

I/O Controller Data Integrity Extensions

Martin K. Petersen, Oracle Linux Engineering <martin.petersen@oracle.com>

Introduction

In order to provide customers with increased data integrity protection, Oracle has worked with industry partners to develop an integrity protection scheme that is standards-based and not specific to the Oracle database.

Work within the T10 committee that governs the SCSI protocols led to the ratification of the Protection Information Model extensions to the SCSI block protocol. The T10 Protection Information Model (also known as Data Integrity Field, or DIF) provides means to protect the communication between host adapter and storage device. However, the T10 specification only describes the protocol between the host adapter and the storage device. How the operating system interacts with the I/O controller is outside the scope of T10.

The Data Integrity Extensions described in this document specify how an I/O controller should interact with the host operating system to engage in exchange of protection information, thus enabling end-to-end data integrity.

T10 Protection Information Model

Enterprise drives using the SCSI protocol have long had the capability to be reformatted to bigger sector sizes such as 520 bytes. The extra 8 bytes of information per sector have traditionally been used by array firmware to store integrity information proprietary to the array. The T10 PIM was introduced as a way to use those extra bytes in an open and standardized fashion.

The 8 bytes of protection information are divided up as follows:

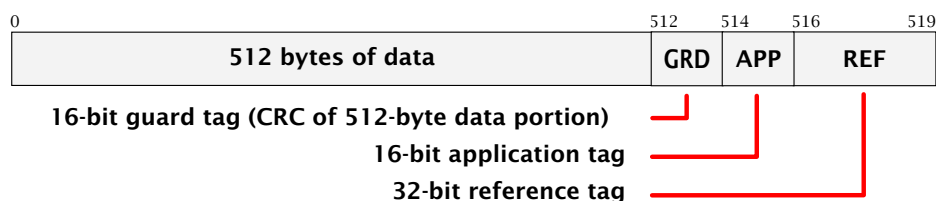


Figure 1 T10 Data Integrity Fields

The guard tag protects the data portion of the sector. The application tag is simply opaque storage. And finally, the reference tag is being used to protect against out-of-order and misdirected write scenarios.

Standardizing the contents of the protection information enables all nodes in the I/O path, including the disk itself, to verify the integrity of the data block.

Comparison of I/O Paths

A typical I/O submission scenario in an enterprise configuration is illustrated in figure 2. The only entity capable of using the 8 bytes of protection information is the array firmware.

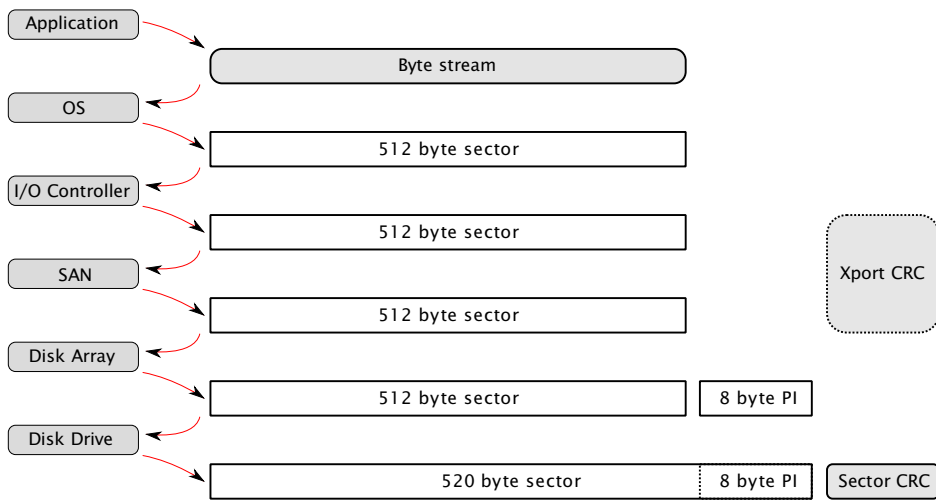


Figure 2 Normal I/O write: Application writes byte stream to OS. Filesystem writes in logical blocks that are multiples of 512-byte sectors. Depending on physical transport, a CRC may be applied on the wire. Array firmware generates 8 bytes of proprietary protection information. Disk stores 520-byte sectors and generates its own CRC.

