Hypervisor Storage Interfaces for Storage Optimization White Paper

June 2010

SNIA

Revision 1.0

Table of Contents

List of Tables

List of Figures

About the SNIA

The Storage Networking Industry Association (SNIA) is a not-for-profit global organization, made up of some 400 member companies and 7,000 individuals spanning virtually the entire storage industry. SNIA's mission is to lead the storage industry worldwide in developing and promoting standards, technologies, and educational services to empower organizations in the management of information. To this end, the SNIA is uniquely committed to delivering standards, education, and services that will propel open storage networking solutions into the broader market. For additional information, visit the SNIA web site at www.snia.org.

Preface

This document is aimed at Hypervisor software providers and providers of storage intended for use with Hypervisor environments. It uses Hypervisor examples to describe the use of commands from the SCSI T10 Standards for performance and utilization enhancements such as copy offload, atomic sequence offload, and space reclamation technology in a blocks environment. This includes using the SCSI EXTENDED COPY command for providing copy offload services, the SCSI COMPARE AND WRITE command for atomic sequence offload, and SCSI unmap operations for space reclamation services.

About the SNIA HSI Technical Working Group

The SNIA Hypervisor Storage Interfaces (HSI) Technical Working Group is a multi-vendor group of Hypervisor vendors, Storage vendors and System vendors. The goal of the group is to define common methods for interaction between hypervisor software and storage devices.

Acknowledgements

We would like to take this opportunity to acknowledge the many contributions that contributed many hours, dedication, and persistence to make this document possible.

- The SNIA's HSI TWG

1 Hypervisor Storage Interfaces

1.1 Terms and Concepts

Virtual Disk – A set of disk blocks presented to an operating environment as a range of consecutively numbered logical blocks with disk-like storage and I/O semantics. In the context of this document, a virtual disk is defined as a set of disk blocks presented by a hypervisor to a guest operating environment as a range of consecutively number logical blocks with a disk-like storage and I/O semantics.

Virtual Machine (VM) – A representation of a physical machine that is used by the guest operating system as the hardware platform on which it executes. A virtual machine is comprised of one or more virtual disks and associated meta-data.

Hypervisor – Software that provides virtual machine environments which are used by guest operating systems.

Perceived logical unit – A unit of storage as seen by the hypervisor.

1.2 Hypervisor Storage

Hypervisor Storage refers to a storage abstraction layer that is used by a Hypervisor to create "Virtual Disks" in the Virtual Machine. It is an intermediate storage abstraction layer located between the guest operating system storage stack running in a VM and the perceived logical units. From the perspective of a guest operating system executing within a VM, a virtual disk is perceived as a logical unit, and is generally indistinguishable from a physical disk. These Virtual Disks are in turn comprised of a collection of blocks that are located on perceived logical units. Virtual Disks are "encapsulated", meaning that they are a self-contained, self-describing resource to the VM. Virtual Disks may be represented on physical storage arrays as Files, Blocks or Objects. The address space of a virtual disk (i.e., the blocks that comprise the virtual disk) appear to the guest operating system as a contiguous address space. These contiguous addresses are mapped by the hypervisor storage layer to the perceived logical unit and may be located on different physical devices/granularities at the discretion of the hypervisor storage layer.

Virtual Disks are completely abstracted by the Hypervisor storage layer. There is no visibility into the composition of a given virtual disk by the storage array(s) on which the blocks are stored. In addition, the guest operating system has no visibility into the physical storage array that hosts a virtual disk.

Multi-layer, or nested virtualization, may also occur. For example, a guest operating system may itself be a hypervisor that uses its perceived logical units to create virtual disks for guest operating systems of its own. An example of this is shown in Figure 1.

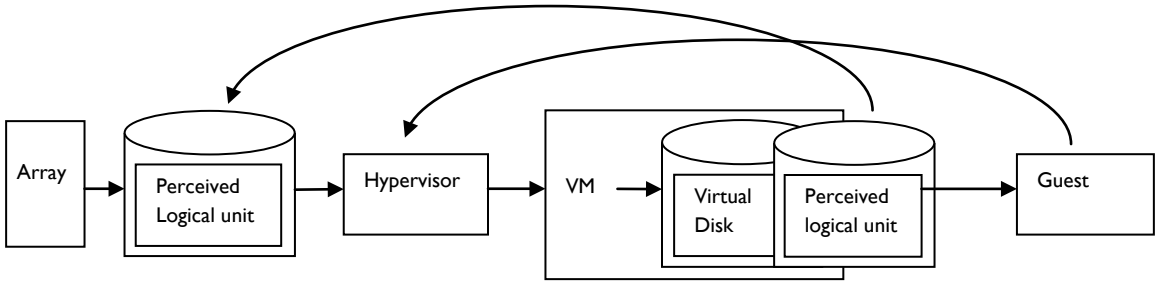

Figure 1 – Nested Perceived Logical Units

1.3 Hypervisor Storage Interfaces (HSI)

To assist with the management and operation of Hypervisor Storage, the SNIA HSI TWG is clarifying the use of, extending, and/or developing standard Hypervisor Storage Interfaces for storage interactions with hypervisors. These interfaces enable the hypervisor to offload certain storage intensive tasks to the underlying storage systems to achieve performance and functional improvements.

It is the purpose of this document to describe use cases where it is desirable to have the storage array perform offload operations and/or have some visibility into virtual disk constructs while maintaining the necessary and appropriate abstractions and isolation.

The first revision of this document focuses on optimizing copy operations and space reclamation operations in block storage environments.

1.4 Offload Use Cases

The offload use cases addressed in this first revision include:

- Copying a Virtual Disk: Create a copy of an individual Virtual Disk with no burden on the server/hypervisor to optimize hypervisor performance.
- Copying an entire Logical Unit: Create a copy of an entire Logical Unit (which may contain one or more virtual disks) with no burden on the server/hypervisor to optimize hypervisor performance.
- Copying a Virtual Machine: Create a copy of a Virtual Machine (which may involve multiple logical units) with no burden on the server/hypervisor to optimize hypervisor performance.
- Virtual Machine Storage Migration: Leverage array-based copy/replication function to non-disruptively migrate the storage for an entire Virtual Machine from one location to another.
- Dividing Virtual Machines: Enable the non-disruptive split of a Virtual Machine set by providing a single procedure that simultaneously relocates a subset of virtual machines from one storage location to another.
- Reclaiming Unused Space After Virtual Disk Deletion: Offload to the array the reclaiming of blocks after a Virtual Disk is deleted,

- Reclaiming Unused Space on behalf of a guest: Offload to the array the reclaiming of blocks when a guest operating system deletes a file or otherwise no longer needs to retain a particular set of data.

2 Copy Offload Model

2.1 Terms and Concepts

Copy Manager – The entity within a logical unit that manages extended copy operations. The copy manager receives the copy request and performs the necessary operations to execute a copy operation.

IP Copy Service – An IP entity outside the SCSI device server that communicates with the copy manager and performs the necessary operations to transfer data through an IP network.

Copy Source Device – The SCSI logical unit from which data is read.

Copy Destination Device – The SCSI logical unit to which the data is written.

Copy Target Device –Either a copy source device or a copy destination device.

Target Descriptor – A data structure used to describe a copy target device.

Segment Descriptor – A data structure that describes a segment (as a starting logical block address and length) to be transferred, and provides an indication of the copy target devices involved in the transfer.

Relative Port Identifier – The unique identifier of the port through which the copy manager accesses an external logical unit.

2.2 Overview

Copy offload operations are performed by a SCSI copy manager. A copy manager is a process in a SCSI logical unit that accepts copy offload requests from a host system, and performs the requested transfers on behalf of the requesting host. The host uses the SCSI EXTENDED COPY command (see SPC-4) to specify the details of the copy request to the copy manager.

The EXTENDED COPY command is one of the most complex commands in the SCSI command set. The EXTENDED COPY command contains at a minimum:

a. an identifier of a device from which data is read (a target descriptor);
b. an identifier of a device to which data is written (a target descriptor); and
c. a description of the data to copy (a segment descriptor).

The complexity of this command results from the multitude of options available through this command (e.g., eleven (11) different types of target descriptors are available, and twenty two (22) different types of segment descriptors are available, creating a huge number of possible combinations). In addition, multiple copy operations may be described in a single command (multiple segments may be copied from multiple sources to multiple destinations). This document focuses on the most common combinations of these descriptors and how they are used to provide copy offload services for hypervisors.

2.2.1 Scope of Copy Managers

Some copy managers may have the ability to provide copy offload services only for blocks on logical units that are contained within the same array that contains the copy manager itself (intra-array copy offload).

Some copy managers may have the ability to provide copy offload services for blocks on logical units that span multiple arrays (inter-array copy offload) as well as the ability to provide intra-array copy offload.

Some copy managers have the ability to use TCP/IP protocols to perform inter-array copy offload using an IP copy service.

While, copy managers are permitted to provide copy offload services exclusively for logical units that span multiple arrays (inter-array copy offload only), these types of copy managers are rare.

This document uses the term "internal" when referencing a logical unit that is within the same array that contains the copy manager. This document uses the term "external" when referencing a logical unit that is not within the same array that contains the copy manager. These terms are not used in the SPC-4 standard where the EXTENDED COPY command is described.

2.2.2 Copy Manager Identifiers

Internal logical units are typically identified using a target descriptor known as an "Identification descriptor target descriptor" (type E4h in SPC-4). This type of descriptor contains a designator that is matched with a designation descriptor in a logical units Device Identification VPD page (page 83h). The logical unit with the matching designation descriptor is used for the copy operation.

Internal logical units may also be identified using target descriptors (e.g., types E0h-E3h, E5h-EAh) containing port identifiers and the logical unit number (or LU Identifier). In this case however, the port identifier in the target descriptor must match a port identifier of a port within the array that is allowed access to the specified internal logical unit. Due to the additional complexity of this type of identifier, use of target descriptors of this type for internal logical units is not recommended.

External logical units may be identified in three ways:

a. Using an Identification descriptor target descriptor (type E4h)
b. Using a port type target descriptor (e.g., type E0h-E3h, E5h,-EAh)
c. Using an IP copy service target descriptor (type EBh).

While other port type target descriptors may be used, this document discusses the following port type target descriptors:

a. N_PORT_NAME (i.e., type E0h),
b. N_PORT_ID (i.e., type E1h),
c. N_PORT_NAME and N_PORT_ID (i.e., type E2h),
d. IPv4 address (i.e., type E5h),
e. SAS address (i.e., type E9h),
f. IPv6 address (i.e., type EAh).

When using a port type target descriptor, the host specifies the identifier (e.g., N_PORT_NAME, N_PORT_ID, IPv4 address, IPv6 address. SAS address) of the port on the external array that is to be used to access the specified logical unit number (LU Identifier). The host may optionally specify which port (i.e., the relative port identifier) on the array containing the copy manager to use to initiate the operation with the external array. The host has the ability to totally control which path to use for the copy operation. The copy manager performs just the operations specified by the host.

When using an identification descriptor target descriptor, the host specifies the designator that is to be matched with a designator in the Device Identification VPD page of the logical unit selected for the operation. The host may optionally specify which port (i.e., the relative port identifier) on the array containing the copy manager to use to initiate the operation with the external array, but the host cannot specify which port on the external array to use for the copy operation. With the identification descriptor target descriptor, the copy manager is responsible for examining the external logical units to which it has access, and finding a logical unit that has a Device Identification VPD page designator entry that matches the designator specified by the host. This examination of external logical units to find a matching Device Identification VPD page entry requires additional operations from the copy manager beyond just the copy operation requested by the host.

