



Hypervisor Storage Interfaces for Storage Optimization White Paper

October 2009

“Publication of this *Working Draft* for review and comment has been approved by the Hypervisor Storage Interfaces (HSI) TWG. This draft represents a “best effort” attempt by the Hypervisor Storage Interfaces (HSI) TWG to reach preliminary consensus, and it may be updated, replaced, or made obsolete at any time. This document should not be used as reference material or cited as other than a “work in progress. Suggestions for revision should be directed to <http://www.sina.org/feedback/>.”

DRAFT – Rev 0.5a

Copy Offload Hypervisor Storage Interfaces

The SNIA hereby grants permission for individuals to use this document for personal use only, and for corporations and other business entities to use this document for internal use only (including internal copying, distribution, and display) provided that:

1. Any text, diagram, chart, table or definition reproduced must be reproduced in its entirety with no alteration, and,
2. Any document, printed or electronic, in which material from this document (or any portion hereof) is reproduced must acknowledge the SNIA copyright on that material, and must credit the SNIA for granting permission for its reuse.

Other than as explicitly provided above, you may not make any commercial use of this document, sell any or this entire document, or distribute this document to third parties. All rights not explicitly granted are expressly reserved to SNIA.

Permission to use this document for purposes other than those enumerated above may be requested by e-mailing tcmd@snia.org please include the identity of the requesting individual and/or company and a brief description of the purpose, nature, and scope of the requested use.

Copyright © 2009 Storage Networking Industry Association.

Table of Contents

About the SNIA	1
Preface	1
About the SNIA HSI Technical Working Group.....	1
Acknowledgements.....	1
1 Hypervisor Storage Interfaces	2
1.1 Terms and Concepts.....	2
1.2 Hypervisor Storage	2
1.3 Hypervisor Storage Interfaces (HSI).....	2
1.4 Offload Use Cases.....	3
2 The Extended Copy Command	3
2.1 Terms and Concepts.....	3
2.2 Operations	4
2.2.1 External Copy Operation	5
2.2.2 Internal to External Copy Operation	6
2.2.3 External to Internal Copy Operation	8
2.2.4 Internal Copy Operation	9
2.2.5 IP Copy Service Operation	11
2.3 Reservation Considerations	12
2.4 General Considerations	12
2.5 Discovery.....	13
2.5.1 XCOPY command discovery.....	13
2.5.2 UNMAP operation discovery	13
2.5.3 UNMAP command discovery	14
2.5.4 WRITE SAME (16) command discovery	14
2.5.5 Logical unit discovery requirement.....	15
2.6 Target Descriptors.....	15
2.6.1 Identification target Descriptors.....	15
2.6.2 Fibre Channel Target Descriptors.....	15
2.6.3 iSCSI Target Descriptors.....	15
2.6.4 Copy Service Descriptor	15
2.7 Completion	15
3 Security Considerations	16
4 Use Sequence	17
4.1 Sequence Diagram.....	17
4.2 Gather Copy Target Descriptor Information	19
4.2.1 Method 1 – Identification Descriptor (E4h).....	19
4.2.2 Method 2 – N_Port_Name (E0h).....	19
4.2.3 Method 3 – N_Port_ID (E1h - uncommon)	19
4.2.4 Method 4 – N_Port_ID with N_Port_Name checking (E2h - uncommon).....	19
4.2.5 Method 5 – iSCSI name descriptors (E5h and EAh).....	20

Copy Offload Hypervisor Storage Interfaces

4.2.6	COPY Service descriptor (E5h and EAh).....	20
4.3	Gather Segment Descriptor Information (02h).....	20
4.4	Build the Extended Copy Parameter list.....	20
4.5	Copying a Virtual Disk.....	21
4.6	Copying an Entire Logical Unit.....	23
4.6.1	CDB and Parameter Data Details.....	25
4.7	Copying a Virtual Machine.....	26
4.8	Virtual Machine Storage Migration.....	27
4.9	Dividing Virtual Machines.....	29
4.10	Reclaiming Unused Space.....	31
4.11	Reclaiming Unused Space on behalf of a guest	32
5	References	32

List of Tables

Table 1 - Single Parameter List Example.....	22
Table 2 – Parameter List to copy Segment 1	22
Table 3 – Parameter List to copy Segment 2.....	23
Table 4 – Parameter List to copy Segment 3.....	23
Table 5 – Whole LUN copy (single segment descriptor)	24
Table 6 – Whole LUN copy (multiple segment descriptors).....	25
Table 7 - Detailed CDB	25
Table 8 - Detailed Parameter Data	26
Table 9 – Copy of multiple segments of a LUN.....	27

List of Figures

Figure 1 – External Copy Offload Operation	5
Figure 2 – Internal to External Copy Offload Operation.....	6
Figure 3 – External to internal Copy Offload Operation.....	8
Figure 4 – Internal Copy Offload Operation.....	9
Figure 5 – IP Copy Service Operation.....	11
Figure 6 - Copy Manager Sequence Diagram.....	18
Figure 7 - Example Copy of a Virtual Disk in a Container.....	21
Figure 8 – Example of Copying an Entire Logical Unit.....	24
Figure 9 – Simple example of dividing virtual machine.....	29
Figure 10 – Freeing Blocks	31

Copy Offload Hypervisor Storage Interfaces

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 paper is aimed at Hypervisor software providers and providers of storage intended for use with Hypervisor environments. It documents best practices for using copy offload and space reclamation technology in a blocks environment. This includes using the SCSI Extended Copy command for providing copy offload services 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

Copy Offload Hypervisor Storage Interfaces

I Hypervisor Storage Interfaces

I.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 paper, 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.

I.2 Hypervisor Storage

Hypervisor Storage refers to a storage abstraction layer that is used by a Hypervisor to present “Virtual Disks” to Virtual Machines. It is an intermediate storage abstraction layer located between the guest operating system storage stack running in a VM and the actual physical storage devices. From the perspective of a guest operating system executing within a VM, a virtual disk is generally indistinguishable from a physical disk. These Virtual Disks are in turn comprised of a collection of blocks that are located on physical storage. 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 actual physical storage devices 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 physical blocks are stored. In addition, the guest operating system has no visibility into the physical storage array that hosts a virtual disk.

I.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.

Copy Offload Hypervisor Storage Interfaces

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 The Extended Copy Command

2.1 Terms and Concepts

Copy Manager – The copy manager is a SCSI application client that is contained within the SCSI device server. The copy manager receives the copy request and performs the necessary operations to execute a copy operation.

Copy Service – A copy engine is outside the SCSI device server and performs the necessary operations to transfer data through an IP network.

Copy Source Device – The copy source device is the device from which data is read.

Copy Offload Hypervisor Storage Interfaces

Copy Destination Device – The copy destination device is the device to which the data is written.

Copy Target Device – A copy target device is either a copy source device or a copy destination device.

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

Segment Descriptor – A segment descriptor is a data structure that describes a segment (as a starting logical block address and length) on a copy target device and an indication of the copy target device on which that segment resides.

Relative Port ID – A Relative Port Id is used to identify which port on the device that contains the copy manager that received a SCSI XCOPY command should be used by the copy manager to access the device which is described by the target descriptor.