A similar T10 PI-enabled configuration will look like figure 3. The I/O controller generates and appends the protection information and every subsequent node in the I/O path can verify the data integrity.

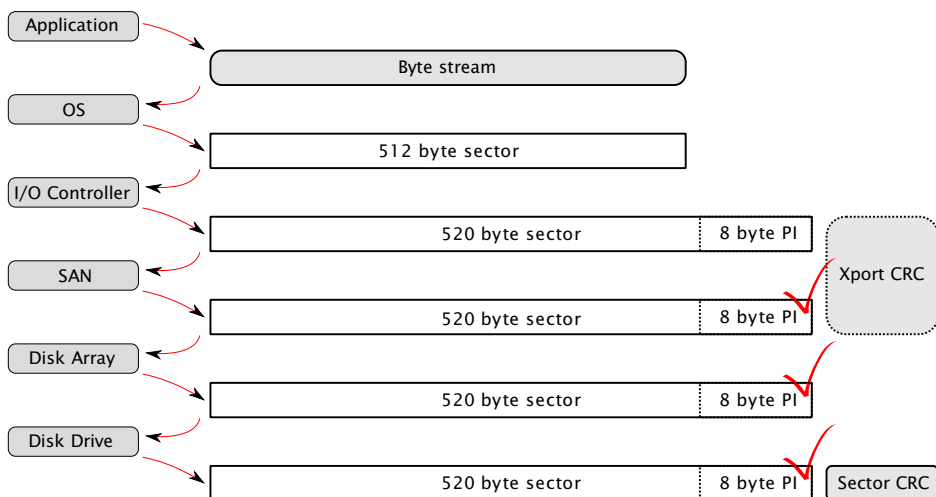


Figure 3 T10 PI I/O write: Application writes byte stream to OS. Filesystem writes in logical blocks that are multiples of 512-byte sectors. HBA generates protection information and sends out 520-byte sectors. SAN switch can optionally check protection information. Array firmware verifies protection information, optionally remaps reference tags and writes to disk. Disk verifies protection information before storing request.

Combining T10 PIM with the Data Integrity Extensions allows the protection information to be attached even higher up in the stack—either in the application or in the operating system. The entire I/O path is protected and true end-to-end data integrity protection is achieved.

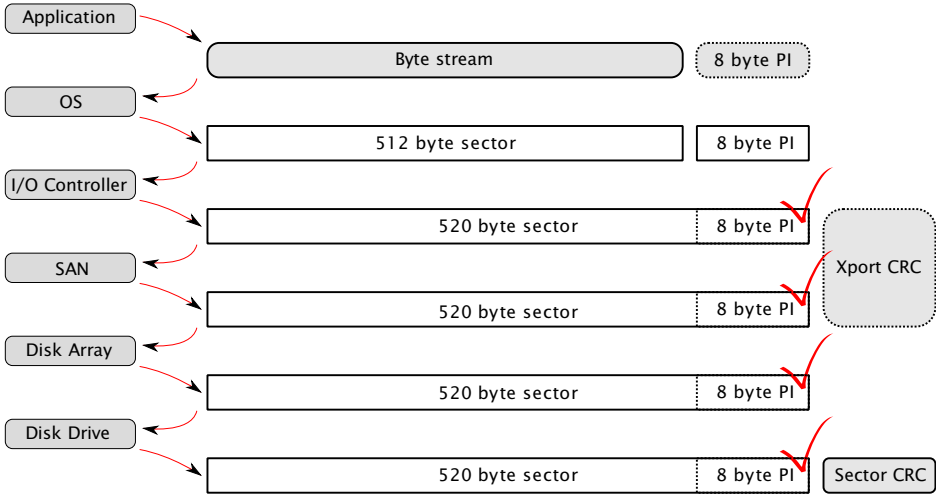


Figure 4 Combined DIX + T10 PI I/O write: Application writes byte stream to OS, optionally including protection information. Filesystem writes in logical blocks that are multiples of 512-byte sectors. If no protection information has been generated, OS automatically does so and attaches it to the I/O. HBA verifies data integrity, merges data and protection scatterlists and sends out 520-byte sectors. The remainder of the I/O path is similar to the T10 PI example above.

