Date: July 1, 2003

To: T10 Technical Committee
From: Keith Holt (LSI Logic)
Subject: End-to-End Data Protection Justification

1. Introduction

The purpose of this document is to provide justification for the End-to-End Data Protection proposal described in document T10/03-111. Data integrity concerns are not new, and protection mechanisms abound. Packet-based storage transport protocols have CRC protection on command and data payloads. Interconnect buses have parity protection. Memory systems have parity detection/correction schemes. I/O protocol controllers at the transport/interconnect boundaries have internal data path protection.

Data availability in storage systems is frequently measured simply in terms of the reliability of the hardware components and the effects of redundant hardware. But the reliability of the software, its ability to detect errors, and its ability to correctly report or apply corrective actions to a failure have a significant bearing on the overall storage system availability. The primary objective of the proposed End-to-End Data Protection mechanism is as follows:

- Define a standardized data integrity protection mechanism that spans transport and device boundaries to protect data as it is transferred from application space to storage media, then back.

There are proprietary mechanisms already in existence that provide end-to-end data protection. The primary motivation behind defining a standardized mechanism based on SCSI command and architecture standards are as follows:

- Improve fault isolation by allowing each device in the data path to understand the protection scheme.
- Increased interoperability between standardized initiator and target devices.

The proposed data protection mechanism is intended to complement, not replace, end-to-end protection schemes defined at the application level. SCSI architecture primarily defines transport-level operations. The proposed mechanism spans transport boundaries by embedding the protection into the data stream itself. This allows target devices to receive data with standardized data integrity protection, store the data and associated data integrity fields in a proprietary fashion, then return the data and associated data integrity fields in a standardized fashion. Intermediate storage devices that bridge or connect SCSI domains can check the standardized data integrity as data is transferred between SCSI domains.

2. Proposed Data Integrity Mechanism Overview

The data integrity mechanism proposed in T10/03-111 embeds data integrity information into the data stream. An 8-byte Data Integrity Field (DIF) is interleaved with each standard block of data as illustrated below in Figure 1.

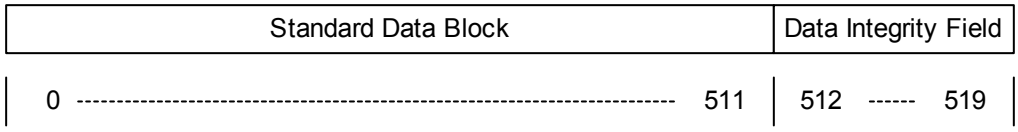

Figure 1 – Data Integrity Field Appended to 512-Byte Standard Block

As shown below in Figure 2, the Data Integrity Field is composed of three sub-fields, the Reference Tag, the Meta Tag and the Guard. The Reference Tag nominally contains information associated with a specific data block within some context, such as the lower 4 bytes of the Logical Block Address. During a multi-block data transfer, this field is incremented by one for each successive block. Since the Reference Tag is not required to be LBA based, the proposal defines additional SCSI read and write block commands that allow the Reference Tag base to be specified on a per I/O basis. The Meta Tag contains additional context information that is nominally held fixed within the context of a given I/O operation. The Guard field is computed from the data in the standard data block. The tag values in the DIF are excluded from the computation.

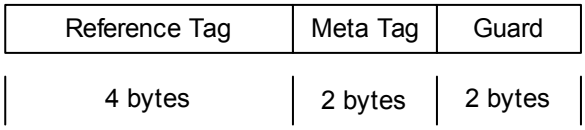

Figure 2 – Data Integrity Field Format

3. Data Integrity Error Classification

For the purposes of this analysis, there are two error classifications, defined as follows:

1. **bit error** – Bit errors are defined to include any bit or byte level error in which the majority of data within the block is correct.
2. **data displacement error** – A data displacement error is defined as a data transfer in which either the source or destination of the data is incorrect. The displacement may occur on part, or all, of a data block. A partial data block displacement may appear to be a bit error.

4. System Example

It is common in target devices to have a buffered architecture in which data is staged in an intermediate buffer after it is received from the initiator, or prior to transfer to the initiator. Figure 3 shows an illustration of a target device that is an intermediate device that appears as a SCSI target to host systems in one SCSI domain, designated "A," and a SCSI initiator to disk drives where the data is actually stored in a second domain, designated "B." This system will be used to illustrate the benefits of end-to-end data protection.