2.2 Operations

The SCSI Extended Copy command may be used to provide copy offload services within a SCSI domain. Figure 1 shows an example of a fully external copy offload operation. Figure 2 shows an example of a copy offload from a logical unit in the same array as the copy manager (where the XCOPY command is received) to a logical unit in an array that is external to the copy manager. Figure 3 shows an example of a copy offload operation from a logical unit external to the copy manager to a logical unit in the same array as the copy manager (where the XCOPY command is received). Figure 4 shows a copy offload between logical units within the same array as the copy manager. Figure 5 shows a copy offload using an IP copy service to perform the transfers over an IP network connected logical unit to a logical unit in the same array as the copy manager (where the XCOPY command is received).

Note: the terms internal and external are relative to the copy manager. If a logical unit is contained in the same SCSI Device that contains the copy manager, that logical unit is referred to as “internal”. If a logical unit is not contained in the same SCSI Device that contains the copy manager, that logical unit is referred to as “external”.

Copy Offload Hypervisor Storage Interfaces

2.2.1 External Copy Operation

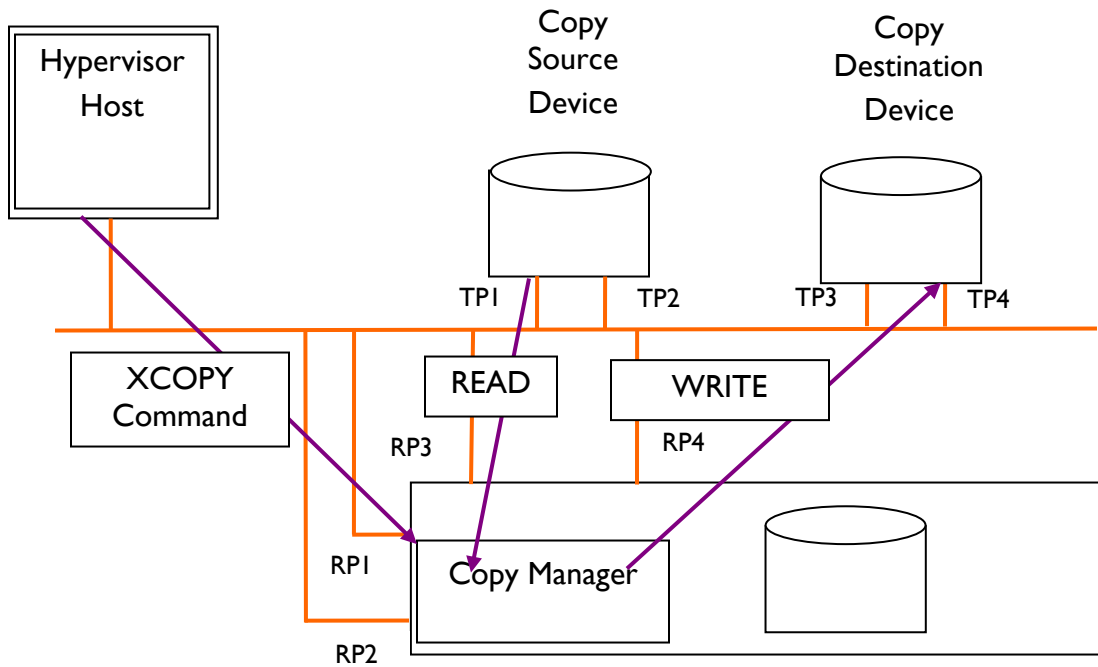


Figure 1 – External Copy Offload Operation

When a host wants to copy blocks from an external source device to an external destination device (see Figure 1) 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. A source target descriptor containing:
 - a. the relative target port over which to issue the READ command (RP3);
 - b. the port ID or name 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.
- B. A destination target descriptor containing:
 - a. the relative target port over which to issue the write command (RP4);
 - b. the port ID or name 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 written.
- C. A segment descriptor 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.

Then the XCOPY command is sent to the logical unit containing the copy manager (via RP1). The copy manager then connects through RP3 to TP1 and sends a READ command (using the starting LBA

Copy Offload Hypervisor Storage Interfaces

and length from the segment descriptor) to the specified logical unit. The copy manager then connects through RP4 to TP4 and sends a WRITE command (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.

2.2.2 Internal to External Copy Operation

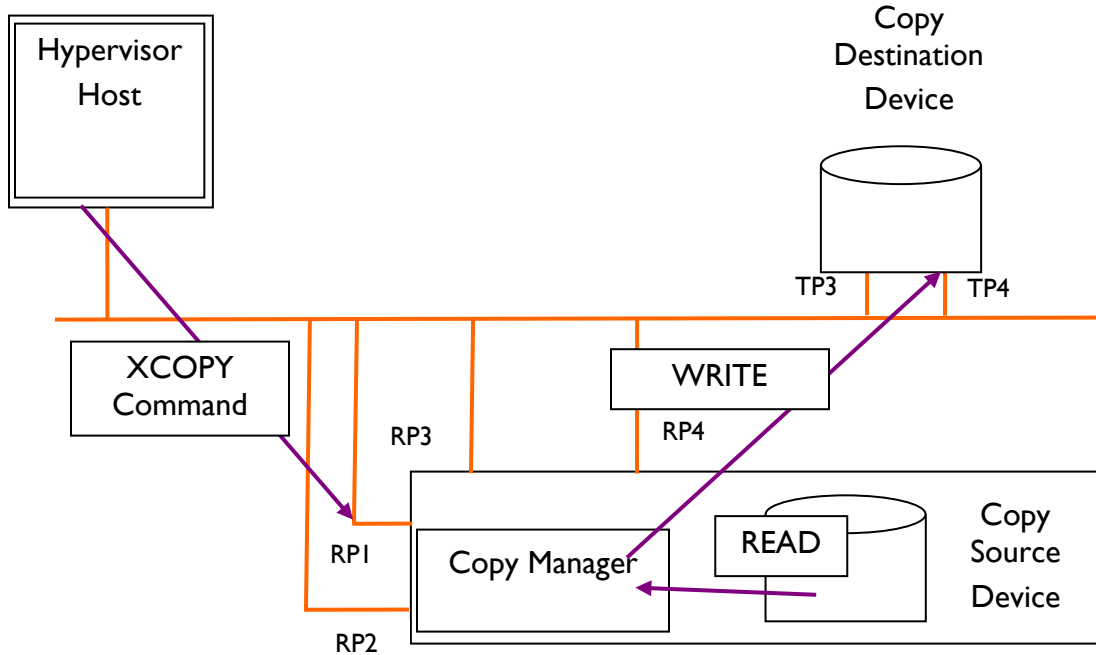


Figure 2 – Internal to External Copy Offload Operation

When a host wants to copy blocks from an internal source device to an external destination device (see Figure 2) 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. A source target descriptor containing:
 - either
 - a. a relative target port of zero;
 - b. the port ID or name of a port on the source device which 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 target port of zero;

Copy Offload Hypervisor Storage Interfaces

- b. an identification descriptor containing the serial number of the logical unit from which to read the data.
- B. A destination target descriptor containing:
 - either
 - a. the relative target port over which to issue the write command (RP4);
 - b. the port ID or name 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 target port of zero;
 - b. an identification descriptor containing the serial number of the logical unit to which the data is to be written.
- C. A segment descriptor 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.

Then the XCOPY command is sent to the logical unit containing the copy manager (via RPI). 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 a WRITE command (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.