When using an IP copy service target descriptor, the host specifies the IP address of the copy service indicated by the external logical unit in its Management Network Addresses VPD page (85h), and a designator that is to be matched with an entry in the Device Identification VPD page of the logical unit selected for the operation. The host cannot specify any portion of the path used for communication with the external array. The IP copy service is responsible for examining external logical units to which it has access, and finding a logical unit that has a Device Identification VPD page entry that matches the designator specified by the host. The copy manager performs just the operations specified by the host.

2.3 Example Operations

Figure 2 shows an example of a fully external copy offload operation. Figure 3 shows an example of a copy offload from an internal logical unit to an external logical unit. Figure 4 shows an example of a copy offload operation from an external logical unit to an internal logical unit. Figure 5 shows a copy offload between logical units within the same array as the copy manager. Figure 6 shows a copy offload using an IP copy service to perform the transfers over an IP network connected logical unit to an internal logical unit.

2.3.1 External Copy Operation

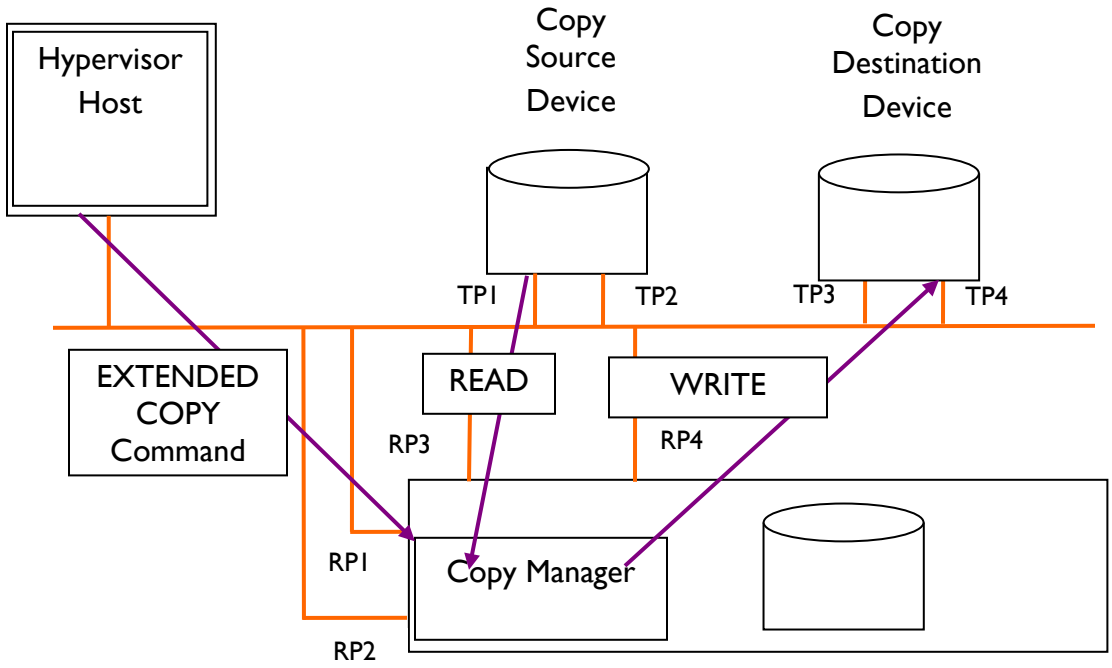

Figure 2 – External Copy Offload Operation

When a host wants to copy blocks from an external source device to an external destination device (see Figure 2) these procedures are used. To perform copy offload, one or more target descriptors and one or more segment descriptors are created to have the copy manager perform this operation on behalf of the requesting host:

A. A source target descriptor containing:
   either:
   a. the relative port identifier of the port over which to issue the READ command (RP3) (or zero);
   b. the port identifier (see 2.2.2) of the port on the source device to process the READ command (TP1); and
   c. the logical unit number of the logical unit from which to read the data.

or

a. the relative port identifier of the port over which to issue the READ command (RP3) (or zero);
b. an identification descriptor target descriptor containing the Device Identification designator of the logical unit from which the data is to be read.

B. A destination target descriptor containing:

either:

a. the relative port identifier of the port over which to issue the WRITE command (RP4);
b. the port identifier (see 2.2.2) of the port on the copy destination device to process the WRITE command (TP4); and
c. the logical unit number of the logical unit to which the data is written.

or

a. the relative port identifier of the port over which to issue the WRITE command (RP3) (or zero);
b. an identification descriptor target descriptor containing the Device Identification designator of the logical unit to which the data is to be written.

C. Segment descriptor(s) containing:

a. the number of logical blocks to be transferred;
b. the starting LBA on the source device to begin the READ operation; and
c. the starting LBA on the destination device to begin the WRITE operation.

If an identification descriptor target descriptor is used, the copy manager is responsible for determining the location of the logical unit with the matching designator. This may involve the copy manager maintaining information about accessible logical units and the paths (e.g., which relative ports and/or target ports) via which those logical units may be accessed.

The EXTENDED COPY command is sent to the logical unit containing the copy manager (via RP1). The copy manager then connects through RP3 to TP1 and sends appropriate READ commands (using the starting LBA and length from the segment descriptor) to the specified logical unit. The copy manager then connects through RP4 to TP4 and sends appropriate WRITE commands (using the starting LBA and length from the segment descriptor) to the specified logical unit.

The data has now been copied from the copy source device to the copy destination device without any movement of data through the host.

2.3.2 Internal to External Copy Operation

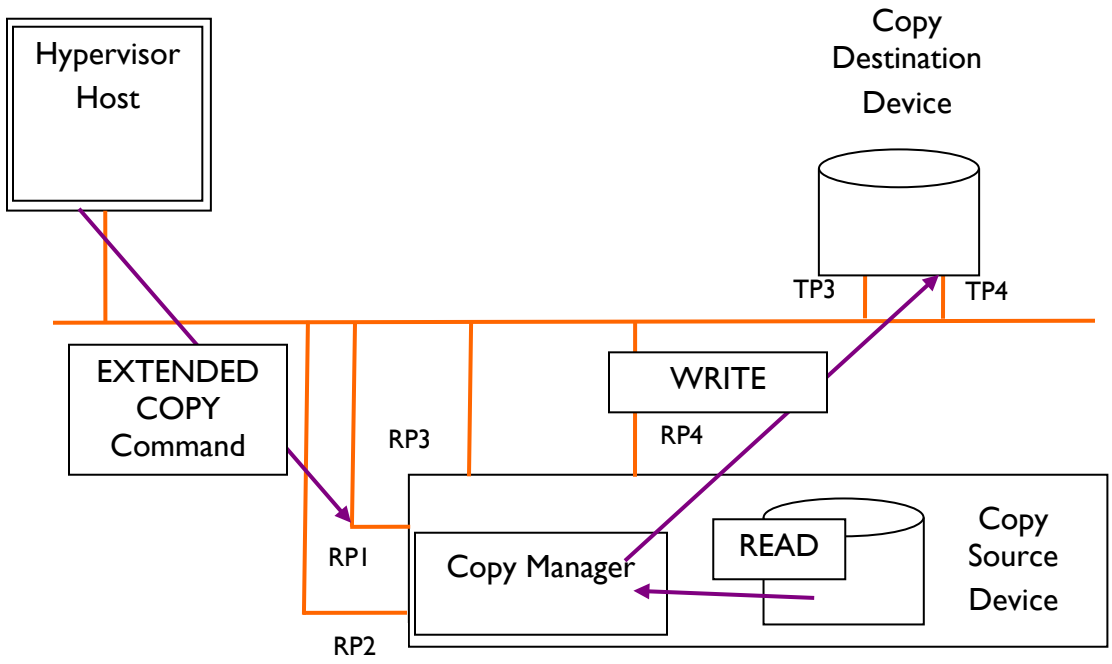

Figure 3 – Internal to External Copy Offload Operation

When a host wants to copy blocks from an internal source device to an external destination device (see Figure 3) these procedures are used. To perform copy offload, one or more target descriptors, and one or more segment descriptors are created to have the copy manager perform this operation on behalf of the requesting host:

A. A source target descriptor containing:
   either:
   a. a relative port identifier of zero;
   b. the port identifier (see 2.2.2) of a port on the source device that has access to the logical unit from which to read the data (RP1, RP2, RP3, or RP4); and
   c. the logical unit number of the logical unit from which to read the data.
   or
   a. a relative port identifier of zero;
   b. an identification descriptor target descriptor containing the Device Identification designator of the logical unit from which to read the data.

B. A destination target descriptor containing:
   either:
   a. the relative port identifier of the port over which to issue the WRITE command (RP4) (or zero);

b. the port identifier (see 2.2.2) of the port on the destination device to process the write command (TP4); and
c. the logical unit number of the logical unit to which the data is to be written.

or

a. a relative port identifier of the port over which to issue the WRITE command (RP4) (or zero);
b. an identification descriptor target descriptor containing the Device Identification designator of the logical unit to which the data is to be written.

C. Segment descriptor(s) containing:
a. the number of logical blocks to be transferred;
b. the starting LBA on the source device to begin the READ operation; and
c. the starting LBA on the destination device to begin the WRITE operation.

If the destination target descriptor is an identification descriptor target descriptor, then the copy manager is responsible for determining the location of the logical unit with the matching designator. This may involve the copy manager maintaining information about accessible logical units and the paths (i.e., the relative ports and/or target ports) via which those logical units may be accessed.

The EXTENDED COPY command is sent to the logical unit containing the copy manager (via RP1). The copy manager then reads from the specified local logical unit (using the starting LBA and length from the segment descriptor). The copy manager then connects through RP4 to TP4 and sends appropriate WRITE commands (using the starting LBA and length from the segment descriptor) to the specified logical unit.

The data has now been copied from the copy source device to the copy destination device without any movement of data through the host.

2.3.3 External to Internal Copy Operation

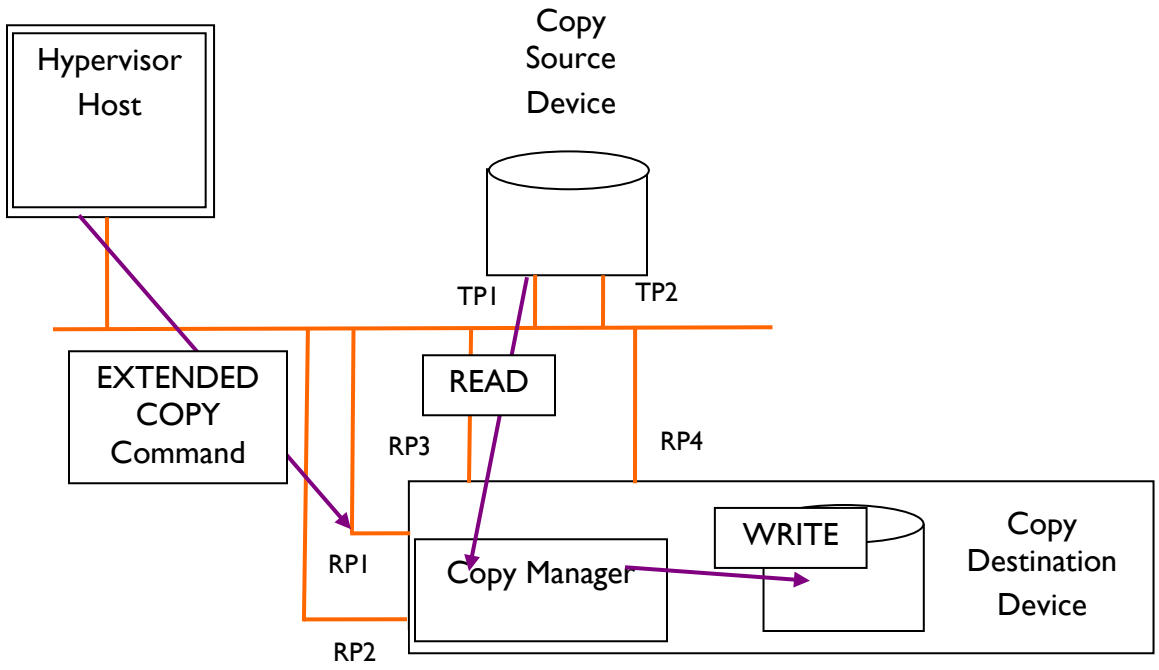

Figure 4 – External to internal Copy Offload Operation

When a host wants to copy blocks from an external source device to an internal destination device (see Figure 4) these procedures are used.