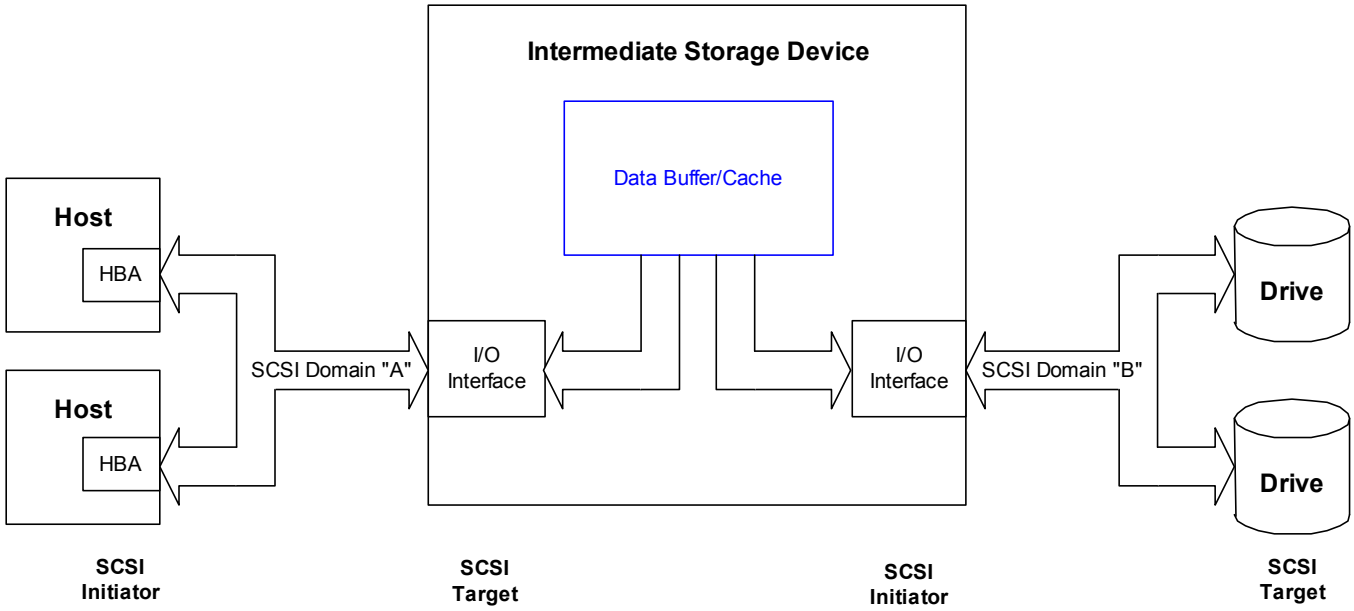

Figure 3 – Target Device Data Buffer Illustration

4.1. Address Model

4.1.1. Host to Intermediate Target Address Model

As data is received by the target from the host system initiator, the source of the data is memory in the host system. From the perspective of the intermediate storage device, the source is the initiator in SCSI domain A. The source address is specified by SCSI protocol addressing methods for that domain. Address elements include a protocol-specific SCSI initiator device address, the Logical Unit Number, and the Logical Block Address. The destination of the transfer is a data buffer in the target. From the perspective of the intermediate storage device, the destination is specified by one or more buffer memory address/length pairs, commonly referred to as a scatter/gather list.

As data is sent from the target device to the host system, the source of the data is buffer memory, specified by a scatter/gather list. The destination of the data is the initiator in the host system, specified by SCSI domain A address elements.

4.1.2. Intermediate Target to Drive Address Model

The address model for data transfers between the intermediate storage device and the disk drives is similar to that described for host to intermediate storage device transfers. The storage device now assumes the role of SCSI initiator. The data buffer is the source for disk write operations, and the

destination for disk read operations, and is addressed via a scatter/gather list. Buffer memory in the disk drive is the destination for disk write operations, and the source for disk read operations. From the perspective of the intermediate storage device, the drive is addressed as a SCSI target device using SCSI domain B address elements.

4.2. Host Write Illustration

The steps below illustrate the operations needed to write data to media using a write-through cache in an intermediate storage device. Figure 4 shows the data flow, including data structures that can reasonably be expected to be associated with this type of system. This example assumes the use of the proposed end-to-end data integrity mechanism. The DIF received from the host is written to media unchanged.