Copy Offload Hypervisor Storage Interfaces

2.2.3 External to Internal Copy Operation

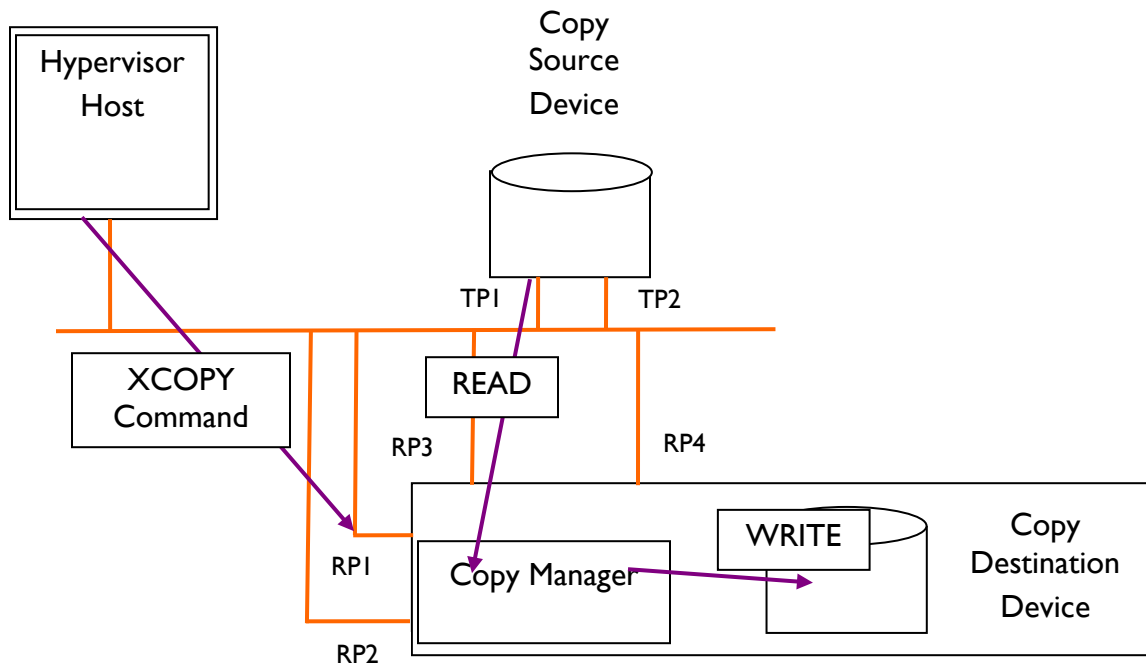


Figure 3 – External to internal Copy Offload Operation

When a host wants to copy blocks from an external source device to an internal destination device (see Figure 3) these procedures are used.

<Ed note: Is this description an appropriate level of detail? Are there too many choices, not enough of the possible options described, or just right? How does a hypervisor decide between the “either/or” list that follows – vendor unique is not the right answer.>

- A. A source target descriptor containing:
 - either
 - a. the relative target port over which to issue the READ command (RP3);
 - b. the port ID or name of the port on the source device to process the read command (TPI); and
 - c. the logical unit number of the logical unit from which to read the data.
 - or
 - a. a relative target port of zero;
 - b. an identification descriptor containing the serial number of the logical unit from which to read the data.
- B. A destination target descriptor containing:
 - either
 - a. a relative target port of zero;

Copy Offload Hypervisor Storage Interfaces

- b. the port ID or name of a port on the destination device which 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 target port of zero;
 - b. an identification descriptor containing the serial number of the logical unit to which the data is to be written.
- C. A segment descriptor 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.

Then the XCOPY command is sent to the logical unit containing the copy manager (via RP1). The copy manager then connects through RP3 to TPI and sends a READ command (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.

2.2.4 Internal Copy Operation

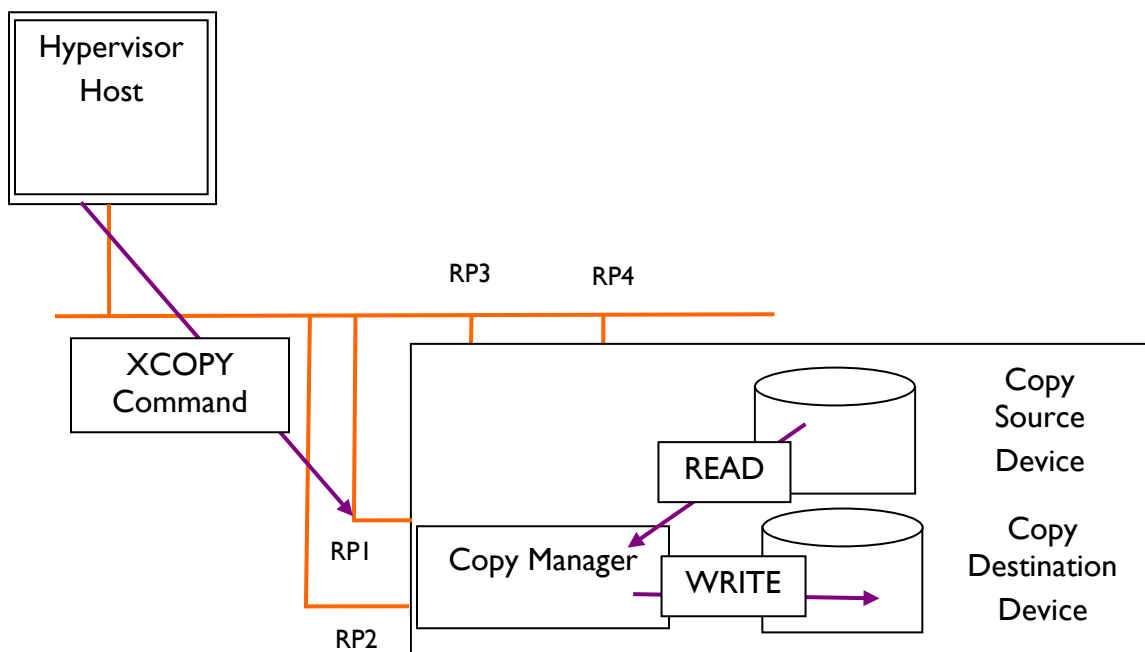


Figure 4 – Internal Copy Offload Operation

Copy Offload Hypervisor Storage Interfaces

When a host wants to copy blocks from an internal source logical unit to another logical unit internal to the same array (see Figure 4). 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 target port of zero;
 - b. an identification descriptor containing the serial number of the logical unit from which to read the data.
- B. A destination target descriptor containing:
 - a. a relative target port of zero;
 - b. an identification descriptor containing the serial number of the logical unit to which the data is to be written.
- C. A segment descriptor 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.

Then the XCOPY command is sent to the logical unit containing the copy manager (via RPI). 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.