A. A source target descriptor containing:
   either:
   a. the relative port identifier of the port over which to issue the READ command (RP3) (or zero);
   b. the port identifier (see 2.2.2) of the port on the source device to process the read command (TP1); and
   c. the logical unit number of the logical unit from which to read the data.
   or
   a. the relative port identifier of the port over which to issue the READ command (RP3) (or zero);
   b. an identification descriptor target descriptor containing the Device Identification designator of the logical unit from which to read the data.
B. A destination target descriptor containing:
   either:
   a. a relative port identifier of zero;

b. the port identifier (see 2.2.2) of a port on the destination device that has access to the logical unit to which the data is to be written (RP1, RP2, RP3, or RP4); and
c. the logical unit number of the logical unit from which to read the data.

or

a. a relative port identifier of zero;
b. an identification descriptor target descriptor containing the Device Identification designator of the logical unit to which the data is to be written.

C. Segment descriptor(s) containing:
   a. the number of logical blocks to be transferred;
   b. the starting LBA on the source device to begin the READ operation; and
   c. the starting LBA on the destination device to begin the WRITE operation.

If the source target descriptor is an identification descriptor target descriptor, then the copy manager is responsible for determining the location of the logical unit with the matching designator. This may involve the copy manager maintaining information about accessible logical units and the paths (i.e., the relative ports and/or target ports) via which those logical units may be accessed.

The EXTENDED COPY command is sent to the logical unit containing the copy manager (via RP1). The copy manager then connects through RP3 to TP1 and sends appropriate READ commands (using the starting LBA and length from the segment descriptor). The copy manager then writes to the specified internal logical unit (using the starting LBA and length from the segment descriptor) to the specified logical unit.

The data has now been copied from the copy source device to the copy destination device without any movement of data through the host.

2.3.4 Internal Copy Operation

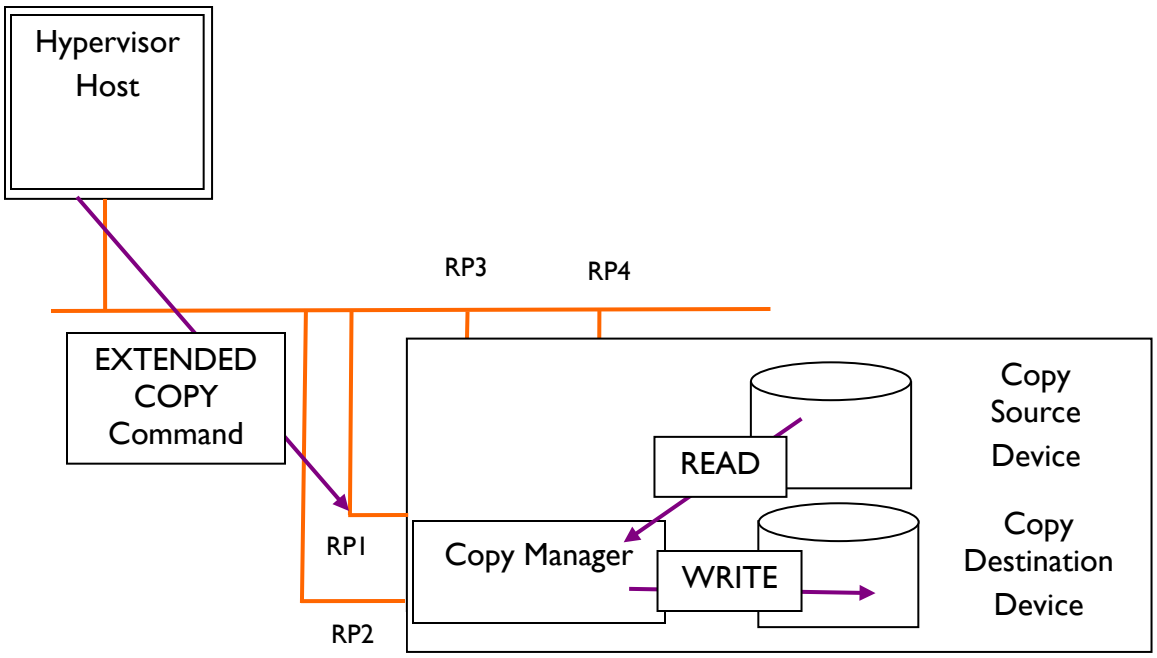

Figure 5 – Internal Copy Offload Operation

When a host wants to copy blocks from an internal source logical unit to another logical unit internal to the same array (see Figure 5). To perform copy offload, multiple descriptors are created to have the copy manager perform this operation on behalf of the requesting host:

A. A source target descriptor containing:
   a. a relative port identifier of zero;
   b. an identification descriptor target descriptor containing the Device Identification designator of the logical unit from which to read the data.
B. A destination target descriptor containing:
   a. a relative port identifier of zero;
   b. an identification descriptor target descriptor containing the Device Identification designator of the logical unit to which the data is to be written.
C. Segment descriptor(s) containing:
   a. the number of logical blocks to be transferred;
   b. the starting LBA on the source device to begin the READ operation; and
   c. the starting LBA on the destination device to begin the WRITE operation.

If the copy source device and the copy destination device are the same logical unit, then a single target descriptor may be used, with the segment descriptor referencing the same descriptor for both the source and the destination.

The EXTENDED COPY command is sent to the logical unit containing the copy manager (via RP1). The copy manager then reads from the specified internal logical unit (using the starting LBA and length from the segment descriptor). The copy manager then writes to the specified internal logical unit (using the starting LBA and length from the segment descriptor).

The data has now been copied from the copy source device to the copy destination device without any movement of data through the host.

2.3.5 IP Copy Service Operation

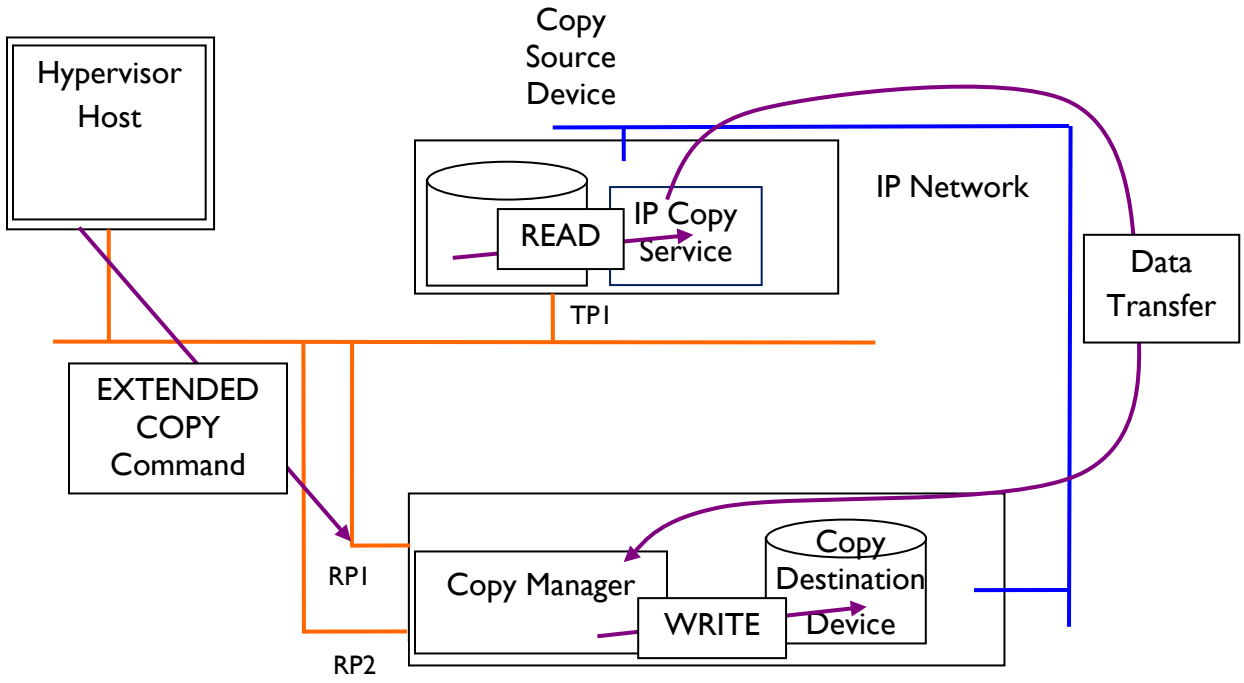

Figure 6 – IP Copy Service Operation

When a host wants to copy blocks from an external source device to an internal destination device (see Figure 6) using an IP copy service, these procedures are used. To perform copy offload, multiple descriptors are created to have the copy manager perform this operation on behalf of the requesting host:

A. An IP Copy Service target descriptor containing:
    a. the IP address of the copy service obtained from the Management Network Address VPD page (85h) of the source logical unit;
    b. an identification description designator containing the Device Identification designator of the logical unit from which to read the data.
B. A destination target descriptor containing:
    a. a relative port identifier of zero;

   b. an identification description designator containing the Device Identification designator of the logical unit to which the data is to be written.

C. Segment descriptor(s) containing:
   a. the number of logical blocks to be transferred;
   b. the starting LBA on the source device to begin the READ operation; and
   c. the starting LBA on the destination device to begin the WRITE operation.

The EXTENDED COPY command is sent to the destination logical unit. The copy manager supplies the necessary information from the EXTENDED COPY command that describes the source device, and the logical blocks to be transferred to the copy service at the specified IP address. The copy manager and copy service then perform the necessary transfers over an IP network to transfer the data from the source device to the destination device. The protocol used to perform this transfer is not standardized, and as such, this sequence may only be used when stated explicitly by an array vendor.

The data has now been copied from the copy source device to the copy destination device without any movement of data through the host.

2.4 Reservation Considerations

SCSI Reservations are used to control access to logical units. These reservations may impact the ability of the copy manager to access a logical unit.

If the RESERVE command (see SPC-2) is issued to a logical unit, that logical unit cannot be accessed by the copy manager because the logical unit is reserved for the exclusive use of the single initiator that issued the RESERVE command.

If reservations are required to restrict access to a logical unit, and a copy manager is also expected to access that logical unit, then the PERSISTENT RESERVE IN/OUT commands must be used to create the reservation. There are two methods available through persistent reservations to enable this capability:

A. The host creating the reservation may set the SPEC_I_PT bit (see PERSISTENT RESERVE OUT parameter data in SPC-4), and supply a TransportID list in the PERSISTENT RESERVE OUT command that creates the reservation. The Initiator identifier of the relative port in the copy manager that will be used to perform the copy operation must be set in the TransportID list. The relative port identifier used in the segment descriptor of the parameter data of the EXTENDED COPY command must be the relative port identifier of the Initiator identifier that was supplied in the TransportID list for the reservation.
B. The host creating the reservation may use the REGISTER AND MOVE service action when creating the reservation. This allows the host to create the reservation, and transfer access ability created by that reservation to a port within the copy manager (the relative port identifier of the port within the target device and the TransportID of the copy managers initiating port

are required for this operation). This method is described in the "Third party persistent reservations" section of SPC-4.