1. Storage controller receives CDB from host via its host I/O interface
    a. Op = Write
    b. LBA = 319
    c. Number of blocks = 2
2. Storage controller allocates cache block(s) in its data buffer to receive the data
    a. Cache block 452, memory address 0x1C4000
    b. Cache block 823, memory address 0x337000
3. Storage controller programs its host I/O interface to receive the data from the initiator
    a. Op = Receive
    b. Reference Tag Base = 319
    c. Scatter/gather element 1 = address 0x1C4000, length = 512
    d. Scatter/gather element 2 = address 0x337000, length = 512
4. I/O interface receives data from the initiator
5. Storage controller receives interrupt upon completion of receive operation
6. Storage controller maps the host data blocks to drive data blocks
    a. Host LBA 319 maps to drive 2, LBA 127
    b. Host LBA 320 maps to drive 3, LBA 64
7. Storage controller programs its drive I/O interface to send data to target 2
    a. CDB
        i. Op = Write
        ii. LBA = 127
        iii. Number of blocks = 1
    b. I/O interface control structure
        i. Op = Transmit
        ii. Reference Tag Base = 319
        iii. Scatter/gather element 1 = address 0x1C4000, length = 512
8. Storage controller programs its drive I/O interface to send data to target 3
    a. CDB
        i. Op = Write
        ii. LBA = 64
        iii. Number of blocks = 1
    b. I/O interface control structure
        i. Op = Transmit
        ii. Reference Tag Base = 320
        iii. Scatter/gather element 1 = address 0x337000, length = 512
9. I/O interface transmits data to the target
10. Storage controller receives interrupt upon completion of each transmit operation
11. Storage controller receives SCSI completion status for the two disk write operations
12. Storage controller sends SCSI completion status to the host

