such as Unisphere and NaviSecCLI. However, the intent of SEC Rule 17a-4(f)
requirement is to prevent modification of individual files without trace or record. VNX FLR-C
meets this requirement, because an administrator cannot target individual files for deletion.
For more information on FLR, refer to the *EMC VNX File-Level Retention White Paper* available on
EMC Online Support.

**MirrorView**

It is also important to protect data that is being replicated for purposes such as disaster
recovery (DR). This ensures the data on the target is consistent with the source if a failover is
required.

MirrorView™ is a block-based storage-system disaster recovery solution for mirroring
local production data to a remote disaster recovery site. MirrorView provides end-to-
end data protection by replicating the contents of a primary volume to a secondary
volume that resides on a different VNX. MirrorView can be deployed in either
synchronous mode (MirrorView/S) or asynchronous mode (MirrorView/A).

On MirrorView/S, data integrity is maintained through the use of write-intent and fracture logs.
The fracture log protects primarily against loss of communication with the secondary images.
The write intent log protects primarily against interruptions to the primary image. The write
intent log is enabled by default but is optional. Both of these structures exist to enable partial
synchronizations in the event of interruptions to the primary and secondary images.
Fracture Log

The fracture log is a bitmap held in the memory of the storage processor that owns the primary image. It indicates which physical areas of the primary image have been updated since communication was interrupted with the secondary image.

The fracture log tracks changes on the primary image for as long as the secondary image is unreachable. It is a bitmap that represents areas of the primary image with regions called extents. The amount of data represented by an extent depends on the size of the mirror images.

When the secondary LUN returns to service, the secondary image must be synchronized with the primary image. This is accomplished by reading those areas of the primary image addressed by the fracture log and writing them to the secondary image. This activity occurs in parallel with any writes coming into the primary image and mirrored to the secondary image. Bits in the fracture log are cleared once the area of the primary image marked by an extent is copied to the secondary image. This ability to perform a partial synchronization can result in significant time savings.

Write-Intent Log

During normal operation, the write-intent log tracks in-flight writes to the primary and secondary images in a mirror relationship. Similar to the fracture log, the write-intent log is a bitmap composed of extents indicating where data is written. However, the write-intent log is always active, and the fracture log is enabled only when the mirror is fractured.

When in use, MirrorView makes an entry in the write-intent log of its intent to update the primary and secondary images at a particular location, and then proceeds with the attempted update. After both images respond that data has been written (governed by normal LUN access mechanisms, for example, written to write cache), MirrorView clears previous write-intent log entries. For performance reasons, the write-intent log is not cleared immediately following the acknowledgement from the primary and secondary images; it will be cleared while subsequent write-intent log operations are performed.

During recoveries, the write-intent log can be used to determine which extents must be synchronized from the primary storage system to the secondary system. For instance, if a single SP becomes unavailable (for example during a reboot or failure), there may be in-flight writes that were sent to the secondary image, but not acknowledged before the outage. These writes will remain marked in the write intent log.

For more information on MirrorView, refer to the MirrorView Knowledgebook: Releases 30-32 available on EMC Online Support.

VNX Replicator

VNX Replicator is an IP-based asynchronous tool that can replicate file systems and Virtual Data Movers (VDMs), as well as making one-time copies of file systems. Replicator includes a CRC on
all data sent over the Data Mover interconnect to ensure integrity and consistency. This is automatically enabled and cannot be disabled.

For more information on VNX Replicator, refer to the *Using VNX Replicator* document available on [EMC Online Support](https://www.emc.com).

**File System Check**

File System Check (FSCK) is a tool used to check the consistency of a file system similar to chkdsk (check disk) on Windows. A FSCK is run automatically when the array detects a file system in an inconsistent state.

**Note:** The FSCK takes the file system offline for the duration of the scan. The tool will attempt to repair damaged files, if possible, or move the inodes to the lost+found directory to prevent them from being read. The output and results of the FSCK are saved in the server_log.