2.5 General Considerations

Hosts typically apply relatively short timers to each I/O operation they issue. In the case of the EXTENDED COPY command, the host must evaluate the request being made and evaluate what would be an appropriate timer. For example, an EXTENDED COPY that requested the transfer of 100 logical blocks would be expected to complete much faster than an EXTENDED COPY command that requested the transfer of 100 million logical blocks.

The EXTENDED COPY command parameter data includes a priority value. This value may be used by the hypervisor to indicate the priority of the offloaded copy operations relative to the priority of other operations being performed on the logical unit. A priority value of zero indicates that the EXTENDED COPY command operations have a higher priority than other operations. A priority value of one indicates that the EXTENDED COPY command operations have the same priority as other operations; with increasing values indicating lower priorities.

The actual impact of the use of the priority field may vary across different storage vendors.

The List Identifier field of the EXTENDED COPY parameter allows the hypervisor to tag each EXTENDED COPY operation with a unique identifier. This is needed if the hypervisor wishes to use the RECEIVE COPY RESULTS command to track the intermediate status of a particular EXTENDED COPY command. This may be useful for long lived copy operations, but only 255 of these identifiers can be used at any point in time. If more than 255 operations are expected, then these unique identifiers cannot be used. In addition, when operations are not expected to be long lived, these unique identifiers are not needed. The discovery process indicates whether this capability is supported by the device or not.

2.6 Discovery

Several methods may be used by a host to determine when the EXTENDED COPY command, COMPARE AND WRITE command, and/or UMAP command are available for use with a particular device.

2.6.1 EXTENDED COPY command discovery

To determine if the EXTENDED COPY command is supported:

A. Issue the INQUIRY command, and examine the 3PC bit in the response. If the 3PC bit is set to one, the device supports the EXTENDED COPY command. The RECEIVE COPY RESULTS command with a service action code of OPERATING PARAMETERS (03h) will return descriptive information about support of the EXTENDED COPY command.
B. Issue a REPORT SUPPORTED OPERATION CODES command (see SPC-4):

a. a generic request may be made to determine information about all supported operation codes (the reporting option set to 000b). This may be useful if a host wishes to determine the supported status of multiple commands with a single request; or
b. a specific request (the reporting option set to 001b) may be made to determine just the supported status of the EXTENDED COPY command. Specifying a reporting option of 001b and a requested operation code of 83h will return the supported status of just the EXTENDED COPY command.

C. Issue an EXTENDED COPY command specifying a parameter list length of zero. Devices that support the EXTENDED COPY command will return status of GOOD (while performing no copy operations), while devices that do not support the EXTENDED COPY command will return a status of CHECK CONDITION, a sense key of ILLEGAL REQUEST, and an ASC/Q of INVALID COMMAND OPERATION CODE.
D. Issue a RECEIVE COPY RESULTS command with a service action code of OPERATING PARAMETERS (03h). If successful, the command will return descriptive information about support of the EXTENDED COPY command by the SCSI Device (such as if unique identifiers are required or not, the maximum number of target descriptors supported, maximum number of segment descriptors supported, maximum transfer size, maximum number of outstanding operations, etc).

2.6.2 COMPARE AND WRITE command discovery

To determine if the COMPARE AND WRITE command is supported:

A. Issue a REPORT SUPPORTED OPERATION CODES command (see SPC-4):
   a. a generic request may be made to determine information about all supported operation codes (the reporting option set to 000b). This may be useful if a host wishes to determine the supported status of multiple commands with a single request; or
   b. a specific request (the reporting option set to 001b) may be made to determine just the supported status of the COMPARE AND WRITE command. Specifying a reporting option of 001b and a requested operation code of 89h will return the supported status of just the COMPARE AND WRITE command.
B. Issue a COMPARE AND WRITE command specifying the number of blocks to transfer as zero. Devices that support the COMPARE AND WRITE command will return status of GOOD (and perform no actual operation), while devices that do not support the COMPARE AND WRITE command will return a status of CHECK CONDITION, a sense key of ILLEGAL REQUEST, and an ASC/Q of INVALID COMMAND OPERATION CODE.

2.6.3 UNMAP operation discovery

To determine if unmap operations are available:

A. Issue a READ CAPACITY (16) command (see SBC-3) and examine the TPE[1] bit in the returned parameter data. A device that does not support READ CAPACITY (16) (i.e., returns a status of CHECK CONDITION, a sense key of ILLEGAL REQUEST, and an ASC/Q of INVALID COMMAND OPERATION CODE) does not support UNMAP operations. If the TPE bit is set to one, then the device supports the UNMAP command, and/or the WRITE SAME(16) command with the unmap bit.
B. The TPRZ[2] bit in the returned parameter data is a useful hint for unmap operations (particularly for unmap operations using the WRITE SAME command method).
C. For devices that support unmap operations (the TPE bit set is to one):
    a. if the device supports the UNMAP command, then the device may or may not support the WRITE SAME(16) command with the unmap bit set to one.
    b. if the device does not support the UNMAP command, then the device does support the WRITE SAME (16) command with the unmap bit set to one.
    c. if the device supports the WRITE SAME(16) command with the unmap bit set to one, then the device may or may not support the UNMAP command.
    d. if the device does not support the WRITE SAME(16) command with the unmap bit set to one, then the device does support the UNMAP command.

2.6.4 UNMAP command discovery

To determine if the UNMAP command is supported:

A. Issue a REPORT SUPPORTED OPERATION CODES command (see SPC-4):
    a. a generic request may be made to determine information about all supported operation codes (the reporting option set to 000b). This may be useful if a host wishes to determine the supported status of multiple commands with a single request; or
    b. a specific request (the reporting option set to 001b) may be made to determine just the supported status of the UNMAP command. Specifying a reporting option of 001b and a requested operation code of 42h will return the supported status of just the UNMAP command.
B. Issue an UNMAP command specifying a parameter list length of zero. Devices that support the UNMAP command will return status of GOOD (while performing no unmap operations), while devices that do not support the UNMAP command will return a status of CHECK CONDITION, a sense key of ILLEGAL REQUEST, and an ASC/Q of INVALID COMMAND OPERATION CODE.
C. Read the Block Limits VPD page (B0h). Devices that support the UNMAP command will report a non-zero value in the maximum unmap lba count field.
D. Read the Thin Provisioning VPD page[3] (B2h). Devices that support the UNMAP command will report the TPU[4] bit set to one.

---

[1] At this time, the name of the TPE bit is being evaluated by the T10 committee, and may be changed. The location however, is bit 7 of byte 14 of the READ CAPACITY (16) returned parameter data.
[2] The location of the TPRZ bit is bit 6 of byte 14 of the READ CAPACITY (16) returned parameter data.

2.6.5 WRITE SAME (16) command discovery

To determine if the WRITE SAME command is supported:

A. Issue a REPORT SUPPORTED OPERATION CODES command (see SPC-4):
   a. a generic request may be made to determine information about all supported operation codes (the reporting option set to 000b). This may be useful if a host wishes to determine the supported status of multiple commands with a single request; or
   b. a specific request (the reporting option set to 001b) may be made to determine just the supported status of the WRITE SAME (16) command. Specifying a reporting option of 001b and a requested operation code of 93h will return the supported status of just the WRITE SAME (16) command.
   c. Examine the returned data and examine the CDB USAGE DATA to determine if the unmap bit is supported by the WRITE SAME(16) command.
B. Read the Thin Provisioning VPD page[5] (B2h). Devices that support the WRITE SAME command with the unmap bit set to one will report the TPWS[6] bit set to one.

Unlike the EXTENDED COPY and UNMAP commands, specifying a number of logical blocks of zero cannot be used to determine if the WRITE SAME command is supported. While a device that does not support the UNMAP command will return a status of CHECK CONDITION, a sense key of ILLEGAL REQUEST, and an ASC/Q of INVALID COMMAND OPERATION CODE, a device that does support the WRITE SAME command treats a logical block count of zero, as a request to write the supplied data from the specified starting logical block address to the end of the media. Therefore, a WRITE SAME command with a number of logical blocks field set to zero cannot be used for discovery.

This use of a zero value to indicate a length of other than zero is unusual to SCSI. For this reason, it is possible devices may have non-compliant behavior related to a zero value in the length field (e,g., the command may return a status of CHECK CONDITION, with a sense key of ILLEGAL REQUEST when a zero transfer length is requested).

Some devices may support the WRITE SAME command only for the purpose of doing unmap operations, and therefore may return a status of CHECK CONDITION, with a sense key of ILLEGAL REQUEST when the data buffer contains any value other than all zeros.

---

[3] At this time, the name of the Thin Provisioning VPD page is being evaluated by the T10 committee, and may be changed. The page is identified by page code B2h.
[4] At this time, the name of the TPU bit is being evaluated by the T10 committee, and may be changed. The location is bit 7 of byte 5 of the VPD page.
[5] At this time, the name of the Thin Provisioning VPD page is being evaluated by the T10 committee, and may be changed. The page is identified by page code B2h
[6] At this time, the name of the TPWS bit is being evaluated by the T10 committee, and may be changed. The location however, is bit 6 of byte 5 of the VPD page.

2.6.6 Logical unit discovery requirement

The host application making use of the EXTENDED COPY command must still be aware of the topology of the storage connectivity. Specifically, the application must know which logical units are within the appropriate domains so a copy manager or copy service may be used to perform the copy operations between the logical units.

2.7 Target Descriptors

2.7.1 Identification target descriptors

The Identification descriptor target descriptor contains a Device Identification designator from VPD page 83h of the logical unit the descriptor is referencing. The designators supplied by the host are copied from INQUIRY VPD page 83h, and should have the association field set to 00b. These descriptors may be used to identify internal logical units, or if the copy manager maintains information about external logical units, these descriptors may also be used to identify external logical units.

2.7.2 Port type descriptors for Fibre Channel Targets

For copy offload to or from a Fibre Channel device, when identification descriptor target descriptors are not used, the target descriptor should use the Fibre Channel N_Port_Name target descriptor (type E0h), or if necessary, the Fibre Channel N_Port_ID with N_Port_Name checking target descriptor (type E2h). Fibre Channel N_Port_ID target descriptors (type E1h) are not recommended.

2.7.3 Port type descriptors for iSCSI Targets

For copy offload to or from iSCSI devices, when identification descriptor target descriptors are not used, the target descriptor may use either the IPv4 target descriptor or the IPv6 target descriptor.

2.7.4 Port type descriptors for SAS targets

For copy offload to or from SAS devices, when identification descriptors target descriptors are not used, the target descriptor should use the SAS Serial SCSI Protocol target descriptor (type E9h).

2.7.5 Copy Service Descriptor

For a copy offload operation using an IP Copy Service, the IP copy service target descriptor should be used.

2.8 Completion

At the completion of an EXTENDED COPY command, the copy manager returns status to the host. If the SCSI Status of the operation is GOOD, then the results of the requested transfer are available for use. If the SCSI Status is not GOOD, then a failure of some kind occurred. Well known SCSI events such as ACA ACTIVE, BUSY, QUEUE FULL, and others should be handled just as they are for other

commands. The status of CHECK CONDITION is returned along with Sense Data to include additional information about the specifics of the failure.

Command processing in the Copy Manager begins by validating portions of the command and parameter data. A failure of these checks (described in SPC4 as "Errors detected before starting processing of the segment descriptors") would mean that no data was transferred. Errors that occur after these checks (described in SPC4 as "Errors detected during processing of segment descriptors") would mean that data may or may not have been transferred; and are reported with the sense key set to COPY ABORTED.

Commands that are terminated with a sense key of ILLEGAL REQUEST and an additional sense code of PARAMETER LIST LENGTH ERROR, TOO MANY TARGET DESCRIPTORS, or TOO MANY SEGMENT DESCRIPTORS must be reformed with a shorter parameter list or fewer descriptors and reissued. The additional sense codes of INVALID FIELD IN CDB and INVALID FIELD IN PARAMETER LIST indicate command processing errors that must be corrected and the command reissued.