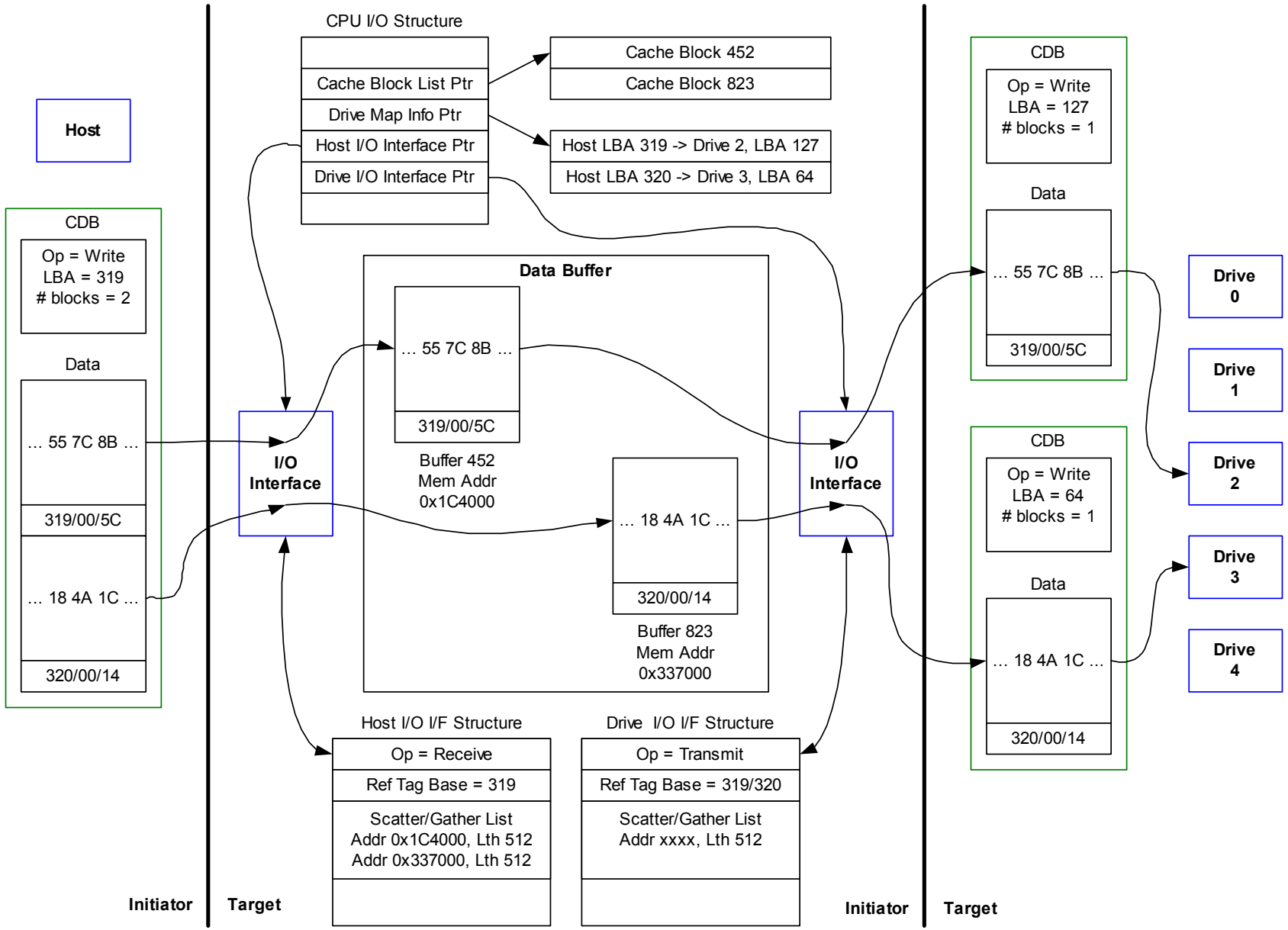

Figure 4 - Host to Drive Data Flow Illustration with Intermediate Storage Device

4.2.1. Host Write Error Examples

Examples of errors that may occur in the intermediate storage controller are as follows:

Case 1: Buffer allocation error

Scenario: A buffer allocation error occurs after data has been received from the host (the error occurs after step 4, but before step 9). Data buffer 452 is incorrectly allocated for another I/O operation for LBA 683, and the entire data block for LBA 319 is overwritten with data for LBA 683 prior to the time that the data is transmitted to the drive in step 9. The drive I/O interface structure, including the associated CDB, is unaffected by the error.

Error type: Data displacement

Detectable: Yes

Detection: Standards based error detection occurs during step 9 when the target determines that the data received for LBA 127 does not contain Reference Tag 319 as specified in the CDB. *(The I/O interface in the intermediate storage controller could also have detected the error as data was transmitted, but such detection is outside of the standard.)*

Case 2: I/O interface data path error

Scenario: A bit error occurs on an internal data path in the I/O interface chip as data is transmitted to the target in step 9. The error occurs after any proprietary DIF check by the I/O interface chip. The SCSI transport CRC is computed by the I/O interface chip after the bit error occurs.

Error type: Bit error

Detectable: Yes

Detection: Standards based error detection occurs during step 9 as the target receives the data. If the bit error occurred in the standard data block, the data guard computed by the target does not match the DIF Guard. If the bit error occurred in one of the Tag fields, the value in the DIF would not match the value specified in the CDB for that I/O operation.

Case 3: "Wild store" error

Scenario: The processor or a bus master on the interconnect bus such as an I/O interface chip writes one or more bytes to a seemingly random memory location in the data buffer. (The error could be due to a hardware error, or a software error such as use of an incorrect pointer.) In this particular case, data in buffer 452 is overwritten after data has been received from the host, but prior to transmission to the drive.

Error type: Bit error

Detectable: Yes

Detection: Standards based error detection occurs during step 9 as the target receives the data. If a byte or multi-byte error occurred in the standard data block, the data guard computed by the target does not match the DIF Guard. If the error occurred in one of the Tag fields, the value in the DIF would not match the value specified in the CDB for that I/O operation.

Similar errors that occur on write errors in the drive are not detectable from a standards viewpoint since the errors occur after data is transferred through the interface. The intermediate storage device can detect these error on a read, however. One benefit of the standards based protection is that the intermediate device can detect an error on protection applied by the host even if the data is being manipulated, moved, or copied to another location without transferring the data to the host.

5. Interoperability

The proposal allows the initiator to arbitrarily specify Reference Tags on a per I/O basis using algorithms that are unknown to the target device. This can be done only in a homogeneous system where the algorithm is controlled by a single entity such as the file system or HBA vendor. Realistically, heterogeneous systems require that the same algorithm be used by all initiators in a given SCSI domain. In the system example shown in Figure 3, if both host systems wish to access a common set of logical units in SCSI domain A, then they must use the same algorithm for Reference Tags. This can be enforced at a system level by the intermediate storage device by forcing the Reference Tags to be LBA based via the Data Integrity Mode Page, hence the expression “LBA locked” for that SCSI domain. Note that this does not mean that the Reference Tags must be LBA locked in SCSI domain B. If the intermediate storage device is a mapping device that stripes data across multiple drives, then this is likely a homogeneous environment, where all initiators in that domain understand the data layouts. In this case, the Reference Tags applied in SCSI domain A can be passed through to SCSI domain B, even if the tags don’t match the drive LBAs.