Copy Offload Hypervisor Storage Interfaces

2.2.5 IP Copy Service Operation

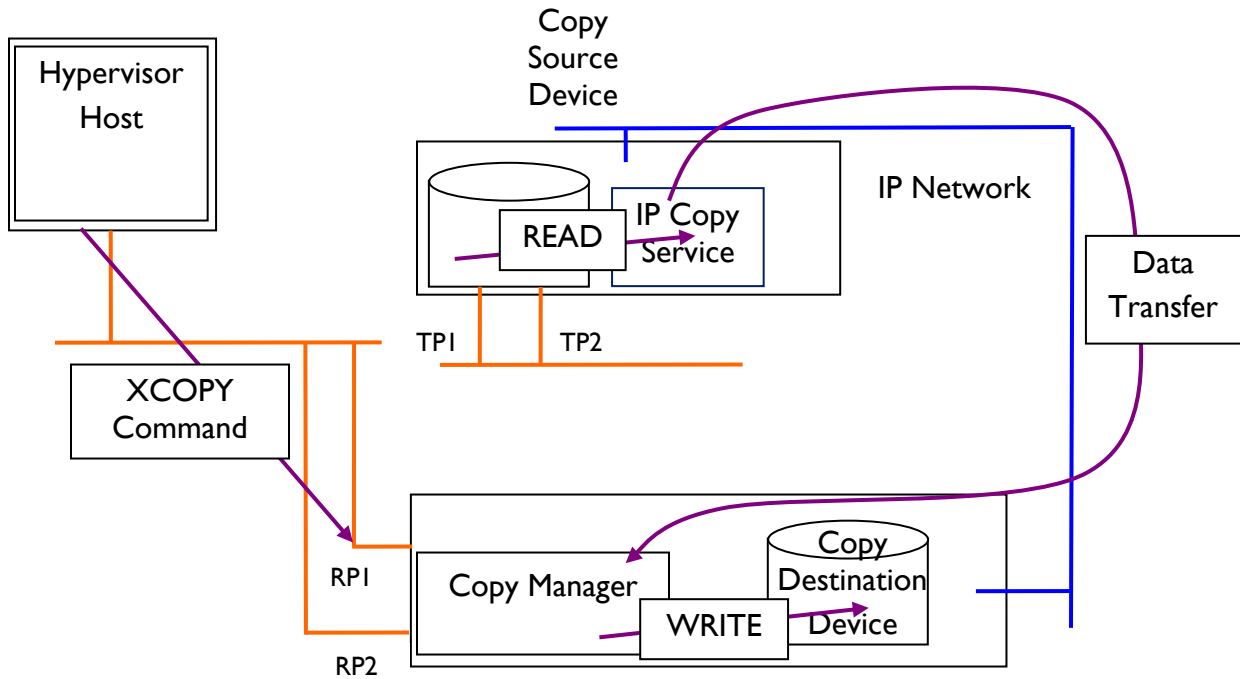


Figure 5 – IP Copy Service Operation

When a host wants to copy blocks from an external source device to an internal destination device (see Figure 5) 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. A source target descriptor containing:
 - a. a relative target port of zero;
 - b. an identification descriptor containing the serial number of the logical unit from which to read the data.
- B. A destination target descriptor containing:
 - a. a relative target port of zero;
 - b. an identification descriptor containing the serial number of the logical unit to which the data is to be written.
- C. An IPv4 or IPv6 target descriptor containing:
 - a. The IP address of the copy service obtained from page 85h of the source logical unit.
- D. A segment descriptor 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.

Copy Offload Hypervisor Storage Interfaces

Then the XCOPY command is sent to the destination logical unit. The copy manager supplies the necessary information from the XCOPY 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 alternate 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.

2.3 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 (6) command (see SCSI-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, and supply a TransportID list in the PERSISTENT RESERVE OUT command that creates the reservation. The Initiator ID 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 ID used in the segment descriptor of the parameter data of the XCOPY command must be the relative port ID of the Initiator ID 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 ID 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.

2.4 General Considerations

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

Copy Offload Hypervisor Storage Interfaces

2.5 Discovery

Several methods may be used by a host to determine when the Extended Copy Command and/or UMAP are available for use with a particular device.

2.5.1 XCOPY command discovery

To determine if the XCOPY 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 XCOPY command. Specifying a reporting option of 001b and a requested operation code of 83h will return the supported status of just the XCOPY command.
- B. Issue a XCOPY command specifying a parameter list length of zero. Devices that support the XCOPY command will return status of GOOD (while performing no copy operations), while devices that do not support the XCOPY command will return a status of CHECK CONDITION, a sense key of ILLEGAL REQUEST, and an ASC/Q of INVALID COMMAND OPERATION CODE.
- C. 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 XCOPY command by the SCSI Device (such as the maximum number of target descriptors supported, maximum number of segment descriptors supported, max transfer size, etc). This command is optional, so a failure of this command cannot be used as a reliable indicator that a SCSI Device does not support the XCOPY command.

2.5.2 UNMAP operation discovery

To determine if unmap operations are available:

- A. Issue a READ CAPACITY (16) command (see SBC-3) and examine the TPE 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 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.

Copy Offload Hypervisor Storage Interfaces

- 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.

2.5.3 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.

2.5.4 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.
- B. Examine the returned data and examine the CDB USAGE DATA to determine if the unmap bit is supported by the WRITE SAME(16) command.

Unlike the XCOPY 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

Copy Offload Hypervisor Storage Interfaces

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.

2.5.5 Logical unit discovery requirement

The host application making use of the XCOPY 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.

2.6 Target Descriptors

2.6.1 Identification target Descriptors

The Identification descriptor target descriptor contains a serial number from page 83h of the logical unit the descriptor is referencing.

2.6.2 Fibre Channel Target Descriptors

For copy offload to or from a Fibre Channel device, the target descriptor should use either

2.6.3 iSCSI Target Descriptors

For copy offload to or from an iSCSI devices, the target descriptor should use either the IPv4 target descriptor or the IPv6 target descriptor and the IPv6 target descriptor extension.

2.6.4 Copy Service Descriptor

For a copy offload operation using an IP Copy Service, the IPv4 target descriptor or the IPv6 target descriptor and the IPv6 target descriptor extension should be used to identify the IP address of the cop service.

2.7 Completion

At the completion of an XCOPY command, the copy manager returns status to the host. If the SCSI Status of the operation is GOOD, then the requested copy has been performed. 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.

Copy Offload Hypervisor Storage Interfaces

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 correct 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).

<Reviewers: Is this sufficient?>

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, a description of those commands is beyond the scope of this white paper. 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 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 masking (the ability to restrict access to a LUN to a subset of the possible initiators), the 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.

When using iSCSI, zoning does not apply, but rather, it is replaced by IP 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 masking considerations also apply to iSCSI connectivity.

Copy Offload Hypervisor Storage Interfaces

Usage of an IP copy service requires IP 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 6 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 XCOPY 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.