Commands that are terminated with a sense key of COPY ABORTED may have transferred some data, and the remainder of the sense data must be examined for the exact error condition. The sense data information field should be examined to determine if residual information has been returned for a partial transfer. In addition, if multiple segment descriptors are used, the third and fourth bytes of the command-specific information field of the sense data may be used to determine the segment number of the segment descriptor being processed at the time the error occurred. See SPC-4 for details.

When a copy operation is aborted (the Sense Key of the Sense Data is COPY ABORTED), the Field Pointer field within the Sense Key specific data of the Sense Data is used to indicate which target descriptor or segment descriptor encountered an error (see SPC4).

3 Security Considerations

Copy managers are initiators of I/O operations within the storage network. As such, the copy manager is able to read from and write to other logical units within the storage network. This ability creates the opportunity for special security considerations. While the SCSI Standard provides the ACCESS CONTROL IN/OUT command, these command are typically not implemented. Basic considerations are listed here to raise awareness of this topic for copy offload implementers.

When using Fibre Channel connectivity, if external copy operations are to be permitted, FC zoning within the fabric, if established, must be defined so that the ports of the storage array containing the copy manager are permitted access to the ports of the storage arrays from which data will be copied (the ports on the source device), and are permitted access to the ports of the storage arrays to which data will be copied (the ports on the destination device). In addition, in arrays that support LUN mapping and masking (the ability to enable or restrict access to a LUN to a subset of the possible initiators), the mapping and masking for source and/or destination logical units must be setup to include the FC initiators of the array that contains the copy manager.

If only internal copy operations are to be utilized, there are no FC zoning considerations. However, the copy manager is responsible to verify that the requesting application client (the host) is permitted read access to the source logical unit and write access to the destination logical unit. EXTENDED COPY commands that attempt access to logical units to which the host does not have appropriate access will be failed by the copy manager.

When using iSCSI, zoning does not apply, but rather, it is replaced by IP networking (including routing) considerations. The copy manager ports must have connectivity and active network paths to any external source and/or external destination devices. In addition, LUN mapping and masking considerations also apply to iSCSI connectivity.

Usage of an IP copy service requires IP networking (including routing) considerations. The copy manager must have connectivity and active network paths to allow communication with the IP copy service. There may also be other vendor unique security considerations involved with these configurations.

Extended Copy commands should be sent to the copy destination (rather than the source). Risks of inappropriate reads (while bad) are less dangerous than risks of inappropriate writes (corruption).

4 Use Sequence

4.1 Sequence Diagram

Figure 7 is a UML sequence diagram showing the data structures and components associated with a copy offload operation. The interactions of the data structures and components are shown, along with the basic sequence of operations involved in a copy offload operation. This figure is intended to convey the interactions with the hypervisor, and the copy related operations performed by the copy manager as a result of processing an EXTENDED COPY command received from the hypervisor. This figure does not show all the necessary transport specific protocol operations required between the copy manager and the source and/or destination devices such as discovery, login, logout, session management, etc.

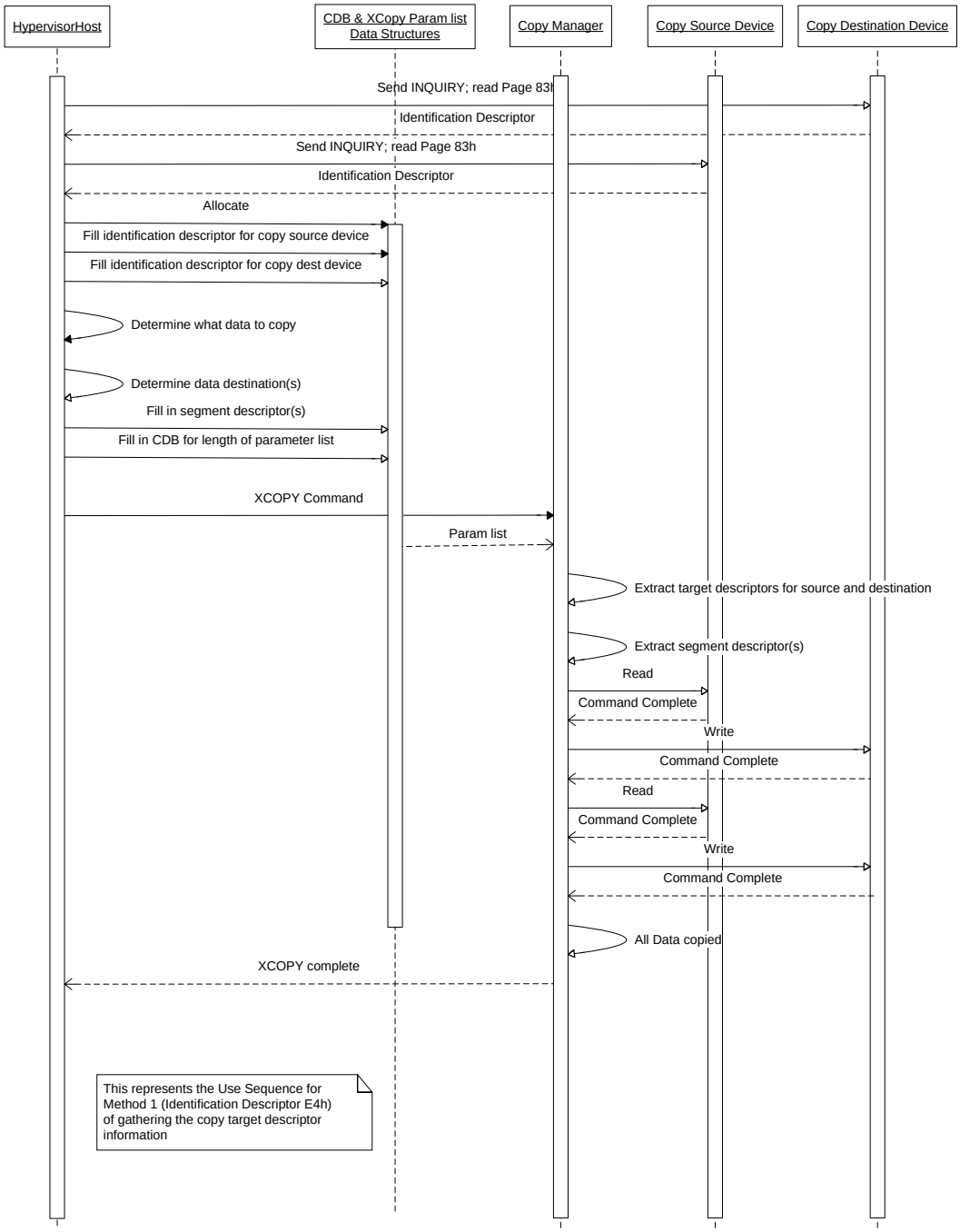

Figure 7 – Copy Manager Sequence Diagram

4.2 Gather Copy Target Descriptor Information

The relative port identifier may be obtained from a port in the SCSI INQUIRY VPD page 83h data that is available from each port of a device. The relative port identifier descriptor has the ASSOCIATION field set to 01b and the DESIGNATOR field set to 4h.

When supplying the relative port identifier in the copy target descriptor, a value of zero indicates that the copy manager is not restricted to using a specific port, and may use any port to access the copy target device.

Method 1 identifies a logical unit using the Device Identification designator and for external logical units, requires that the copy manager maintain information about how to access those external logical units.

Methods 2, 3, 4, and 5 identify a logical unit using its logical unit number and for external logical units, the path to the external logical unit (by specifying the remote port name through which the logical unit is accessed).

Method 6 identifies a logical unit using the Device Identification designator and for external logical units, includes an IP address descriptor that identifies how to access those external logical units.

4.2.1 Method 1 – Identification Descriptor (E4h)

1) Determine the relative port identifier of the initiator Port within the device to be used by the copy manager for access to the copy target device.
2) Examine SCSI INQUIRY VPD page 83h of the copy target device. Search the list of Device Identification designators for a suitable descriptor.
3) Create an Identification descriptor target descriptor using this information.

4.2.2 Method 2 – N_Port_Name target descriptor (E0h)

1) Determine the relative port identifier of the initiator Port within the device to be used by the copy manager for access to the copy target device.
2) Determine the Logical Unit Identifier of the logical unit when accessed through the relative port identifier determined in step 1.
3) Determine the N_Port_Name defined by the PLOGI of the VN_Port for the copy target device used to access the logical unit ID determined in step 2.
4) Create a Fibre Channel N_Port_Name target descriptor using this information.

4.2.3 Method 3 – N_Port_ID target descriptor (E1h – not recommended)

1) Determine the relative port identifier of the initiator Port within the device to be used by the copy manager for access to the copy target device.

2) Determine the Logical Unit Identifier of the logical unit when accessed through the relative port identifier determined in step 1.
3) Determine the D_ID to be used in the transport frames of the copy target device used to access the logical unit ID determined in step 2.
4) Create a Fibre Channel N_Port_ID target descriptor using this information.

4.2.4 Method 4 – N_Port_ID with N_Port_Name checking target descriptor (E2h – non-fabric environments only)

1) Determine the relative port identifier of the initiator Port within the device to be used by the copy manager for access to the copy target device.
2) Determine the Logical Unit Identifier of the logical unit when accessed through the relative port identifier determined in step 1.
3) Determine the D_ID to be used in the transport frames of the target device used to access the logical unit ID determined in step 2.
4) Determine the N_Port_Name defined by the PLOGI of the VN_Port for the copy target device used to access the logical unit ID determined in step 2.
5) Create a Fibre Channel N_Port_ID target descriptor with this information.

4.2.5 Method 5 – iSCSI target descriptors (E5h and EAh)

1) Determine the relative port identifier of the initiator port within the device to be used by the copy manager for access to the copy target device.
2) Determine the Logical unit Identifier of the logical unit on the iSCSI device when accessed through the relative port identifier determined in step 1.
3) Determine the iSCSI target device name.
4) Create the appropriate (IPv4 or IPv6) iSCSI target descriptor with this information.

4.2.6 Method 6 – IP COPY Service descriptor (EBh)

1) Examine SCSI INQUIRY VPD page 85h of the copy target device. Search the list of Network Service Descriptors for a Copy Service (type 06h).
2) Examine SCSI INQUIRY VPD page 83h of the copy target device. Search the list of Device Identification designators for a suitable descriptor.
3) Create an IP Copy Service target descriptor with this information.

4.2.7 Method 7 – SAS Serial SCSI Protocol target descriptor (E9h)

1) Determine the relative port identifier of the initiator port within the device to be used by the copy manager for acess to the copy target device.
2) Determine the Logical unit Identifier of the logical unit on the SAS device when accessed through the relative port identifier determined in step 1.
3) Determine the SAS Address of the target device.

4) Create a SAS Serial SCSI Protocol target descriptor with this information.

4.3 Gather Segment Descriptor Information (02h)

1) Determine the starting LBA on the copy source device to begin the read operation.
2) Determine the starting LBA on the copy destination device to begin the write operation.
3) Determine the length of the copy (the number of logical blocks to copy). This value is used both for the number of blocks to read and the number of blocks to write.

4.4 Build the Extended Copy Parameter list

1) Allocate space for parameter list
2) Fill in target descriptor for copy source device
3) Fill in target descriptor for copy destination device (if different than the copy source)
4) Fill in segment descriptor(s):
    a) index of the copy source device target descriptor
    b) index of the copy destination device target descriptor
    c) number of logical blocks to read and write
    d) starting LBA on the copy source device
    e) starting LBA on the copy destination device
5) Fill in the CDB with the length of the parameter list
6) Send the command to the copy destination device

4.5 Copying a Virtual Disk

The EXTENDED COPY command may be used to create a copy of a virtual disk with no burden on the server/hypervisor to optimize hypervisor performance. This may be used to provision a new virtual machine. For example, an administrator may wish to copy a known good "golden image".