If the Reference Tags are LBA locked in SCSI domain A, then the intermediate storage device can allow legacy and DIF capable systems to share access to a volume as illustrated in Figure 5.

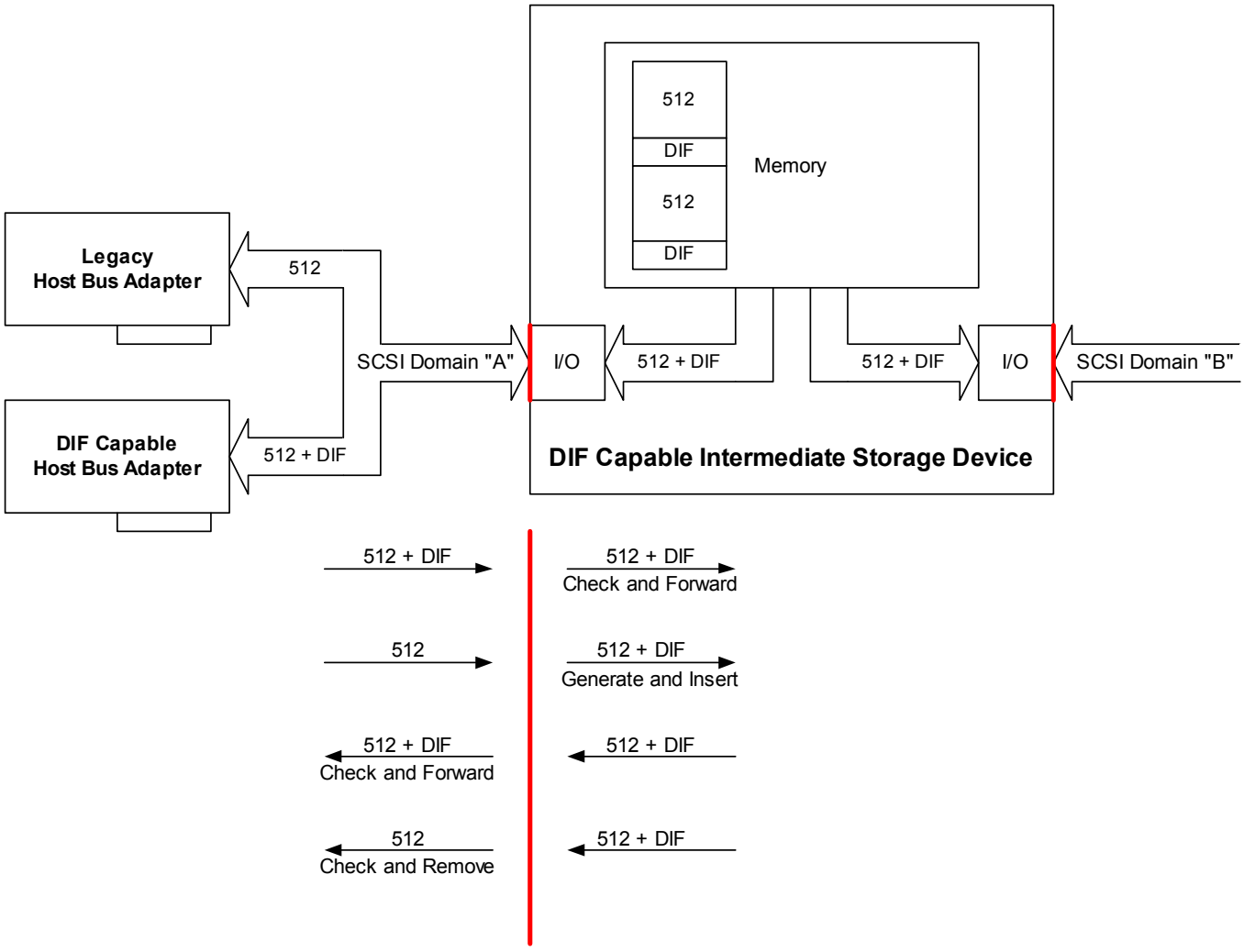

Figure 5 – Interoperability with Dissimilar Host Capabilities