Copy Offload Hypervisor Storage Interfaces

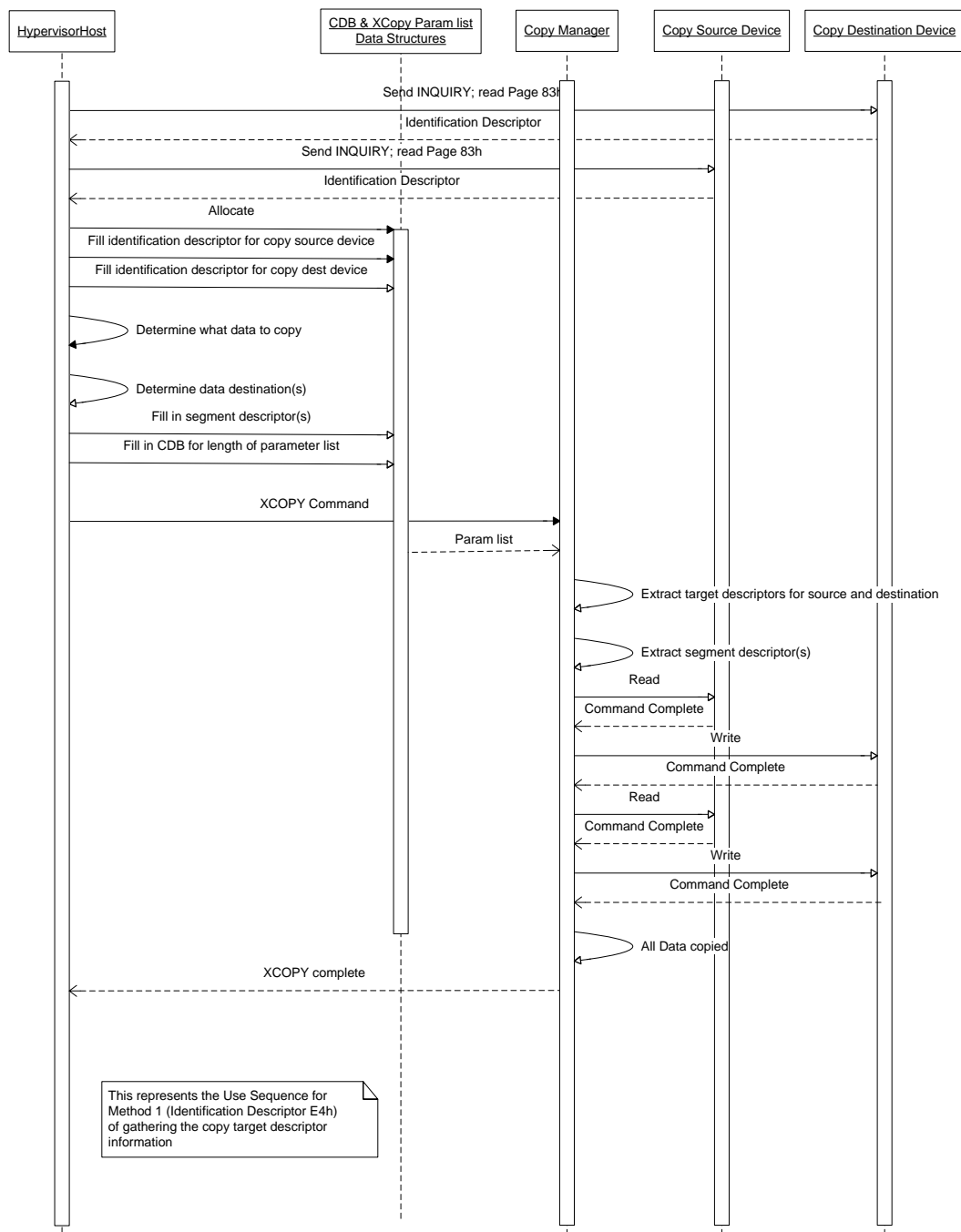


Figure 6 - Copy Manager Sequence Diagram

Copy Offload Hypervisor Storage Interfaces

4.2 Gather Copy Target Descriptor Information

The Relative Port ID 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 ID descriptor has the ASSOCIATION field set to 01b and the DESIGNATOR field set to 4h.

When supplying the Relative Port ID 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.

4.2.1 Method 1 – Identification Descriptor (E4h)

- 1) Determine the relative port ID 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 serial numbers for a suitable descriptor.
- 3) Create an Identification descriptor target descriptor.

4.2.2 Method 2 – N_Port_Name (E0h)

- 1) Determine the relative port ID 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 ID 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.

4.2.3 Method 3 – N_Port_ID (E1h - uncommon)

- 1) Determine the relative port ID 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 ID 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.

4.2.4 Method 4 – N_Port_ID with N_Port_Name checking (E2h - uncommon)

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

Copy Offload Hypervisor Storage Interfaces

- 2) Determine the Logical Unit Identifier of the logical unit when access through the relative port ID 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.

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

- 1) Determine the relative port ID of the initiator Port within the device to be used by the copy manager for access to the copy target device. [<In the iSCSI architecture \(RFC3720\), the initiator and target device are different, as are initiator ports and target ports; so, how can you get the RP-ID of an initiator port, when you can't talk to it?>](#)
- 2) Determine the Logical unit Identifier of the logical unit on the iSCSI device when accessed through the relative port ID determined in step 1.
- 3) Determine the iSCSI target device name.
- 4) Create the appropriate (IPv4 or IPv6) iSCSI target descriptor.

4.2.6 COPY Service descriptor (E5h and EAh)

When an IP based copy service is used, the address of that service is obtained by examining INQUIRY VPD Page 85h to discover the Network Service Descriptor that reports a service type of Copy Service (06h).

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
- 4) Fill in the appropriate IP address descriptor for the Copy Service (if applicable)
- 5) 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

Copy Offload Hypervisor Storage Interfaces

- d) starting LBA on the copy source device
- e) starting LBA on the copy destination device
- 6) Fill in the CDB with the length of the parameter list
- 7) Send the command to the copy destination device

4.5 Copying a Virtual Disk

The XCOPY 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 7), 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.

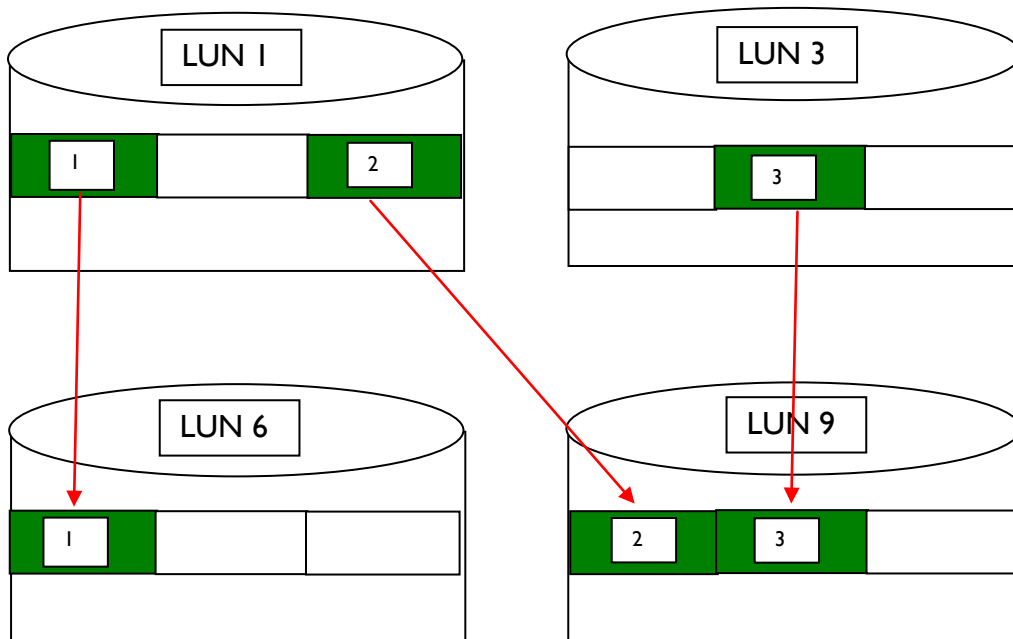


Figure 7 - Example Copy of a Virtual Disk in a Container

A single XCOPY command parameter list may be build (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

Copy Offload Hypervisor Storage Interfaces

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.