The steps above are used to build the appropriate copy parameter list to copy a virtual disk. In the following example (see Figure 8), a virtual disk occupies 3 segments of a container that resides on 2 logical units (segment 1 and 2 reside on logical unit number 1, while segment 3 resides on logical unit number 3). This virtual disk is to be copied to a container that resides on logical unit number 6 and logical unit number 9.

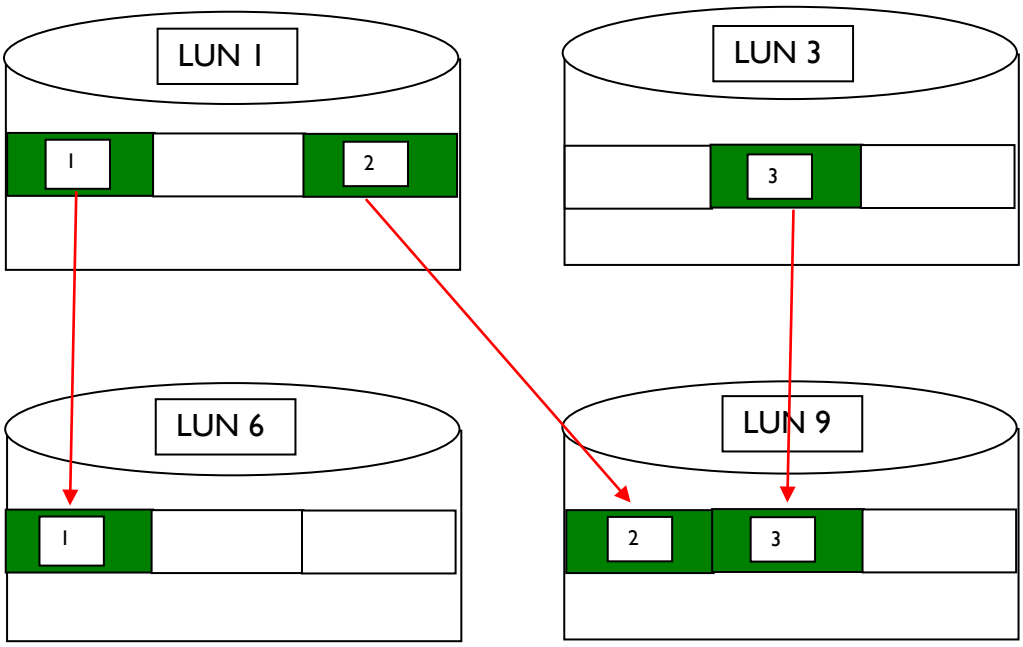

Figure 8 – Example Copy of a Virtual Disk in a Container

A single EXTENDED COPY command parameter list may be built (see Table 1) that contains multiple segment descriptors. The segment descriptor for segment 1 points to the source target descriptor for LUN 1 and the destination target descriptor for LUN 6. The segment descriptor for segment 2 points to the source target descriptor for LUN 1 and the destination target descriptor for LUN 9. The segment descriptor for segment 3 points to the source target descriptor for LUN 3 and the destination target descriptor for LUN 9. Each segment descriptor contains two pointers to copy target descriptors (one for the source logical unit, and one for the destination logical unit). In addition, the starting LBA on the source logical unit, the starting LBA on the destination logical unit, and the number of logical blocks to transfer are included in the segment descriptor.

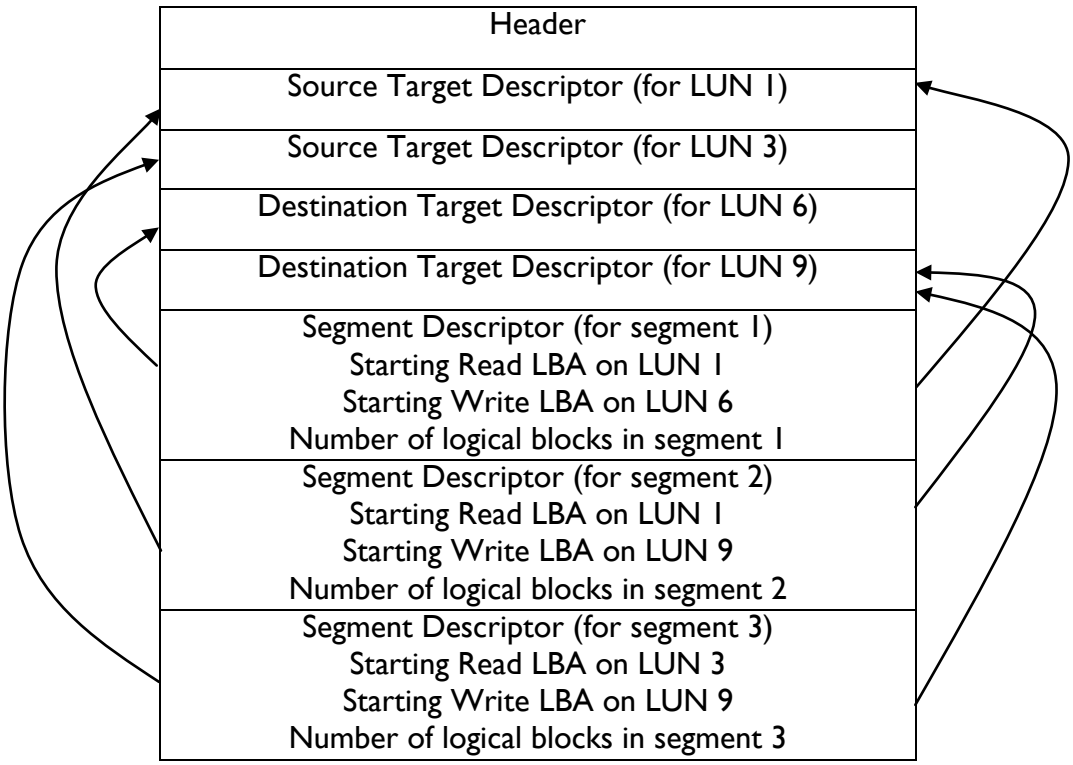

Table 1 – Single Parameter List Example

Multiple EXTENDED COPY command parameter lists may also be built where a unique EXTENDED COPY command is sent to the copy manager that describes only one portion of the copy offload operation per EXTENDED COPY command. This is shown in Table 2, Table 3, and Table 4.

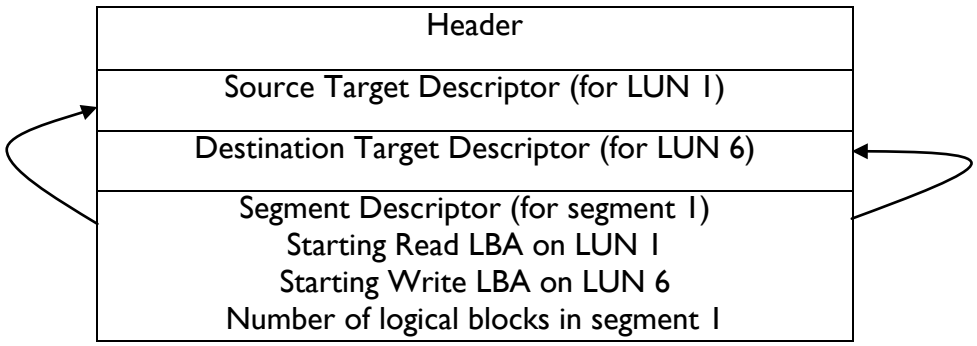

Table 2 – Parameter List to copy Segment 1

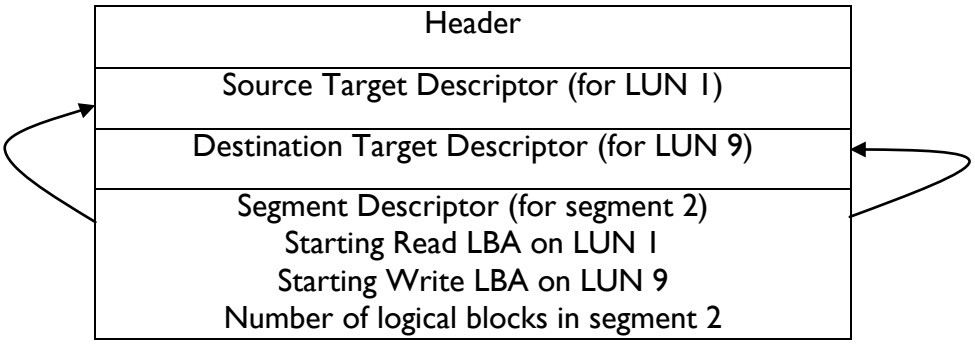

Table 3 – Parameter List to copy Segment 2

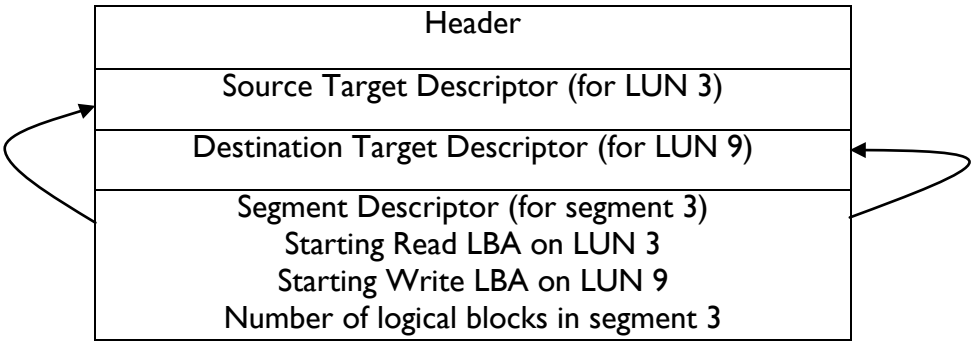

Table 4 – Parameter List to copy Segment 3

In general, the more information contained within a single EXTENDED COPY command, the more efficient and higher performance will be the copy offload operation. However, the more information contained within a single EXTENDED COPY command, the more complex the processing, and error tracking should a failure occur. This tradeoff must be made by the application performing the copy offload operation.

4.6 Copying an Entire Logical Unit

The EXTENDED COPY command may be used to copy an entire logical unit with no burden on the server/hypervisor to optimize hypervisor performance. This operation may be used, for example, when a new virtual infrastructure is provisioned for a new office, business unit, user group, or test/development environment.

This sequence may be done via a single EXTENDED COPY command, where the starting LBA is set to zero, and the length is set to the total number of logical blocks on the logical unit, or this copy may be done using a single EXTENDED COPY command that contains multiple segment descriptors, where each segment is described independently. Alternatively, multiple independent EXTENDED COPY commands may be used; each copying just a segment of the blocks on the logical unit.

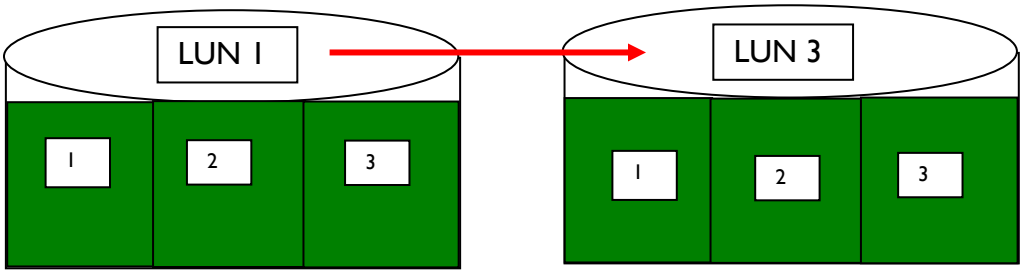

Figure 9 – Example of Copying an Entire Logical Unit

In this example, the entire capacity of the logical unit is copied from LUN 1 to LUN 3 (see Figure 9). A single EXTENDED COPY command may be used (see Table 5) that describes the entire capacity of the logical unit. A multi-segment descriptor example is shown in Table 6. This copy offload could also be performed using multiple EXTENDED COPY commands each containing a single segment descriptor.