A FSCK can also be initiated manually if corruption is suspected. If corruption is suspected on a production file system, EMC recommends a FSCK can be run on a writeable checkpoint to confirm the status while the production file system remains online. If corruption is found on the writeable checkpoint, the production file system needs to be taken offline to be repaired. Any checkpoints of the corrupted file system should be deleted, because they may also contain corruption.

**Mirrored SP Write Cache**

The SP write cache on a VNX is mirrored between the two SPs. Any incoming write request to an SP will be copied to the other SP through the Common Management Interface (CMI). The host will receive confirmation that the write has completed once the data has been successfully stored on both SPs. This allows the write cache data to be preserved through hardware and/or software faults and SP reboots. The data will be flushed from cache to disk later. Figure 4 shows the VNX SP cache architecture.
The only way the cached data can be lost is if both SPs fail at exactly the same time. Cached data is protected as long as the surviving SP has enough time to detect the failure on the other SP. In the event of an emergency, such as a power outage or temperature alarm on both SPs, the Standby Power Supply (SPS) will kick in, and the cache will be de-staged to the VAULT drives. The SPS is designed to keep the array up long enough for this de-staging to complete. This operation can take anywhere from several seconds to several minutes, depending on the model, cache capacity, and type of VAULT drives. Once power is restored, any writes that were de-staged are reconciled and persisted to the target backend disks to ensure no data is lost.

VNX features Greater Write Cache Availability (GWCA), where the write cache remains enabled through a wide range of failures, including a single SP failure, single VAULT drive failure, and single SPS or power supply failure. In cases of a single SP failure, the cache is no longer be mirrored across the SPs. To reduce the risks of losing write cache data if an SP reboots, the software saves the write cache data at the time of the reboot for retrieval on the next boot. Write cache is disabled only in circumstances where there is a power loss, when only a single fan remains, or both SPs become overheated.
The VAULT

The VNX VAULT is located on the first four drives on bus 0, enclosure 0. They are disks 0 through 3 on either the DPE for VNX5100/5300/5500 or the first DAE in VNX5700/7500. These disks are protected with proprietary EMC parity protection and hold the reserved area for write cache de-staging. In the event of a cache dump, this can hold the contents of cache indefinitely without data loss. In addition, these disks are used for system information, such as storing the operating system, Persistent Storage Manager (PSM), File Control LUNs, and operating environment database.

RAID Protection

RAID (Redundant Array of Independent Disks) is a technology that allows multiple disks to work together as a single logical unit for performance and availability. RAID uses disk striping, mirroring, parity calculations, or a combination of the three to save data across multiple drives.

VNX supports RAID 0, 1, 1/0, 3, 5, and 6.

- **RAID 0**—Striped. RAID 0 has no redundancy but provides additional performance. Data blocks are written to multiple drives in parallel which increases bandwidth. However, a single drive failure will destroy the group. This RAID level is not recommended by itself but can be paired with RAID 1.

- **RAID 1**—Mirrored. Data is mirrored across two drives, creating a duplicate of the data. RAID 1 only provides the usable capacity of a single drive, because the second drive is designated as the copy. If either drive in the pair fails, no data is lost, because it is still available on the replica.

- **RAID 1/0**—Mirrored and Striped. Disks are grouped into mirrored pairs, and then the pairs are striped together. This combines the benefits of RAID 0 and RAID 1 for performance and availability. Due to the mirrors, only half the capacity of the drives will be usable, but RAID 10 can sustain multiple drive losses as long as they are not part of the same mirror.

- **RAID 3**—Striping with Dedicated Parity. Data is striped across multiple drives, and a single drive is dedicated for storing parity data. The parity data is calculated with a XOR operation on the user data. If a disk fails, the missing user data can be rebuilt from the parity, or the parity can be recalculated if that is lost. The parity drive contains no usable capacity.

- **RAID 5**—Striping with Distributed Parity. Data and parity are striped across multiple drives. RAID 5 can survive a single drive failure due to the parity data. The parity drive contains no usable capacity.