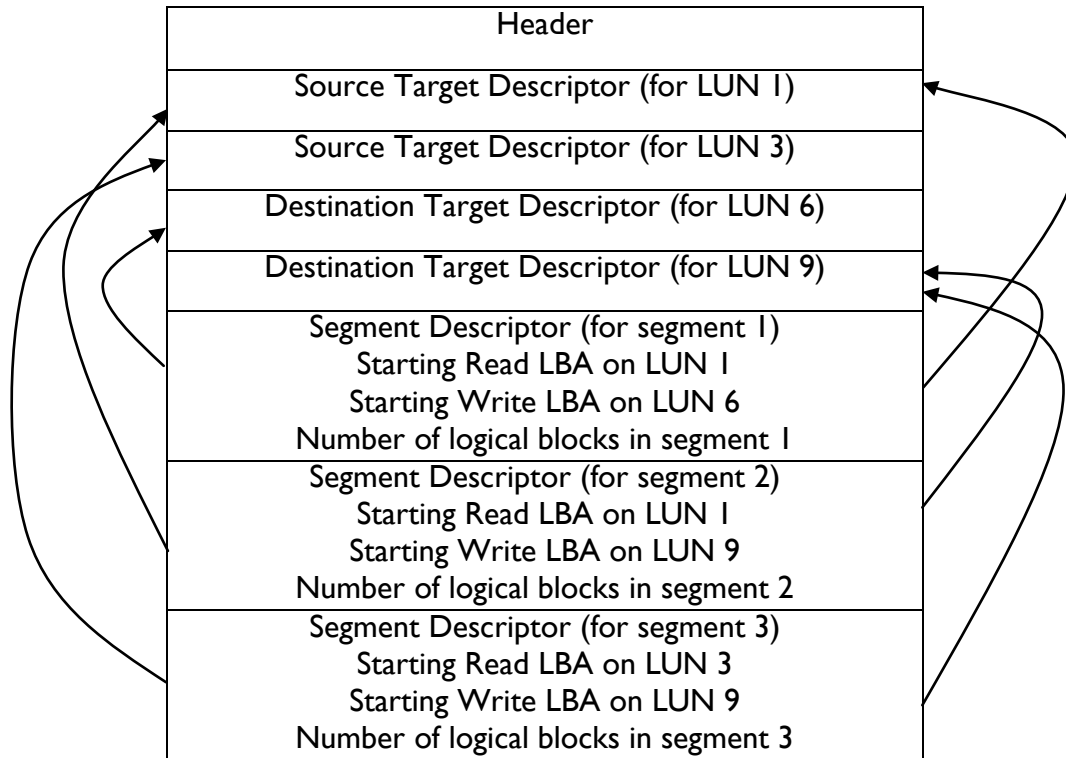


Table 1 - Single Parameter List Example

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

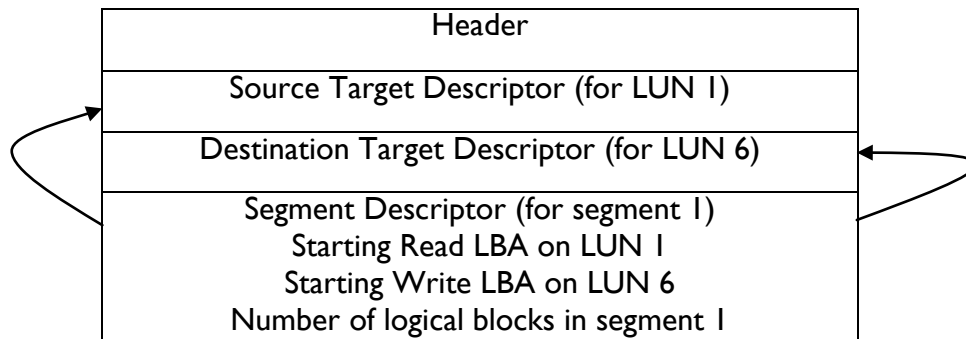


Table 2 – Parameter List to copy Segment 1

Copy Offload Hypervisor Storage Interfaces

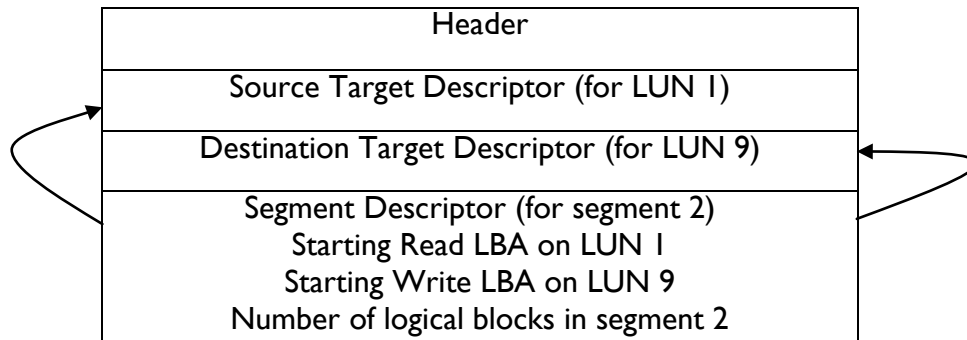


Table 3 – Parameter List to copy Segment 2

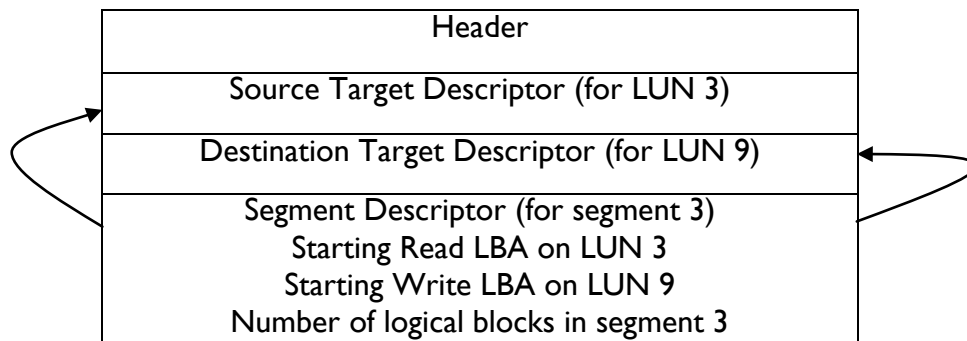


Table 4 – Parameter List to copy Segment 3

In general, the more information contained within a single XCOPY command, the more efficient and higher performance will be the copy offload operation. However, the more information contained within a single XCOPY 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 XCOPY 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 XCOPY 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 XCOPY command that contains multiple segment descriptors, where each segment is described independently. Alternatively, multiple independent XCOPY commands may be used; each copying just a segment of the blocks on the logical unit.

Copy Offload Hypervisor Storage Interfaces

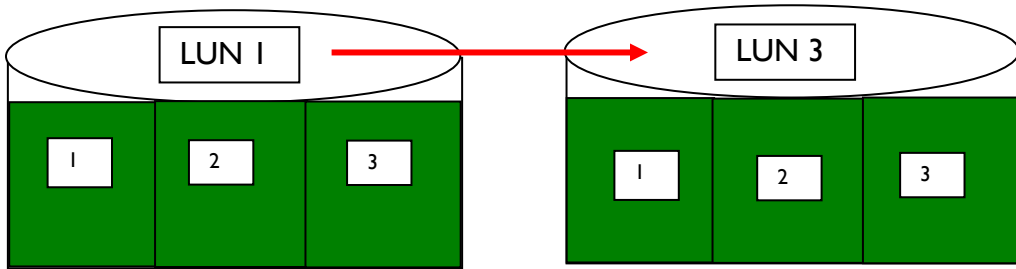


Figure 8 – Example of Copying an Entire Logical Unit

In this example, the entire capacity of the logical unit is copied from LUN 1 to LUN 3 (see Figure 8). A single XCOPY 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 XCOPY 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)