Header
Source Target Descriptor (for LUN 1)
Destination Target Descriptor (for LUN 3)
Segment Descriptor (for the entire LUN) Starting Read LBA on LUN 1 (LBA 0) Starting Write LBA on LUN 3 (LBA 0) Number of logical blocks on the LUN

Table 5 – Whole LUN copy (single segment descriptor)

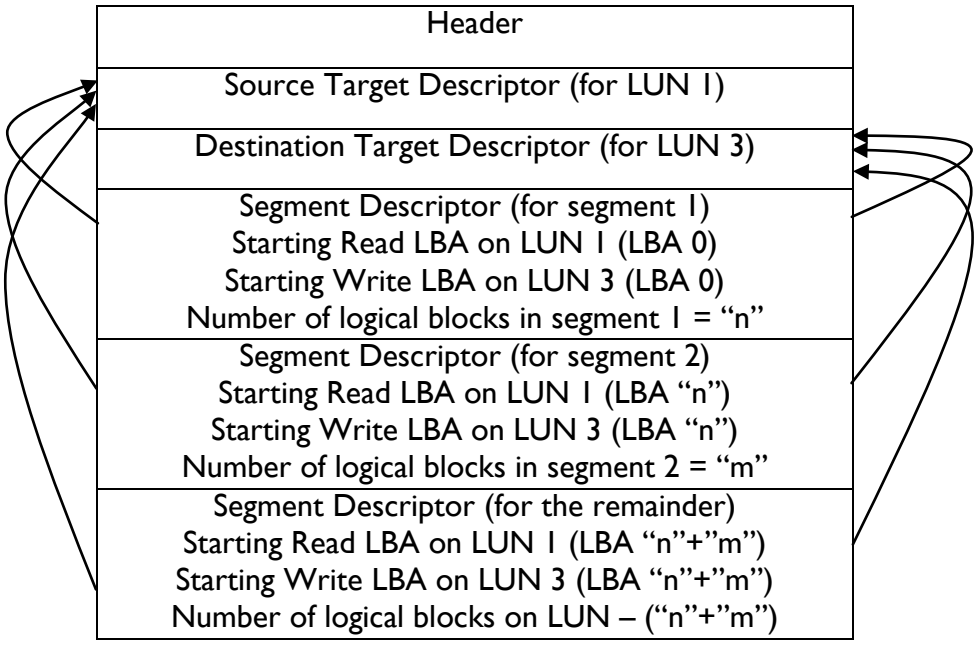

Table 6 – Whole LUN copy (multiple segment descriptors)

4.6.1 CDB and Parameter Data Details

Table 7 shows a more detailed example CDB used with the parameter data shown in Table 8. This example shows a copy offload from an internal logical unit to another internal logical unit where one segment of data is copied (note that the CDB contents shown here are specific to this example, and not to be construed as generic). To copy an entire logical unit, the starting LBA of the source device and the starting LBA of the destination device would both be set to zero, and the number of blocks field would be set to the number of blocks on the logical unit.

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (83h)							
1-9	All Zeros (0h)							
10-12	Zeros (000000h)							
13	Parameter Length (112 / 70h)							
14	Zero (0h)							
15	Control Byte							

Table 7 – Detailed CDB

If the application does not wish to allow the length of time that such a large transfer may require, the operation should be broken into multiple EXTENDED COPY commands with shorter transfers

specified in the parameter data (segment descriptor) of each command. The parameter data shown in table 8 is specific to this example, and should not be construed as generic.

Bit / Byte	7	6	5	4	3	2	1	0
0								
1								
2-3	Target descriptors length (64 / 0040h)							
4-7	All Zeros (00000000h)							
8-11	Segment descriptors length (32 (0020h)							
12-15	All Zeros (00000000h)							
Target Descriptor 0								
16	Descriptor type (E4h)							
17	Zero (00h)							
18-19	Relative port identifier (0000h)							
20-n	Device Identification designator from source device page 83h							
n+1-47	Zeros (00h)							
Target Descriptor 1								
48	Descriptor type (E4h)							
49	Zero (00h)							
50-51	Relative port identifier (0000h)							
52-n	Device Identification designator from destination device page 83h							
n+1-79	Zeros (00h)							
Segment Descriptor 0								
80	Descriptor type (02h)							
81	Zero (00h)							
82-83	Length (0018h)							
84-85	Source target descriptor index (0000h)							
86-87	Destination target descriptor index (0001h)							
88-89	Zero (0000h)							
90-91	Number of Blocks to transfer (xxh)							
92-99	Starting LBA on source device (xxxxxxxxxxxxxxxxh)							
100-108	Starting LBA on destination device (xxxxxxxxxxxxxxxxh)							
109-111	Zeros (000000h)							

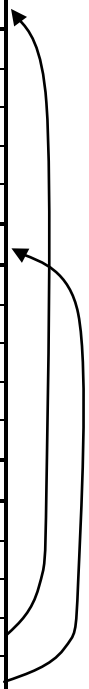

Table 8 – Detailed Parameter Data Example

4.7 Copying a Virtual Machine

A common practice to generate a new virtual machine is to create a golden image, then copy the golden image to provision the new virtual machine. The virtual machine generated with this procedure

is expected to be ready for deployment in production with the least possible configuration steps. The virtual machine copy operation may also be used as a possible option for back up.

To copy a virtual machine, all of the information (e.g., metadata, virtual disks, if any) associated with that virtual machine must be copied. All meta-data and virtual disks data are stored on one or more logical units. On each logical unit, the data are stored in one or more ranges of contiguous LBAs. The virtual machine copy may be composed of multiple copy operations, as required to transfer all corresponding data segments.

The hypervisor, that has knowledge of the data layout, should identify the LBA addresses and ranges on the logical unit, The hypervisor then constructs the appropriate requests specifying the appropriate LBAs to be copied, and passes the request(s) to the underlying storage.

This may be done via one or more EXTENDED COPY command(s), the choice being dependent on performance considerations and complexity. Each EXTENDED COPY command may contain multiple independent segment descriptors where each segment descriptor may be associated with the metadata or with a virtual disk associated with the virtual machine.

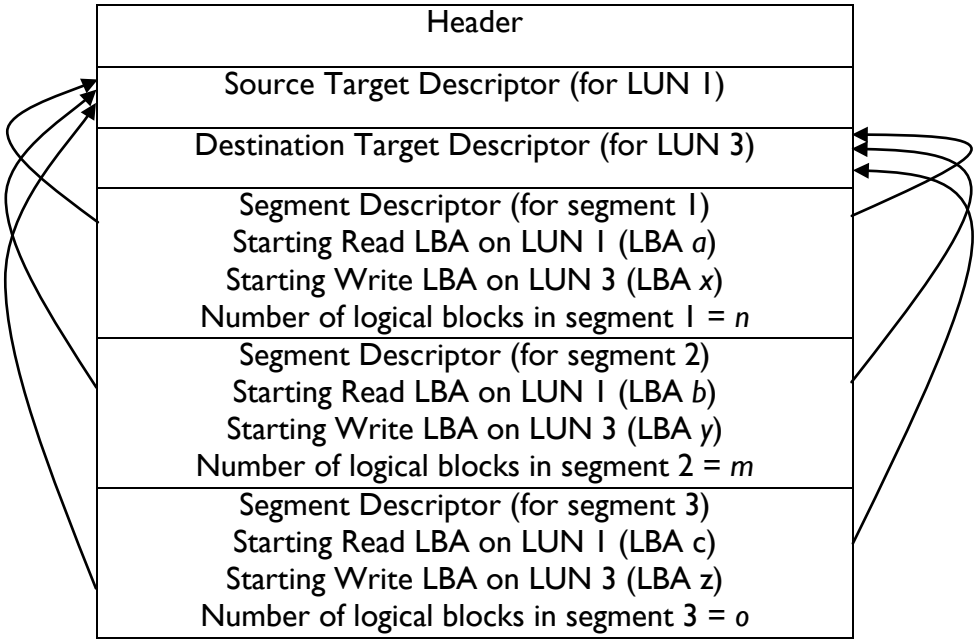

Table 9 – Copy of multiple segments of a LUN

4.8 Virtual Machine Storage Migration

This use sequence enables virtual machine storage migration by providing an offload function to copy all of the storage associated with the virtual machine from one array to another. The basic operation is similar to the Copying a Virtual Machine sequence above, except that the value in the destination copy descriptor contains an identifier of a destination that is in a different array. This use sequence

may also be performed manually (through means outside the scope of this document) by leveraging an array's mirroring capabilities.

First the entire virtual machine must be copied from the source target array to destination target array. The location of the Copy Manager is immaterial: it may be external to both arrays or contained within either the Source or Destination Array.

To copy an entire virtual machine, multiple virtual disks or entire logical units may need to be copied. The hypervisor, that has knowledge of the data layout, should identify the LBA addresses, ranges, and port/array identifiers for the storage. The hypervisor then constructs the appropriate requests specifying the appropriate LBAs to be copied, and passes the request(s) to the underlying storage.

The requests are passed to the underlying storage system via one or more EXTENDED COPY command(s), the choice being dependant on performance and complexity considerations. Each EXTENDED COPY command may contain multiple independent segment descriptors, where each segment descriptor may be associated with the metadata or with a virtual disk associated with the virtual machine.

After the virtual machines have been copied, the previously used space on the source storage may now be reclaimed as described in section 4.10 below.

The operation details in section 1 describe how to construct the Source Target Descriptors and Destination Target Descriptors for the EXTENDED COPY command, such as in the example below:

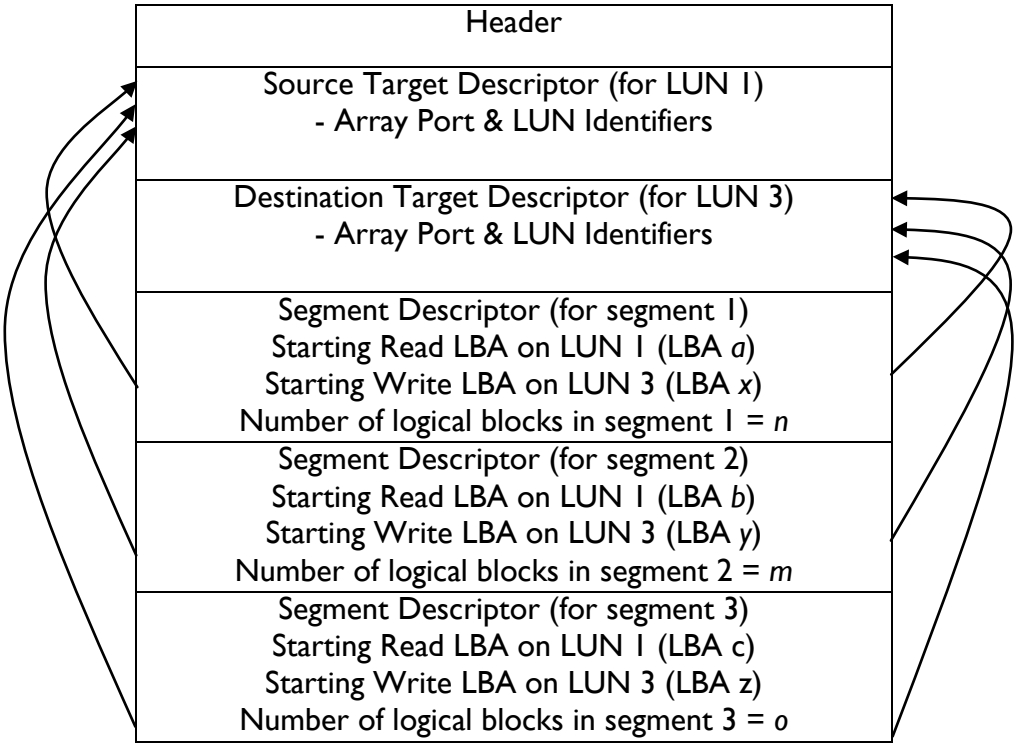

Table 8 – Copy of multiple segments of a LUN across arrays.