Figure 5 illustrates the protection coverage for the protection schemes mentioned above:

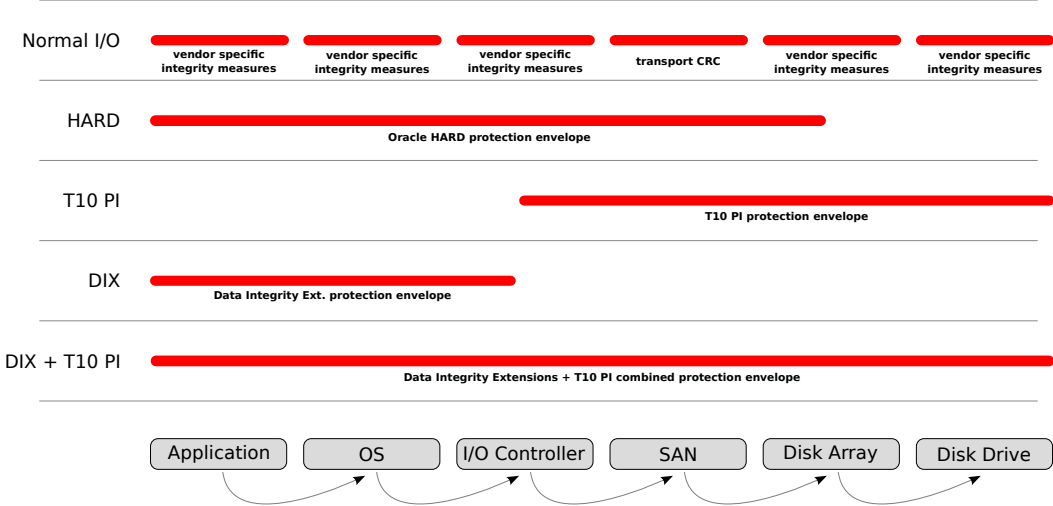


Figure 5 The Normal I/O line illustrates the disjoint integrity coverage offered using a current operating system and standard hardware. The HARD line shows the protection envelope offered by the Oracle Database accessing a disk array with HARD capability. T10 PI shows coverage using the integrity portions of the SCSI protocol. DIX shows the coverage offered by the Data Integrity Extensions. And finally, DIX + T10 PI illustrates the full coverage provided by the Data Integrity Extensions in combination with T10 PI.

Data Integrity Extensions

As shown above, in order to facilitate true end-to-end data integrity between application and storage device, integrity metadata must be made accessible to the operating system. We have developed a model in which the T10 protection information can be transferred to and from the operating system using standard DMA.

While the functionality of the operating system-level interfaces can be implemented on top of most currently available controllers, the Data Integrity Extensions have been designed to provide extra features that drastically improve performance. Most of these features are optional but if they are not present they put additional load on the system processor. We encourage vendors to implement the full feature set described below.

Scatter-Gather List Separation

On the wire between initiator and target the T10 SCSI Block Commands specification mandates that the protection information is interleaved with the data sectors. For instance 512 bytes of data followed by 8 bytes of protection information. This is very inconvenient for the operating system which usually stores application data in 4KB pages. Doing memory copies to interleave data and protection data wastes system resources. Our benchmarking shows that manual interleaving by way of scatterlists also impacts system performance.

Consequently, in our model interleaving of data and protection information has been delegated to the I/O controller. The controller driver will receive a request with two scatter-gather lists attached. One containing the data in/out buffer as usual and one containing the matching protection data in/out buffer.

When a WRITE request is received from the operating system the controller must verify that the data buffer and the protection buffer are in agreement and subsequently interleave the two buffers to get the desired 520 (512 + 8) byte sectors mandated by the T10 Protection Information Model.