Copy Offload Hypervisor Storage Interfaces

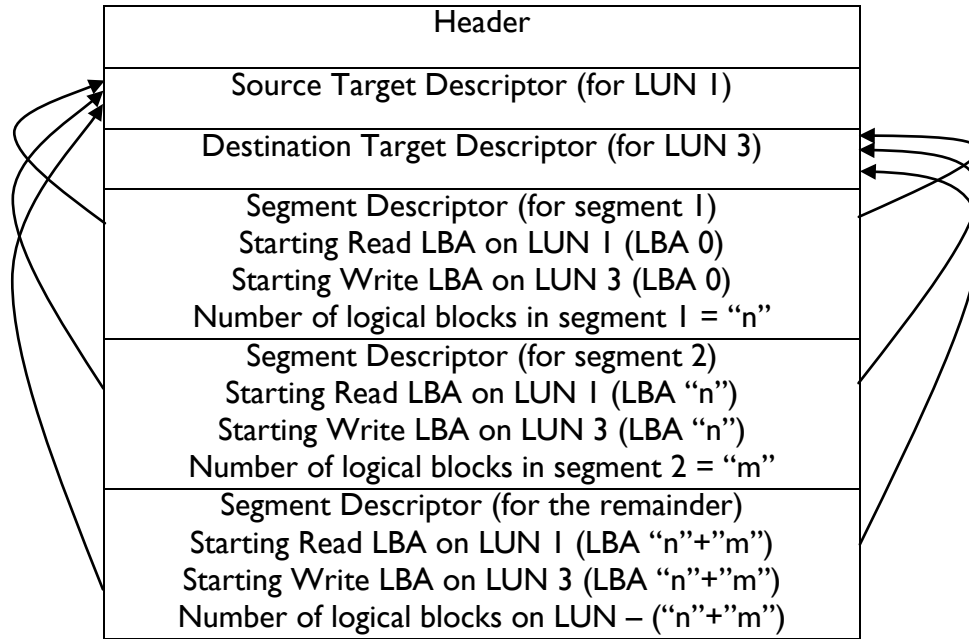


Table 6 – Whole LUN copy (multiple segment descriptors)

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. 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.

Byte	Bit	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 XCOPY commands with shorter transfers specified in the parameter data (segment descriptor) of each command.

Copy Offload Hypervisor Storage Interfaces

Byte	Bit	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 ID (0000h)							
20-n		Serial number from source device page 83h							
n+1-47		Zeros (00h)							
Target Descriptor 1									
48		Descriptor type (E4h)							
49		Zero (00h)							
50-51		Relative port ID (0000h)							
52-n		Serial number 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 (xxxxxxxxxxxxxxxxxxh)							
100-108		Starting LBA on destination device (xxxxxxxxxxxxxxxxxxh)							
109-111		Zeros (000000h)							

Table 8 - Detailed Parameter Data

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.

Copy Offload Hypervisor Storage Interfaces

To copy an entire virtual machine, multiple virtual disks or entire logical units may need to be copied. A virtual machine is stored as a set of meta-data, associated to one or more virtual disks. All meta-data and virtual disks data are stored on one or more physical LUN(s). On each LUN, 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, which has knowledge of the data layout, should identify the LBA addresses and ranges on the physical storage, in such requests to the appropriate LBAs on the physical storage. Then the hypervisor assembles the request and pass those requests to the underlying storage for operation.

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

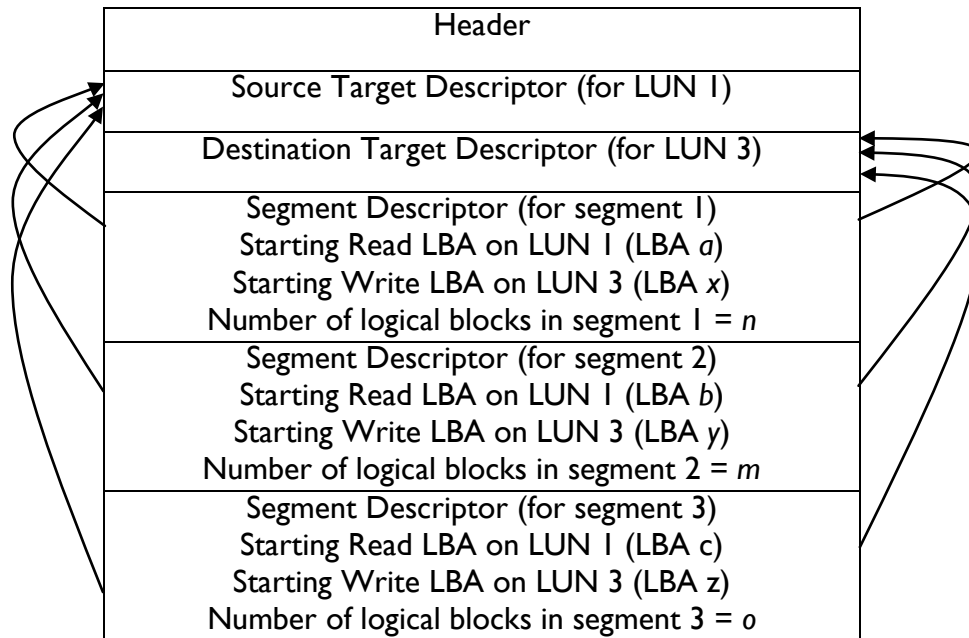


Table 9 – Copy of multiple segments of a LUN

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.

Copy Offload Hypervisor Storage Interfaces

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, which has knowledge of the data layout, should identify the LBA addresses, ranges, and port/array identifiers for the physical 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 XCOPY command(s), being the choice being dependant on performance and complexity considerations. Each XCOPY 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 I describe how to construct the Source Target Descriptors and Destination Target Descriptors for the XCOPY command, such as in the example below:

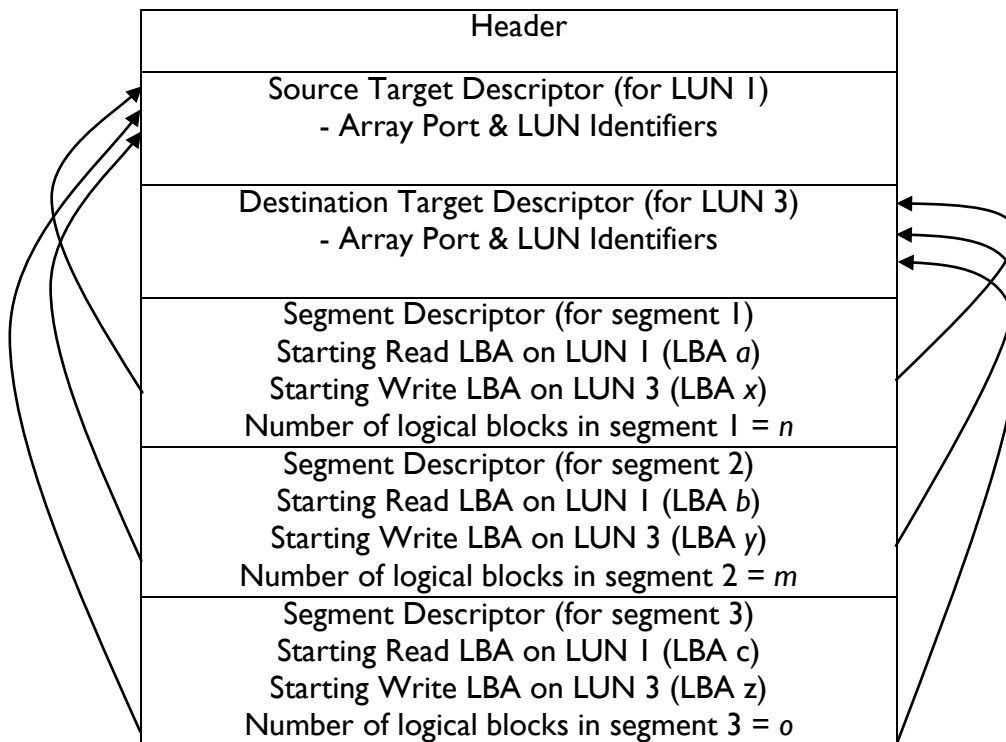


Table 8 – Copy of multiple segments of a LUN across arrays.