- **RAID 6**—Striping with Dual Distributed Parity. RAID 6 is similar to RAID 5 but with a second drive dedicated for parity. This allows RAID 6 to survive two disk failures. The two parity drives contain no usable capacity.
Using RAID technology (other than RAID 0) allows data integrity to be preserved through disk failures. Media errors, such as bad sectors, can be corrected by RAID instead of resulting in data loss.

**Parity Shedding**

When a disk has failed but has not yet been replaced, power interruptions can cause data corruption on parity protected disks. Figure 5 shows a disk failure scenario.

1. This is a RAID 5 (4+1) stripe in a consistent state. There are four data sectors and a parity sector with a value of 12.
   
   **Note:** Parity calculations are actually done with XOR, but we are using a sum here for simplicity.

2. Disk 2 fails.

3. An incoming write request changes the 3 to a 4.

4. This change is committed, but a power outage occurs before the parity block could be updated. The parity still has a value of 12, even though it should be 13.

5. Power is restored and a read is requested from the failed disk. The data is calculated using the incorrect parity, and a 0 is returned instead of a 1.

---

**Figure 5. Disk Failure Scenario**
VNX1 systems offer enhanced RAID parity protection with parity shedding to protect against these events. Figure 6 shows the same scenario with parity shedding.

1. This is a RAID 5 (4+1) stripe in a consistent state. There are four data sectors and a parity sector with a value of 12. Note, parity calculations are actually done with XOR but we are using a sum here for simplicity.
2. Disk 2 fails.
3. An incoming write request changes the 3 to a 4.
4. Before the write is executed, the data on the failed drive is recalculated and written over the parity.
5. The write request is committed, and the stripe is again in a consistent state. Any read request to the failed drive is now redirected to disk 4. Because there is no parity now, there is no point where a power outage can lead to data/parity incoherence.
6. When the failed disk is replaced, the parity is recalculated, and the stripe returns to standard RAID 5.
Write Journaling

VNX2 systems do not use parity shedding. Parity shedding worked well when the drive could not splice a sector (i.e. the enclosures were all on battery). However, the shed of parity was susceptible to spliced writes during the parity shed. Instead of parity shedding, VNX2 systems leverage write journaling instead. Just like parity shedding, write journaling is designed to solve the RAID Write Hole problem where a spliced write on a power failure to a drive in a degraded RG could cause data loss. Parity shedding is no longer used because write journaling offers a better overall solution.

Write journaling is a Multicore RAID technology which writes a journal entry onto the same drive before proceeding with an actual write, as shown in Figure 7. If the journal write is committed without issues, the Multicore RAID system proceeds to commit the write to its intended address.
on the drive. After the write is successfully committed to the drive, the journal entry is invalidated and the write is marked as clean in cache (Multicore Cache).

![Write Journaling](image)

**Figure 7. Write Journaling**

Uncommitted dirty Multicore Cache pages are protected by the vaulting mechanism. After recovering a RAID group from any fault condition (for example, powering on after an unscheduled power off, or recovering after several drives go offline, and so on), the journal area is examined for active journal entries. If any are found, they are played back to proper locations.

Write journaling is active only when a RAID group is in a degraded state, just like rebuild logging—the two technologies coexist. Write journaling writes the journal to the RAID group metadata area within the destination drive. The journal contains the actual write plus metadata. The journal space is limited and managed on the circular buffer basis. When journal space is full, incoming writes are delayed until first entries are invalidated.

Write journaling comes with some performance degradation: every disk write is now doubled. However, the performance cost is an acceptable disadvantage given the benefit.

**Non-Volatile Random-Access Memory**

VNX uses non-volatile random-access memory (NOVRAM) to track the consistency state of all the stripes on the array. If there is a power outage after the data has been written, but before the parity has been updated, it will be reflected in the NOVRAM entries for that stripe. When power is restored, the stripe states are checked and the parity will be recalculated for any stripes with inconsistent states. In a worst-case scenario where there is a power failure and the NOVRAM is lost (such as an SP failure), a complete background verify is run to ensure all stripes are consistent.