When a READ request is received from the storage device the controller must verify that the data portions match the protection information, split the 520-byte sectors and DMA to the data and protection buffers respectively.

The scatter-gather list elements of the protection buffer will honor the same alignment constraints as those of the data buffer.

Protection Data Endianness

The protection information format is network-endian as per T10. This includes the protection data exchanged with the host.

Guard Tag Format

The format of the guard tag between initiator and target is specified in the T10 SBC standard as a cyclic redundancy check using a well-defined polynomial. A CRC is expensive to calculate in software and benchmarks show that it has a significant impact on the performance of a system.

Various algorithms were tested and the IP checksum was selected for the first implementation of this. The IP checksum is cheap to calculate and also has a few other properties that make it suitable for this purpose.

It should be noted that the T10 CRC checksum is mandatory and the alternate guard tag format is an optional feature. An I/O driver, utility or management tool can indicate to the operating system whether to use the T10 CRC, the IP checksum or (in the future) another format. The operating system is responsible for negotiating the best performing algorithm in the case of for instance multipathing using controllers with different checksum capabilities.

This negotiation capability is also used to pick a suitable checksum format when a software RAID device spans devices with different hardware sector size or guard tag capabilities.

The guard tag type is required to be a per-request property, not a global setting.

Error Detection and Isolation

For error isolation purposes the I/O controller is expected to verify data integrity before passing it on. This is an optional feature when the T10 CRC is used. However, it is mandatory when any guard tag conversion is taking place.

In case of mismatch in the protection data the I/O controller must return a suitable error to the operating system. The error notification must include in which request, at which LBA the mismatch occurred, and whether it was a guard, reference or an application tag failure.

If a storage device returns a value of `0xFFFF` in the application tag and the device is formatted with T10 PI Type 1 or 2 protection, the I/O controller must disable integrity checking for that sector.

Similarly, if a storage device returns a value of `0xFFFF` in the application tag, a value of `0xFFFFFFFF` in the reference tag and the device is formatted with T10 PI Type 3 protection, integrity checking must be disabled for that sector.

For all other errors the controller is expected to abort the I/O, report the error to the OS, and let the OS deal with (SPC/SBC-level) retries. Physical transport (SPI, SAS, FC) retries are usually the responsibility of the I/O controller firmware.

DIX Operations

In the case of both READ and WRITE requests the CDB passed to the I/O controller has a RDPROTECT / WRPROTECT field which indicates whether to transfer protection data between initiator and target. Given the OS-to-I/O controller interface is outside the scope of T10, there are no similar controls for this piece of the I/O path.

The OS will always prepare a CDB with appropriate RDPROTECT / WRPROTECT information depending on target format and capabilities. The request passed to the controller driver will also include information about which protection type the target has been formatted with, and which checksum is used for communication between OS and I/O controller.

Finally, the OS will also provide the I/O controller driver with an operation code which tells the controller which type of I/O to perform. The operation codes are:

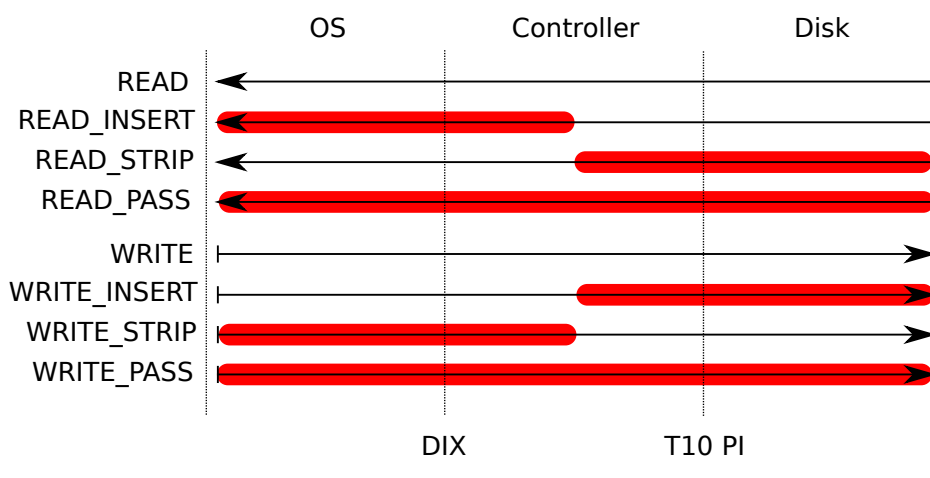


Figure 6 DIX operations: Red bars indicate protected path.