4.9 Dividing Virtual Machines

Multiple virtual machines may be grouped together because they are associated with the same application, the same set of users, the same security context, or require a specific service level. For these and other reasons, the administrator may need to simultaneously relocate, a subset of virtual machines from one storage location to another using a single procedure. When this operation is performed, virtual machines may be viewed to be in one of two groups: the one group that is left in the original location, or a second group that is relocated to a new location.

Virtual machine division may be performed between storage belonging to the same storage array or across different storage arrays.

This procedure is actually composed of multiple virtual machine storage migration operations, as described in section 4.8. After the virtual machines have been migrated, the unused space on the source storage may be reclaimed as described in section 4.10 below. What is unique in this procedure, with respect to the previous cases, is the option to create EXTENDED COPY segment descriptors combining the data of multiple virtual machines.

As described in the previous scenarios, the hypervisor constructs the appropriate requests specifying the appropriate LBAs to be copied, and passes the request(s) to the Copy Manager, via one or more

EXTENDED COPY command(s), the choice mostly depending on performance and complexity considerations. Each EXTENDED COPY command may contain multiple independent segment descriptors; each segment descriptor is associated with the data of one or more virtual machines.

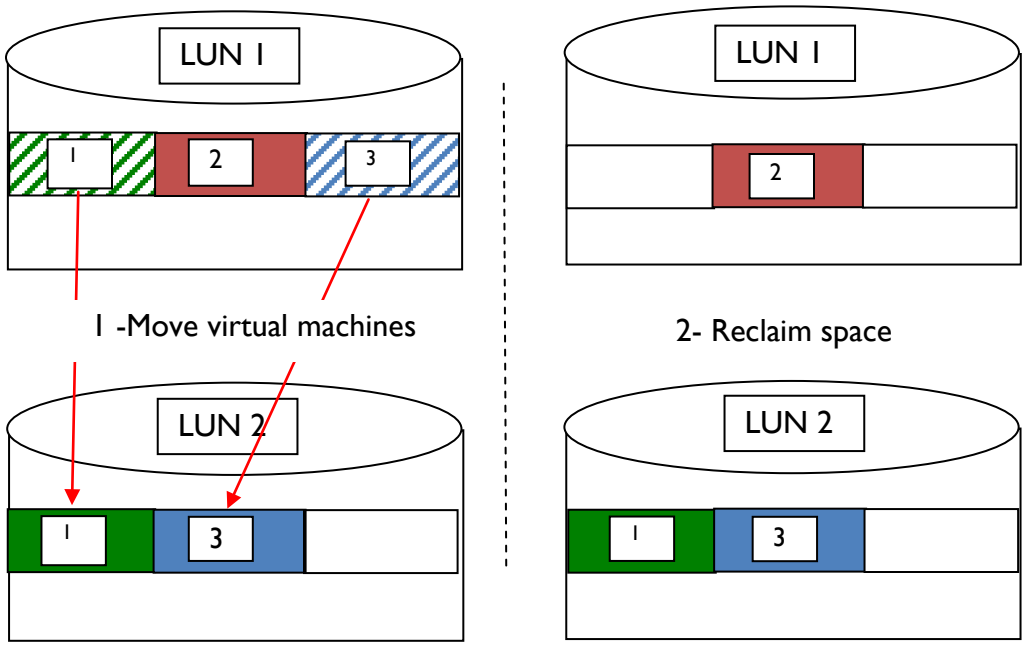

Figure 10 – Simple example of dividing virtual machine

Figure 9 shows a simple example of dividing virtual machines; table 9 shows the EXTENDED COPY segment descriptors contained in a single EXTENDED COPY command to relocate both virtual machines in a single operation. In this simple example the blocks assigned to each virtual machines are contiguous, so two segment descriptors are sufficient to divide two virtual machines from a third one.

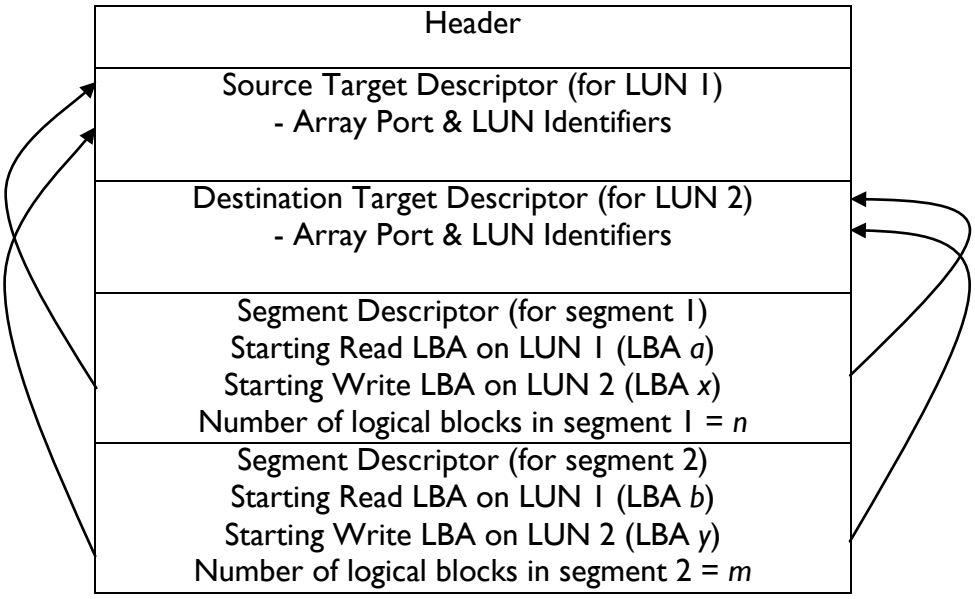

Table9 – Example of segment descriptors for dividing virtual machines

4.10 Reclaiming Unused Space

4.10.1 Reclaiming space overview

When logical blocks no longer contain useful data, storage devices may gain increased storage efficiency if the storage device is notified. SCSI devices may use unmap operations to perform this notification. ATA devices may use the TRIM function of the DATA SET MANAGEMENT command to perform this notification.

4.10.2 Reclaiming space on virtual disk deletion

When a virtual disk is deleted, the space may eventually be reused by the host system, however, increased storage efficiency could be gained if the storage device were told when the space is being reclaimed.

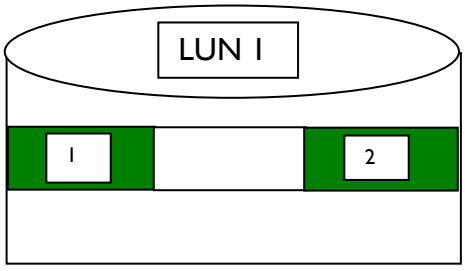

Figure 11 – Reclaiming Blocks

There are two SCSI methods that may be used to notify the storage device when data is deleted, and that space may be reclaimed:

A. The WRITE SAME(16) command (see SBC-3):
   This command provides a method to inform the storage device when data in a single set of contiguous logical blocks is no longer needed and may be reclaimed to improve storage efficiency. In Figure 11, to reclaim blocks solely in segment 1 and segment 2 would require issuing two commands (one command for segment 1 and a second command for segment 2). To cause actual storage space to be reclaimed, the UNMAP bit in the CDB must be set to one. In addition, a data buffer must be supplied to satisfy the WRITE portion of this command. While that buffer would typically contain all zeros, the actual data required to allow the device to reclaim the now unused space is vendor specific (for example, some SSD type devices require a buffer of all ones). If the TPRZ bit in the returned parameter data of the READ CAPACITY (16) is set to one, then a buffer of zeros is required to reclaim space. If the TPRZ bit is cleared to zero, then there is no method available to determine the necessary non-zero vendor specific data pattern required to reclaim space.
B. The UNMAP command (see SBC-3):
   This command provides a method to inform the storage device when data in multiple sets of logical blocks are no longer needed and may be reclaimed to improve storage efficiency. In Figure 11, to reclaim blocks in segment 1 and segment 2 would require just 1 command (with a parameter list that contained 2 UNMAP block descriptors (one descriptor for segment 1 and one descriptor for segment 2). There is no write data supplied with this command.

If the storage device is an ATA device, the notification to the device is done using the TRIM function of the DATA SET MANAGEMENT command.

In addition it is recommended that hosts honor the granularity and offset values specified in the Block Limits VPD Page (see SBC-3) to achieve optimal performance for UNMAP operations.

4.10.3 Reclaiming Unused Space on behalf of a guest

When a guest operating system deletes a file or otherwise no longer needs to retain a particular set of data, it may choose to use the WRITE SAME (16) command with the UNMAP bit set or the UNMAP command to indicate to a virtual disk that reclaim operations may be performed.

To support enhanced storage optimizations throughout the entire virtual storage stack and further increased storage efficiency, the hypervisor should translate the LBA addresses in such requests to the appropriate LBAs on the logical unit and pass those requests to the underlying storage for operation. In some cases, command translation may also be required (e.g., if a guest uses the WRITE SAME (16) command, but the underlying storage supports only the UNMAP command; or the hypervisor presents ATA storage to the guest, but the underlying storage is SCSI).

4.11 Atomic operation offload

Some hypervisors use disk based locking structures that require atomic access. To perform such operations, hosts have relied on a sequence of SCSI RESERVE, SCSI READ, SCSI WRITE, SCSI RELEASE commands to create non-interruptible sequences. This sequence is time consuming, requires host cpu cycles (to compare/modify the data before rewriting), and exclusively locks the entire logical unit preventing any other access. To improve scaling (by not exclusively locking the logical unit), and offload the compare operation, the COMPARE AND WRITE command can be used.

The COMPARE AND WRITE command provides an atomic read, compare, and write function for data on the disk. The host must supply 2 data buffers (one for the compare operation, and the second for the write operation). The device reads data from the specified location, and compares the data with the first set of data. If that data matches, then the device will transfer the second set of data and write that data to the specified locations. In this way, an atomic read/modify/write can be accomplished.

Use of this command also requires that the host not use other write commands to access the same logical blocks accessed using the COMPARE AND WRITE command.

Due to operational periods when older hypervisor versions (using the old RESERVE / RELEASE method) and newer hypervisor versions (using the COMPARE AND WRITE method) must coexist, the device server must guarantee that the RESERVE command does not return success if any COMPARE AND WRITE commands remain in the queue for possible execution after the completion of the RESERVE command is sent (i.e., the status for the RESERVE command must not be sent until after the status for all COMPARE AND WRITE commands in the queue has been sent).

5 References

SPC-4 – SCSI Primary Commands – 4
http://www.t10.org/cgi-bin/ac.pl?t=f&f=spc4r24.pdf

SBC-3 – SCSI Block Commands – 3
http://www.t10.org/cgi-bin/ac.pl?t=f&f=sbc3r22.pdf

ACS-2 – ATA/ATAPI Command Set – 2
http://www.t13.org/Documents/UploadedDocuments/docs2009/d2015r2-ATAATAPI_Command_set_-_2_ACS-2.pdf

UML reference
http://www.omg.org/technology/documents/formal/uml.htm