**520-Byte Sectoring**

A drive sector typically contains 512 bytes of data. On the VNX, data is stored on 520-byte sectors—512 bytes for data and 8 bytes for metadata. While this reduces the usable capacity of a disk by less than two percent, it has a large positive effect on availability. The additional 8
The eight additional bytes consist of:

- 2-byte shed/LBA stamp
- 2-byte checksum
- 2-byte write stamp
- 2-byte time stamp

**Shed/LBA Stamp**

On VNX1 systems, the shed stamp is a 16-bit field that is used to maintain coherency when a parity drive is missing. Shed stamps are used in conjunction with parity shedding on the parity drive on RAID-3, RAID-5, and RAID-6. Under normal circumstances, the shed stamp has a value of 0.

In RAID-3 and RAID-5, after a disk fails and parity shedding occurs, the shed stamp is changed to indicate the drive is now holding data instead of parity. The shed stamp is a 16-bit hexadecimal value used as a bit map to indicate to which disk the data actually belongs. The maximum RAID group size is 16, and the bit assignments are as follows:

- Disk 0—0x0001
- Disk 1—0x0002
- Disk 2—0x0004
- Disk 3—0x0008
- Disk 4—0x0010
- Disk 5—0x0020
- Disk 6—0x0040
- Disk 7—0x0080
- Disk 8—0x0100
- Disk 9—0x0200
- Disk 10—0x0400
- Disk 11—0x0800
- Disk 12—0x1000
- Disk 13—0x2000
- Disk 14—0x4000
- Disk 15—0x8000

On a read request, the shed stamp is examined so that the SP knows where the data belongs in the stripe. When the failed disk is replaced, the shed stamp is used to indicate that this data belongs to the failed disk. After the rebuild is complete, the parity is recalculated and stored back on the parity drive.

The shed stamp works slightly differently for RAID-6, because it does not use parity shedding. In RAID-6, the shed stamp holds the parity of checksums on the parity drives only. The checksum is
calculated with the same parity algorithm that is used in RAID. This allows the parity data to be maintained even when there are regions with bad blocks.

Figure 8 shows how shed stamps work for RAID-3 and RAID-5.

1. The starting condition is an existing LUN with a write pending to disk 0. All shed stamps are set to 0. Disk 1 fails, which puts this stripe in a degraded and valid state.
2. The shed occurs, which replaces the parity data with the data from disk 1. The shed stamp on disk 4 is updated to 0x0002 to indicate that the data belongs to disk 1.
3. A write is committed to disk 0, and no parity needs to be updated. The stripe remains in a degraded and valid state.

Since shed stamps are only used on parity drives, this 16-bit field is available for use on data drives. On both VNX1 and VNX2 systems, this field is used to hold the LBA stamp for RAID-3, RAID-5, and RAID-6 implementations. For any write to a data block, the LBA of that block is written to this field prior to the block being written to disk. The purpose of this is to detect if a data block has been written to the wrong location on the disk. The software can compare the actual location (LBA) of a block with the LBA value that is in the LBA stamp field. If the values do not match, then the system knows that the block was written to the wrong position on the disk, overwriting the data of an unrelated block. The system can now rebuild the proper data for the block that was overwritten and also correct the block and stripe where the block should have been written (according to the LBA stamp value).
Checksum

The checksum is a 16-bit CRC calculated by running an XOR operation on all 512 bytes of user data. This is generated by the array any time data is written and stored within the sector. The checksum is always attached to the data as it travels between the cache and disk. It is used to detect any corruption that might occur in memory, on disk, or in transit. In addition, the checksum is calculated and compared to the stored checksum on all critical operations, including host reads, writes, and rebuilds.

Write Stamp

The write stamp is used only for RAID 5 and is designed to protect against incomplete non-full stripe writes. The write stamp uses the same bit map as the shed stamp. Each drive toggles the bit to which it is assigned when a write is committed, and the parity drive holds the total for all the drives in the stripe.