- READ and WRITE are for normal (unprotected) I/O
- READ_INSERT Read data from target, generate protection data and transfer both to OS.
- WRITE_STRIP Transfer data and protection data from OS, verify data integrity, discard protection data and write data to target
- READ_STRIP Read data and protection data from target, verify data integrity, discard protection data and transfer data to OS
- WRITE_INSERT Transfer data from OS, generate protection data and write both data and protection data to target
- READ_PASS Read data + protection data from target, optionally verify protection data, optionally convert checksum, transfer both data and protection data to OS
- WRITE_PASS Transfer data + protection data from OS, optionally verify protection data, optionally convert checksum, write both data and protection data to target

READ_INSERT and WRITE_STRIP are for DIX in combination with legacy (non-PI) devices. These two operations enable protection between OS and controller only. The contents of the guard tag and the reference tag are defined to be the same as in Type 1. Given that there is no place to persistently store the application tag the contents of the application tag must be ignored.

READ_STRIP and WRITE_INSERT are for regular T10 PI operation where no protection information is exchanged with the operating system.

READ_PASS and WRITE_PASS are for passing protection information through to the target. Checksum conversion may be taking place in the I/O controller if something other than the T10 CRC is used when communicating with the host.

Protection Modes

The controller driver must indicate its T10 PI and DIX capabilities to the operating system so that requests can be prepared accordingly. The following protection modes are defined:

Mode	Description
T10 Type 0	Normal, unprotected I/O.
T10 Type 1	Protection supported between controller and a storage device formatted with T10 Type 1 protection information.
T10 Type 2	Protection supported between controller and a storage device formatted with T10 Type 2 protection information.
T10 Type 3	Protection supported between controller and a storage device formatted with T10 Type 3 protection information.
DIX Type 0	Protection DMA enabled between OS and controller. Storage device is not formatted with protection information.
DIX Type 1	Protection DMA enabled between OS and controller. Storage device is formatted with T10 Type 1.
DIX Type 2	Protection DMA enabled between OS and controller. Storage device is formatted with T10 Type 2.
DIX Type 3	Protection DMA enabled between OS and controller. Storage device is formatted with T10 Type 3.

T10 Type 1-3 indicate that the controller implements the READ_STRIP and WRITE_INSERT operations.

DIX Type 0 indicates that the controller implements the READ_INSERT and WRITE_STRIP operations.

DIX Type 1-3 indicate that the controller implements the READ_PASS and WRITE_PASS operations.

Request Routing

It is important to emphasize that there are two distinct protection envelopes to consider:

- Transfer of protection information between operating system and I/O controller
- Transfer of protection information between I/O controller (initiator) and disk (target)

Whether to protect the path between OS and controller is up to the application, the operating system or system administrator preference. Thus there is no guarantee that an I/O request bound for a controller supporting the Data Integrity Extensions will provide a scatter-gather list for integrity metadata. It is a per-I/O property.

Similarly, whether the path between controller and disk should be protected with T10 PI is controlled by the RDPROTECT / WRPROTECT field in the CDB.

The two protection envelopes are completely *orthogonal*. And any combination can be expected on hardware that supports it.

Example: A WRITE request could include an integrity metadata scatter-gather list despite being bound for a storage device that is not formatted using T10 PI. In that case the controller must read and verify the protection information and then use standard 512 byte sectors when communicating with the target. The operation in this case is called WRITE_STRIP because it is a *write* request and the protection data must be *stripped* off of the I/O after verification. We refer to this mode of operation as DIX Type 0.