Copy Offload Hypervisor Storage Interfaces

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 by multiple virtual machine storage migration operations, as described above, performed in sequence. After the virtual machines have been migrated, the unused space on the source storage may be reclaimed as described in 4.10 below. What is unique in this procedure, with respect to the previous cases, is the option to create XCOPY 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 XCOPY command(s), the choice mostly depending on performance and complexity considerations. Each XCOPY command may contain multiple independent segment descriptors; each segment descriptor is associated with the data of one or more virtual machines.

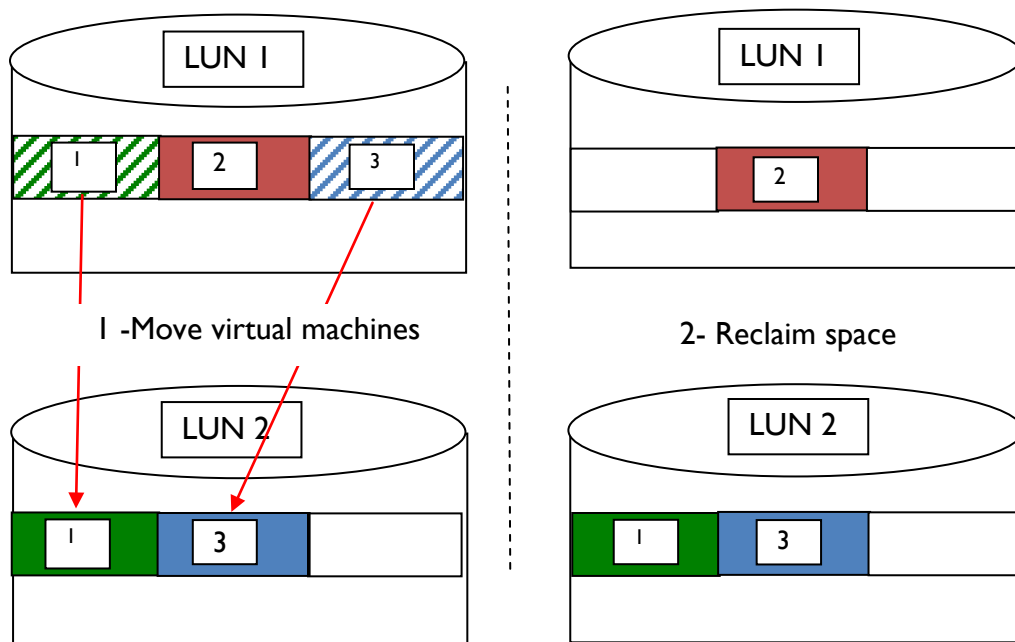


Figure 9 – Simple example of dividing virtual machine

Copy Offload Hypervisor Storage Interfaces

Figure 9 shows a simple example of dividing virtual machines; table 9 shows the XCOPY segment descriptors contained in a single XCOPY 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.

Copy Offload Hypervisor Storage Interfaces

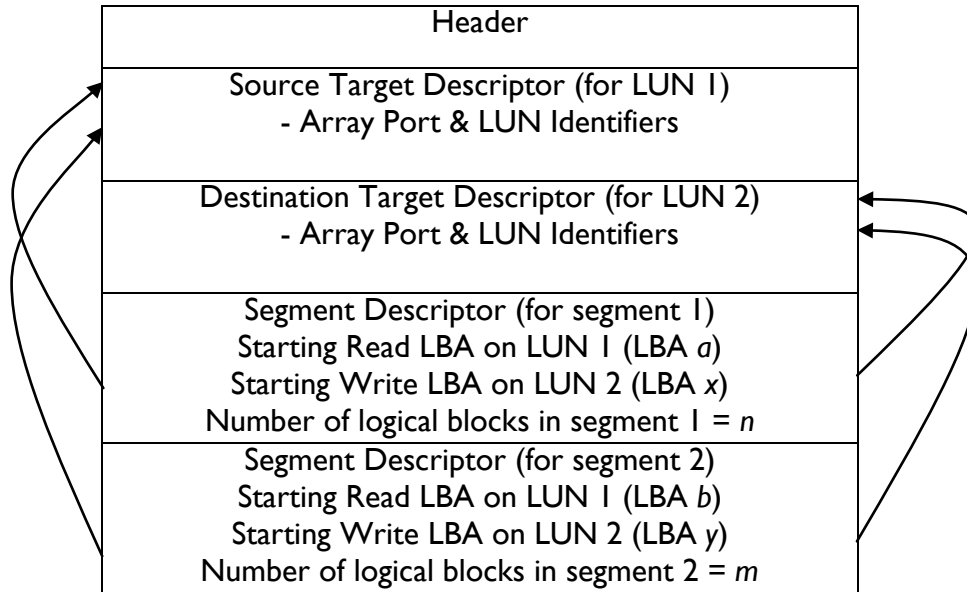


Table9 - Example of segment descriptors for dividing virtual machines

4.10 Reclaiming Unused Space

When a virtual disk is deleted, the space could 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.

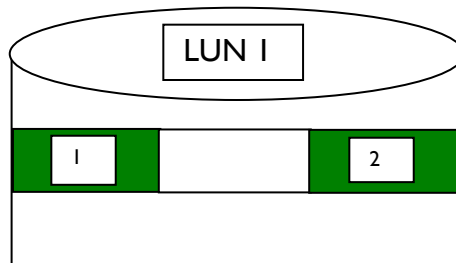


Figure 10 – Freeing Blocks

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

A. The WRITE SAME(16/32) 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 10, to free blocks 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

Copy Offload Hypervisor Storage Interfaces

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). The TPRZ bit in the returned parameter data of the READ CAPACITY(16) indicates when a buffer of zeros is required to reclaim space. There is no method available to determine any necessary non-zero vendor specific data pattern.

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 10, to free 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.

In addition it is important to honor the granularity and offset requirements specified in the Block Limits VPD Page (see SBC-3) for UNMAP operations.

4.1.1 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 (16/32) command with the UNMAP bit set or the UNMAP command to indicate to a virtual disk that reclaim operations may be performed.

To fully support enhanced storage optimizations and increased storage efficiency, the hypervisor should translate the LBA addresses in such requests to the appropriate LBAs on the physical storage and pass those requests to the underlying storage for operation.

5 References

SPC-4 – SCSI Primary Commands – 4

<http://www.t10.org/cgi-bin/ac.pl?t=f&f=spc4r21.pdf>

SBC-3 – SCSI Block Commands – 3

<http://www.t10.org/cgi-bin/ac.pl?t=f&f=sbc3r20.pdf>

UML reference

<http://www.omg.org/technology/documents/formal/uml.htm>