Figure 9 shows how write stamps work.

1. The starting condition is an empty LUN. All the data and write stamps are set to 0.
2. Data is written to disks 1 and 2, and their respective write stamp bits are toggled.
3. The parity and write stamp are calculated and written to disk 4 (parity).

![Write Stamps Diagram]

The write stamp adds another layer of protection by detecting when data is missing from the sectors. Figure 10 shows an example of how write stamps work in a worst case failure scenario.

1. The starting condition is an empty LUN. All the data and write stamps are set to 0.
2. Data is written to disks 1 and 2, and their respective write stamp bits are toggled.
3. Power is suddenly lost before the parity drive can be updated, and disk 2 fails while the power is down. When power is restored, the data from disk 2 is requested to be read. Because disk 2 is not available, the SP must read the rest of the stripe to reconstruct the data. Before initiating the reconstruction, the SP inspects the write stamps to confirm the validity of the stripe. Because the write stamp on disk 1 and 4 (parity) are not consistent, the SP determines that the stripe is in an inconsistent state. The SP cancels the reconstruction, because this would result in corrupted data.

![Figure 10. Write Stamps in a Failure](image)

**Example (Drive 4 holds Parity; 16 bits of data shown on each drive)**

1. The starting condition is an empty LUN. All the data and time stamps are set to 0.
2. A full stripe write occurs on disks 0, 1, 2, and 3. The parity is calculated and written to disk 4. All disks have the same time stamp, and the stripe is in a consistent state.

**Time Stamp**

The time stamp is a 16-bit field used to detect incomplete full stripe writes on RAID-3 and RAID-5. The contents of the time stamp are based on an internal SP clock. This value is written to all the disks, along with the data. This allows the SP to later verify that there were no changes to the stripe after it was written. Because this field is limited to 16 bits, the maximum value of the time stamp is 65535. When the internal clock reaches this number, it will automatically restart from 0.

Figure 11 shows how time stamps work.

1. The starting condition is an empty LUN. All the data and time stamps are set to 0.
2. A full stripe write occurs on disks 0, 1, 2, and 3. The parity is calculated and written to disk 4. All disks have the same time stamp, and the stripe is in a consistent state.
Figure 12 shows how time stamps work in a worst-case failure scenario.

1. The starting condition is an empty LUN. All the data and time stamps are set to 0.
2. A full stripe write occurs. Data is written to disks 1 and 2, and the calculated parity is written to disk 4.
3. The power fails before disks 0 and 3 are updated. Disk 2 fails while the power is out. When power is restored, the data from disk 2 is requested to be read. Because disk 2 is not available, the SP must read the rest of the stripe to reconstruct the data. Before initiating the reconstruction, the SP inspects the time stamps to confirm the validity of the stripe. Because the time stamps on disks 0 and 3 are not consistent with the rest of the stripe, the SP determines that the stripe is in an inconsistent state. The SP cancels the reconstruction, because this would result in corrupted data.
Whenever time stamps are used, write stamps cannot be used, so the write stamp will be set to 0 on all drives. This is because time stamps are only used for full stripe writes, and write stamps are only used for non-full stripe writes.

**SNiiFFER**

Media problems or contamination from microscopic particles during normal operations can sometimes affect sectors on a disk. When this occurs, the controller might still be able to read the sector with its own embedded error-correcting algorithms, and a disk-recovered error is reported. The VNX automatically remaps this sector by rewriting the data to a new physical sector in case the bad sector continues to deteriorate.

If a media defect causes the sector to be completely unreadable, a medium error is reported. When this occurs, the array uses RAID technology, such as mirroring or parity, to reconstruct the data. If successful, this is known as a RAID-correctable error. If another disk has a second unreadable sector or invalid CRC on the same stripe, this results in an uncorrectable error.