The following charts illustrate how to route READ and WRITE requests respectively:

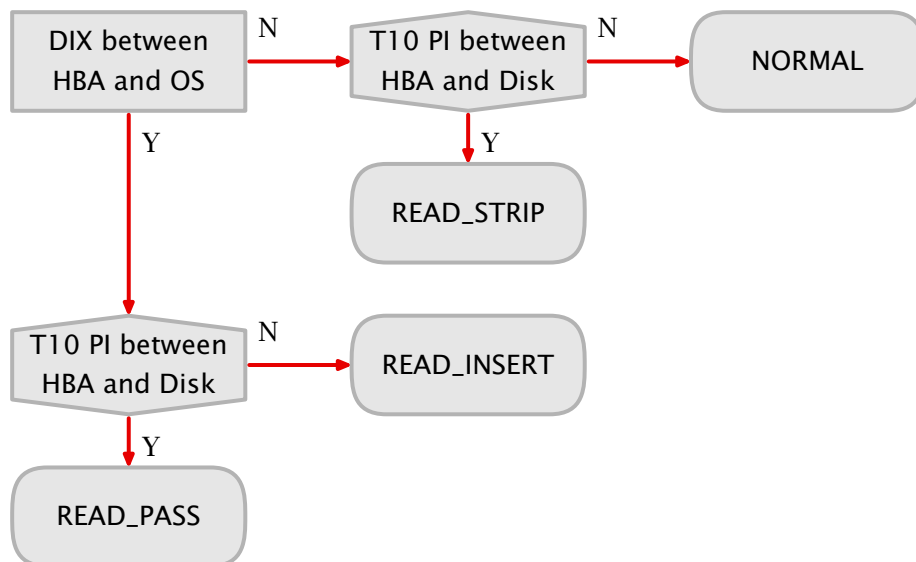


Figure 7 READ request

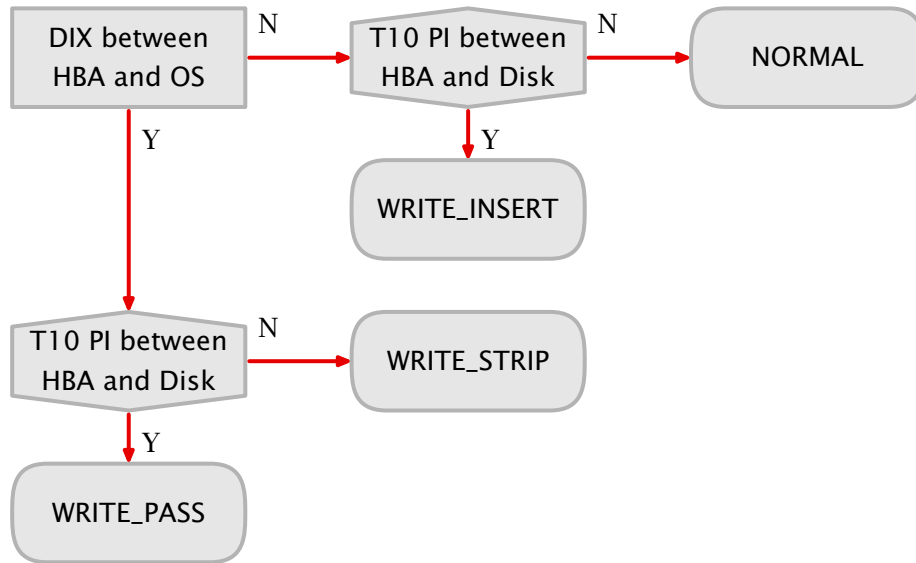


Figure 8 WRITE request

Revision History

Date	Change
2009-11-19	Deprecated CONVERT operations and started referring to "T10 PI" instead of "DIF".
2008-07-18	Added illustrations and incorporated request routing document
2008-03-31	Clarify handling of uninitialized DIF data (0xFFFF in the application tag)
2008-01-10	Require protection info to be network-endian
2007-10-02	Initial version