Early detection of problematic sectors provides the array the best opportunity to correct them before they have a chance to become disk medium errors. Likewise, early detection of any medium errors provides the array the best opportunity to correct them before a second issue causes an uncorrectable error. This is done when the data is accessed and by a low-priority, continuously-running background process called SNiiFFER.

SNiiFFER performs a media check by asking the drive to read a series of blocks. When a new LUN is bound, SNiiFFER is started automatically on a low priority and will not impact back-end bus bandwidth. However, this means it could take several days to complete a full pass on a
single LUN. If the LUN is idle, SNiFFER automatically increases its verification rate and reduces the time to complete the scan. SNiFFER performs this check on all disks used for the LUN simultaneously, so the number of disks does not matter. If bad sectors are detected, the sector is rebuilt using the redundant information that is available from RAID protection. This reduces the possibility that a defect remains undetected and causes data loss due to a disk fault in the array.

SNiFFER is not designed to verify the state of the LUNs, because it does not perform any consistency checks of the data or parity. Instead, it provides a mechanism to check the state of the media in the background without any significant impact to host I/O performance. The benefit is that drive errors can be detected proactively before they are sent to a client.

**Background Verify**

Background verify is a data consistency check that calculates the parity and compares it to the stored parity. Background verify runs automatically after a LUN is bound, when a LUN is trespassed, and when the SP detects a difference between the stripe’s calculated parity and the stored parity. When this occurs, the SP will attempt to reconstruct the data using the stored parity.

A manual background verify can be started when the data integrity of a LUN is suspected to be compromised. The time required to run a full background verify varies depending on many factors, including I/O, LUN size, RAID type, and disk type. By default, background verify runs with a medium priority. When initiating a manual background verify, four priorities are available: low, medium, high, and ASAP (ASAP is available on VNX1 systems only). Selecting high or ASAP allows the process to complete faster but also requires more storage system resources, which may impact performance.

**Uncorrectable Errors**

When the SP detects an inconsistency between the user data and the 8 bytes of metadata, the SP attempts to correct the inconsistency by using RAID technology. The SP attempts to read the data from the mirror on RAID-1 and RAID-10 and reconstructs the data from parity on RAID-3 and RAID-5.

In rare cases, multiple drives in a RAID group with CRC mismatches can result in an uncorrectable error. This indicates that the invalid data could not be read from the mirror, nor could it be reconstructed from parity. The array has determined the contents are no longer valid and invalidates the sector to ensure it will not be sent to a host. Any attempts to read an invalidated location results in a hard error returned to the host. Any attempts to write to an invalidated location complete successfully and overwrite the void. Because of this, there are times when previously uncorrectable sectors can disappear after valid data is written over the voided data.

When an uncorrectable error is detected, it is logged in the SP logs. This error will indicate drive(s) from which the array could not successfully read. The invalidated error indicates drive(s) that the array then marked as void of valid information in a specific location.
If uncorrectable errors are detected, it is critical that these are addressed as soon as possible to prevent additional data loss. A background verify should be initiated to check for any additional uncorrectable sectors. This will locate any other uncorrectable sectors that might not have been detected yet.

**Conclusion**

Data is at risk for corruption at any time, while in transit or at rest. End-to-end data integrity on a hardware and software level is crucial to prevent catastrophic events that can lead to extended periods of data unavailability or even data loss.

In this paper, we discussed how data integrity is guaranteed as it travels from the client, through the wire, and throughout the VNX. We also looked at how data integrity is verified on previously stored data to ensure it is still valid before it is read. This is accomplished by integrating data integrity functionality into advanced services, mirroring SP cache, parity shedding, using NVRAM, 520-byte sectoring, the SNiiFFER process, and background verify. VNX features additional measures on top of protection already offered by standard protocols to further protect your data.
References

Websites

White Papers
Refer to the following white papers, available on EMC Online Support, for more information:
- MCx - Multicore Everything
- Using VNX Event Enabler
- EMC VNX File-Level Retention White Paper
- MirrorView Knowledgebook: Releases 30-32
- Using VNX Replicator